الجمهوريـة الجزائـريـة الديمقراطيـة الشعبيـة
People's Democratic Republic of Algeria
وزارة التعليـم العالـي والبحـث العلمـي
Ministry of Higher Education and Scientific Research

N° Ref: …………

University AbdElhafid Boussouf Mila

Mathematics & Computer Science Institute

Computer Science Department

Thesis Submitted for the Award of the MASTER/STARTUP Degree
In accordance with Ministerial Decision No. "008", amending and supplementing Ministerial Decision "1275"

Specialty: Artificial Intelligence and its Applications (I2A)

Motif-Based Time Series Clustering for Deep Learning in Urban Traffic Forecasting

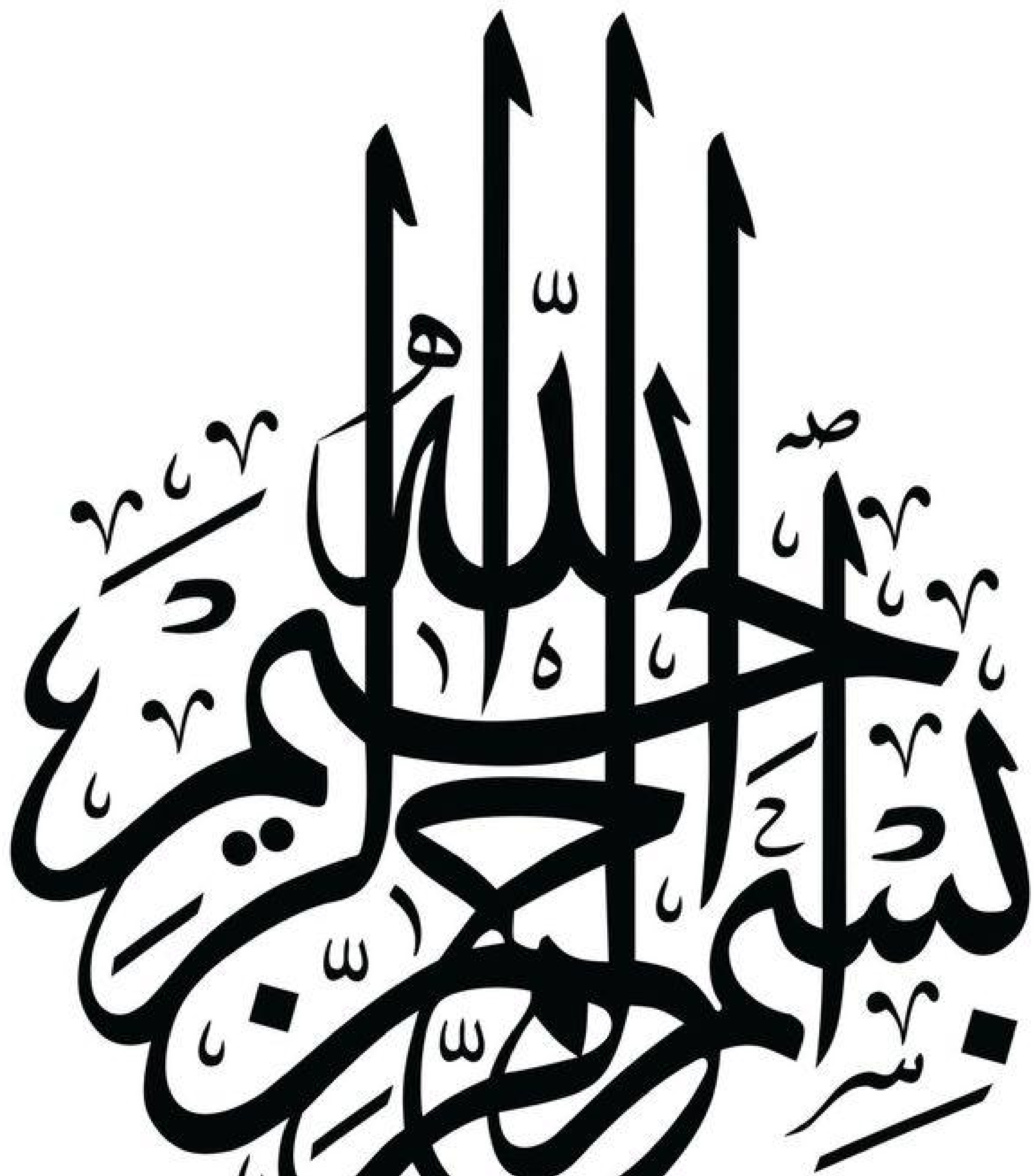
Prepared by:

- ZOUAGHI Amira
- LALMI Norhen

Jury Reviewed :

Mr. Djaaboub Salim	……………	President
Mr. Kimouche Abdelkader	……………	Examiner
Mrs. Talai Meriem	……………	Supervisor

Academic Term: 2024/2025


بِسْمِ اللَّهِ الرَّحْمَٰنِ الرَّحِيمِ

Acknowledgement

First of all, I would like to express my deepest gratitude to Almighty God for granting me the strength, patience, and perseverance needed to complete this work.

I would like to express my sincere thanks and appreciation to my supervisors, ***Mrs. Talai Meriem*** *and* ***Mr. Medjdoub Alaa Eddine****, for their availability, valuable advice, rigorous guidance, and continuous support throughout this project.*

Finally, I dedicate this modest work to my family, my friends, and all those who supported me, whether directly or indirectly, during the preparation of this thesis.

ZOUAGHI AMIRA * LALMI NORHEN**

إ هـ د ا ء

لكل من كان سببًا في أن أصل إلى هذه اللحظة،

لكل من منحني دعمًا، كلمة، أو نظرة أمل في طريقي،

إلى أمي، نبع الحنان وصوت الطمأنينة في أيامي،

إلى أبي، الذي علّمني معنى القوة والإصرار،

إلى أخي، السند والصديق في كل المواقف،

إلى أختيّ العزيزتين، اللتين جعلتا لحياتي طعم الفرح والضحك،

إلى عائلتي الكبيرة، التي كانت دائمًا داعمة وفخورة بي من بعيد أو قريب،

إلى نفسي، التي واصلت رغم التعب، وآمنت أن لكل جهد ثمرة،

لكم جميعًا، أهدي هذه الثمرة المتواضعة، عرفانًا بالجميل، وحبًا لا يوصف بالكلمات.

أميرة

إلى أمي وأبي،

أكتب هذه الكلمات تعبيرًا عن أعمق مشاعر الحب والامتنان. أنتما يا منارة حياتي، ورمز العطاء اللامتناهي. بفضل توجيهاتكما ودعمكما، أصبحت الشخص الذي أنا عليه اليوم. كلمات الشكر تعجز عن وصف ما قدمتموه لي، لكنني آمل أن تعبر هذه المذكرة عن بعض من حبي وتقديري.

إلى إخوتي وأخواتي،

إلى الأخوة والأخوات، يا رفاق الدرب، وشركاء الحياة. شكراً لوجودكم في حياتي، لدعمكم، ولضحكاتكم التي تضيء أيامي. أنتم السند والعون، والرفقة التي لا تقدر بثمن.

إلى صديقاتي،

يا رفيقات الدرب، ويا من يشاركنني الأفراح والأحزان. شكراً لكن على الصداقة، وعلى كل لحظة جميلة قضيناها معاً. أنتنّ كنز لا يقدر بثمن.

إلى زوجي، إلى زوجي،

يا حب حياتي، وشريك دربي. شكراً لك على حبك، على دعمك، وعلى كل ما تبذله من أجل سعادتي. أنت الحضن الدافئ، والملاذ الآمن.

أتمنى أن تعبر هذه الكلمات عن مدى حبي وتقديري لكم جميعاً. أنتم عائلتي، أنتم عالمي، وأنتم كل شيء

نورهان

Abstract

Across most parts of the world, the number of vehicles is rapidly increasing, while road infrastructures remain relatively unchanged. This imbalance has led to growing traffic congestion in many urban areas. Urban traffic control systems can play a vital role in reducing congestion and optimizing traffic flow. Their effectiveness, however, depends on continuous access to accurate traffic flow information, which is typically collected through sensors distributed across the road network.

Since the installation and maintenance of such sensors can be costly, a more economical alternative is to deploy a limited number of sensors and rely on artificial intelligence techniques to predict traffic intensity at various locations throughout the city.

In this study, we propose a deep learning–based architecture that is both complex and efficient for short-term urban traffic forecasting. The proposed approach incorporates clustering algorithms, time series clustering, motif discovery, and the matrix profile, which together enhance the understanding of the internal structure of traffic flow and improve the solution search process for this supervised regression problem. Integrating these techniques enables the system to identify recurrent patterns, structural similarities, and meaningful subsequence relationships within traffic flow data, thereby strengthening the prediction model and improving forecasting accuracy.

Our experimental results show that traffic flow can be accurately predicted by analyzing its variation over the previous hour. Notably, the findings indicate that the system can produce reliable forecasts without requiring contextual information such as the current time, day of the week, or type of day (holiday, working day, etc...).

Keywords : Urban traffic forecasting, Deep learning, Traffic flow prediction, time series clustering, Motif discovery, Matrix profile, Traffic congestion, Sensor data, Short- term prediction, Pattern recognition, Clustering algorithms, Traffic flow variation.

Résumé

Dans la plupart des régions du monde, le nombre de véhicules augmente rapidement, tandis que les infrastructures routières restent relativement inchangées. Ce déséquilibre a entraîné une congestion croissante dans de nombreuses zones urbaines. Les systèmes de contrôle du trafic urbain peuvent jouer un rôle essentiel dans la réduction de la congestion et l'optimisation de la circulation. Leur efficacité dépend toutefois de l'accès continu à des informations précises sur le flux de trafic, généralement collectées via des capteurs répartis sur le réseau routier.

Étant donné que l'installation et la maintenance de ces capteurs peuvent être coûteuses, une alternative plus économique consiste à déployer un nombre limité de capteurs et à utiliser des techniques d'intelligence artificielle pour prédire l'intensité du trafic à différents endroits de la ville.

Dans cette étude, nous proposons une architecture basée sur l'apprentissage profond, à la fois complexe et efficace pour la prévision du trafic urbain à court terme. L'approche proposée intègre des algorithmes de clustering, le clustering de séries temporelles, la découverte de motifs (motif discovery) et le matrix profile, qui ensemble améliorent la compréhension de la structure interne du flux de trafic et optimisent le processus de recherche de solutions pour ce problème de régression supervisée. L'intégration de ces techniques permet au système d'identifier les motifs récurrents, les similarités structurelles et les relations significatives entre les sous-séquences des données de trafic, renforçant ainsi le modèle de prédiction et améliorant sa précision.

Les résultats expérimentaux montrent que le flux de trafic peut être prédit avec précision en analysant ses variations au cours de l'heure précédente. De plus, les résultats indiquent que le système peut produire des prévisions fiables sans nécessiter d'informations contextuelles telles que l'heure actuelle, le jour de la semaine ou le type de journée (jour férié, jour ouvrable, etc...).

Mots-clés : Prévision du trafic urbain, Apprentissage profond, Prédiction du flux de trafic, Clustering de séries temporelles, Découverte de motifs, Matrix profile, Congestion du trafic, Données de capteurs, Prévision à court terme, Reconnaissance de motifs, Algorithmes de clustering, Variation du flux de trafic.

ملخص

في معظم أنحاء العالم، يزداد عدد المركبات بوتيرة سريعة، في حين تبقى البُنى التحتية الطرقية شبه ثابتة دون توسّع يُواكب هذا الارتفاع. وقد أدّى هذا الخلل إلى تفاقم الازدحام المروري في العديد من المناطق الحضرية. يمكن لأنظمة التحكم المروري الحضرية أن تلعب دورًا مهمًا في الحد من الازدحام وتحسين انسيابية حركة المرور، غير أن فعاليتها تعتمد بشكل أساسي على توفر معلومات دقيقة ومستمرة حول تدفّق المرور، والتي يتم جمعها عادةً عبر مجسّات موزعة عبر شبكة الطرق.

وبما أنّ تركيب هذه المجسّات وصيانتها يُعدّ مكلفًا، يُعدّ الحل الاقتصادي الأنسب هو نشر عدد محدود منها والاعتماد على تقنيات الذكاء الاصطناعي للتنبؤ بشدّة الحركة المرورية في مواقع مختلفة داخل المدينة.

في هذه الدراسة، نقترح بنية تعتمد على التعلّم العميق، تتميز بالتعقيد والفعالية في التنبؤ المروري قصير المدى داخل المدن. تعتمد المنهجية المقترحة على خوارزميات التجميع، وتجميع السلاسل الزمنية، واكتشاف الأنماط المتكررة(motif discovery) ، ومصفوفة المؤشرات (Matrix Profile)، والتي تعمل معًا على تعزيز فهم البنية الداخلية لتدفّق المرور وتحسين عملية البحث عن الحلول في هذا النوع من مشكلات الانحدار المُشرف. ويتيح دمج هذه التقنيات للنظام تحديد الأنماط المتكررة، والتشابهات البنيوية، والعلاقات المهمة بين المقاطع الفرعية داخل بيانات تدفق المرور، مما يُقوّي نموذج التنبؤ ويحسّن دقته.

تُظهر نتائج التجارب أنّ حركة المرور يمكن التنبؤ بها بدقة من خلال تحليل تغيّرها خلال الساعة السابقة .كما تُشير النتائج إلى قدرة النظام على إنتاج تنبؤات موثوقة دون الحاجة إلى معلومات سياقية مثل الوقت الحالي، أو يوم الأسبوع، أو نوع اليوم (عطلة، يوم عمل، إلخ).

الكلمات المفتاحية :التنبؤ بالمرور الحضري، التعلم العميق، التنبؤ بتدفق المرور، تجميع السلاسل الزمنية، اكتشاف الأنماط المتكررة، مصفوفة المؤشرات(Matrix Profile) ، الازدحام المروري، بيانات المجسّات، التنبؤ قصير المدى، التعرف على الأنماط، خوارزميات التجميع، تغيّر تدفق المرور.

Contents

List of Figures

List of Tables

General Introduction

Modern urban mobility systems are under growing pressure from population growth, increased vehicle usage, and the demand for sustainability in transport operations. Accurate road traffic forecasting plays a critical role in intelligent transportation systems (ITS), enabling proactive traffic control, efficient routing, congestion mitigation, and improved traveler information services.

Road traffic forecasting using collected data from road sensors refers to the task of predicting future traffic conditions such as vehicle flow, speed, or density based on historical observations and possibly contextual or external factors. It is a time series regression problem where forecasting horizons vary depending on the duration needed to obtain the response (latency) and the planned use of this forecast. In shorter forecasting horizons, the task is often characterized by the need for high temporal resolution and precise modeling of local dynamics, including sudden fluctuations due to traffic signals, incidents, or short-term demand surges (e.g., after large events). In contrast, forecasting traffic over longer horizons requires modeling more abstract, slowly evolving components of the data, such as seasonality (e.g., weekly or monthly cycles), long-range dependencies, and behavioral trends. Despite their differences, forecasting in both regimes benefits from a deeper understanding of underlying temporal structures. Time series subsequence analysis allows studying dynamic behaviors such as congestion formation, rush-hour peaks, or cyclic disruptions. A commonly used technique for processing time series subsequences is motif discovery, which automatically identifies repeated subsequences using the Matrix Profile technique or other methods.

Time series subsequence clustering, when performed without precautions, can yield misleading or trivial results. However, recent methodological advances, especially the use of the Matrix Profile technique, have enabled more robust and interpretable clustering. This motivates the integration of data-driven learning models to enhance both robustness and interpretability of time series subsequences. In particular, representations of local behaviors, such as subsequence clustering based on motifs discovered within a deep forecasting pipeline, provide a structured and explainable path toward more reliable ITS.

In this context, this work explores the integration of Matrix Profile- driven temporal pattern discovery with deep learning models to improve forecasting accuracy and interpretability in road traffic systems. By incorporating motif-level information in meaningful clustering operations when processing historical traffic time series, we aim to guide neural models toward learning more meaningful temporal patterns, enhance their capacity to adapt and generalize across regimes, and facilitate adaptation across both short- and long-term forecasting scenarios. This thesis is structured into three main parts.

Part 1 *"Elementary Time Series and Deep Learning Forecasting Concepts"*: introduces fundamental concepts of time series and deep learning forecasting, providing the theoretical foundations required to understand traffic prediction models.

Part 2 *"Motif-Based Time Series Clustering for Deep Learning"*: focuses on motif-based time series clustering for deep learning, explaining how the Matrix Profile technique is used to identify and cluster recurring patterns to enhance model interpretability.

Part 3 *"TraffiCost Web Application"*: presents the implementation of the proposed methodology through the TraffiCost website, along with the evaluation of forecasting results in real-world scenarios.

Chapter 1

Elementary Time Series and Deep Learning Forecasting Concepts

In the following an introduction of elementary concepts related to Time Series data and their processing by clustering, motif discovery, and forecasting with deep learning architectures especially in road traffic context.

1.1 Time series elementary concepts

1.1.1 Time Series and Time Series Subsequences

A time series (or raw data) is defined as an ordered sequence of real-valued observations recorded at regular time intervals,

$$T = [t_1, t_2, \ldots, t_n] \tag{1.1}$$

where n denotes the length of the time series and each t_i corresponds to an observation at discrete time step i. The time series is univariate if $t_i \in \mathbb{R}$ and it is a multivariate if $t_i \in \mathbb{R}^d$ with d variables observed at each time step i. In traffic applications, T represents collected data specific to roads as vehicle counts, speed, or density measurements and could represent additional information like temperature, humidity, pollution measurements, all recorded at same regular intervals (e.g., every 5 minutes). In our work, we study univariate time series.

A time series subsequence is a contiguous subset of the time series defined as:

$$T_{(i,m)} = \{t_i, t_{(i+1)}, \ldots, t_{(i+m-1)}\} \tag{1.2}$$

where $1 \leq i \leq n - m + 1$ and m is the fixed subsequence length. The collection of all subsequences of length m in T is denoted:

$$S_T^{(m)} = \{T_{(i,m)} \mid 1 \leq i \leq n - m + 1\} \tag{1.3}$$

1.1.2 Time Series Distance Measures

In time series data mining, including tasks such as clustering and motif discovery, quantifying the similarity or dissimilarity between time series is a fundamental step [1]. A distance function $D : \mathbb{R}^n \mathbb{R}^n \to \mathbb{R}^+$ maps a pair of time series or subsequences to a non-negative real number, where a smaller value implies greater similarity.

Several distance functions exist which use depends on the application. In the following the mathematical expressions of Euclidean, Z-normalized Euclidean and Dynamic Time Warping distances. Considering $T^{(1)} = [t_1^{(1)}, t_2^{(1)}, \ldots, t_n^{(1)}]$ and $T^{(2)} = [t_1^{(2)}, t_2^{(2)}, \ldots, t_n^{(2)}]$ two univariate time series of equal length n.

1.1.2.1 Euclidean Distance

The Euclidean distance is the most commonly used metric for comparing two time series of equal length. It is defined as:

$$D_{\text{EUC}}(T^{(1)}, T^{(2)}) = \sqrt{\sum_{i=1}^{n} (t_i^{(1)} - t_i^{(2)})^2} \tag{1.4}$$

While computationally efficient, Euclidean distance is sensitive to misalignments and does not perform well under temporal distortions or shifting [2].

1.1.2.2 Z-normalized Euclidean Distance

To compare time series that may differ in amplitude or offset but are structurally similar, z-normalization is typically applied. It ensures invariance to scaling and shifting, which is crucial in tasks like motif discovery. The z-normalization of a time series $T^{(k)}$ is given by :

$$\tilde{T}^{(k)} = \frac{T^{(k)} - \mu_{T^{(k)}}}{\sigma_{T^{(k)}}} \quad , \quad k \in \{1, 2\} \tag{1.5}$$

where $\mu_{T^{(k)}}$ and $\sigma_{T^{(k)}}$ are the mean and standard deviation of $T^{(k)}$, respectively. The z-normalized Euclidean distance is then computed as:

$$D_{\text{ZEUC}}(\tilde{T}^{(1)}, \tilde{T}^{(2)}) = \sqrt{\sum_{i=1}^{n} (\tilde{t}_i^{(1)} - \tilde{t}_i^{(2)})^2} \tag{1.6}$$

For subsequence series analysis (e.g., with Matrix Profile), distances are often computed between z-normalized subsequences $T_{(i,m)}$ and $T_{(j,m)}$:

$$D(T_{(i,m)}, T_{(j,m)}) = \sqrt{\sum_{k=0}^{m-1} (\tilde{t}_{(i+k)} - \tilde{t}_{(j+k)})^2} \tag{1.7}$$

This formulation is invariant to mean and variance, enabling the detection of motifs that differ in absolute scale but are of similar shape [3].

1.1.2.3 Dynamic Time Warping (DTW)

Dynamic Time Warping is a non-linear distance measure that allows temporal misalignments between sequences by computing an optimal alignment path. It is defined via dynamic programming as:

$$DTW(i, j) = \|t_i^{(1)} - t_j^{(2)}\| + \min \begin{cases} DTW(i-1, j) \\ DTW(i, j-1) \\ DTW(i-1, j-1) \end{cases} \tag{1.8}$$

with boundary conditions $DTW(0,0) = 0$ and $DTW(i,0) = DTW(0,j) = \infty$. The final distance is $DTW(n,n)$. DTW is effective for handling local warping and misalignment but is computationally more expensive (typically $O(n^2)$) [4] , [5].

1.2 Time Series Clustering and Subsequence Clustering

Time series clustering aims to partition a set of time series $\Psi = \{T^{(1)}, T^{(2)}, \ldots, T^{(N)}\}$ into k clusters $C = \{C_1, C_2, \ldots, C_k\}$ such

$$C_k \subset \Psi, \quad C_i \cap C_j = \emptyset \quad \text{for } i \neq j \tag{1.9}$$

Time series subsequence clustering, in contrast to time series clustering, operates on a single time series T by extracting overlapping subsequences $T_{(i,m)}$ and clustering them based on their mutual similarity. Each subsequence $T_{(i,m)} \in \mathbb{R}^m$ can be treated as a point in an m-dimensional space.

1.2.1 k-means for time series

The literature presents several clustering methods for time series clustering and subsequences clustering, each with its advantages and drawbacks such as : k-means , density-based methods, Hidden Markov Model (HMM)-based clustering,Hierarchical clustering, Spectral Clustering [6].

Let a set of subsequences $S_T^{(m)} = \{T_{(i,m)} = [t_i, t_{(i+1)}, \ldots, t_{(i+m-1)}] \mid 1 \leq i \leq n - m + 1\}$ with a given window length m. The goal of k-means clustering is to partition these subsequences into k clusters $\{C_1, C_2, \ldots, C_K\}$ by minimizing the within-cluster variance. Formally, k-means minimizes the objective function:

$$J = \sum_{j=1}^{k} \left(\sum_{T_{(i,m)} \in C_j} \|T_{(i,m)} - \mu_j\|^2 \right) \tag{1.10}$$

where μ_j is the centroid of cluster C_j; defined as:

$$\mu_j = \frac{1}{|C_j|} \sum_{T_{(i,m)} \in C_j} T_{(i,m)} \tag{1.11}$$

The algorithm of k-means proceeds iteratively on the following steps:

1. Initialize k centroids randomly or based on previous knowledge.
2. Assign each subsequence $T_{(i,m)}$ to the nearest centroid according to a distance measure (e.g., Euclidean distance or z-normalized Euclidean distance).
3. Update centroids μ_j based on current assignments.
4. Repeat steps (2) and (3) until convergence measured by assignments stabilize or objective J decreases below a tolerance.

Also, in time series clustering, z-normalization of subsequences is often applied prior to distance computation [1]:

$$\tilde{T}_{(i,m)} = \frac{T_{(i,m)} - \mu_{T_{(i,m)}}}{\sigma_{T_{(i,m)}}} \tag{1.12}$$

1.2.2 Silhouette Coefficient

The clustering effectiveness is measured by the intra-cluster similarity which should be maximized and a minimized inter-cluster similarity. Several indexes are used to assess the effectiveness of a time series clustering, such as Silhouette Coefficient, Davies–Bouldin Index, Dunn Index, and WCSS (Within-Cluster Sum of Squares) [7]. Internal clustering validation metrics focus on the compactness and separation of clusters. In particular, the Silhouette Coefficient evaluates the degree of confidence in the assignment of each point to its cluster.

For a subsequence $T_{(i,m)}$ assigned to cluster C_p:

- Let $a(i)$ denote average distance from $T_{(i,m)}$ to all other subsequences in C_p:

$$a(i) = \frac{1}{|C_p| - 1} \sum_{T_{(i,m)} \in C_p} D(T_{(i,m)}, T_{(j,m)}) \tag{1.13}$$

  Where $D(.,.)$ is a distance function (e.g., Euclidean or DTW).

- Let $b(i)$ denote the minimum average distance from $T_{(i,m)}$ to all subsequences in other clusters C_q, $q \neq p$:

$$b(i) = \min_{q \neq p} \left(\frac{1}{|C_q|} \sum_{T_{(j,m)} \in C_q} D(T_{(i,m)}, T_{(j,m)}) \right) \tag{1.14}$$

- The silhouette score of $T_{(i,m)}$ is defined as:

$$s(i) = \frac{b(i) - a(i)}{\max\{a(i), b(i)\}} \quad \text{-1} \leq s(i) \leq 1 \tag{1.15}$$

The overall silhouette score is the mean over all data points:

$$S = \frac{1}{N} \sum_{i=1}^{N} s(i) \tag{1.16}$$

Near to 1 values mean well-clustered and near to 0 values mean overlapping clusters. It provides a per-sample and global measure of clustering quality. Its main drawback is that it is computationally expensive for large datasets.

1.3 Motif Discovery in Time Series

A motif is defined as a pair or set of pairs of similar subsequences in a time series. Motif discovery aims to identify k most similar pairs of subsequences:

$$\text{motif}_k = \arg \min_{\substack{k \\ |i-j|>w}} D(T_{(i,m)}, T_{(j,m)}) \tag{1.17}$$

Where $D(.,.)$ is a distance measure (typically Euclidean distance), and w is an exclusion zone to avoid trivial matches (overlapping similar subsequences).

1.3.1 Matrix Profile

The Matrix Profile method is often used to guide meaningful subsequence comparisons efficiently. It is based on MASS (Mueen's Algorithm for Similarity Search) and provides a scalable solution for motif and discord discovery.

The resulting matrix profile $MP \in \mathbb{R}^{n-m+1}$ is a vector that stores the distance to the nearest non-trivial matching subsequence for each $T_{(i,m)}$, formally defined as:

$$MP_T^{(m)}[i] = \min_{\substack{1 \le j \le n-m+1 \\ |i-j| \ge m}} \text{dist}(T_{(i,m)}, T_{(j,m)}) \tag{1.18}$$

where dist is the z-Euclidean distance between $T_{(i,m)}$ and its nearest non-trivially overlapping neighbor in $S_T^{(m)}$. The default exclusion zone is $w = m/2$.

Additionally, the matrix profile index vector PI stores the location (index) of the nearest neighbor to each subsequence:

$$PI_T^{(m)}[i] = \arg \min_{\substack{1 \le j \le n-m+1 \\ |i-j| \ge m}} \text{dist}(T_{(i,m)}, T_{(j,m)}) \tag{1.19}$$

Motifs M correspond to the lowest values in MP while discords correspond to the highest values:

$$M = \{T_{(j,m)} \in S_T^{(m)} \mid \text{dist}(T_{(j,m)}, T_{(i^*,m)}) \le \epsilon\} \tag{1.20}$$

Matrix Profile algorithms such as STOMP [8], SCRIMP++ [9], and SCAMP [10] reduce the computational problem of motif discovery, lowering complexity from the naive $O(n^2)$ pairwise distance computation to amortized near-linear time. These algorithms enable exact, scalable motif discovery on long time series that were previously intractable [11].

In the road traffic context, motifs often correspond to recurring congestion patterns or temporal traffic behaviors such as morning rush hours.

1.4 Fundamentals of Deep Learning Architectures

Effective traffic forecasting models must handle challenges such as non-stationarity, missing values, and anomalies. Deep learning methods have shown particular effectiveness in learning such complex, non-linear patterns from time series data. Several architectures can be used in a deep learning model including: Recurrent Neural

Networks (RNNs), Long Short-Term Memory networks (LSTMs), Temporal Convolutional Networks (TCNs), and more recently Transformer-based models, commonly used for forecasting tasks.

1.4.1 Artificial Neural Networks (ANN)

Artificial Neural Networks (ANNs) are computational models inspired by the structure and functioning of biological neural systems. Formally, an ANN is a directed graph composed of layers of units (neurons), where each unit performs a weighted summation followed by an activation function.

Let $X = [x_1, x_2, \dots, x_n]^T$ be the input vector and $W = [w_1, w_2, \dots, w_n]^T$ the weight vector for a given neuron. The output of a single artificial neuron can be expressed as:

$$z = \sum_{i=1}^{n} w_i x_i + b = W^T X + b, \tag{1.21}$$

$$y = \phi(z), \tag{1.22}$$

where b is the bias term and $\phi(\cdot)$ is a nonlinear activation function such as sigmoid, tanh, or ReLU [12].

The general structure of ANNs is organized into three main types of layers:

- **Input Layer:** Each neuron corresponds to an input feature. For example, with three features, the input layer contains three neurons.
- **Hidden Layers:** A multilayer perceptron (MLP) consists of one or more hidden layers, each containing several neurons that transform and refine the information received from the input layer.
- **Output Layer:** This layer produces the final prediction. If multiple outputs are required, the number of neurons corresponds to the number of predicted values.

ANNs include perceptrons, MLPs, and recurrent neural networks (RNNs), which are described in the following sections, as well as other architectures such as convolutional neural networks (CNNs).

1.4.2 Single-Layer Neural Network (Perceptron)

The perceptron is the simplest ANN, consisting of one input layer directly connected to one output neuron (or output layer) with no hidden layers. It can only learn linearly separable patterns and performs binary classification based on a linear threshold function. The decision rule is given by:

$$y = \begin{cases} 1, & \text{if } W^T X \geq t, \\ 0, & \text{otherwise,} \end{cases} \tag{1.23}$$

where t is the decision threshold. Although limited to linearly separable problems, the perceptron forms the foundation for deeper architectures [12].

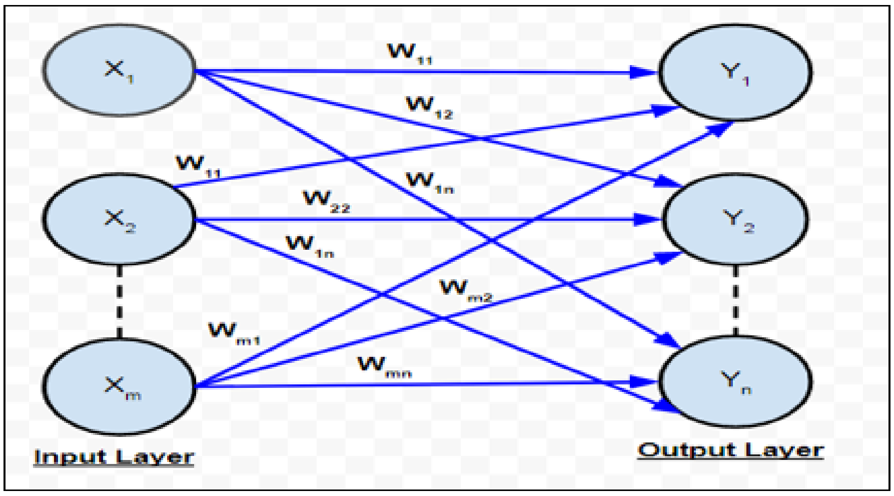


Figure 1.1: Single layer perceptron [13]

1.4.3 Multi-Layer Perceptron (MLP)

A Multi-Layer Perceptron (MLP) is a feedforward neural network with one or more hidden layers between the input and output. Each hidden layer applies an affine transformation followed by a nonlinear activation. Let $X \in \mathbb{R}^n$ be the input and $y \in \mathbb{R}^m$ the output. For a network with L layers, the transformations are recursively defined as:

$$h^{(0)} = X \tag{1.24}$$

$$h^{(l)} = \varphi\left(W^{(l)} h^{(l-1)} + b^{(l)}\right) \quad \text{for } l = 1..L \tag{1.25}$$

$$\hat{y} = h^{(L)} \tag{1.26}$$

Each layer is fully connected, meaning every neuron in layer $l-1$ is connected to every neuron in layer l. MLPs are universal function approximators and are widely used in regression and classification tasks.

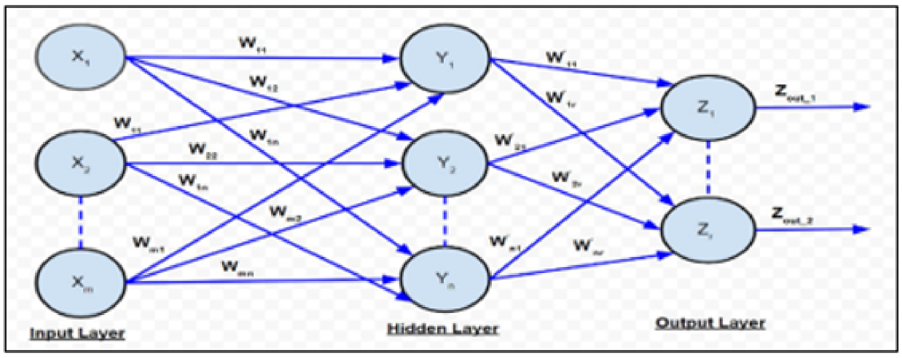


Figure 1.2: Multi-Layer Perceptron [13]

1.5 Recurrent Neural Networks (RNNs)

Recurrent Neural Networks (RNNs) are neural architectures designed for modeling sequential data, such as time series, speech, and text. Unlike feedforward networks, RNNs maintain an internal state that captures dependencies over time. Let $X_t \in \mathbb{R}^d$ denote the input at time t and $h_t \in \mathbb{R}^h$ the hidden state. The standard RNN updates are given by:

$$h_t = \varphi(W_{xh}x_t + W_{hh}h_t + b_h) \tag{1.27}$$

$$\hat{y}_t = \psi(W_{hy}h_t + b_y) \tag{1.28}$$

Where $\varphi(\cdot)$ is typically a tanh or ReLU activation, and $\psi(\cdot)$ is the output activation (e.g., softmax for classification). The hidden state acts as a memory, enabling temporal pattern learning.

However, standard RNNs suffer from the vanishing gradient problem, limiting their ability to model long-term dependencies. To address this, gated variants such as LSTM and GRU have been developed.

1.5.1 Long Short-Term Memory (LSTM)

LSTM architecture introduces a memory cell with gated mechanisms to control information flow. The gates allow selective reading, writing, and forgetting of information, enabling LSTMs to capture long-term patterns across thousands of timesteps. The LSTM updates consist of:

$$f_t = \sigma\left(W_f[h_{t-1}, x_t] + b_f\right) \quad \text{(Forget gate)} \tag{1.29}$$

$$i_t = \sigma\left(W_i[h_{t-1}, x_t] + b_i\right) \quad \text{(Input gate)} \tag{1.30}$$

$$\tilde{C}_t = \tanh\left(W_C[h_{t-1}, x_t] + b_C\right) \quad \text{(Cell candidate)} \tag{1.31}$$

$$C_t = f_t \odot C_{t-1} + i_t \odot \tilde{C}_t \quad \text{(Cell state update)} \tag{1.32}$$

$$o_t = \sigma\left(W_o[h_{t-1}, x_t] + b_o\right) \quad \text{(Output gate)} \tag{1.33}$$

$$h_t = o_t \odot \tanh(C_t) \quad \text{(Hidden state)} \tag{1.34}$$

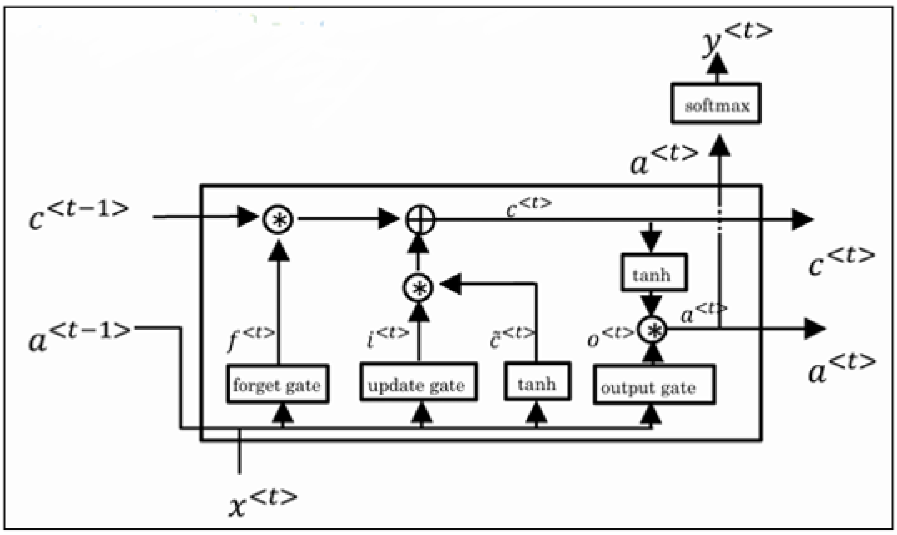


Figure 1.3: LSTM neuron architecture [13]

1.5.2 Gated Recurrent Unit (GRU)

GRUs are a simplified variant of LSTM that merge the forget and input gates into a single update gate, and use a reset gate to control the contribution of the previous state:

$$z_t = \sigma\left(W_z[h_{t-1}, x_t] + b_z\right) \quad \text{(Update gate)} \tag{1.35}$$

$$r_t = \sigma\left(W_r[h_{t-1}, x_t] + b_r\right) \quad \text{(Reset gate)} \tag{1.36}$$

$$\tilde{h}_t = \tanh\left(W_h[\, r_t \odot h_{t-1},\, x_t] + b_h\right) \tag{1.37}$$

$$h_t = (1 - z_t) \odot h_{t-1} + z_t \odot \tilde{h}_t \tag{1.38}$$

GRUs reduce computational complexity while preserving long-term memory capacity and are commonly used in sequence modeling tasks with limited training data.

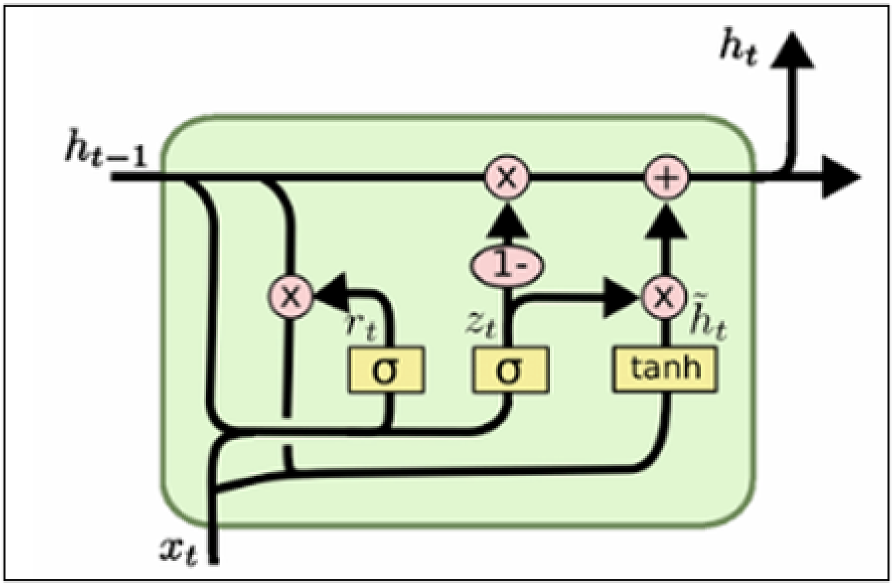


Figure 1.4: Architecture of Gated Recurrent Unit [13]

1.6 Formalisation of road Traffic Forecasting with time series data

Let $T \in \mathbb{R}^{nd}$ be a time series with n timesteps and $d = 1$ for a single studied variable, and let $x_t \in \mathbb{R}$ denote the observation at time t. The goal of traffic forecasting is to estimate a sequence of future values

over a prediction horizon H.

Formally, the forecasting function f maps a historical subsequence (window) of length m to a future sequence:

$$\hat{X}_{T+1:T+H} = f(X_{T-m+1}, X_{T-m+2}, \ldots, X_T), \tag{1.39}$$

where f can be implemented using statistical models (ARIMA, Kalman Filters) or data-driven models (LSTM, TCN, Transformer-based architectures).

The objective of these models is to reduce error metrics (loss functions) such as Mean Squared Error, Root Mean Squared Error (RMSE), or Mean Absolute Error (MAE):

$$l(\theta) = \frac{1}{N} \sum_{i=1}^{N} \| f_\theta(X^{(i)}) - y^{(i)} \|^2 \tag{1.40}$$

$$\text{MAE} = \frac{1}{H} \sum_{h=1}^{H} |\hat{x}_{T+h} - x_{T+h}| \tag{1.41}$$

$$\text{RMSE} = \sqrt{\frac{1}{H} \sum_{h=1}^{H} (\hat{x}_{T+h} - x_{T+h})^2} \tag{1.42}$$

These metrics are commonly used to evaluate forecasting performance across different horizons and traffic conditions.

A general deep learning model can be formulated as a function $f_\theta : \mathbb{R}^{nd} \rightarrow \mathbb{R}^h$ parameterized by weights θ, mapping a time-series input $T = [t_1, t_2, \ldots, t_n]$ to a prediction:

$$\hat{y}_{t+1:t+h} = f_\theta([t_1, t_2, \ldots, t_t]) \tag{1.43}$$

where h is the forecasting horizon.

In short-term forecasting, h is small and typically ranges from a few minutes up to one hour ahead (e.g., 5–15 minutes). Models such as ARIMA, SVR, LSTM, and TCNs are commonly used.

Long-term forecasting covers horizons from several hours to multiple days. The forecasting function becomes:

$$\hat{X}_{t+1:t+H} = f_\theta(x_{1:t},\, E_{1:t}) \tag{1.44}$$

since long-term prediction is more sensitive to error accumulation E and requires more robust generalization.

Chapter 2

Motif-Based Time Series Clustering for Deep Learning

2.1 Introduction

In this work we aim to improve DL models generalisation for time series forecasting task based on subsequence clustering. The goal of time series subsequence clustering is to group similar subsequences extracted from a longer time series into meaningful clusters.

However, achieving meaningful clustering of time-series subsequences faces several challenges. First, as highlighted by [14] , subsequences extracted using a sliding window tend to be highly overlapping and therefore extremely similar, which leads to trivial matches and prevents the discovery of truly meaningful patterns. In addition, meaningful clustering requires an appropriate normalization strategy to ensure scale invariance. Yet, as noted by [15], improper normalization may distort the underlying patterns and mislead the clustering process. Finally, specifying a predefined number of clusters (k) may not reflect the true structure of the data, and the resulting clusters must remain interpretable within the specific application domain.

Our approach addresses these issues by assuring an appropriate subsequence processing. It normalizes subsequences to zero mean and unit variance with z-normalization. Also, it employs the silhouette validation metric to select an optimal number of clusters (k) with semantically meaningful patterns. Matrix profile method is integrated into the k-means algorithm employed for clustering to fit with time series clustering characteristics. More details are provided in the following sections.

2.2 Related work

The authors of [16] proposes a forecasting framework that enhances an attention-based LSTM by adding features derived from the Matrix Profile (MP), computed on each COVID-19 indicator to capture weekly patterns. The MP reveals similar subsequences (motifs) and anomalies (discords), and these structural signals are transformed into features and concatenated with the model inputs. This helps the attention mechanism focus on historically similar patterns and detect deviations. Two variants (LSTM-MatAtt and LSTM-RelAtt) are evaluated, both outperforming classical baselines and LSTMs without MP.

The pipeline of their approach is composed of three steps:

1. MP-based subsequence clustering.
2. Cluster-specific LSTM models with attention.
3. Validation on U.S. COVID-19 data. Results show superior accuracy and practical use for resource planning and emerging-variant detection.

In contrast, study [17] clusters full time-series subsequences to build specialized datasets trained with LSTMs optimized using the ARDE algorithm. Although results are acceptable, the methodology suffers from a major theoretical flaw: subsequences are clustered with k- means despite overlapping segments, making the clustering unreliable. However, the ARDE Adaptive Random Differential Evolution based hyperparameter optimization remains valid and effective.

2.3 Overview of the traffic flow forecasting based on motif discovery approach

Figure 2.1 shows the two main components of our proposed approach which are explained in the following:

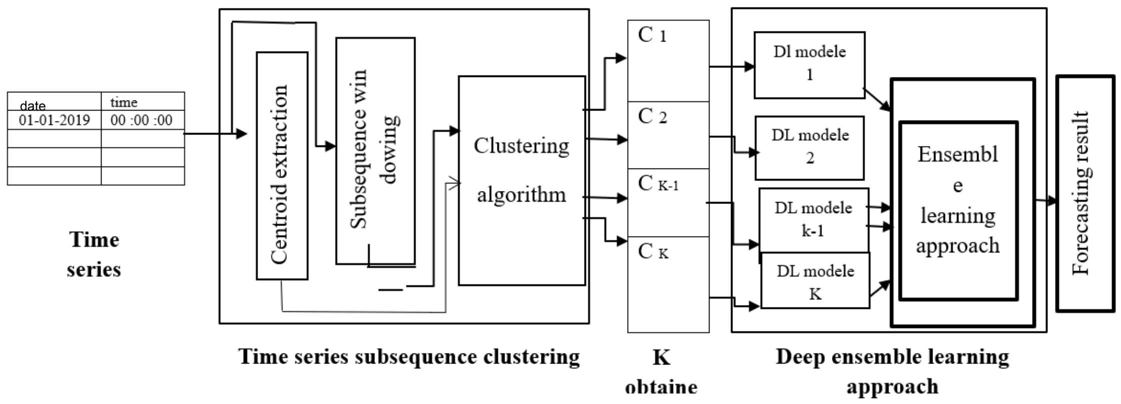


Figure 2.1: The two main components of the proposed approach

2.3.1 Motif discovery-based clustering

Based on the complete time series, the objective of this first stage is to group structurally similar subsequences. The Matrix Profile is employed as the primary tool for extracting motifs, i.e., low-distance subsequences. A filtering process is then applied, where only the most representative motifs are retained, while high-distance subsequences are discarded as they typically correspond to anomalous behaviors (discords). The retained motifs are aggregated into groups according to the presence of a shared common subsequence, producing small initial clusters that serve as proto-clusters for the subsequent clustering phase. This is justified by the fact that all subsequences within each group exhibit similarity through transitivity. Once these initial groups are established, their representative centroids are computed, forming the basis for the final clustering step, which is performed using the k-means algorithm.

2.3.2 Deep ensemble learning

(k)-LSTM-based models are fitted to the obtained (k) clusters. A splitting preprocessing step is performed to split each cluster's subsequences into training, validation, and test sets. The LSTM hyperparameters are initially extracted from the results. Each LSTM model is then trained on its corresponding cluster independently of the others. depending on the chosen ensemble learning strategy.

During inference, a new subsequence is first assigned to the nearest cluster by measuring its distance to the cluster centroid. Once the belonging cluster is determined, the subsequence is forwarded to the specific LSTM model trained for that cluster to produce the forecast. The final behavior (whether models are combined or used separately) depends on the ensemble design chosen in advance.

This approach goes on the following steps using specific methods as shown in figure 2.2:

1. After dataset preprocessing step, windowing the time series into overlapping subsequences is an elementary step of our approach.

2. Matrix Profile (MP) and Matrix Profile Index (PI) computation using z-normalized Euclidean distance.

3. Motif discovery via best-neighbor pairs and index-similarity grouping to form motif sets.

4. Centroid extraction from each motif set to made initial cluster prototypes.

5. Subsequence clustering, initialized by motif centroids and using extracted subsequences into (k) clusters using DTW.

6. Local deep models trained one per cluster.

7. At inference, ensembling to produce the final forecast.

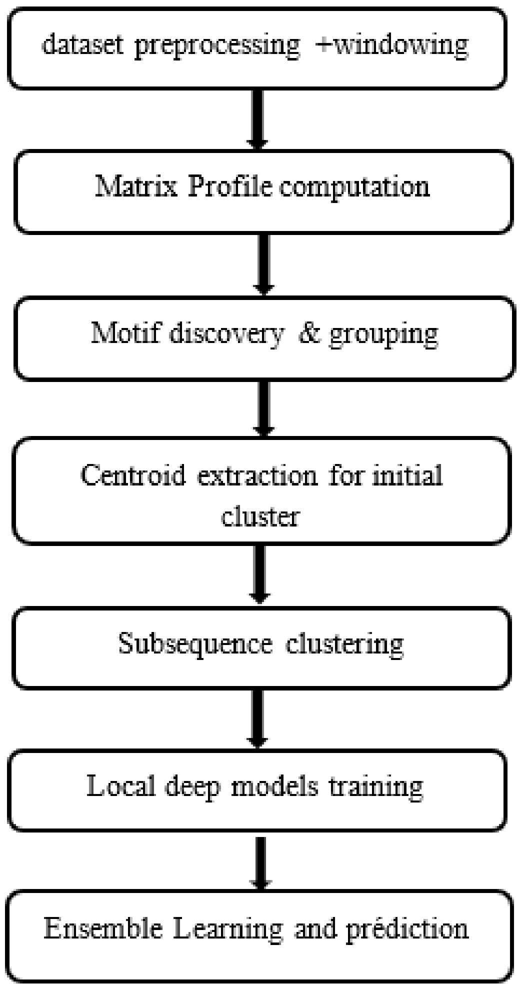

dataset preprocessing +windowing
Matrix Profile computation
Motif discovery & grouping
Centroid extraction for initial cluster
Subsequence clustering
Local deep models training
Ensemble Learning and prédiction

Figure 2.2: Steps of the proposed forecasting approach

2.4 Details of the proposed approach

2.4.1 Datasets and preprocessing

Three datasets of univariate traffic flow collected from different locations, namely 1280- S, M50-N, and M1-N, were employed in this study. The first dataset corresponds to a road segment in San Francisco, USA, while the other two represent traffic flow from motorways in Dublin, Ireland. The San Francisco dataset was obtained from the Caltrans Performance Measurement System (PeMS), whereas the Irish datasets were sourced from the Transport Infrastructure Ireland (TII) database. The corresponding access links to these repositories are provided below:

- TII: `https://trafficdata.tii.ie`
- PeMS: `http://pems.dot.ca.gov`

The data collection period extends from March 28, 2018 to March 31, 2019. Each dataset consists of three attributes: Date, Time, and Count, where the Count denotes the average number of vehicles recorded by road sensors at regular 15-minute intervals. For consistency in the subsequent analysis, the time intervals were resampled into 15-minute windows, resulting in an equal number of samples across identical time spans. Further details about these datasets are provided in Table 2.1.

Dataset	Segment	Location	Detector No.	Detector Type	Direction	Source (Database)
1280-S	Highway 1280	San Francisco, USA	#S-1280	Inductive Loop	Southbound	PeMS (Caltrans)
M50-N	M50	Dublin, Ireland	#M50-07	Automatic Traffic Counter	Northbound	TII (Ireland)
M1-N	M1	Dublin, Ireland	#M1-12	Automatic Traffic Counter	Northbound	TII (Ireland)

Table 2.1: Datasets Description

Before using this raw data, required verifications and preprocessing steps are performed:

1. Resampling and alignment: Done implicitly, because the dataset already has a fixed time step [$\Delta = 15$ minutes].
2. No missing values were detected in the inspected dataset, and therefore no imputation was required.
3. No outlier handling since data is collected from normal flow without anomalies like accidents.

The output of these steps is a cleaned time series ready to be used for further rigorous processing.

2.4.2 Subsequence extraction

The subsequence length m is fixed to 96, which corresponds to one full day of traffic measurements sampled at a 15-minute interval ($\Delta = 15$). A stride of 1 is used to ensure that all possible subsequences are captured without any loss of temporal information during the deep learning model training phase.

Selecting an appropriate window length is crucial. Larger values of m generate high-dimensional subsequences where **distance measures lose discriminative power**, a known effect referred to as *distance concentration* in high-dimensional spaces (Beyer et al., 1999). As a result, motif discovery becomes less frequent, the number of available samples decreases, and the computational cost increases.

Conversely, smaller values of m produce a larger number of subsequences and increase the sensitivity to short-term fluctuations. However, they also amplify redundancy, trivial matches, and instability in clustering, often capturing noise rather than meaningful temporal motifs—as highlighted by [10] .

The overlapping subsequences extracted from a time series

$$S_T^{(m)} = \{T_{(i,m)} = [t_i, \ldots, t_{i+m-1}] \mid i = 1, \ldots, n-m+1,\ \text{stride} = 1\} \tag{2.1}$$

where n is the total number of time points.
Thus, the total number of overlapping subsequences is computed as:

$$\text{Number of subsequences} = \frac{(n-m)}{\text{stride}} + 1 \tag{2.2}$$

For the dataset used in this study, $n = 35{,}425$. With a stride of 1, the total number of generated subsequences is:

$$\text{Number of subsequences} = (35{,}425 - 96)/1 + 1 = 35{,}330 \tag{2.3}$$

The choice of a stride equal to 1 ensures that all possible subsequences are included, enabling a comprehensive and lossless exploration of temporal patterns.

2.4.2.1 Normalization

Normalization is applied to ensure that all features operate on a comparable scale and to prevent biases caused by heterogeneous value ranges. In this work, Min-Max scaling was used to linearly transform the traffic Count values into the $[0, 1]$ range. This technique preserves the relative structure of the data while standardizing its amplitude, making it more suitable for subsequent deep learning and clustering tasks. The Min-Max normalization is defined as:

$$x' = \frac{x - \min x}{\max x - \min x} \tag{2.4}$$

Where x represents the original value and x' the normalized value. This scaling enhances numerical stability, accelerates training, and prevents features with large magnitudes from dominating the learning process.
Alternatively, standardization (Z-score normalization), defined as

$$x' = \frac{x - \mu}{\sigma} \tag{2.5}$$

Rescales the data using the global mean μ and standard deviation σ. Unlike Min-Max scaling, Z-score normalization produces values centered around zero with unit variance. Its main advantage is robustness to outliers and suitability for models that assume Gaussian distributions. However, Min-Max scaling is often preferred for LSTM-based models and distance-based clustering because it preserves the original data boundaries and prevents distorting temporal patterns. Considering these advantages and the characteristics of traffic flow data, Min-Max scaling was selected for this study.

2.4.3 Matrix Profile and Motif Index

To compute the Matrix Profile (MP) and the Profile Index (PI) using the z-normalized Euclidean distance, we employ the STOMP/SCRIMP++ algorithms as implemented in the stumpy library. These algorithms run in $O(n \log n)$ time in practice due to FFT-based convolution, although the theoretical worst-case remains $O(n^2)$ for a time series of length n [18] . An exclusion zone of $w = m/2$ is applied to eliminate trivial overlaps between neighboring subsequences.

The Matrix Profile and its index are defined as:

$$MP[i] = \min_{j:\,|i-j|\geq} \left\| \tilde{T}_{(i,m)} - \tilde{T}_{(j,m)} \right\|_2, \tag{2.6}$$

$$PI[i] = \arg\min_{j:\,|i-j|\geq} \left\| \tilde{T}_{(i,m)} - \tilde{T}_{(j,m)} \right\|_2, \tag{2.7}$$

where $\tilde{T}_{i,m}$ denotes the i^{th} z-normalized subsequence of length m.

For our dataset (n = 35,425, m = 96), the total number of extracted subsequences is:

$$n - m + 1 = 35{,}330. \tag{2.8}$$

The resulting Matrix Profile values are then sorted in ascending order to identify the closest pairs of subsequences, i.e., the motifs. Table 2.2 reports the top-5 motifs with the smallest profile distances.

Datetime	Profile distance	Profile index
2018-06-20 19:00:00	0.408901	6796
2018-06-06 19:00:00	0.408901	8140
2018-06-20 18:45:00	0.417791	6795
2018-06-06 18:45:00	0.417791	8139
2018-06-06 18:30:00	0.422036	8138

Table 2.2: motifs extracted from the Matrix Profile.

These results show that each subsequence in the Datetime column is highly similar to the subsequence referenced by its corresponding Profile index, with Matrix Profile distances below 0.5. The similarity is therefore established between the two matched subsequences (each having its own timestamp), and not necessarily between identical times of the day. This confirms the presence of recurrent motifs in the traffic flow data and validates the Matrix Profile as an effective tool for discovering repeated temporal patterns.

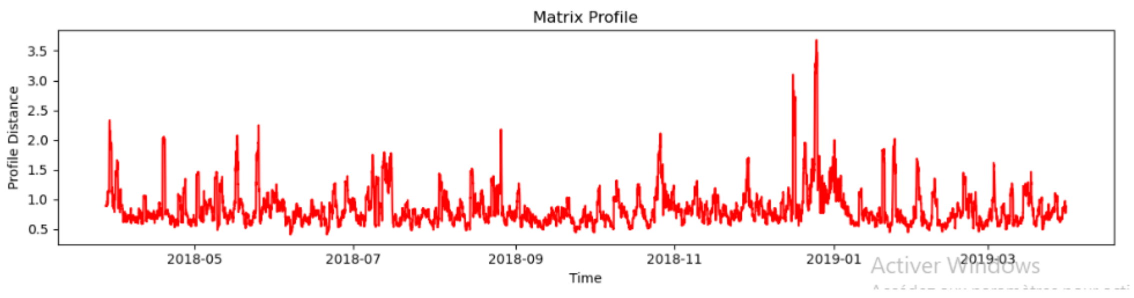


Figure 2.3: Matrix Profile

2.4.4 Motif discovery and index-similarity grouping

The motif discovery phase is carried out through several processing staps.

First, only motif pairs whose Matrix Profile distance is below a similarity threshold τ are retained. In this work, τ is defined as the mean Matrix Profile value:

$$\tau = \frac{1}{|MP|} \sum_{i=1}^{|MP|} MP[i] \tag{2.9}$$

where:

- $|MP|$ is the length of the Matrix Profile,
- $MP[i]$ is the profile distance at index i.

A motif pair (i, j) is therefore kept if:

$$MP[i] < \tau \quad \wedge \quad |i - j| \geq \frac{m}{2} \tag{2.10}$$

m: is the subsequence length.

In our implementation, the notion of pairs is expanded into groups based on transitive similarity.If subsequence i is similar to subsequence j, and j is also similar to k, then i and k are considered part of the same motif group.

Third, trivial matches caused by fully overlapping subsequences are removed.These occur when both subsequences of a motif pair overlap significantly with another motif pair, producing redundant matches. Such pairs are discarded.

Additionally, not all subsequences below the threshold are considered. To ensure only the strongest motif candidates, we restrict the selection to distances belonging to the first quartile of the Matrix Profile distribution.

Finally, motif groups are constructed using an undirected graph representation. Each subsequence index is treated as a node, and an edge is added between nodes belonging to the same motif pair.

The connected components of this graph correspond directly to the final motif groups.

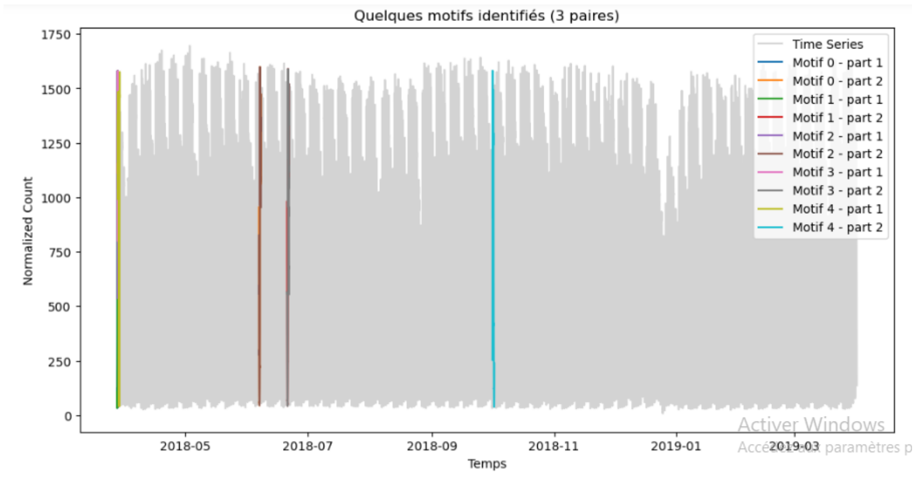


Figure 2.4: Detected Motifs in the Traffic Time Series

Figure 2.4 illustrates the motifs identified within the normalized traffic time series. The gray curve represents the complete time series, while each colored vertical segment corresponds to a subsequence belonging to a motif pair detected through the Matrix Profile. For every motif, two subsequences with highly similar temporal shapes are highlighted ("part 1" and "part 2").

Although these subsequences occur on different dates, they exhibit nearly identical traffic-flow behavior, reflecting repeated daily patterns in the dataset. This visual representation confirms the presence of recurrent motifs and supports the use of Matrix-Profile-based motif discovery in the proposed methodology.

Algorithm 1: Motif discovery from matrix profile

```
Input: Matrix profile file $mp\_df$, time series $time\_series$, subsequence
       length $m$, max distance $max\_distance$
Output: List of motifs and used pairs, saved CSV file
IMPORT libraries: pandas, numpy, networkx, matplotlib,
  sklearn.preprocessing.MinMaxScaler;
SET parameters: $m \leftarrow 96$, $max\_distance \leftarrow mean\_distance$ ;
Function find_all_motifs($mp\_df, time\_series, m, max\_distance$):
    $motifs \leftarrow []$;
    $used\_pairs \leftarrow \{\}$;
    $used\_indices \leftarrow \{\}$;
    for each row $(idx, row)$ in $mp\_df$ do
        $i \leftarrow$ idx;
        $j \leftarrow row['profile\_index']$;
        $distance \leftarrow row['profile\_distance']$;
        if $distance > max\_distance$ then
            continue;
        end
        $overlap\_found \leftarrow FALSE$;
        for each $(k, r)$ do
            if $(k \geq i$ and $k < i + m/2)$ and $(r \geq j$ and $r < j + m/2)$ then
                $overlap\_found \leftarrow TRUE$;
                break;
            end
            if $(r \geq i$ and $r < i + m/2)$ and $(k \geq j$ and $k < j + m/2)$ then
                $overlap\_found \leftarrow TRUE$;
                break;
            end
        end
        if $overlap\_found = FALSE$ then
            ADD $(i, j, distance)$ to $motifs$;
            ADD $(i, j)$ and $(j, i)$ to $used\_pairs$;
            ADD index ranges $[i, i + m/2)$ and $[j, j + m/2)$ to $used\_indices$;
        end
    end
    return $motifs, used\_pairs$;
```

2.5 Subsequence clustering

A basic step before any clustering algorithm is to decide the number of clusters. Classic methods perform with a prior knowledge of data behaviour and a brute-force search with experimental study [19].

In our work, , the determination of k also follows theses steps :

1. a measure of the best sets extracted from the previous step. This evaluation uses the Silhouette score to sort the previously obtained groups according to their internal cohesion and external separation.

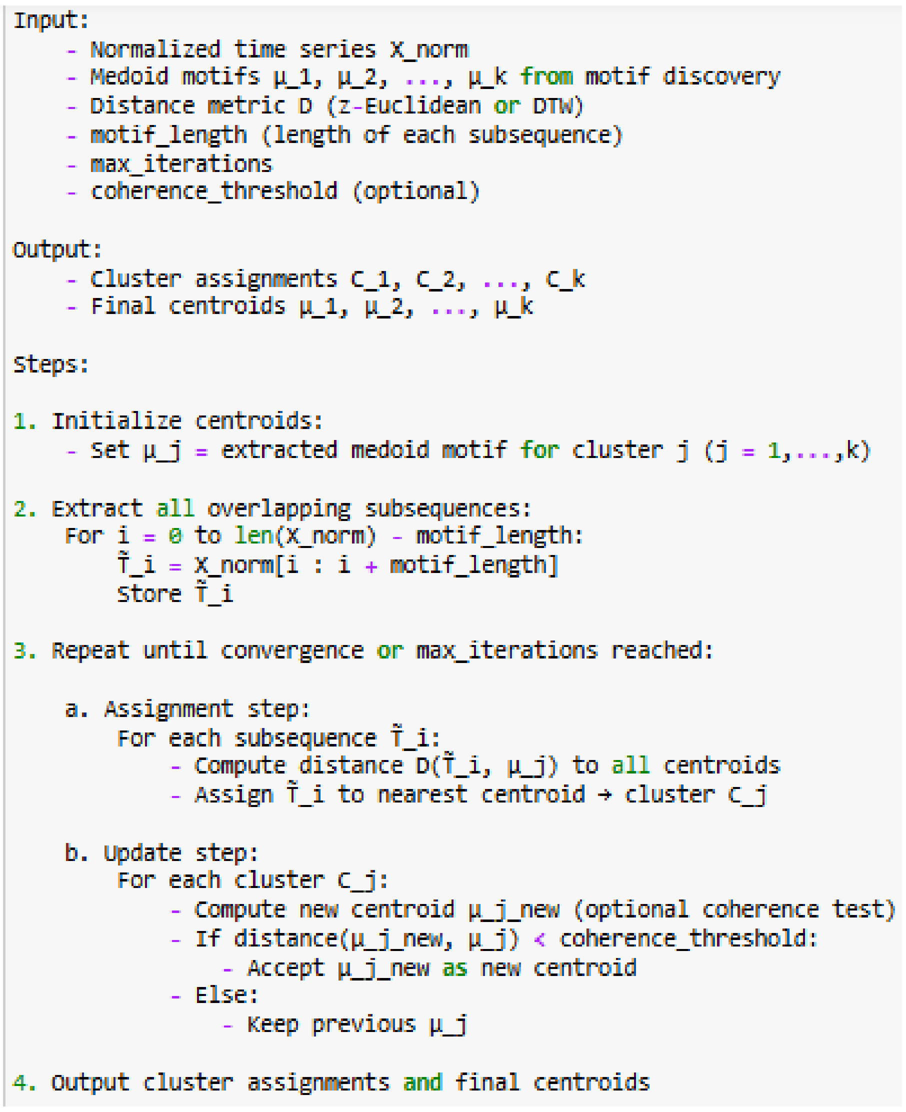
2. since we work on m time steps corresponding to a day duration, we suppose three main types of traffic : weekends, working days, abnormal traffic (specific events, accidents, roads maintenac, . . .)

3. - however, experiments to assess this (k) value are conducted also with different other values under constraints of limited available computational capacities.

K-means is the algorithm used for subsequence clustering. However, calculating centroids for each iteration is not based on the means of overlapping subsequences, which is proven inconsistent. Instead, centroids $\{\mu_c\}$ are initialized with extracted medoid motifs from the previous step.

$$\min_{\{C_1,\ldots,C_k\},\{\mu_1,\ldots,\mu_k\}} \sum_{j=1}^{k} \sum_{\tilde{T}_{i,m} \in C_j} D(\tilde{T}_{i,m}, \mu_j)^2 \tag{2.11}$$

With $D = DTW$ Also, the number of iterations is limited and a coherence test before accepting a new calculated centroids are added to the default behaviour of k-means.



```
Input:
    - Normalized time series X_norm
    - Medoid motifs μ_1, μ_2, ..., μ_k from motif discovery
    - Distance metric D (z-Euclidean or DTW)
    - motif_length (length of each subsequence)
    - max_iterations
    - coherence_threshold (optional)

Output:
    - Cluster assignments C_1, C_2, ..., C_k
    - Final centroids μ_1, μ_2, ..., μ_k

Steps:

1. Initialize centroids:
    - Set μ_j = extracted medoid motif for cluster j (j = 1,...,k)

2. Extract all overlapping subsequences:
    For i = 0 to len(X_norm) - motif_length:
        T̃_i = X_norm[i : i + motif_length]
        Store T̃_i

3. Repeat until convergence or max_iterations reached:

    a. Assignment step:
        For each subsequence T̃_i:
            - Compute distance D(T̃_i, μ_j) to all centroids
            - Assign T̃_i to nearest centroid → cluster C_j

    b. Update step:
        For each cluster C_j:
            - Compute new centroid μ_j_new (optional coherence test)
            - If distance(μ_j_new, μ_j) < coherence_threshold:
                - Accept μ_j_new as new centroid
            - Else:
                - Keep previous μ_j

4. Output cluster assignments and final centroids
```

Figure 2.5: Pseudo-code K-means Clustering

Algorithm 2: K-Means pseudocode for clustering subsequences

```
Input: Clusters dictionary containing subsequences
Output: K-Means model and results saved to disk
# Step 1: Prepare K-Means data;
# Initialize an empty list: all_subsequences = [];
for each cluster_name, cluster_data in clusters.items() do
    if 'subsequences' in cluster_data then
        subseqs = cluster_data['subsequences'];
        # Append to global dataset:
         all_subsequences.append(subseqs);
    end
end
# Concatenate all subsequences: X =
 np.vstack(all_subsequences);
# Step 2: Apply K-Means;
# Import: from sklearn.cluster import KMeans;
# Set number of clusters: K = 5;
# Train model: kmeans = KMeans(n_clusters=K, random_state=42);
kmeans.fit(X);
# Obtain labels and centroids:;
labels = kmeans.labels_;
centroids = kmeans.cluster_centers_;
# Step 3: Organize results;
# Create DataFrame:;
kmeans_results = pd.DataFrame({'Cluster_Label': labels});
# Add source info:;
kmeans_results['Source_Cluster'] = source_cluster_list;
```

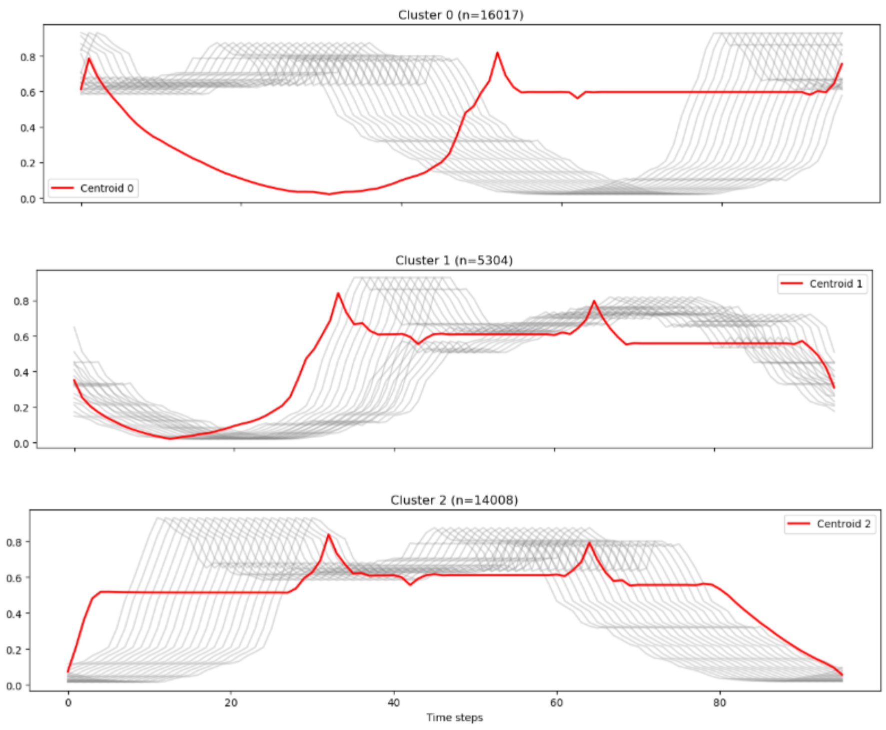


Figure 2.6: Resulting clusters of subsequences and their centroids

2.6 Local deep models

For each cluster C_j, form supervised pairs by sliding input windows of length m and forecast horizon $[H]$ across members of C_j

Input: $x_t = [\,\mathrm{t}1_{(t-L+)}, \ldots, t_t\,]$

Target: $Y_t = [\,\mathrm{t}1_{(t-L+)}, \ldots, t_{(t+H)}\,]$

Model family: LSTM with parameters [hidden size], [layers], [dropout].

Loss: MSE/MAE/...

Optimizer: [Adam $\beta_1 = \ldots, \beta_2 = \ldots, lr = \ldots$]

Batch: $[B]$;

Epochs: $[E]$.

Split: [train/val/test over time] to avoid leakage.

Regularization: early stopping / weight decay.

Rationale: Each expert/model focuses on a homogeneous shape regime, simplifying the mapping from local history to future behaviour.

2.7 True vs Prediction Values for Each Cluster

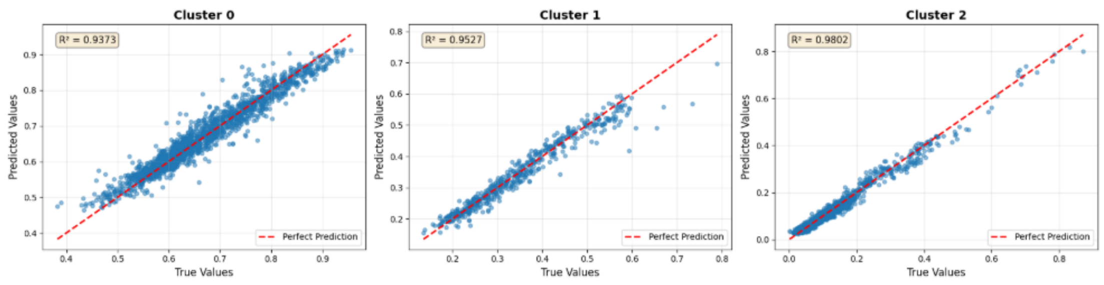


Figure 2.7: True vs Prediction Values for Each Cluster

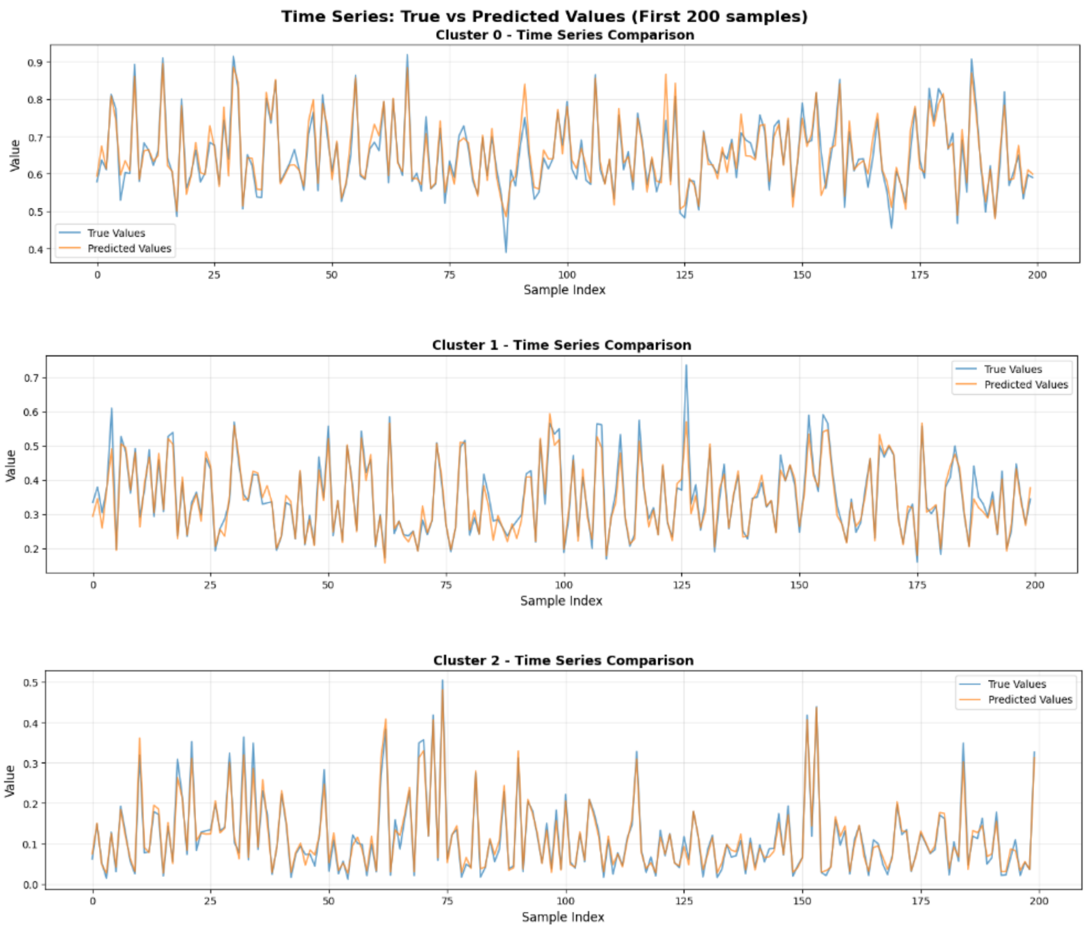


Figure 2.8: True Vs Predicted Values (First 200 Samples)

2.8 Prediction accuracy

To evaluate the predictio accuracy of our models within each cluster, the following four metrics are used MAE, RMSE, MSE, MAPE. MAE measures the average amount of error between the true and predicted values, while MSE computes the mean of the squared errors and is more sensitive to large mistakes. RMSE is the square root of MSE and provides the error in the original scale of the data, whereas MAPE gives the percentage error relative to the true values. In the implementation, the true and predicted values for each cluster are processed separately, and these metrics are calculated and stored in the metrics results dictionary. The results for each cluster are then displayed, and a summary DataFrame is generated to compare performance across all clusters, helping to identify which cluster achieves the best accuracy and which one shows weaker performance.

Cluster	MAE	MSE	RMSE	MAPE
Cluster0	0.019626	0.000685	0.026171	3.015792
Cluster1	0.017482	0.000669	0.025871	5.063007
Cluster2	0.011592	0.000249	0.015776	19.902829

Table 2.3: Prediction metrics for each cluster

2.9 Ensembling

The ensembling component combines the outputs of several independently trained models to produce a single prediction that is typically more stable and accurate than any individual model. In our implementation, three pre-trained models (loaded models[0..2]) are used to forecast the next value using the same input window composed of the last 95 data points. Each model generates its own prediction, and these outputs are then averaged through a simple arithmetic mean to obtain the final ensemble result.

The rationale behind this approach is that different models usually make different types of errors due to variations in their training process or internal parameter updates. By averaging their predictions, some of these individual errors can cancel out, leading to improved robustness and higher overall accuracy. This approach, however, assumes that all models contribute equally and that their outputs are expressed on the same scale. For this reason, it is crucial that all models rely on the same data pre-processing, particularly the same normalization (scaler), during both training and prediction.

It is important to note that the predictions are not random ("parachuted"), but are fully derived from the historical data in the last 95 points. Each cluster model sees these points differently depending on the patterns it has learned, which explains the differences among individual predictions.

The following are prediction results from the three clusters:

Cluster	**Prediction normalized**
Cluster 0	0.441353
Cluster 1	0.105143
Cluster 2	0.080818
Ensemble (average)	0.209105

Table 2.4: Normalized cluster prediction results

After inverse transformation to the original scale of the data, the ensemble prediction corresponds to approximately 360.97, representing the expected value of the next time step.

This version now:

- Explains the 95-point input window.
- Clarifies that predictions are data-driven, not random.
- Connects the cluster models and ensemble logic.
- Presents results clearly.

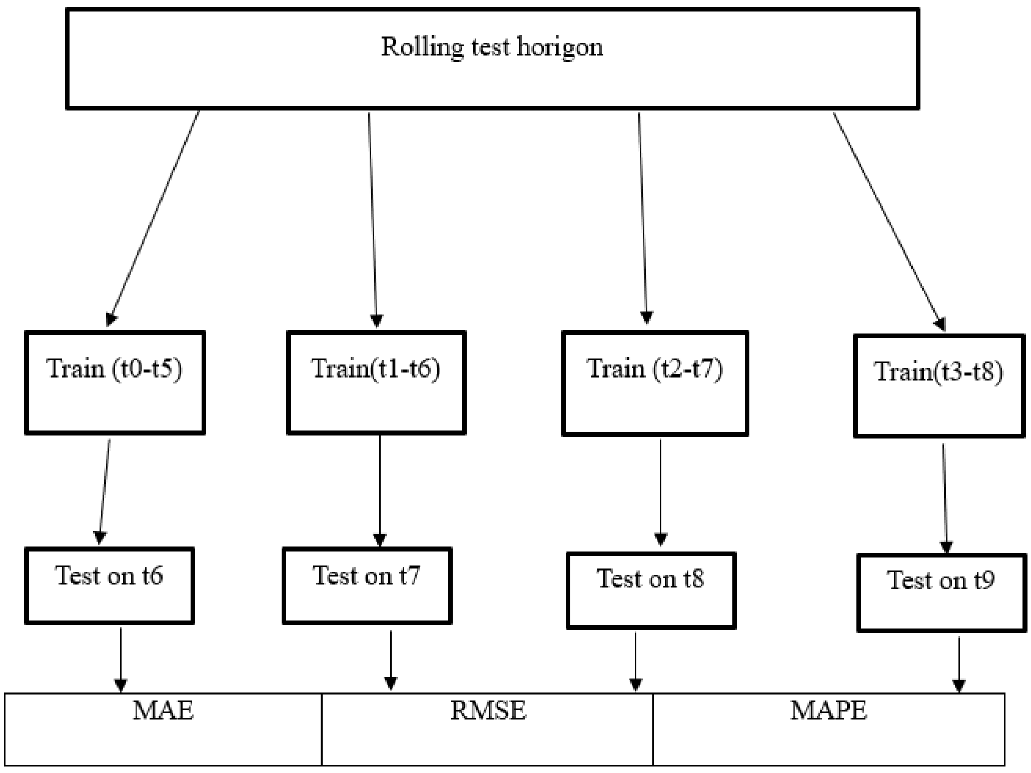


Figure 2.9: Rolling forecast evaluation

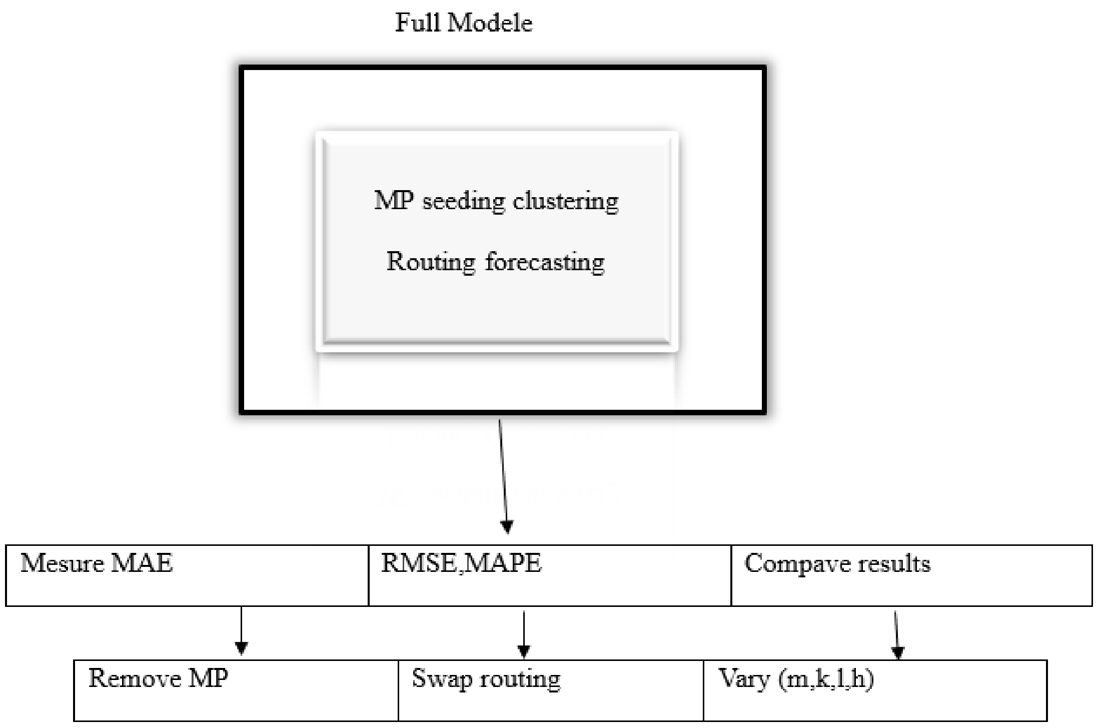


Figure 2.10: Compoment impact

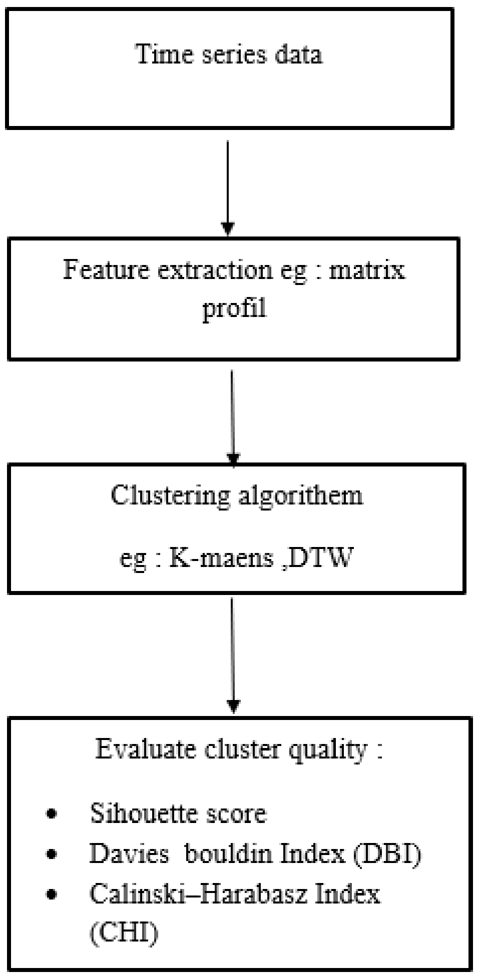

Time series data
Feature extraction eg : matrix profil
Clustering algorithem
eg : K-maens ,DTW
Evaluate cluster quality :
Sihouette score
Davies bouldin Index (DBI)
Calinski–Harabasz Index (CHI)

Figure 2.11: Clustering Evaluation workflow

Chapter 3

TraffiCast Web Application

3.1 Brief presentation of TraffiCast website

The TraffiCost website is an online platform that provides users with services and tools for traffic prediction and analysis, helping them make smarter decisions. The website includes:

- User-friendly interface for easy navigation.
- Traffic prediction module based on live and past data.
- Date and time integration, allowing users to view or choose predictions for a specific moment.
- User account options, such as registration, login, viewing prediction history, and profile management.

Its main goal is to aid trip planning with accurate information and so save time and fuel consumption.

3.2 Website design

At this stage, we utilize UML diagrams to represent and describe our system.

3.2.1 Use case diagram

A Use Case Diagram is a UML diagram that provides a broad overview of a system's functional requirements.

3.2.1.1 Identification of Actors

Identification of actors refers to the process of determining all the external entities that interact with a system. These actors can be human users, external systems, or devices that exchange information or perform actions with the system.

We define two main actors in TraffiCost website :

- **Users (Guest)** : View traffic information and forecasts without the need to create an account.

- **Registered** : Access detailed traffic information and forecasts, receive personalized alerts, and use additional features available only to registered accounts.

3.2.1.2 Identification of the use cases

This process consists of defining specific use cases as shown on figure bellow, with each one depicting a distinct action performed by one or more actors.

- **View landing page** : this use case describes how a user accesses the system's homepage, which provides general information, navigation options, and entry points to main features without requiring login.
- **Register Account** : the "Register Account" use case describes the process in which a new user creates an account in the system by providing required information such as name, email, username, and password.
- **Login** : the "Log In" use case describes the process in which an existing user accesses the system by entering valid credentials, such as a username and password.
- **Predict Traffic** : the "Predict Traffic" use case describes when a user requests traffic forecasts from the system. Using live and historical data, the system predicts road conditions, congestion levels, and estimated travel times to help the user plan trips more efficiently.
- **View prediction history** : the "View Prediction History" use case describes the process in which a user accesses and reviews previously saved traffic predictions.
- **View profile** : the "View Profile" use case describes the process in which a user accesses their personal profile within the system.
- **Toggle dark mode** : allows the user to switch the interface between light and dark themes.

3.2.1.3 Presentation of the use cases

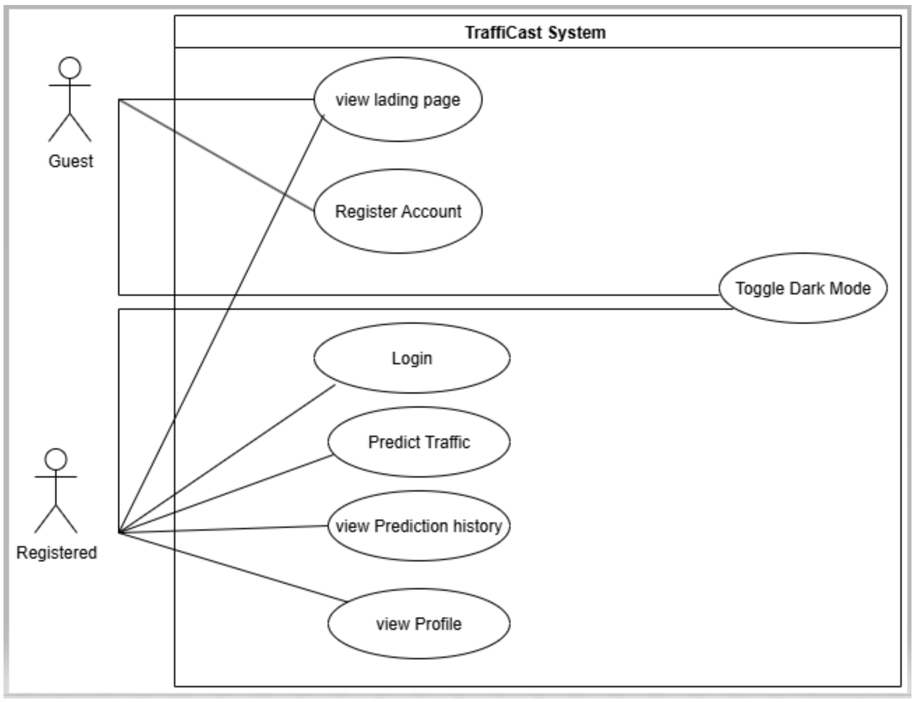


Figure 3.1: Use case diagram of TraffiCost website

3.2.2 Use case specification

The use case specification for the Trafficost system describes each function of the system in detail, defining the actors, goals, conditions, and scenarios for every use case.

3.2.2.1 Register Account

- **Text description of the case Register Account**

Use Case	Register Account
Actors	Guest , registered .
Goal	Allow new users to create an account in the system.
Pre-condition	The user has access to the registration page.
Post-condition	A new user account is successfully created and saved in the system.
Nominal Scenario	1. The user accesses the system and selects the "Register" option. 2. The system displays the registration form. 3. The user fills in required information (e.g., name, email, password). 4. The user submits the form. 5. The system validates the input data. 6. If the data is valid, the system creates the new account.

Alternative Scenario	The system displays an error message and prompts the user to correct the input (returns to step 2).

Table 3.1: Text description « Register Account »

- **Sequence diagram of the case Register Account**

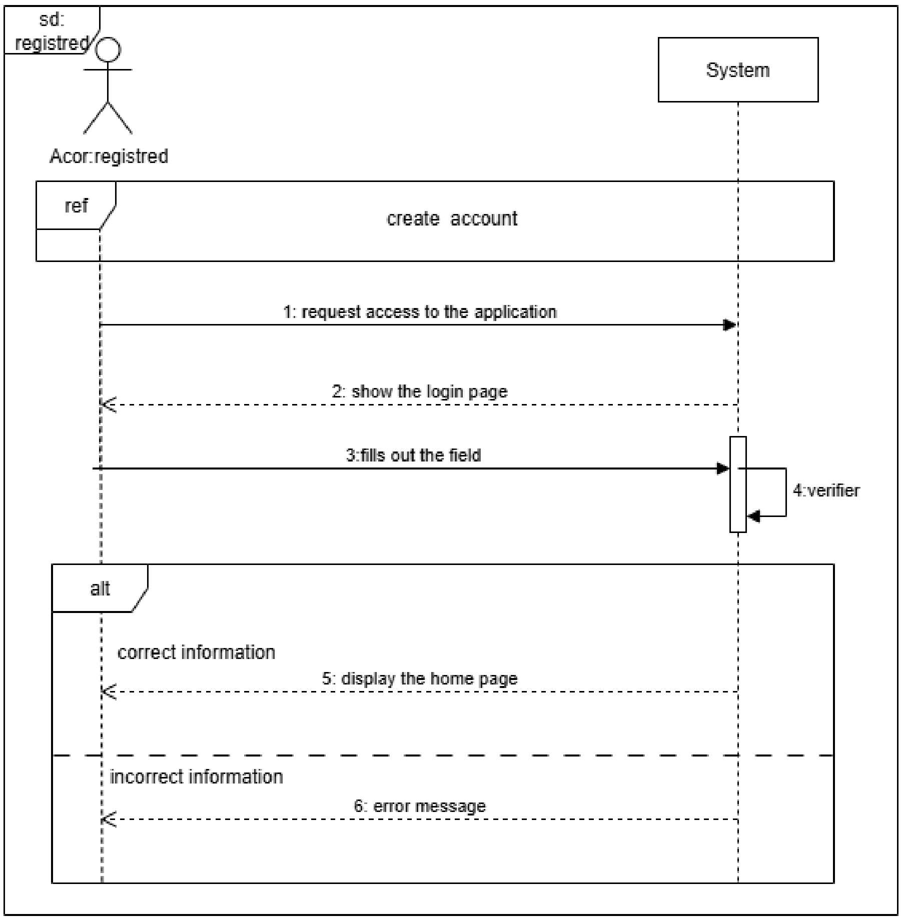


Figure 3.2: Sequence diagram « Register Account »

3.2.2.2 Login

- **Text description of the case login**

Use Case	Login
Actors	Guest user.
Goal	To allow authenticated users to access the system by entering valid login credentials.
Pre-condition	The user must already have a registered account in the system.
Post-condition	The user successfully logs into the system and is redirected to the home page or dashboard.

Nominal Scenario	1. The user opens the system and selects the "Login" option. 2. The system displays the login form. 3. The user enters their login credentials (email and password). 4. The system verifies the entered information. 5. If the credentials are correct, the system grants access. 6. The user is redirected to the home page or appropriate dashboard based on their role.
Alternative Scenario	if the email or password is incorrect, the system displays an error message (e.g."Invalid email or password") (returned step2).

Table 3.2: Text description « Login »

- **Sequence diagram of the use case : Login**

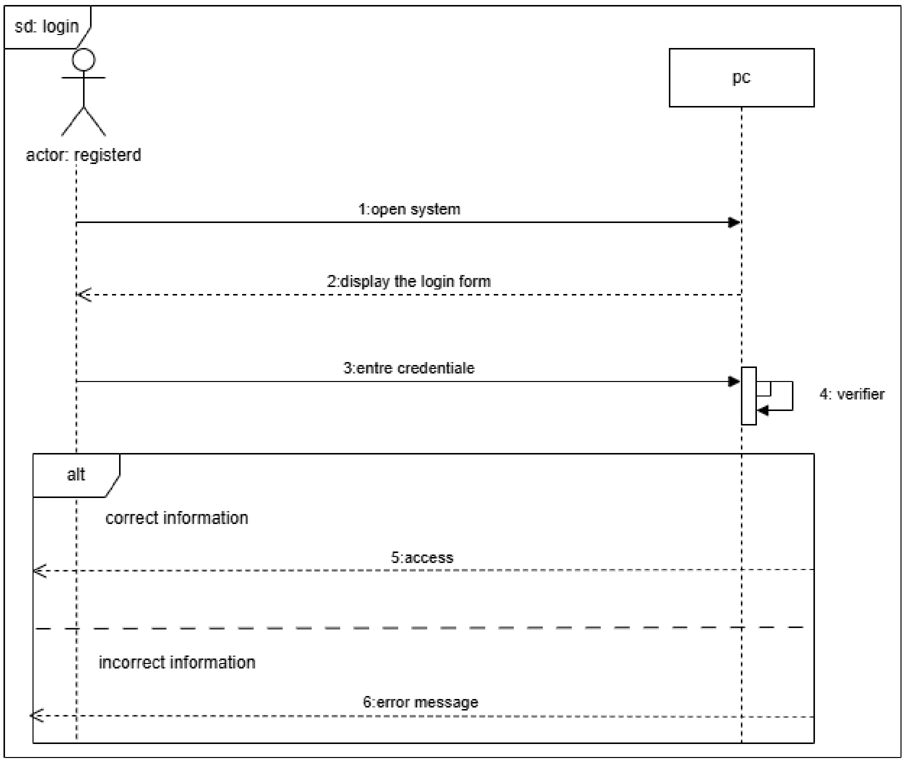


Figure 3.3: Sequence diagram « Login »

3.2.2.3 Predict Traffic

- **Text description of the case Predict Traffic**

Use Case	Predict Traffic
Actors	Registered User.
Goal	To allow actors to view or retrieve traffic predictions based on real-time or historical data.

Pre-condition	The user is authenticated (if required) and has access to the prediction interface.
Post-condition	The system displays or returns traffic prediction results for a specified area and time
Nominal Scenario	1. The user logs in and accesses the dashboard. 2. The user selects traffic prediction and enters parameters (location, date, time). 3. The system applies the prediction model 4. The system displays the results 5. The prediction is automatically saved (allow user to view the history.
Alternative Scenario	The user submits incomplet data , the system displays a message requesting all required fields to be filled.

Table 3.3: Text description «Predict Traffic»

- **Sequence diagram of the use case Predict Traffic**

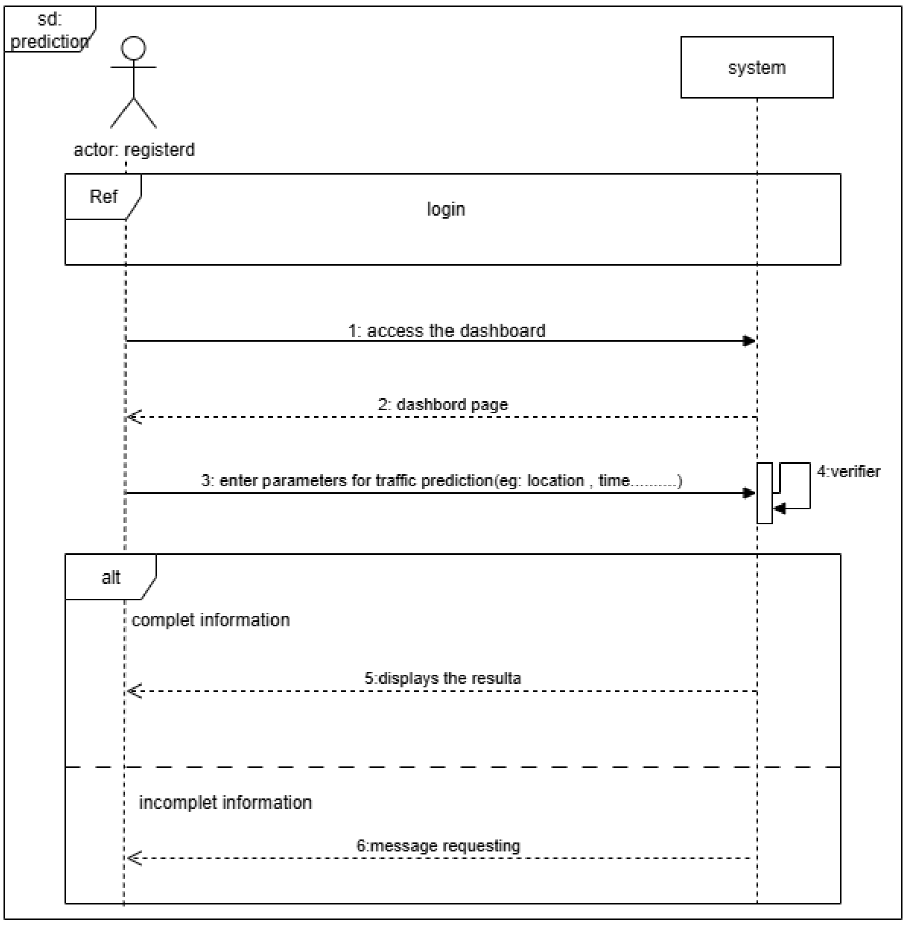


Figure 3.4: Sequence diagram « Predict Traffic »

3.2.2.4 View prediction history

- **Text description of the case View prediction history**

Use Case	View prediction history
Actors	Registered User.
Goal	Allow users to access and review their past traffic prediction results.
Pre-condition	The user is logged into the system. The system has stored previous prediction requests made by the user.
Post-condition	The user can see a list of past predictions with relevant details (e.g., date, time, location, predicted traffic conditions).
Nominal Scenario	1. The user navigates to the "Prediction History" section. 2. The system retrieves the stored prediction records related to the user. 3. The user selects a specific record to view more details if needed. 4. The system displays details.
Alternative Scenario	If no past predictions exist, the system displays a message such as “No prediction history available.”

Table 3.4: Text description « View prediction history »

- **Sequence diagram of the case View prediction history**

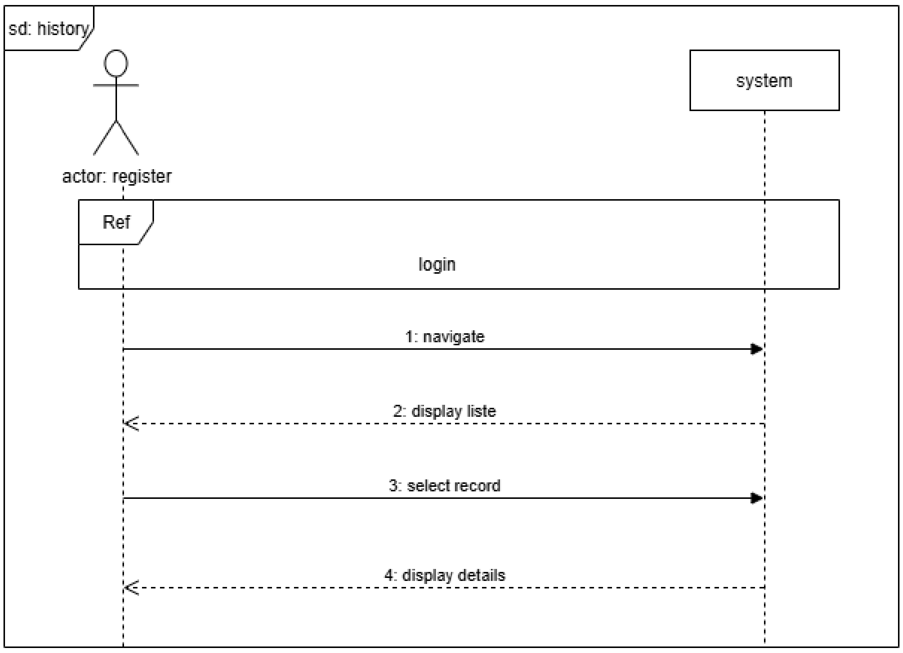


Figure 3.5: Sequence diagram « View prediction history »

3.3 Global Class diagram

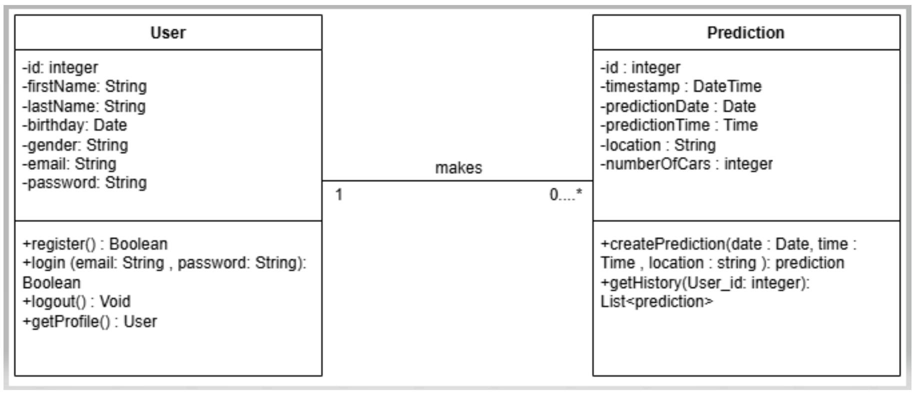


Figure 3.6: Global class diagram

3.4 Relational Model

- **User** (id, firstName, lastName, birthday, gender, email, password)
- **Prediction** (id, timestamp, predictionDate, predictionTime, location, numberOfCars)

3.5 The interfaces of the application

3.5.1 Landing

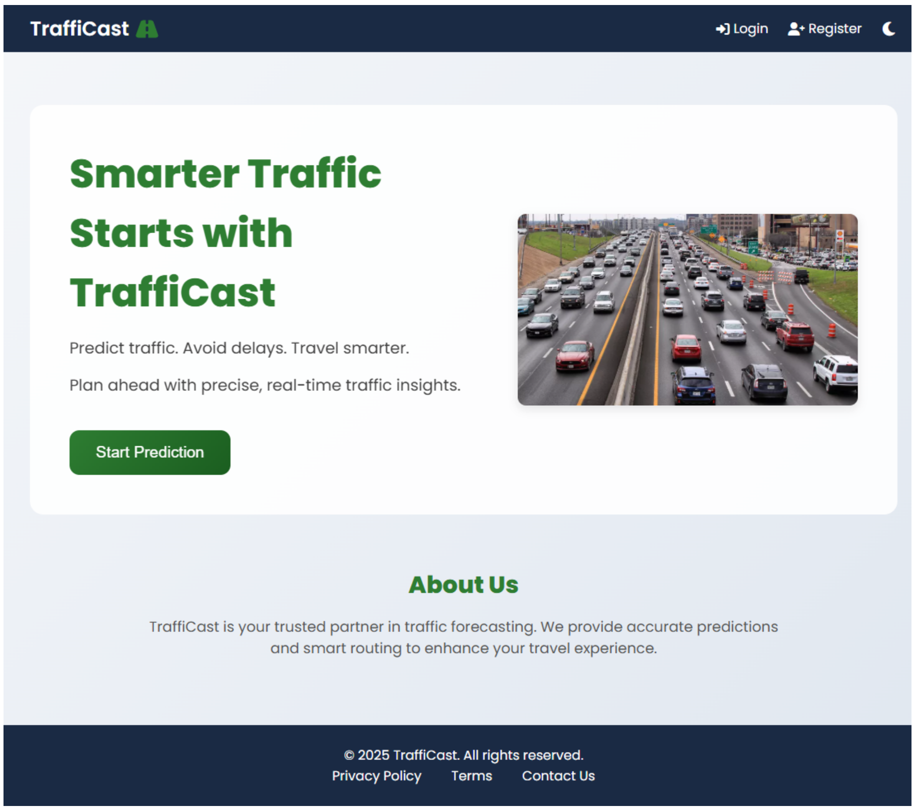

TraffiCast
Login
Register
Smarter Traffic Starts with TraffiCast
Predict traffic. Avoid delays. Travel smarter.
Plan ahead with precise, real-time traffic insights.
Start Prediction
About Us
TraffiCast is your trusted partner in traffic forecasting. We provide accurate predictions and smart routing to enhance your travel experience.
© 2025 TraffiCast. All rights reserved.
Privacy Policy
Terms
Contact Us

Figure 3.7: landing Interface

3.5.2 Login

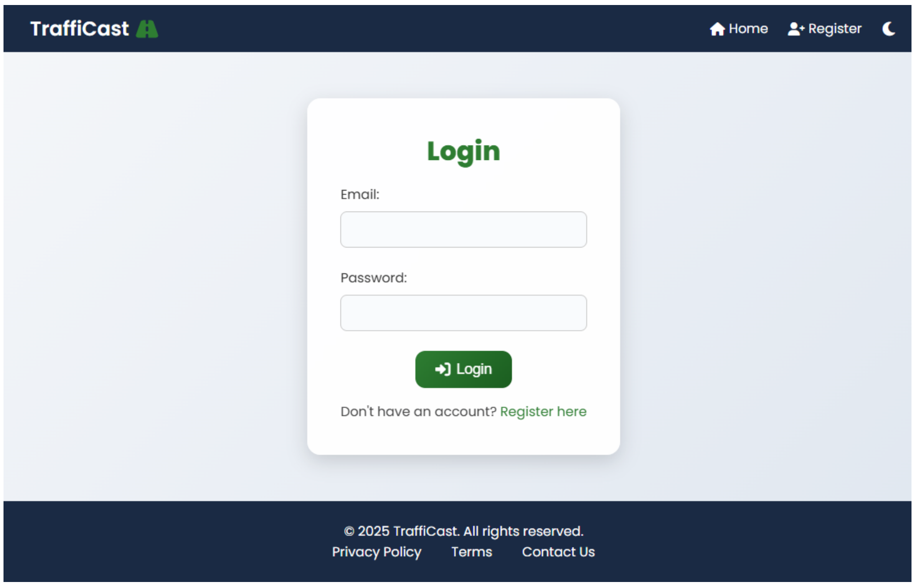


Figure 3.8: Login interface

3.5.3 Register

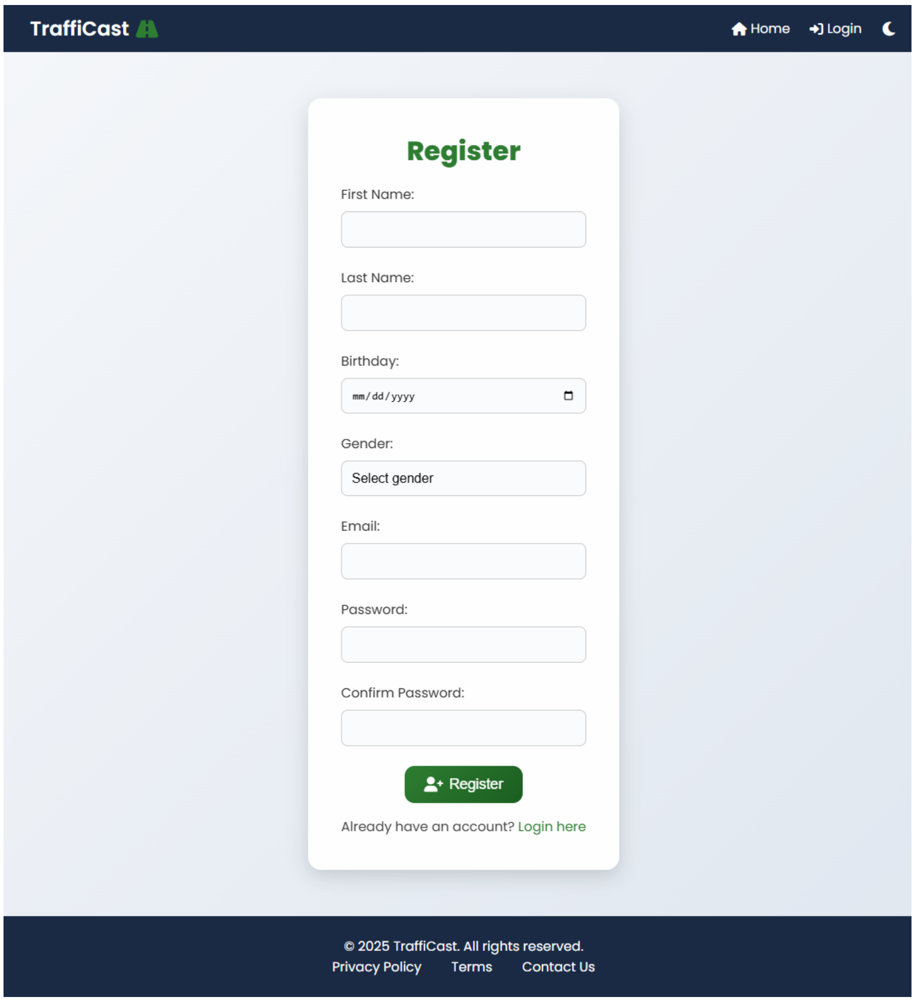
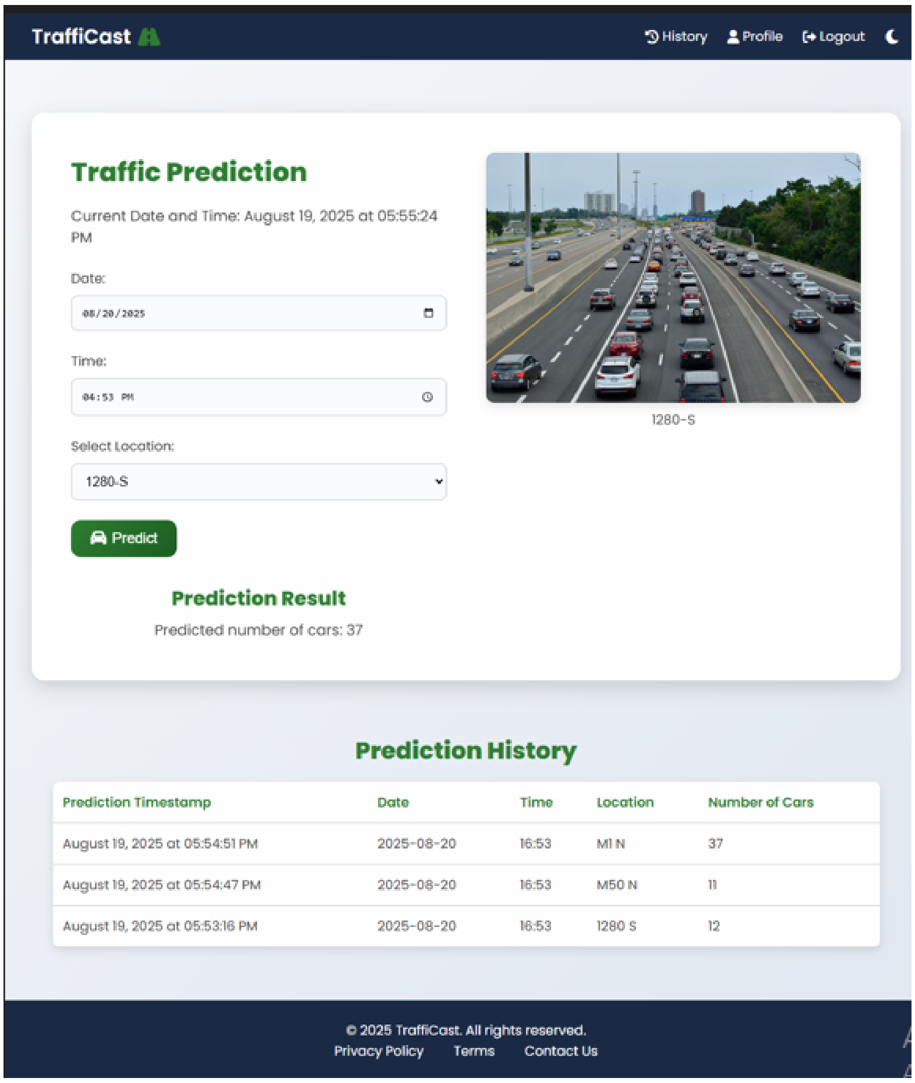


Figure 3.9: Register interface

3.5.4 Traffic Prediction



TraffiCast

History Profile Logout

Traffic Prediction

Current Date and Time: August 19, 2025 at 05:55:24 PM

Date: 08/20/2025

Time: 04:53 PM

Select Location: 1280-S

Predict

1280-S

Prediction Result

Predicted number of cars: 37

Prediction History

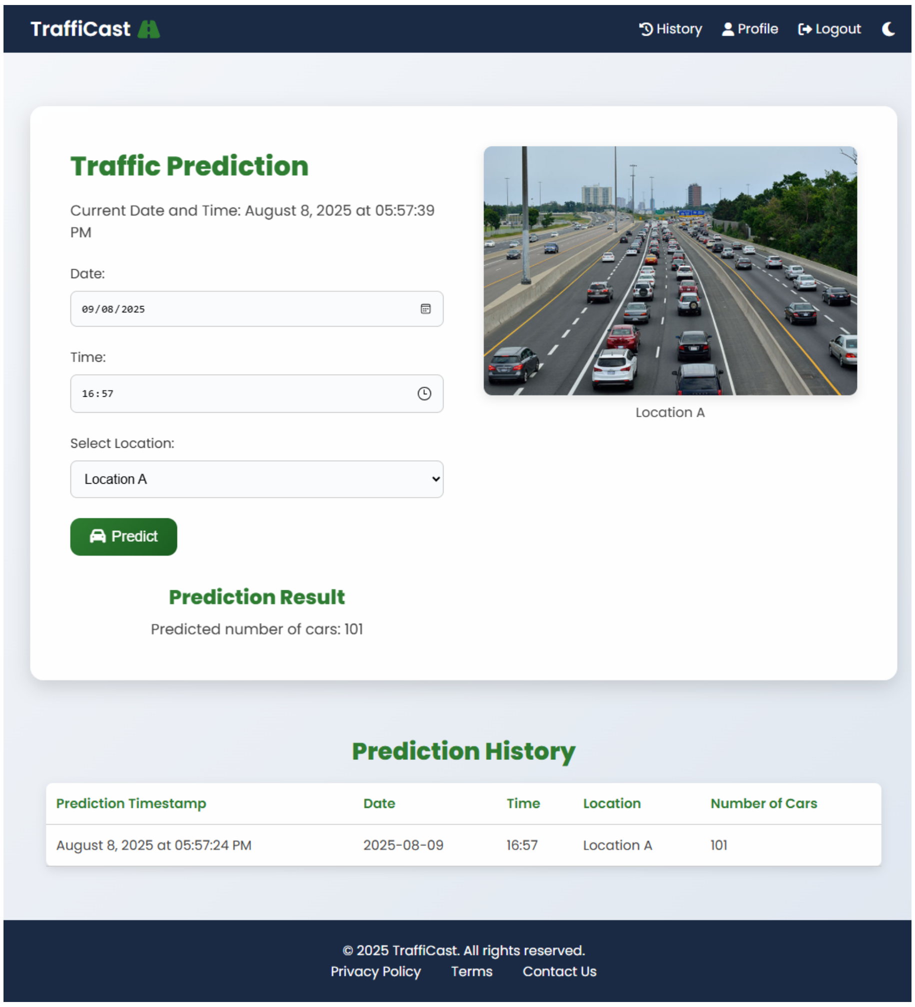
Prediction Timestamp	Date	Time	Location	Number of Cars
August 19, 2025 at 05:54:51 PM	2025-08-20	16:53	M1 N	37
August 19, 2025 at 05:54:47 PM	2025-08-20	16:53	M50 N	11
August 19, 2025 at 05:53:16 PM	2025-08-20	16:53	1280 S	12

© 2025 TraffiCast. All rights reserved.

Privacy Policy Terms Contact Us

Figure 3.10: traffic prediction intreface

3.5.5 Prediction History



Current Date and Time: August 8, 2025 at 05:57:39 PM

Date: 09/08/2025

Time: 16:57

Select Location: Location A

Prediction Result

Predicted number of cars: 101

Prediction History

Prediction Timestamp	Date	Time	Location	Number of Cars
August 8, 2025 at 05:57:24 PM	2025-08-09	16:57	Location A	101

Figure 3.11: Prediction history interface

3.6 Frameworks

In the following a brief presentation of the frameworks, environments and languages used to implement our system.

3.6.1 Python Programming Language

Python is a high-level, interpreted, and dynamically typed programming language known for its simplicity, readability, and versatility. It supports multiple programming paradigms, including object-oriented, procedural, and functional programming. Python is widely used in web development, data science, artificial intelligence, automation, and more, thanks to its extensive libraries and strong community support.

Figure 3.12: Python logo

3.6.2 TensorFlow

TensorFlow is an open-source machine learning framework developed by Google Brain for building and deploying deep learning and artificial intelligence (AI) models. It provides a flexible ecosystem of tools, libraries, and community resources for tasks such as neural network training, computer vision, natural language processing (NLP), and reinforcement learning. TensorFlow supports CPU, GPU, and TPU acceleration, making it scalable for both research and production environments.

Figure 3.13: Tensorflow logo

3.6.3 Keras

Keras is an open-source, high-level deep learning API written in Python that runs on top of TensorFlow. It is designed to provide an easy-to-use, modular, and user-friendly interface for building and training neural networks. Keras supports deep learning tasks such as computer vision, natural language processing (NLP), and reinforcement learning, making it popular for rapid prototyping and research.

Figure 3.14: Keras logo

3.6.4 Google Colaboratory (Colab)

Google Colaboratory (Colab) is a cloud-based Jupyter Notebook environment provided by Google that allows users to write, execute, and share Python code in a web browser. It provides free access to GPUs and TPUs, making it ideal for machine learning, deep learning, and data science projects. Colab requires no setup, supports Google Drive integration, and allows collaboration among multiple users in real time.

Figure 3.15: Colab logo

3.6.5 Flask framework

Flask is a micro web framework for Python, used for building web applications. It is classified as a "microframework" because it provides a core set of functionalities for web development, such as routing, templating, and request handling, but it does not include features like database abstraction layers, form validation, or object-relational mappers (ORMs) by default. This minimalistic design allows developers to choose the specific tools and libraries they need for their projects, offering greater flexibility and control.

Figure 3.16: Flask logo

3.6.6 sQLite database

SQLite is a C-language library that provides a lightweight, self-contained, high-reliability, full-featured, SQL database engine. Unlike traditional client-server database systems (like MySQL or PostgreSQL), SQLite is an embedded database, meaning it doesn't require a separate server process. Instead, the SQLite library is linked directly into the application, and the database itself is stored in a single, cross-platform file on disk.

Figure 3.17: SQLite logo

3.6.7 Visual Studio code

Visual Studio Code (VS Code) is a lightweight yet highly capable source code editor developed by Microsoft. Supporting a broad range of programming languages, it is particularly well-regarded for Python-based machine learning projects due to its adaptability and comprehensive extension ecosystem. In this study, VS Code was employed as the primary development environment, providing advanced features such as intelligent code completion, real-time debugging, Git integration, and extensions like Python and Jupyter, which facilitated the seamless development and evaluation of the deep learning model. Its intuitive interface, combined with powerful tools, played a crucial role in enhancing coding efficiency and supporting effective model experimentation.

Figure 3.18: Visual Studio Code logo

General Conclusion

In this work, we focused on road traffic forecasting, a key component of intelligent transportation systems and urban mobility management. We proposed a hybrid methodology combining Matrix Profile analysis to detect recurring local patterns (motifs) in univariate time series with subsequence clustering, followed by a deep learning framework based on LSTM networks.

Each identified cluster was associated with a specialized LSTM model trained to capture its unique temporal behavior. During inference, new traffic data windows were routed to one or more of these models, and their outputs were aggregated to produce the final forecast. The experimental results confirmed the effectiveness of this approach, demonstrating its ability to discover structural temporal patterns and improve predictive accuracy. Future work may focus on extending the methodology to multivariate traffic data for more comprehensive forecasting, exploring advanced architectures such as Transformers or Graph Neural Networks to better capture complex temporal and spatial relationships, integrating real-time traffic data with adaptive learning for dynamic updates, considering external factors such as weather and special events to enhance prediction reliability, and applying the approach to anomaly detection to support traffic management and safety measures.

Bibliography

[1] John Paparrizos et al. “A survey on time-series distance measures”. In: *arXiv preprint arXiv:2412.20574* (2024).

[2] Eamonn Keogh, Stefano Lonardi, and Chotirat Ann Ratanamahatana. “Towards parameter-free data mining”. In: *Proceedings of the tenth ACM SIGKDD international conference on Knowledge discovery and data mining*. 2004, pp. 206–215.

[3] Chin-Chia Michael Yeh, Helga Van Herle, and Eamonn Keogh. “Matrix profile III: the matrix profile allows visualization of salient subsequences in massive time series”. In: *2016 IEEE 16th international conference on data mining (ICDM)*. IEEE. 2016, pp. 579–588.

[4] Pavel Senin. “Dynamic time warping algorithm review”. In: *Information and Computer Science Department University of Hawaii at Manoa Honolulu, USA* 855.1-23 (2008), p. 40.

[5] Eamonn Keogh. *RATANAMC*. Accessed: 2025-11-15. n.d. URL: https://www.cs.ucr.edu/~eamonn/RATANAMC.pdf.

[6] Xiaozhe Wang, Kate Smith, and Rob Hyndman. “Characteristic-based clustering for time series data”. In: *Data mining and knowledge Discovery* 13.3 (2006), pp. 335–364.

[7] Peter J Rousseeuw. “Silhouettes: a graphical aid to the interpretation and validation of cluster analysis”. In: *Journal of computational and applied mathematics* 20 (1987), pp. 53–65.

[8] Chin-Chia Michael Yeh et al. “Matrix profile I: all pairs similarity joins for time series: a unifying view that includes motifs, discords and shapelets”. In: *2016 IEEE 16th international conference on data mining (ICDM)*. IEEE. 2016, pp. 1317–1322.

[9] Yan Zhu et al. “Matrix Profile XI: SCRIMP++: Time Series Motif Discovery at Interactive Speeds”. In: *Proceedings of the 2018 IEEE International Conference on Data Mining (ICDM)*. IEEE, 2018, pp. 837–846.

[10] Eamonn Keogh. *The UCR Matrix Profile Page*. Accessed: 2025-11-15. n.d. URL: https://www.cs.ucr.edu/~eamonn/MatrixProfile.html.

[11] Qing Zhang and Hong Yu. “CiteGraph: A citation network system for MEDLINE articles and analysis”. In: *Studies in Health Technology and Informatics* 192 (2013). PMID: 23920674, pp. 832–836. URL: https://pubmed.ncbi.nlm.nih.gov/23920674/.

[12] saloni1297. *Introduction to Artificial Neural Networks (ANNs)*. Accessed: 2025-11-15. Sept. 2025. URL: https://www.geeksforgeeks.org/deep-learning/introduction-to-artificial-neutral-networks/.

[13] Sepp Hochreiter and Jürgen Schmidhuber. "Long Short-Term Memory". In: *Neural Computation* 9.8 (1997), pp. 1735–1780. DOI: 10.1162/neco.1997.9.8.1735.

[14] Eamonn Keogh, Jessica Lin, and William Truppel. "Clustering of Time Series Subsequences is Meaningless: Implications for Previous and Future Research". In: *Proceedings of the International Conference on Data Mining*. 2003.

[15] T. Warren Liao. "Clustering of Time Series Data—A Survey". In: *Pattern Recognition* 38.11 (2005), pp. 1857–1874.

[16] Qian Liu et al. "A Novel Matrix Profile-Guided Attention LSTM Model for Forecasting COVID-19 Cases in USA". In: *Frontiers in Public Health* 9 (2021), p. 741030. DOI: 10.3389/fpubh.2021.741030.

[17] K. Iyad K. Mohand. "Urban traffic flow forecasting using deep learning". Master's degree. Mila, Algérie: University Center Abd Elhafid Boussouf Mila, 2024.

[18] Chin-Chang Michael Yeh. "The Matrix Profile: Scalable Algorithms and New Primitives". PhD thesis. University of California, Riverside, 2018. URL: https://escholarship.org/content/qt2x4419fp/qt2x4419fp_noSplash_63b12fd93e27078e9e7
pdf.

[19] Ana Pegado-Bardayo et al. "A review of unsupervised k-value selection techniques in clustering algorithms". In: *Journal of Industrial Engineering and Management* 17.3 (2024), pp. 641–649. DOI: 10.3926/jiem.6791. URL: https://www.jiem.org/index.php/jiem/article/view/6791.