CHALMERS
UNIVERSITY OF TECHNOLOGY

UNIVERSITY OF GOTHENBURG

Machine Learning-based Lane-level Localization

Hypothesis inference for localization of autonomous vehicles

Master's thesis in Mathematical Sciences

Amogha Udayakumar
Bragadesh Bharatwaj Sundararaman

Department of Computer Science and Engineering
CHALMERS UNIVERSITY OF TECHNOLOGY
UNIVERSITY OF GOTHENBURG
Gothenburg, Sweden 2023

Master's thesis 2023

Machine Learning-based Lane-level Localization

Hypothesis inference for localization of autonomous vehicles

Amogha Udayakumar
Bragadesh Bharatwaj Sundararaman

UNIVERSITY OF
GOTHENBURG

CHALMERS
UNIVERSITY OF TECHNOLOGY

Department of Mathematical Sciences
Chalmers University of Technology
University of Gothenburg
Gothenburg, Sweden 2023

Machine Learning-based Lane-level Localization
Hypothesis inference for localization of autonomous vehicles
Amogha Udayakumar
Bragadesh Bharatwaj Sundararaman

Supervisor: Marina Axelson-Fisk, Department of Mathematical Sciences
Advisor: Axel Beauvisage, Zenseact
Advisor: Junsheng Fu, Zenseact
Examiner: Marina Axelson-Fisk, Department of Mathematical Sciences

Master's Thesis 2023
Department of Mathematical Sciences
Chalmers University of Technology and University of Gothenburg
SE-412 96 Gothenburg
Telephone +46 31 772 1000

Typeset in LaTeX
Gothenburg, Sweden 2023

Machine Learning-based Lane-level Localization
Hypothesis inference for localization of autonomous vehicles
Amogha Udayakumar
Bragadesh Bharatwaj Sundararaman
Department of Mathematical Sciences
Chalmers University of Technology and University of Gothenburg

Abstract

In autonomous driving, for the vehicle to make a decision on its own it has to have precise knowledge of its location (lane) with respect to its environment. This problem of determining the lane on which the vehicle is travelling is called Lane-Level Localization (LLL). HD maps with localization algorithms are used to solve the problem of Lane-Level Localization. However, during the initialization phase, there is uncertainty in the confidence of the lane in which the vehicle is driving. Our thesis aims to overcome this problem by using the Multi-Hypothesis Tracking (MHT) approach. Multi-Hypothesis Tracking has two parts which are tracking multiple hypotheses and inference of the correct hypothesis by eliminating the wrong hypotheses. Our thesis focuses on the latter part which can be solved using the early classification of time series technique. The early classification of time series technique tends to make an early and accurate classification of the hypothesis by rejecting wrong hypotheses based on the class probabilities assigned to different hypotheses and using multi-objective optimization to find a trade-off between earliness and accuracy. Our model produced an accuracy of 99.053% with an earliness of 0.109.

Keywords: Autonomous driving, Lane-Level Localization, HD maps, Multi-Hypothesis Tracking (MHT), Early Classification of Time Series (ECTS).

Acknowledgements

We would like to thank and acknowledge Zenseact and our supervisors Axel Beauvisage, Junsheng Fu, and Marina Axelson-Fisk for their support and guidance through all the stages of this project.

Amogha Udayakumar & Bragadesh Bharatwaj Sundararaman,

Gothenburg, 2023-07-25

Contents

List of Figures

List of Tables

1

Introduction

Autonomous vehicle technology is a fast-developing and critically important field [1]. As of 2010, the number of automobiles estimated to be traveling the streets and roads in the world reached 1.015 billion. With the increase in vehicle ownership comes the associated societal cost of traffic crashes, increasing financial cost of congestion, and health cost of congestion from premature deaths due to pollution inhalation [2]. The reduction of the associated costs is the motivation for increased research in the field of autonomous vehicles.

The six different levels of automation outlined by [3] are:

- Level 0 − The driver is responsible for the driving task.
- Level 1 − 2 − The driver is assisted with longitudinal and lateral control by the driver assistance system.
- Level 3 − Automated driving but needs attention from the driver under certain circumstances.
- Level 4 − Automated driving and does not need attention from the driver.
- Level 5 − Fully automated driving that can handle all circumstances and does not require the driver.

Perception, planning, and control are the core components to achieve full autonomy [4]. Perception refers to the information obtained about the surrounding environment from hardware components such as cameras, RADAR, LiDAR, and external sources such as HD maps. Perception can be split into two categories: Environmental perception and Localization. Environmental perception implies understanding contextual information about the environment such as obstacle location, and road sign/marking detection among others. Localization is the ability of the robot to estimate its position with respect to its environment [5]. Autonomy is impossible without precise knowledge of the vehicle's location with respect to its surroundings [6]. Hence, an accurate and robust localization system is in need to build a safe intelligent vehicle.

1.1 Background

In [7], the author(s) breaks down localization into smaller categories such as Road-Level localization (RLL), Ego-Lane Level Localization (ELL), and Lane-Level Localization (LLL). Road-Level Localization is the estimation of the road on which the vehicle is traveling, Ego-Lane Level Localization is finding the vehicle position in terms of lateral and longitudinal position on the lane, and Lane-Level localization is estimating the lane on which the vehicle is traveling.

In [7], the authors further distill LLL into two distinct topics. The first one is determining the lane on which the ego-vehicle is currently traveling. The second one is detecting the lateral position of the vehicle on the overall road. The focus of this thesis is solving the first part of Lane Level Localization which is finding the ego lane of the vehicle.

Recent AVs can estimate their precise positions and generate collision-free trajectories with high-definition (HD) maps [8]. However, one challenge with HD map localization is the initialization phase where the sensor inputs are not reliable enough to determine with high confidence the lane in which the car is driving.

To overcome this problem and to find the ego-lane of the vehicle in the initialization phase, *Multiple-Hypothesis Tracking (MHT)* approach is used. MHT has been used in localization in cases such as [9], [10]. MHT is the process of creating and tracking multiple hypotheses of the target object and eliminating each hypothesis based on its likelihood to the state of the target. MHT can be divided into two parts: (1) Tracking of multiple hypotheses and (2) Inference of the correct hypothesis by eliminating the wrong hypotheses. Our thesis is more focused on building an inference system that can select the correct hypothesis with respect to the target object in the initialization phase.

This thesis is conducted in collaboration with Zenseact, a software company specializing in designing software stacks for autonomous driving and advanced driver-assistance systems to end car accidents and provide car safety [11]. Zenseact has an existing system that is currently being used for this purpose. Apart from this, a machine-learning system was built as part of the thesis in 2022 for lane-level localization in the initialization phase of the HD maps. However, there are a few limitations in this previous thesis system [12] that our thesis aims to resolve or find better solutions to mitigate them.

1.2 Aim

The system developed by the previous year's thesis [12] for the task of online lane-level vehicle localization had limitations in producing correct and early predictions in challenging driving scenarios. Since the hypothesis inference has to be accurate and early at the initialization phase for the vehicle to make safer decisions, this thesis aims to create an improved inference model that can tackle the problems encountered by the previous system. This can be achieved by:

- Performing extensive data analysis and exploration on a bigger and more diverse dataset to identify challenging situations where the previous system can be improved.
- Investigate other state-of-the-art machine learning and localization approaches.
- Propose an improved solution to the former system that can tackle the current problems or propose and examine a new hypothesis inference model that produces an accurate and early inference.

2

Literature Review

2.1 Lane-level Localization

As mentioned in the introduction, lane-level localization is the estimation of the the lane on which the vehicle is traveling. The knowledge of the host lane can provide autonomous navigation systems with the most desirable instructions such as better overtaking strategy [7].

Various systems assist autonomous vehicles with lane-level localization. One such system mentioned in [7] is GNSS (Global Navigation Satellite System) which uses signals from satellites to locate the ego-vehicle on the road. However, GNSS lacks accuracy due to poor satellite signals. GNSS cannot also position the AVs in tunnels where satellite signals cannot be received. Alternatively, Differential GNSS/IMU (Inertial Measurement Unit) integrated navigation can be used to achieve centimeter-level dynamic positioning [13],[14]. However, in areas where weak signals are received, it is difficult to use GNSS as a fixed solution and the accuracy of positioning degrades to decimeter or meter levels which is not desirable in situations requiring high precision localization such as detecting the ego-lane in multi-lane roads in urban areas [15]. A digital map provides information about the road environment such as lane markings and matching the detected landmarks from the sensors to the map data helps to achieve robust localization and perception [7]. This approach is called the map-matching method. Usually a precise high-definition (HD) map is used for this purpose, an HD map is a highly accurate map that contains information more than the traditional map like lane markings, lane boundaries, and road geometry. HD maps have been used for localization purposes in autonomous vehicles in various cases such as [16], [17], and [18].

Map-based localization techniques like Particle Filters and Kalman Filters are used in autonomous vehicles. [19] demonstrates localization by fusing the digital map and outputs of various sensors such as low-cost GPS, low-priced IMU, built-in wheel speed sensor, and single front camera based on particle filter. Particle Filter initializes N number of particles in the map (possible vehicle positions) and at each time step when the vehicle moves, it takes sensor information into account to estimate the current state of the vehicle. At every iteration, a prediction step is made, by moving each particle in the same direction and distance according to a certain motion model. After the prediction step, sensors such as GPS, front camera, and digital map are used and the distance of the vehicle to the known landmarks on the map is mea-

sured to correct the position of the vehicle [20]. [21] demonstrates the application of the Kalman Filter to improve 3D LiDAR signals of autonomous vehicles in adverse weather. Kalman Filter uses sensor outputs to estimate the vehicle's state, position, and velocity. Unlike the particle filter, it operates on two assumptions: (1) target motion and observation models are linear (2) initial estimated probability distribution, and the target errors are Gaussian [22]. However, there are situations where non-linearity exists in target motion and contradicts the first assumption. Therefore, there are variants of the Kalman Filter such as the Extended Kalman Filter (EKF) and Unscented Kalman Filter (UKF) that are developed in case of non-linearity.

Kalman Filter works in two steps [23]: (1) Prediction and (2) Update. In the prediction step, based on some initial state a new prediction of the target state is made. Along with it, an uncertainty/variance prediction is calculated. Uncertainty/variance in autonomous vehicles, for example, is the change in acceleration of the car in front of the ego vehicle. In the update step, Kalman gain is calculated which is the difference in predicted value to the actual measured value from devices such as RADAR or LiDAR on the ego-vehicle. Based on the Kalman gain, the proportion of predicted and measured values to use to update the state is determined. Then, a new prediction and new uncertainty/ variance value are calculated and fed to the prediction step. This continues until the difference between the predicted and the measured value is zero. This calculated value will be the guess made by the Kalman filter.

Using Kalman Filter, lane-level localization can be solved by initializing each lane with one filter which denotes the possible state of the ego-vehicle, and using an inference system to predict the correct hypothesis among the different hypotheses.

2.2 Time series classification

Data is a collection of information, or facts such as observations, numerical values, measurements, descriptions or details about things. Data can be quantitative or qualitative in nature. Quantitative data is numerical information like measurements, qualities, population information, and many more whereas qualitative data is descriptive information like weather, road type, color, and country.

Quantitative data can be further divided into sequential and non-sequential data. Sequential data is a form of data where data points are correlated and have dependencies on each other. Non-sequential data are data where data points are uncorrelated and are independent of each other. One example of sequential data is *Time series*.

In [24], the author(s) describes time series data as a set of observations recorded at different time steps. One factor that differentiates time series data from other sequential data is that it is always presented with time as one of its variables. Time series can be further categorized as univariate and multivariate time series based on the availability of variables. Univariate time series data is time series data with only one variable changing over time, for example, temperature changes recorded by a sensor in a day. Multivariate time series data has multiple variables changing over time. Since time is a ubiquitous factor, time series data is present in various fields

such as medical, automotive, aerospace, weather forecasting, stocks, and engineering. The resulting outcomes of analyzing time series data are prediction, classification, segmentation, clustering, and anomaly detection among others.

Time series classification (TSC) is different from traditional classification problems because in traditional classification the attribute order of the input objects is irrelevant, whereas TSC has temporally correlated attributes defined by complete ordered sequence [25]. In time series classification, the most basic algorithm used is the 1NN-DTW model. DTW (Dynamic Time Wrapping) is a distance measurement method that computes the minimum distance between two sequences with different lengths. The 1NN-DTW model uses 1-Nearest Neighbor(1NN) algorithm to find the nearest training sample of the current instance using DTW as a distance measurement and assigns the same class label to the nearest training sample. It has proved to be highly accurate and does not require training of parameters [26].

Based on the availability of labeled data, time series classification can be approached using supervised or semi-supervised learning methods. One example of a supervised time series classification method is 1NN-DTW which has shown exceptional performance across various datasets from different application domains [27]. 1NN-DTW was considered to be the best method for a prolonged time in the research field [28], [29], [30], [31]. However lately with an increase in research towards time series, algorithms such as ROCKET [32] have been developed that have better performance across various datasets compared to 1NN-DTW [26]. ROCKET [32] transforms time series into features using a large number of random convolutional kernels and used these transformed features to train the linear classifiers. ROCKET obtained the best mean rank over the 85 ′bake off′ datasets in the UCR archive compared to other existing state-of-the-art methods for time series classification such as BOSS, Shapelet Transform, Proximity Forest, ResNet, and HIVECOTE [32].

In semi-supervised methods, the classifiers are constructed using a majority of unlabelled data and a small amount of labeled data. A simple semi-supervised learning approach for time series classification was proposed by [33] where unlabeled data was added to the training set by 1NN if the distance between the labeled and unlabeled data was close by using Euclidean distance as the distance measurement. Improved model performance of the same using DTW as distance measurement was proposed by [34]. Apart from the distance approach for semi-supervised time series classification, label-based and stopping criterion-based approach is also used to optimize the classification task. In [35], the author(s) proposed a label-based approach where small clusters are formed for all sequences, and labels are assigned to the unlabeled data using these seeds. In the classification task, more false negatives or false positives tend to occur if the stopping criterion is too safe or too flexible therefore [36] proposed a parameter-free algorithm to find the stopping criterion using Minimum Description Length (MDL) technique [26].

New requirements for time series classification tasks in specific application scenarios in different domains such as early medical diagnosis of the diseases like asthma, viral infection, and abnormal Electrocardiogram (ECG) in the medical field, for early minimization of maintenance cost in industrial process monitoring and early

classification of driving pattern to reduce accidents in the transportation industry has lead to an extension of TSC called *Early Classification of Time Series* [26], [37].

2.3 Early Classification of Time Series

Early Classification of Time Series (ECTS) is the task of classifying time series data and predicting the class labels as soon as possible using a minimum number of measurements with the highest possible accuracy. The first literature to mention the term 'Early Classification of Time Series' was [38] where intervals were made by segmenting time series and relative predicates, region-based predicates were used to describe these intervals. Base classifiers were constructed using these predicates as features and Ada boost was used to ensemble the base classifiers. The ensemble classifier treated unavailable suffixes of sequences as missing features and made predictions on incomplete data [39]. [40] proposed a case-based reasoning method to classify time series to monitor the system failure in a simulated dynamic system. [40] classified the incomplete time series using KNN classifier and various distances measurements, such as Euclidean distance and Dynamic Time Warping (DTW). The results from this method showed that an increase of classification accuracy occurs on the prefixes through thirty to fifty percent of the full length. [38] and [40] proposed reliable approaches to provide good classification based on incomplete information. However, ways to choose the shortest prefix to provide stable classification results were not considered in [38] and [40].

In [41], the author(s) introduced the novel concept of MPL (Minimum Prediction Length) and developed an ECTS method extending the 1-nearest neighbor (1NN) classification by finding the MPL for a cluster of similar time series instead of a single time series. It provided reliable classification results with high accuracy and effectiveness. However, since the early time series classification have application in domains such as medical and health informatics, security, and safety management there is a need for interpretability of classification results. Therefore, [42] proposed EDSC (Early Distinctive Shapelet Classification) which used a shapelet-based approach of extracting local shapelets as features that are subsequences of time series belonging to the same space of input data that can distinctively demonstrate a target class early and regionally. In EDSC, the best shapelet for classification was selected using kernel density estimation or Chebyshev inequality which provided the threshold value for each shapelet. [43] extended EDSC to multivariate time series and proposed the MSD (Multivariate Shapelet Detection) extracts patterns from all dimensions of the time series. MSD used an information gain-based distance threshold to create local shapelets which outperformed the Chebyshevs inequality method. The utility score method was used to rank the shapelets based on earliness and class discrimination. The shapelet pruning method was used to select the subset of the shapelets for classification. This method had limitations such as only fixed length sequences could be handled, multivariate shapelets were constructed from segments extracted from the same sliding window time frame at the exact time. However, patterns that can help better distinguish between target classes are present in different time intervals for different variables. Therefore, longer intervals could have

been used for segments but it would defeat the purpose of early classification.

To deal with the imbalance of sub-category data within the same class and find distinctiveness within the multivariate time series data [44] proposed MCFEC (Mining Core Feature for Early Classification) method. MCFEC was developed to enhance the interpretability of classification and discover the inner characteristics of Multivariate time series (MTS) data by extracting feature candidates of each variable independently. This method deals with the imbalance within the same class problem by mining core features using feature selection and evaluation strategies. These core features are then used to train two classifiers that classify the unlabeled MTS data in an interpretable way. Further, [45] developed a method using probabilistic classifiers for the problem of early time series classification. A significant feature of this approach is the identification of the time length at which the prediction can be accurately made. In this approach for an unseen time series, prediction is made only when the difference between the two largest predicted class probabilities is over a certain threshold which is found during the training phase. This method demonstrated higher accuracy and earliness when validated against 45 benchmark time series databases as well as in comparison to other state-of-the-art methods. In 2019, [46] presented a new approach to the problem by making it a task of simultaneous optimization of earliness and accuracy. A trigger function was used as a stopping rule to decide whether to output a prediction or wait for a longer period by optimizing earliness and accuracy. This multi-objective optimization approach has dominated the state-of-the-art methods in ECTS. A two-tier architecture approach 'TEASER'[47] was published in 2020 and aimed to solve the problem of identifying the optimal and individual decision time using a combination of primary-secondary probabilistic classifiers.

2.3.1 Early Classification of Time Series by Simultaneously Optimizing the Accuracy and Earliness by Mori et al.(2019)

The paper presents a novel approach of multi-objective optimization over accuracy and earliness to solve the problem of early classification of time series. The proposed method uses a combination of probabilistic classifiers and stopping rule/trigger function which is optimized using a genetic algorithm. The distinctive part of this method is the stopping rule which is a process of simultaneous optimization to find a trade-off between accuracy and earliness.

In the learning phase, a probabilistic classifier $h_{t\,t=1}^{L}$ is trained on a subset of time steps $t \in 1, 2, ..., L$ which is equidistant based on domain knowledge or user-defined time steps. The Stopping rule/trigger function defined in [46] is used to evaluate the reliability of the probabilistic classifier output that is to decide if the prediction made by the classifier at a given time step is reliable and if the class label with the highest probability can be output as the predicted label or to wait for more time steps to have a reliable prediction. The stopping rule defined by [46] is:

$$TR_\gamma(p_t, t) = \begin{cases} 0 & \text{if } \gamma^1 p_t^1 + \gamma^2(p_t^1 - p_t^2) + \gamma^3 \frac{t}{L} \leq 0 \\ 1 & \text{otherwise} \end{cases} \tag{2.1}$$

Here, p_t refers to the probability at time step t for different classes, p_t^1 is the highest probability and p_t^2 is the second highest probability. L is the full length of the time series. $\gamma = (\gamma^1, \gamma^2, \gamma^3)$ are the variable parameters that can range between -1 and 1.

The interpretation of the stopping rules is that '0' indicates that the output of the classifier is not reliable and the class label with the highest probability is not reliable to be predicted as the label that may correspond to the true label and '1' indicates that the output is reliable and the class label with the highest probability can be relied upon to correspond to the true label.

The optimization of the variable parameters $\gamma = (\gamma^1, \gamma^2, \gamma^3, ..., \gamma^k)$ in both the stopping rules/ trigger functions are made using a genetic algorithm where the objective is to optimize between earliness cost and accuracy cost.

The earliness cost function to determine the optimal parameters $\gamma = (\gamma^1, \gamma^2 ... \gamma^k + 1)$ for the unseen time series can be defined as the average elapsed length of the time series in the training set for which the stopping rule outputs 1.

$$C_e(TS, TR_\gamma) = \frac{1}{|TS|} \Sigma_{ts \in TS} \frac{t_{ts}^*}{L} \tag{2.2}$$

where, TS represents all the time series in the training set. $\frac{t_{ts}^*}{L}$ is the earliest time step of the time series ts where TR_γ outputs 1.

The accuracy cost function can be defined as the average error in classification or misclassification rate.

$$C_a(TS, TR_\gamma) = \frac{1}{|TS|} \Sigma_{ts \in TS} f(y_{ts}^* \neq TL_{ts}) \tag{2.3}$$

where, y_{ts}^* is the class label corresponding to the highest probability, TL is the true class label and f is the boolean function that outputs '1' if the condition is true otherwise it outputs '0'.

2.4 Probabilistic Classifier : XGBoost

XGBoost is a gradient-boosting library that is optimized and distributed. It is designed to be portable, highly efficient, and flexible. XGBoost uses the Gradient Boosting framework to implement machine learning algorithms and providing parallel tree boosting which solves numerous data science problems in an accurate and fast manner [48].

Some of the significant features of XGBoost are [49]:

- The model provides parallelization and uses multiple CPU cores for training.
- To avoid overfitting, XGBoost has different regularization penalties which help in successful training and help the model to generalize well.
- XGBoost can detect and learn from non-linear patterns.
- It has built-in cross-validation.
- It can process large amounts of data because of its distributed nature to run on servers and clusters such as Hadoop and Spark.
- It is also available for many programming languages such as Python, Julia, C++, and JAVA.

The working of XGBoost is explained in [50] as follows: XGBoost is based on decision trees and uses other methods such as random forest and gradient boosting to improve its performance. In XGBoost, a training dataset is fit to the model to make an initial prediction, and similarity scores of residuals are used to create a decision tree. The residuals are computed using the predicted and observed values. In XGBoost, when a split is made, the similarity of the data in the leaf and gain in similarity in the subsequent split is calculated and compared to determine the feature to be selected for the split and the threshold to be set for the node. Each leaf output value is also calculated using residuals. In the case of classification, the output values are computed using probabilities or a log of odds. The output of this decision tree is used as a new residual for the dataset to construct another tree. This process continues until the residuals are consistent or for a user-specified number of iterations. Each tree learns from the previous tree and is assigned different weights. The final prediction or classification from this model is calculated by multiplying a learning rate by each tree and adding the initial prediction to it.

2.5 Multi-objective Optimization

Multi-objective optimization is the process of optimizing over two conflicting objectives at the same time [51]. The solutions obtained from the multi-objective optimization is called the pareto optimal solutions or non-dominating solutions. A Pareto optimal solution is defined as the solution where no one objective improves without degrading another objective [51]. The set of all the pareto optimal solutions is called the Pareto front. Mathematically, Pareto optimal solution is defined as:

A pareto optimal solution is a point $z^* \in Z$ *(a feasible search space), if and only if there exists no such point* $z \in Z$ *such that* $F(z) \leq F(z^*)$ *with at least one* $F_i(z) > F_i(z^*)$

The figure 2.1 is a visual representation of the pareto front. The black circles' represent the pareto optimal solutions. The green curve represents the pareto front. The $F_1(z^*)$ represents the optimal point (extreme point) of objective function F_1. The $F_2(z^*)$ represents the optimal point (extreme point) of objective function F_2. It must be noted that the optimal point of one objective function represents the worst point of another objective function in multi-objective optimization consisting

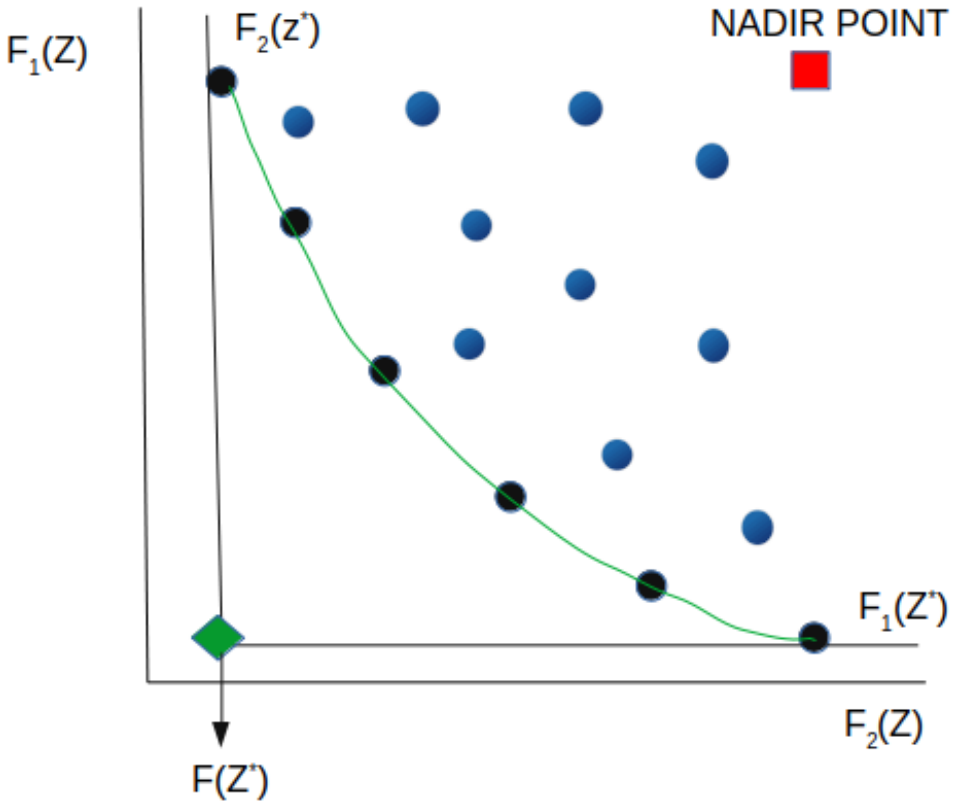

Figure 2.1: Pareto Front

of two objectives. $F(z^*)$ is the ideal point for both objective functions called the 'Utopia point'. The red point is the 'Nadir point' which represents the range of the non-dominated solution set and is obtained from the extreme points of the objective functions F_1 and F_2. The region between the pareto front and the nadir point is a region consisting of feasible solutions which are represented as blue circles. Hypervolume is used as the performance indicator which helps to determine the convergence of the algorithm.

In [51], the author(s) categorized multi-objective optimization mainly into two techniques: evolutionary and swarm-based. The Evolutionary method makes use of natural evolution concepts and is made up of a class of methods. This method produces a set of non-dominating solutions in a single execution of the algorithm and requires less computational power. Swarm-based is a simple and robust technique to determine optimal solutions which are self-organized and decentralized systems.

In [51], the author(s) further distills multi-objective evolutionary algorithms (MOEA) into dominance-based, indicator-based, and decomposition-based algorithms. In a dominance-based algorithm, the pareto dominance principle is used to assign fitness to solutions. Some examples of dominance-based algorithms are Multi-Objective Genetic Algorithm (MOGA) [52], and NSGA-II [53]. In indicator-based algorithms,

the selection of individuals is determined by the value of the indicator measure. Some of the examples of indicator-based algorithm are Portfolio Selection Multi-objective Evolutionary Algorithm (POSEA)[54] and Hypervolume Estimation Algorithm (HypE) [55]. In decomposition-based algorithms, scalarization is used to decompose a problem into several simpler sub-problems and is solved in a collaborative manner. Some of the examples of decomposition-based algorithms are Multi-objective Evolutionary algorithms based on decomposition (MOEA/D) [56] and Non-Dominated Sorting Genetic Algorithm-III (NSGA-III) [57].

In [51], Sharma et al.(2022) explains about MOEAs as: MOEAs work on 'population' which are a set of individuals representing a point in a search space of possible solutions. The population is randomly seeded and new generations are produced through selection, recombination and mutation across generations so that a more optimal search space region is generated. The fitness levels of the entire population is evaluated and new people with better fitness levels are formed by merging the individuals with highest fitness in the population. After a few generations, the algorithm converges and the best fitness individuals represent the optimal solutions. The optimization algorithm used in our proposed system is AGE-MOEA [58].

2.5.1 Adaptive Geometry Estimation based Multi-Objective Evolutionary Algorithm (AGE-MOEA)

AGE-MOEA is an adaptive evolutionary algorithm based on non-euclidean geometry for many-objective optimization introduced by Annibale Panichella in 2019 [58]. The framework of AGE-MOEA defined by [58] inherits the overall framework of NSGA-II but uses survival score that combines diversity and proximity of the non-dominated fronts instead of the crowding distance. In each generation, AGE-MOEA estimates the geometry of the front using a fast heuristic with $O(M \times N)$ computational complexity. The proximity is estimated as the distance between each member of the population and the ideal point and the diversity is measured as the distance between the population members. To compute proximity and diversity, the distance used corresponds to the $L_p norm$ which is associated to the estimated geometry. The advantage of AGE-MOEA is that it does not make any assumption about the geometry of the pareto front and the non-dominated front produced in each generation.

3

Data

The data used for this project is provided by Zenseact. It is an internal dataset generated by Map-based Road Estimation from the field test. It is a geographically diverse dataset spanning several countries in Western Europe and the United States of America. It was collected over the last two years by specialized data collection vehicles.

Section 3.1 describes the data, explaining the main features, the metadata, and the ground truth. In Section 3.2, the data analysis is provided which is followed by the data pre-processing in Section 3.3. Finally, Section 3.5 contains the details regarding the split of the data into various sets (training, validation, optimization, and test).

3.1 Data description

The dataset consists of time series of processed sensor data corresponding to how well each of the perceived sensor data matched with the map, based on the estimated vehicle pose and its uncertainties. The time series data is segmented in the form of sequences. There are a total of 2766 sequences with each sequence identified by a unique sequence name and number. The sequences total around 50 hours of driving data. The data is fetched from driving logs in different countries such as:

- Sweden: 767 sequences
- France: 653 sequences
- USA: 581 sequences
- Germany: 423 sequences
- Belgium: 328 sequences
- Denmark: 11 sequences
- Luxembourg: 3 sequences

Each sequence is of varying length and starts when the hypotheses are initialized. The sequences were not cut short to the same length since in a practical scenario the model must be able to handle all such varying-length sequences. The length of the sequences varies from 12.5 seconds to 27.05 seconds with a mean sequence length of 24.71 seconds. Each sequence consists of 6 hypotheses, with the values

(Model Qualities and Confidence together) of the hypothesis not active being 0. Each sequence has several rows of features ordered by their timestamps. The histogram containing the length of the sequences is shown below in Figure 3.1:

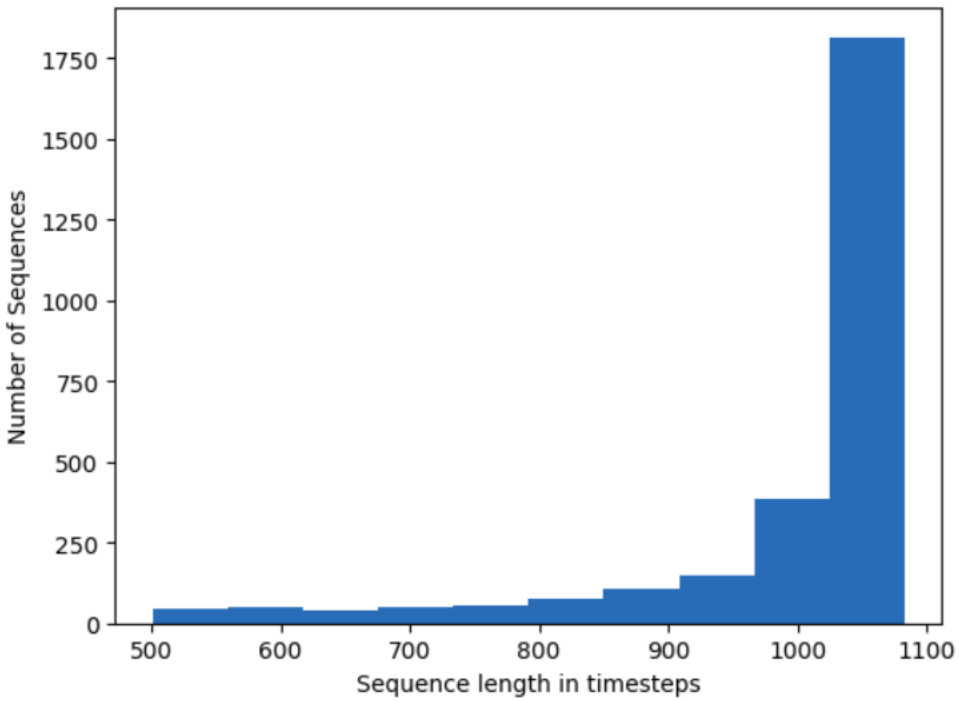

Figure 3.1: Sequence Length

3.1.1 Features

There are a total of nine main features used in the model followed by other metadata. The main features are the Model Qualities (MQs) which are obtained by perceiving how well the sensor data matches with the HD map. The MQs belong in the numerical domain of (-∞, 0] where a higher value closer to zero means that the sensor data and the HD map match well. They are obtained at the rate of 40 Hz. The following MQs comprise the main feature set:

- Primary Left camera
- Primary Left lane type
- Primary Right camera
- Primary Right lane type
- Secondary Left camera
- Secondary Left lane type
- Secondary Right camera
- Secondary Right lane type

- Adjacent vehicle information

The primary left camera MQ pertains to how well the lane marking geometry of the left camera matches with the pre-defined HD map geometry in the lane the vehicle is in currently. The lane type refers to the type of the lane which can be solid, dashed, etc. Similarly, the secondary denotes the next adjacent lane to the one the vehicle is in currently. Adjacent vehicle information denotes how well the geometry of adjacent vehicles perceived by the sensors matches with the HD map.

The primary and secondary left and right lane features overlaid on the map for a correct hypothesis and an incorrect hypothesis are depicted in Figures 3.2 and 3.3. The chosen hypothesis is depicted with a green diamond on top of it. The model qualities will be close to zero if the HD map and the sensor data match like in the case of the correct hypothesis. In the first figure, since the hypothesis chosen is the correct hypothesis we can see that the lane marker types are matching while in the second figure, the lane marker types do not match.

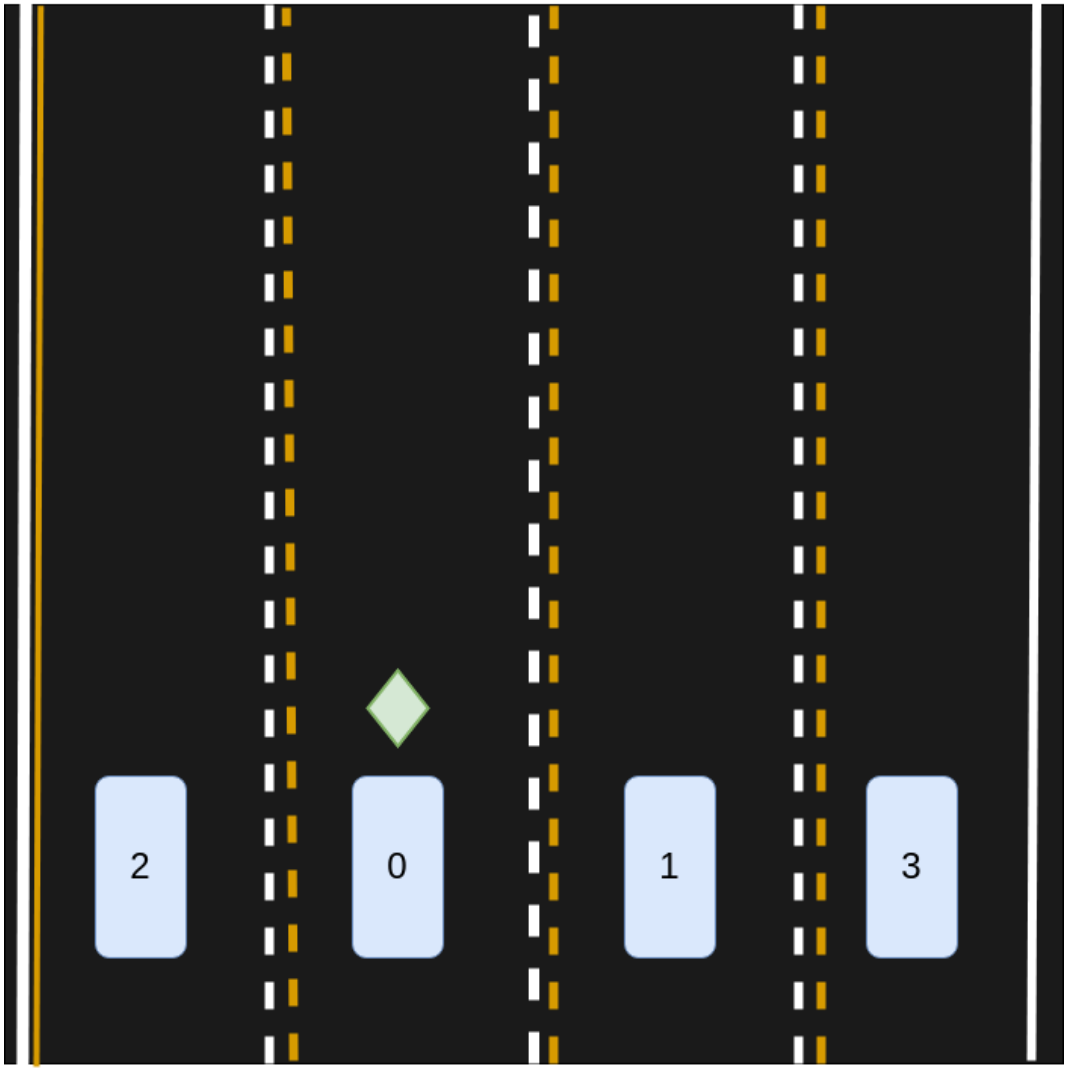

Figure 3.2: Correct hypothesis

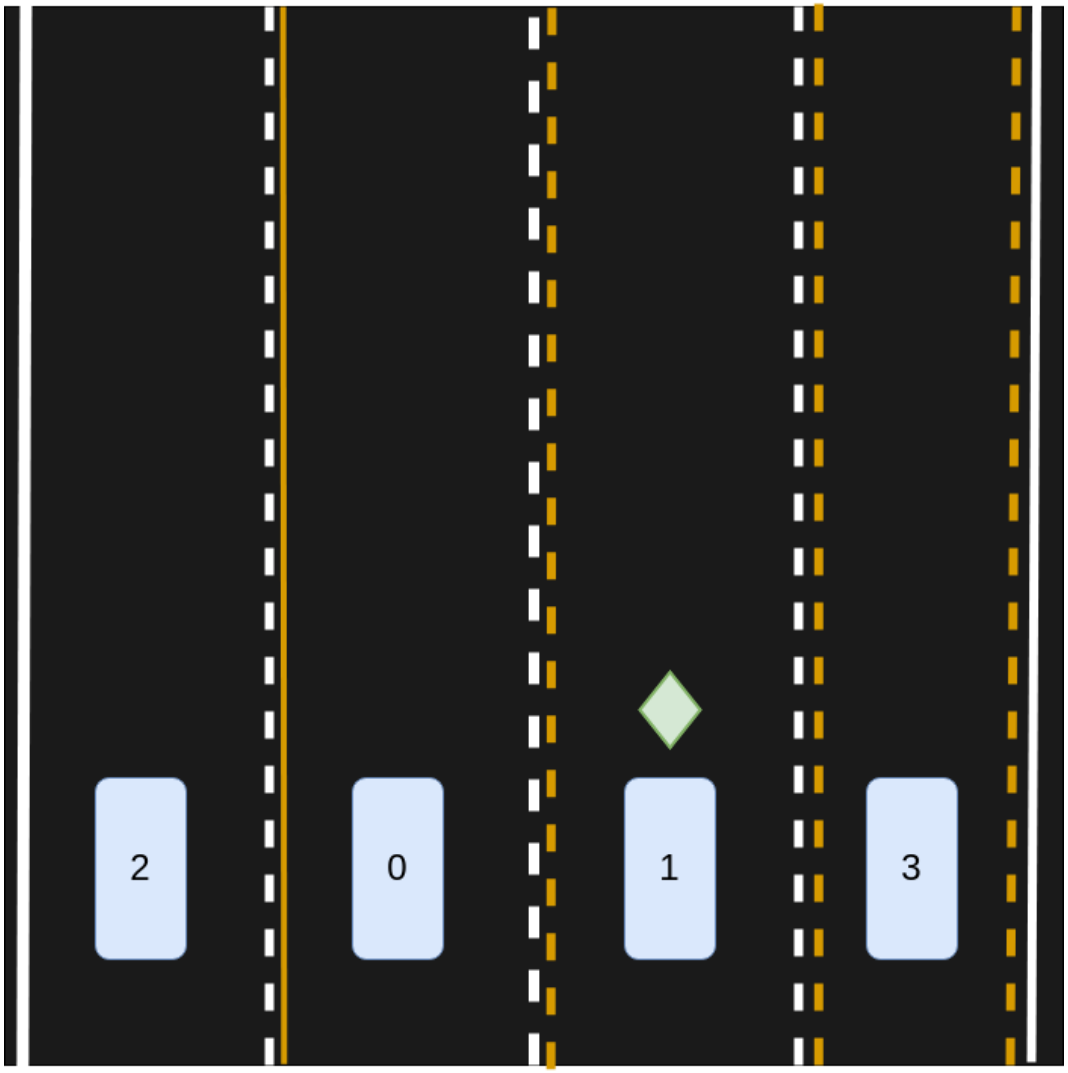

Figure 3.3: Incorrect hypothesis

Apart from the main features, there are a number of secondary features or meta-data. The meta-data includes:

- Latitude
- Longitude
- Country
- Weather
- Road-type

The latitude and longitude are coordinates that depict where the vehicle was in that timestamp. The country denotes the country where the data was collected. The weather contains weather information such as "Snow, Rain, Cloudy, etc". The road-type denotes the type of the road categorized as 0, 1, 2, 3, or 4 as mentioned in the highway definition from OpenStreetMap [59]:

- 0: highway - motorway
- 1: highway - trunk, raceway
- 2: highway - primary, motorway_link, trunk_link, primary_link,
- 3: highway - secondary, secondary_link
- 4: highway - tertiary, residential, unclassified, living_street, pedestrian, service, tertiary_link

3.1.2 Ground Truth

In the data collection vehicles that are used to collect the data, high-precision GNSS/INS systems are present. These GNSS/INS systems are used to get the Ground truth trajectory. The HD map that we have is not globally aligned but only locally aligned. Thus there is an extra filtering tool that corrects the trajectory from the OxTS (Oxford Technical Solutions) INS to align it with the map.

Each hypothesis is assigned to a lane in the map. Thus the label is obtained by checking which hypothesis is assigned to the same lane as the ground truth. There are also some additional tests to ensure that the ground truth is not located right in the middle of two lanes or to detect if a hypothesis is drifting into another lane.

3.2 Data Analysis

Using the XGBOOST gradient boosting method the main features were analyzed and the feature importance was obtained for every feature. The feature importance is plotted below in Figure 3.4:

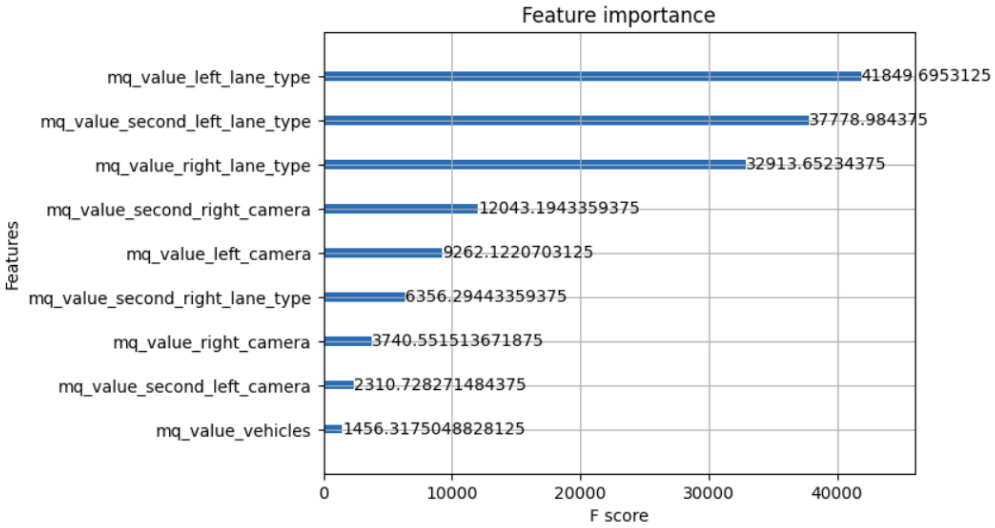

Figure 3.4: Feature Importance

From this, we can see that the Left lane type is the most important feature, followed by the second left lane type and the right lane type. It is evident that the lane types

play a more important role than the lane geometries which is expected since an incorrect lane type will easily be grounds for rejecting an hypothesis.

The mean, standard deviation, median, minimum, and maximum of the entire data for the chosen 9 features are given in the following table:

Feature	Mean	Std	Median	Min	Max
P left camera	-0.260674	1.107839	-0.007736	-69.771599	0.0
P left lane type	-1.609927	2.897804	-0.116209	-8.583140	0.0
P right camera	-0.838204	2.264026	-0.012831	-154.290283	0.0
P right lane type	-1.315685	2.855347	0.0	-8.583140	0.0
S left camera	-0.715610	1.924063	0.0	-26.149216	0.0
S left lane type	-1.267987	2.912863	0.0	-8.583140	0.0
S right camera	-0.968566	2.423541	0.0	-22.332125	0.0
S right lane type	-0.677366	2.218062	0.0	-8.583140	0.0
Adjacent vehicle info	-0.701159	1.633593	0.0	-8.988458	0.0

Table 3.1: Data metrics. P - Primary and S - Secondary

Figures 3.5 and 3.6 show the distribution of the features as a histogram containing 50 bins. The distributions are split into two colors based on the label. This depicts the density of the model quality values contributing to 0 or 1 labels i.e., the density of the model quality values coming from a correct or wrong hypothesis. Looking at the images we can come to the conclusion that the hypotheses with label 1 have model quality values that are very close to zero, i.e., close to ideal value.

A box plot showcasing the feature values is presented in Figure 3.7. Here the Q1-Q3 is shown in a blue box while the median is depicted by a green line. The outliers are depicted as circles. We can see that the majority of the values are close to zero while some features like the primary and secondary left camera and primary and secondary right camera have a lot of outliers.

3.3 Data Pre-processing

For many machine learning models, the features used will be in different scales and thus will require scaling to prevent the model from unknowingly giving more weight to a specific feature. But in our case, all the nine features used are model qualities that are in the same range. Thus scaling was not necessary for our model.

For handling the missing data in the dataset they are two main practices: (i) Removing the rows with missing data and (ii) Imputing the average value of the feature for the missing values.

If we consider removing rows then the total data size will reduce. This can also introduce an inherent bias i.e., certain scenarios that produce more missing data will not be present in the data. On the other hand, imputing values also comes with

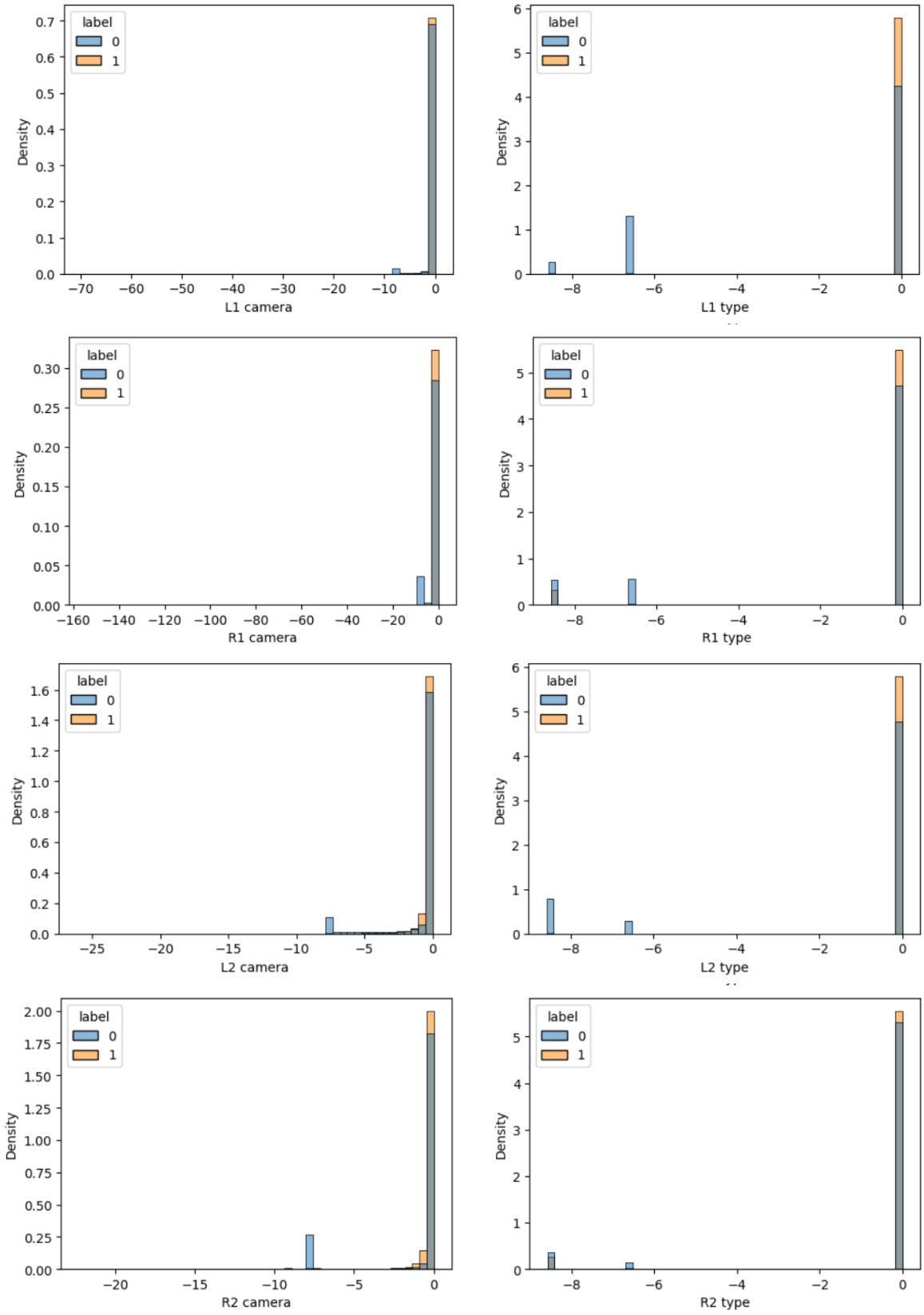

Figure 3.5: Feature Distribution

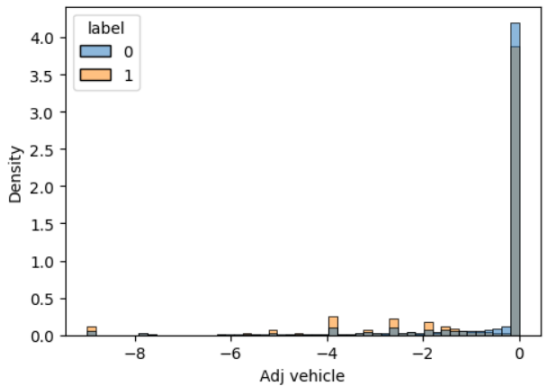

Figure 3.6: Feature Distribution (cont)

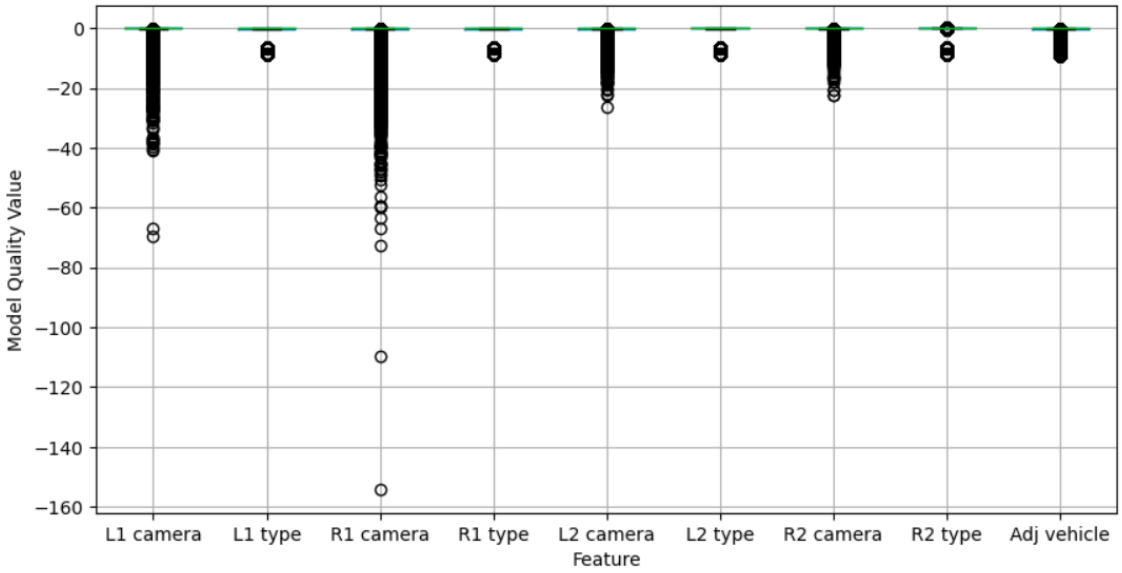

Figure 3.7: Box plot - Features

its own set of disadvantages. Adding values that were not present originally will alter the dataset and create a new bias.

Experiments were conducted for both these possibilities and since they didn't cause any significant improvement, missing rows were not altered. This was also done due to the choice of the probabilistic classifier that we have in our model. XGBOOST is a classifier that has the built-in capability to handle missing data very well. Thus we did not have to impute or remove rows with missing data and were also able to preserve the originality of the data.

3.4 Data Split

The entire data set is split into Training, Optimization, Validation, and Test set. The exact split is as follows:

- Training: 1469 sequences
- Optimization: 490 sequences
- Validation: 490 sequences
- Test: 317 sequences

The data in train, val and opt should be mutually independent with the data in Test. For evaluation to be fair there should not be any overlap between these two sets.

For this reason, we clustered the data using K-means clustering. This was done using the latitude and longitude coordinates. Then the sequences were selected for the test set without having any overlap with the rest of the sets. This was done manually since we didn't want just random sequences without overlap in the test set but for it to actually have sequences that are difficult in real-life scenarios. Thus, we selected sequences from the middle of the cities in busy highways, ones with several lanes, etc. to account for various types of difficult scenarios in the test set.

To create a diverse and complete dataset, we made sure the datasets contained data from all the available countries (Sweden, France, USA, Germany, Belgium, Denmark, and Luxembourg). In this way, the model will be robust and ready to predict in different countries.

4

Methodology

4.1 Proposed system architecture

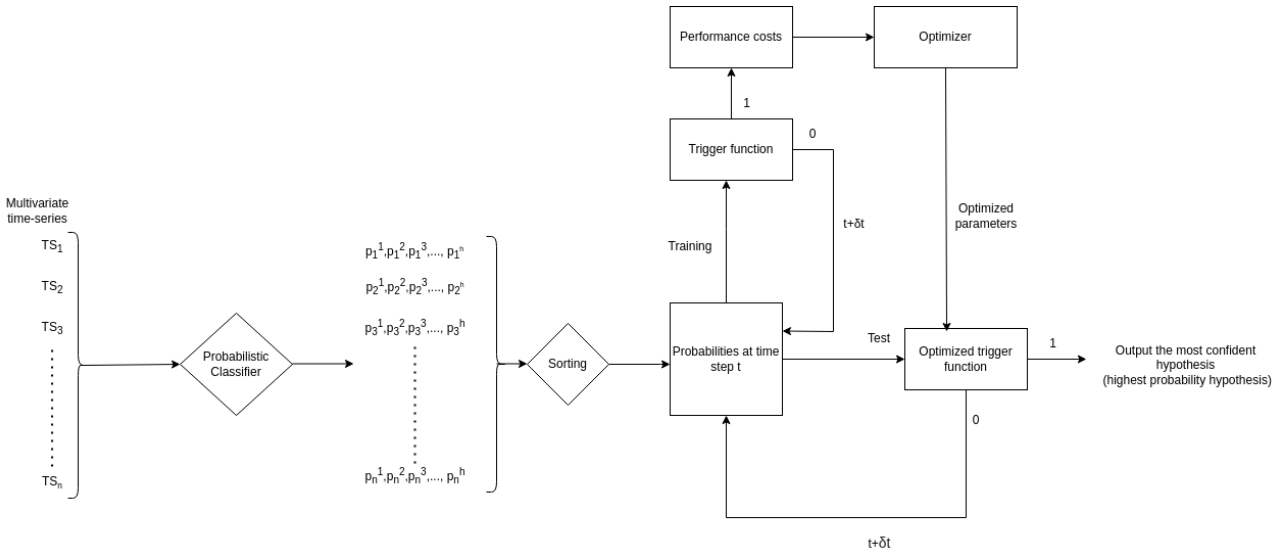

Figure 4.1: Proposed system architecture based on Mori et al.(2019)[46]

The architecture is based on the solution design proposed by [46] but uses a different classifier, trigger function, and optimization algorithm that are suitable to the project's goals, domain, and data to achieve the desired results. The main components of the system are the probabilistic classifier, trigger function, and optimization algorithm. The input to the system is a set of n multivariate time series TS_i, $i \in [1, n]$ that are tested at time point t. The output of the system is the hypothesis with the highest probability of being correct. The system works as follows: The input is given to the probabilistic classifier which outputs the probabilities for each hypothesis which are then sorted according to the probability values. The hypothesis with the highest probability is the predicted label. However, it must be noted that the task at hand is a binary classification problem that checks whether the predicted label corresponds to the true label. This predicted label is input to the trigger function which decides whether the predicted classifier output is reliable or if the system has to wait for more data (time steps) to be confident that the prediction made by the classifier is correct. In the training phase, the probability at

time step t is input to the trigger function, if it outputs '0' then the probability at $t + \delta t$ is fetched and sent to the trigger function. This process continues until the trigger function outputs '1'. Then, the performance costs (accuracy and earliness costs) are sent to the optimizer for obtaining the optimal parameters. During the testing phase, the same process is carried out but a trigger function with optimal parameters is used to take the decision. The role of the optimization algorithm is to obtain optimal parameters for the trigger function so that the system can make an accurate and early decision.

4.1.1 Training the classifier

The initial step is to obtain a trained classifier which can output class probabilities at different time steps. As seen in figure 4.1, the classifier is trained using a labeled set of multivariate time series TS_i, $i \in [1, n]$ where n is the number of sequences. A separate validation set which is also a multivariate time series is used to optimize the training of the model. The minimum and maximum length of all the time series TS_i, $i \in [1, n]$ used in training are 505 time steps and 1083 time steps respectively. Since the momentary data points have fluctuations in value, a rolling window of size 100 is used to model the data to obtain a stabilized input for training the model. Finally, the trained classifier is obtained. In the proposed system, the learning algorithm used to train the classifier is XGBoost. The output from the trained classifier are class probabilities (probability of each hypothesis being the correct hypothesis).

4.1.2 Trigger function

As mentioned above, the trigger function is responsible for deciding whether the output from the classifier at time step t is reliable enough to make a decision or wait for more time steps (more data). The class probabilities obtained from the trained classifier are sorted as per the highest magnitude. All the n multivariate time series TS_i, $i \in [1, n]$ are truncated to time step t. Here, the time step t ranges between [40, L_i] with a step size of 100, where L_i is the length of TS_i with $i \in [1, n]$. The first and the second highest probabilities (p_t^1, p_t^2) along with the the time step t are used as the input for the trigger function. The output is '1' or '0' where '1' represents that the output is reliable whereas '0' represents that the output is unreliable and the system has to wait for more time steps. A new trigger function is used in the proposed system, which is a simplified and modified version of the trigger function defined by [46] and is given as:

$$TR_\gamma(\mathbf{p_t}, t) = \begin{cases} 0 & \text{if } \gamma^1(p_t^1 - p_t^2) + \gamma^2 \frac{t}{L_i} \leq 0 \\ 1 & \text{otherwise} \end{cases} \tag{4.1}$$

Here, p_t^1, p_t^2 are the first and the second highest class probabilities respectively. t is the time step at which the probabilities are being evaluated. L_i is the full length of the time series TS_i with $i \in [1, n]$. γ^1, γ^2 are the optimized parameters obtained from the optimization algorithm. The first term ensures that the hypothesis with the highest probability is the reliable choice by making sure that the difference between

the two highest probabilities is significant. If there are two contending hypotheses, the system waits to obtain more clarity. The second term containing the ratio of the elapsed timestep to the total length of the sequence is used to ensure earliness by adding a cost for waiting longer. This trigger function is the same as that of [46] without the term consisting of only the highest probability. This trigger function is used during both the training and testing phases.

During the selection of the trigger function for the proposed system, trigger functions used in [46] and the previous thesis [12] were experimented with. The performance of the proposed system using these trigger functions are mentioned in the Experiments and Results section 5. The trigger functions that were experimented with are defined as:

$$TR^1_\gamma(\mathbf{p_t}, t) = \begin{cases} 0 & \text{if } \gamma^1 p^1_t + \gamma^2(p^1_t - p^2_t) + \gamma^3 \frac{t}{L} \leq 0 \\ 1 & \text{otherwise} \end{cases} \tag{4.2}$$

$$TR^2_\gamma(\mathbf{p_t}, t) = \begin{cases} 0 & \text{if } \gamma^1 p^1_t + \gamma^2(p^1_t - p^2_t) \leq 0 \\ 1 & \text{otherwise} \end{cases} \tag{4.3}$$

$$TR^3_\gamma(\mathbf{p_t}, t) = \begin{cases} 0 & \text{if } \gamma^1 p^1_t + \gamma^2(p^1_t - p^2_t) + \gamma^3 \frac{p^1}{p^2} + \gamma^4 \leq 0 \\ 1 & \text{otherwise} \end{cases} \tag{4.4}$$

$$TR^4_\gamma(\mathbf{p_t}, t) = \begin{cases} 0 & \text{if } \gamma^1 p^1_t + \gamma^2 p^2_t + \gamma^3 p^3 + \gamma^4 p^4 + \gamma^5 p^5 + \gamma^6 p^6 + \gamma^7 \frac{t}{L} \leq 0 \\ 1 & \text{otherwise} \end{cases} \tag{4.5}$$

Here, TR^1 is the trigger function defined by [46]. Similar to the working of the proposed trigger function, it has an additional term specifying the highest probability thereby adding another term for ensuring accuracy. TR^2 and TR^3 are trigger functions defined in the previous year's thesis [12]. TR^2 is a simplified version of TR^1 without the last term that represents the percentage of elapsed time. TR^3 is similar to TR^2 but includes an additional variable parameter and a term that represents the ratio of the first and second-highest probabilities. This term emphasizes the difference between the two highest probabilities ensuring reliability in the prediction. TR^4 is the trigger function used by [45] which consists of all the class-probabilities (hypothesis) and the ratio of the elapsed time length to the full length of time series.

4.1.3 Optimization of the trigger function

Here, the optimization problem is the minimization of the earliness and accuracy cost functions simultaneously. This multi-objective approach used in the proposed system is based on [46]. The earliness and accuracy cost functions are the same as equation 2.2 and equation 2.3 mentioned in section 2.3.1. The optimization problem is defined as:

$$min_{\gamma}(C_a(TS, TR_{\gamma}), C_e(TS, TR_{\gamma})) \tag{4.6}$$

To solve this optimization problem, optimal parameters $(\gamma_1, \gamma_2 ... \gamma_n)$ have to be found for the trigger function. An optimization algorithm AGE-MOEA is used to obtain the optimal parameters for the trigger function. The first and the second highest probabilities from the probabilistic classifier and initial parameter values [-1, 1] as suggested in [46] are used to get the initial earliness and accuracy cost from the trigger function. The initial earliness and accuracy cost are used as input to the optimization algorithm to obtain a set of pareto optimal solutions. Hypervolume with arbitrary reference point set to [1.1, 1.2] that is slightly worse than the nadir point is used to determine the convergence of the optimization algorithm. The Achievement scalarization function is used from the pymoo package [60] and is defined in [61]. It is used with weights [0.25, 0.75] chosen by experimenting with different values and is used to obtain the optimal parameters for earliness and accuracy.

4.2 Evaluation & Metrics

The two main metrics used to evaluate the proposed system are accuracy and earliness. Accuracy is a simple and widely used evaluation metric for classification tasks. Accuracy is defined as the fraction of correct predictions made by the model. Formally, accuracy is defined as:

$$Accuracy = \frac{Number\ of\ correct\ predictions}{Total\ number\ of\ predictions} \tag{4.7}$$

Accuracy for binary classification is defined as:

$$Accuracy = \frac{TP + TN}{TP + TN + FP + FN} \tag{4.8}$$

where, TP- True positive, TN- True negative, FP- False positive, FN- False negative.

Earliness is defined as the average portion of the elapsed time sequence before a decision is triggered. The performance of the optimization algorithm is evaluated using hypervolume with a reference point set to [1.1, 1.2]. The *Achievement Scalarization Function (ASF)* is used to determine the optimal solution using weights [0.25, 0.75] for earliness and accuracy respectively. The weights represent the importance given to each of the cost functions. The reason for accuracy being given more importance is that the prediction made in the case of localization is required to be more accurate than fast.

5

Experiments and Results

5.1 Previous Thesis System - Baseline

The system built by the previous year's thesis [12] was used as the baseline for comparison. The previous system was tested and it produced an accuracy of 92.744 % and an earliness of 0.020. When analyzing the system it produced 23 False Positives and 64 Late predictions all of which were no predictions meaning the system didn't take a decision until the entire sequence was seen.

Overall the major problem faced by the former system was that it prioritized taking a decision quickly over an accurate decision. This led it to produce more false positives. In a real-world scenario, a false positive is even more dangerous than a late prediction. A false positive can lead to a serious altercation.

Another problem with the previous system was that it nearly always predicted after a similar number of rows around 19 or 20 rows. The model had learned to predict at that time irrespective of the MQ values situation. This is not correct since when two hypotheses are very close in confidence the model needs to wait to take the decision.

5.2 Classifier comparison

As part of the system, a probabilistic classifier has to be chosen to make the predictions. The previous thesis work [12] came to the conclusion that Gradient boosting was the best of Gaussian Naive Bayes, Logistic Regression, Linear Support Vector Classifier, Kernel Support Vector Classifier, Random Forest Classifier, and Gradient Boosting Classifier for this problem.

Instead of just a standard Gradient boosting classifier, there are more advanced boosting variations that we believe will improve the performance of the model. Thus, we compare Gradient boosting, XGBoost, and LightGBM.

We choose Gradient boosting to be part of the testing group since it is a classifier with proven performance and we keep it as a benchmark. XGBOOST is chosen as it is a state-of-the-art algorithm for tabular data [62], [63]. Light GBM has also proven to be highly competitive with tabular and time-series data, hence it was chosen as part of the test.

Gradient Boosting was implemented and run using the Python scikit-learn library [64] while XGBOOST and LightGBM were implemented and run using their respective Python libraries [65] and [48]. All three of them are run with their default configuration and trained on the same data and tested on a separate test data set. For the optimizer, we use the AGE-MOEA and for the trigger function we use our new trigger function, to keep it consistent, although, the tests for finalizing the other aspects such as the optimizer and trigger functions are enumerated in the subsequent subsections.

The performance of the optimization algorithm using the respective classifiers and their non-dominated solution sets are depicted below in the Figures 5.1, 5.2, and 5.3 for each classifier. The performance and the convergence are on the left while the non-dominated solution sets are on the right. The selected solution set is depicted with a red 'x' mark. The ideal solution and the nadir point are also depicted with a red star and gray hexagon respectively.

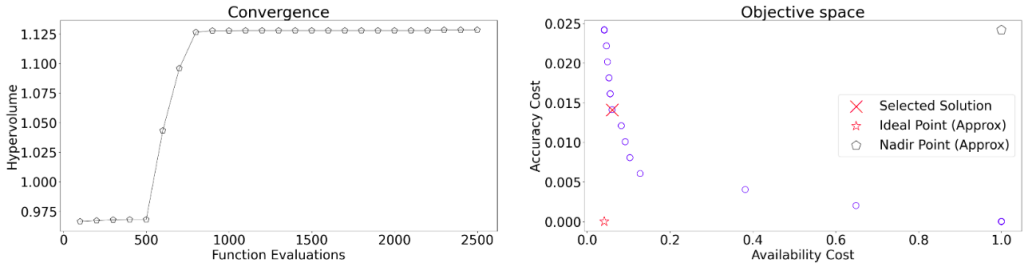

Figure 5.1: Gradient Boosting

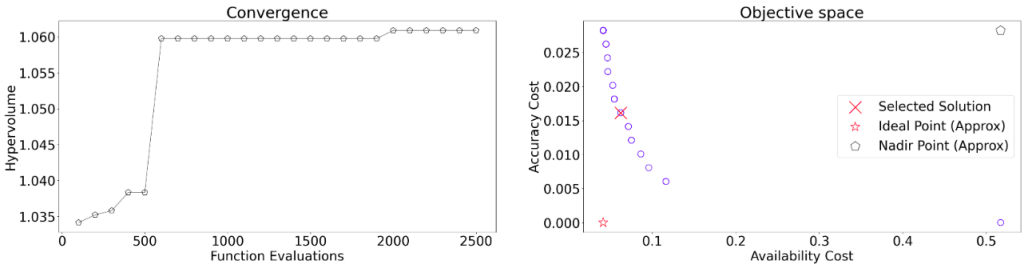

Figure 5.2: LightGBM

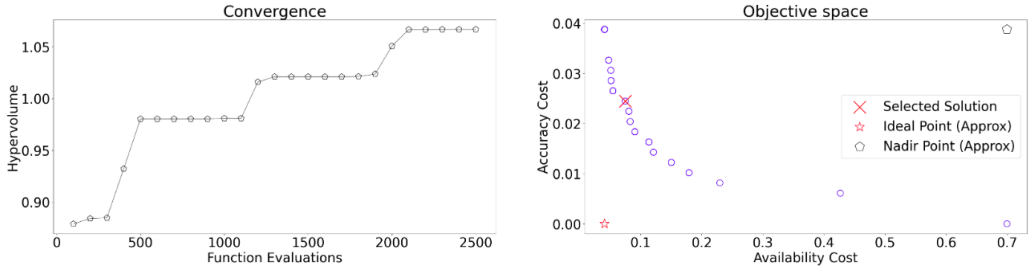

Figure 5.3: XGBOOST

Each model's performance measured by their accuracy and earliness is reported in the table 5.1 below:

Classifier	Accuracy	Earliness	FPs	LPs
Gradient Boosting	96.529 %	0.083	11	13
Light GBM	96.845 %	0.090	10	16
XGBOOST	99.053 %	0.109	3	23

Table 5.1: Classifier comparison, where FP - False Positive and LP - Late Predictions

It is clear from the table that XGBOOST has the highest accuracy in the given setting. LightGBM and Gradient boosting have similar accuracy in this setting. We can see that the earliness is better with LightGBM and gradient boosting but we need to understand that these models make a wrong prediction without waiting for more data in situations where there are two close hypotheses. Whereas the XGBOOST model waits for more data to take a clearer decision. In these important situations, false positives are far more important than earliness. Thus, we choose XGBOOST as our classifier.

The advantage of Light GBM over XGBOOST is its processing time is very low for a high volume of data. Also, XGBOOST can automatically handle missing data, a feature that Gradient boosting lacks. But since we have the processing power and the data is not too large we can afford to choose XGBOOST for the better performance. Based on this we proceed with the next set of experiments and from here on we only use XGBOOST.

5.3 Optimizer Comparison

In this section, we perform experiments to figure out the best optimizer for our model. The optimizers in contention are NSGA-II (Non-dominated Sorting Genetic Algorithm), AGE-MOEA (Adaptive Geometry Estimated based Multi-Objective Evolutionary Algorithm), and C-TAEA (Two-Archive Evolutionary Algorithm for constrained multi-objective optimization).

NSGA-II has been an effective and popular optimization algorithm for multi-objective optimization for several years. We thus take it as a benchmark. AGE-MOEA is a newer algorithm that is promised to be faster and better at optimization. It follows the general outline of NSGA-II but then has a lot of further modifications that make it better. C-TAEA is also a relatively newer algorithm but it is based on reference directions which have to be provided while the algorithm is initialized.

The performance of each of the chosen optimization algorithms and their non-dominated solution sets are depicted below in Figures 5.4, 5.5, and 5.6 for each optimization algorithm. The performance and the convergence are on the left while the non-dominated solution sets are on the right. The selected solution set is depicted with a red 'x' mark. The ideal solution and the nadir point are also depicted with a red star and gray hexagon respectively.

The performance of each of the optimizers with their accuracy, earliness, false posi-

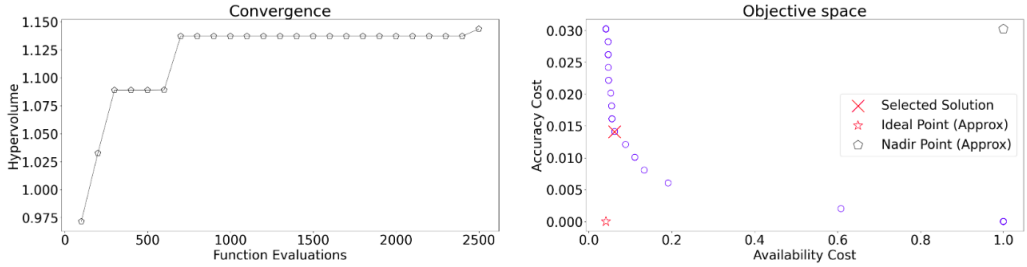

Figure 5.4: NSGA-II

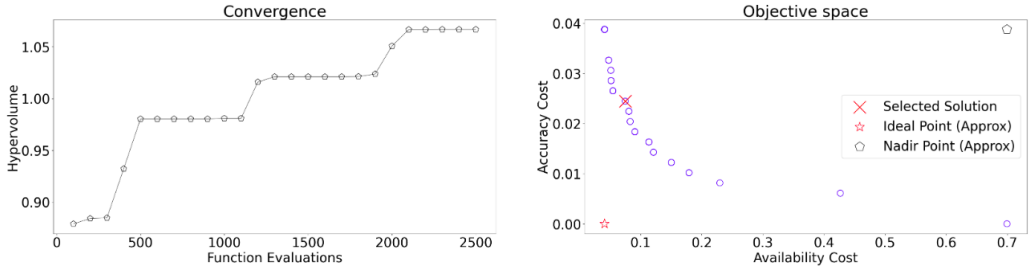

Figure 5.5: AGE-MOEA

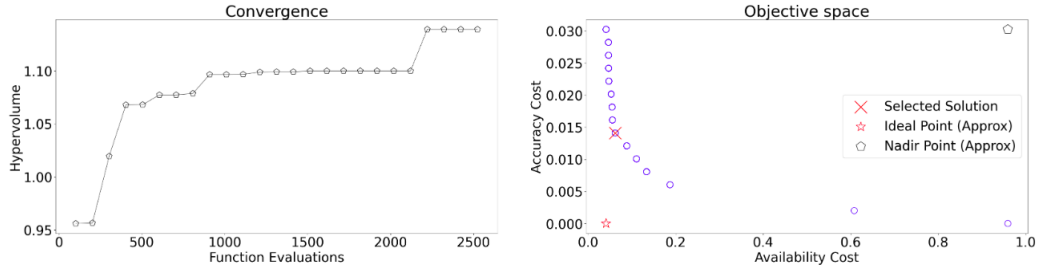

Figure 5.6: C-TAEA

tives, and late predictions are listed in Table 5.2:

Optimizer	Accuracy	Earliness	FPs	LPs
NSGA-II	97.791 %	0.130	7	28
AGE-MOEA	99.053 %	0.109	3	23
C-TAEA	98.738 %	0.133	4	28

Table 5.2: Optimizer comparison, where FP - False Positive and LP - Late Predictions

From the table, it is evident that AGE-MOEA has the best performance. It has the highest accuracy as well as the best earliness. C-TAEA also has similar performance, but since AGE-MOEA beats it by a slight margin we choose AGE-MOEA for our model and proceed with the rest of the experiments.

5.4 Trigger function Comparison

The trigger functions as described in section 4.1.2 are used to decide whether to make a decision at that given timestep or wait for more data to give a better prediction. Thus, a good trigger function is very important for the robust working of the model.

We compare and test a total of five trigger functions. The trigger function equations and their origins are clearly described in section 4.1.2. Similar to other experiments the variable aspect in this experiment is the trigger function while the other settings such as the Classifier and Optimizer are fixed. We select the XGBOOST classifier and AGE-MOEA optimizer as they performed the best in their respective tests.

The performance of the optimization algorithm using the respective trigger functions and their non-dominated solution sets are depicted below in Figures 5.7, 5.8, 5.9, 5.10, and 5.11 for each trigger function. The performance and the convergence are on the left while the non-dominated solution sets are on the right. The selected solution set is depicted with a red 'x' mark. The ideal solution and the nadir point are also depicted with a red star and gray hexagon respectively.

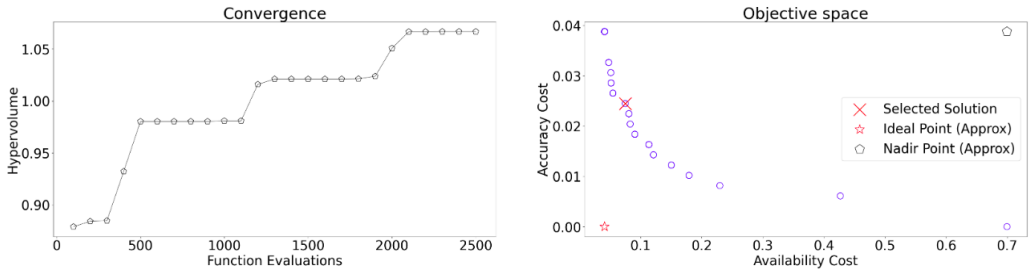

Figure 5.7: Developed Trigger Function (TR)

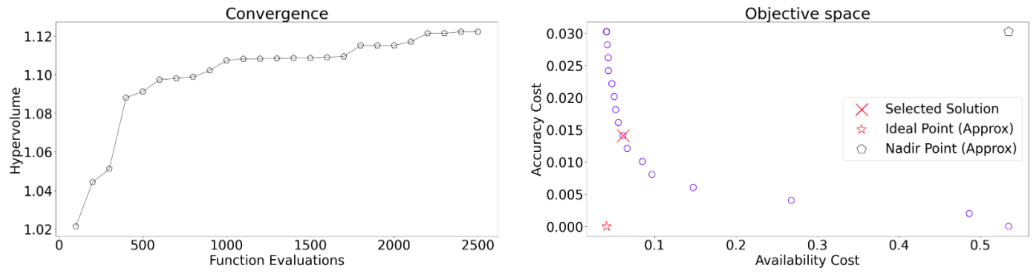

Figure 5.8: Trigger Function 1 (TR1)

Thus, the performance of the trigger functions is listed in Table 5.3.

When we look at the table we can come to the conclusion that our new trigger function has the highest accuracy while not compromising too much on earliness out of the chosen ones. The trigger function (TR1) from Mori et al (2019) [46] is the second best in accuracy but has the highest earliness value and the highest number of late predictions and is pretty close in performance to our trigger function in terms of accuracy.

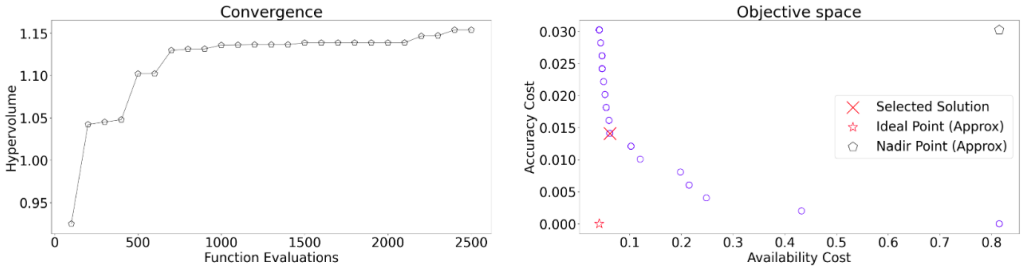

Figure 5.9: Trigger Function 2 (TR2)

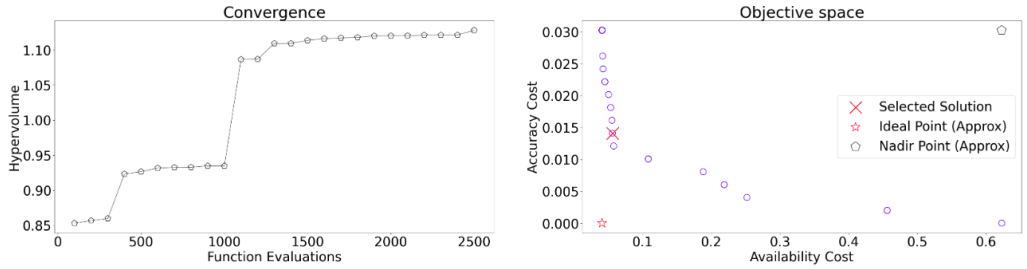

Figure 5.10: Trigger Function 3 (TR3)

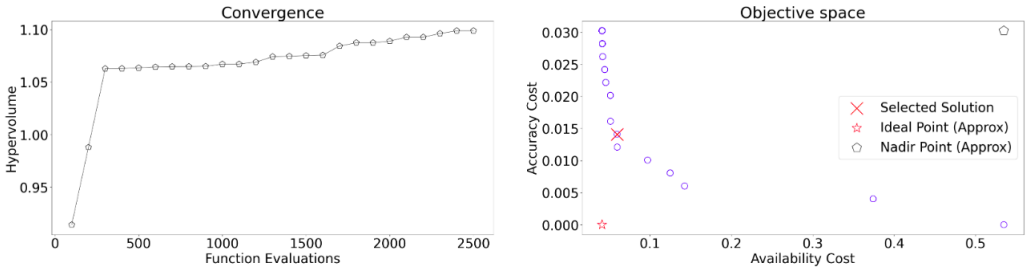

Figure 5.11: Trigger Function 4 (TR4)

Trigger function	Accuracy	Earliness	FPs	LPs
TR	99.053 %	0.109	3	23
TR1	98.107 %	0.129	6	27
TR2	97.476 %	0.088	8	22
TR3	97.476 %	0.087	8	22
TR4	96.214 %	0.091	12	26

Table 5.3: Trigger function comparison, where FP - False Positive and LP - Late Predictions

5.5 Window sizes comparison

While training the classifier we can either train on momentary values or train over an averaged window. To determine if it is better to use a window and if so to determine the window size we perform this experiment.

We test for momentary values (i.e., no window), window sizes of 50, 100, 200, 500, and entire sequence averages.

The results of the experiments are shown in Table 5.4 below:

Window size	Accuracy	Earliness	FPs	LPs
Momentary values	97.160 %	0.113	9	24
50	96.845 %	0.081	10	15
100	99.053 %	0.109	3	23
200	98.107 %	0.107	6	24
500	98.107 %	0.098	6	23
Sequence average	95.583 %	0.067	14	13

Table 5.4: Window size comparison, where FP - False Positive and LP - Late Predictions

Looking at the table, we can see that window sizes of 50, 100, 200, and 500 all perform relatively well but it is evident that a window size of 100 will be the best for our model. Thus, we choose 100 as the window size for our model.

5.6 General classifier model

An autonomous vehicle should be able to operate in different regions and adapt to various driving scenarios. A general classifier model is created to facilitate this requirement. In the further sections, country-specific models are developed and discussed and then compared with the general classifier model to see the difference in working performance. Finally, models are created using the different country data and then tested on an unseen country to evaluate the robustness of the model.

The General Classifier model is the result of all the above tests and experiments. It is the final model which was created using the best possible settings. We name it the General classifier since it is trained on data from all the available country data (as mentioned in 3.1) and can be used to predict on new data from any country. It is thus a robust model.

The settings of the General classifier model are as follows:

- Classifier: XGBOOST
- Optimizer: AGE-MOEA
- Trigger function:

$$TR_{\gamma}(\mathbf{p_t}, t) = \begin{cases} 0 & \text{if } \gamma^1(p_t^1 - p_t^2) + \gamma^2 \frac{t}{L} \leq 0 \\ 1 & \text{otherwise} \end{cases}$$

  the details of which are elucidated in section 4.1.2

- Window size: 100

The model achieved an accuracy of 99.053 % and an earliness of 0.109. There were 3 False Positives and 23 late predictions. The test set used for the model was not just from random splitting, it specifically had difficult sequences which were handpicked to challenge the model.

5.7 Country-specific Model

This experiment is to determine if a general overall model or a tailored specific country model is better on a case-by-case basis. As mentioned in section 3.1 the available countries are Sweden, France, the United States of America, Germany, Belgium, Denmark, and Luxembourg. But out of these, the number of sequences for Belgium and Luxembourg is negligible. Thus, we create country-specific models for each of the remaining five countries.

Here we create five models and each model is specific to one country. This means that for each country the train, validation, optimization, and test sets are from that country only. The test is split carefully to avoid any overlap and as we follow for the general classifier the test set contains hand-picked difficult scenarios.

The performance of the optimization algorithm using the respective country models and their non-dominated solution sets are depicted below in Figures 5.12, 5.13, 5.14, 5.15, and 5.16 for each country. The performance and the convergence are on the left while the non-dominated solution sets are on the right. The selected solution set is depicted with a red 'x' mark. The ideal solution and the nadir point are also depicted with a red star and gray hexagon respectively.

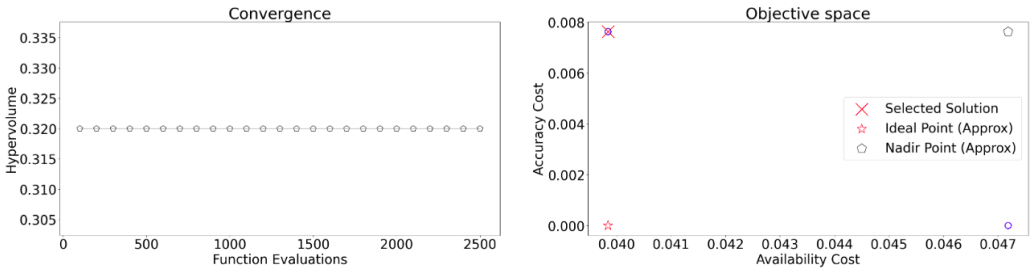

Figure 5.12: Sweden-specific model

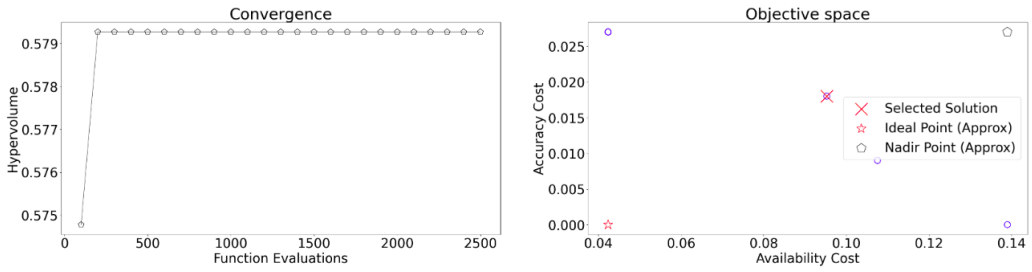

Figure 5.13: France-specific model

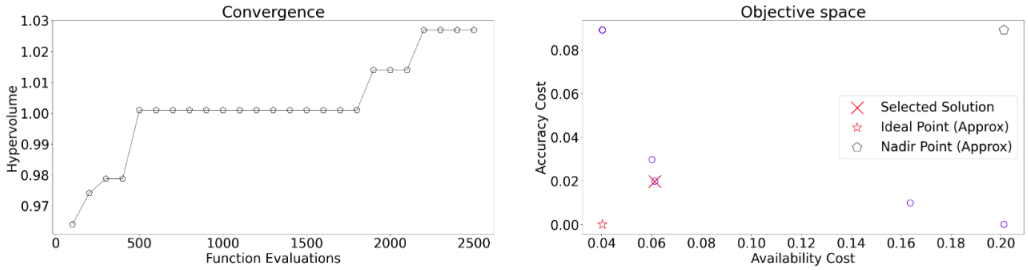

Figure 5.14: USA-specific model

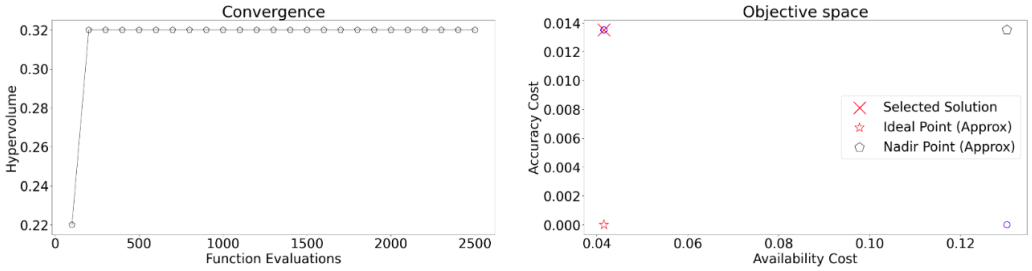

Figure 5.15: Germany-specific model

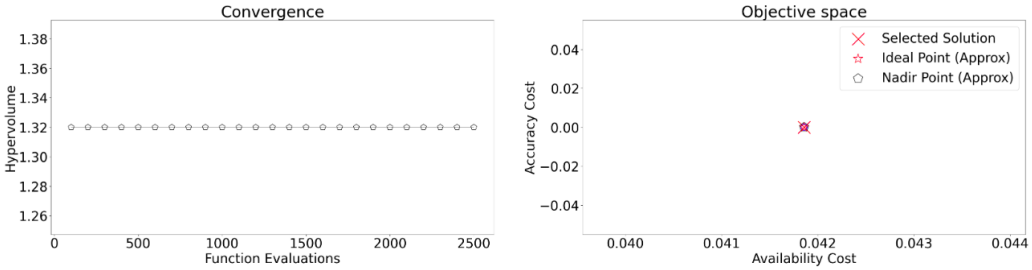

Figure 5.16: Belgium-specific model

The performance of each of the country-specific models is depicted in table 5.5.

On further analysis of the table, we can see that the models of countries: Sweden, Germany, and Belgium perform exceptionally well. While the models of France and the USA drop significantly in performance.

When we take the average of each of these country-specific models and compare these results with the general classifier model we can see that the general classifier model performs significantly better than the country-specific model. This points out that there are unique and valuable information from each of the countries and the model when trained on all the different countries can learn a lot of this valuable information and get enriched which in turn makes the model strong and robust.

5.8 Testing on an unseen country

To test the robustness and generalisability of the model, a test was conducted by creating a new model by keeping the data from a specific country entirely for the

Country	Accuracy	Earliness	FPs	LPs
Sweden	98.290 %	0.040	2	0
France	89.423 %	0.162	11	14
USA	78.481 %	0.040	17	0
Germany	100.0 %	0.041	0	0
Belgium	96.666 %	0.039	1	0

Table 5.5: Country-specific model, where FP - False Positive and LP - Late Predictions

test set. In this way, the chosen country's data is unseen to the model until test time. This was done for all the available countries (except Denmark and Luxembourg as there was very little data from these two countries).

The performance of the model for each of the unseen countries is elucidated in table 5.6.

Unseen Country	Accuracy	Earliness	FPs	LPs
Sweden	98.958 %	0.040	8	1
France	89.786 %	0.101	67	46
USA	96.385 %	0.208	21	122
Germany	99.290 %	0.043	3	1
Belgium	99.695 %	0.050	1	3

Table 5.6: Unseen country performance, where FP - False Positive and LP - Late Predictions

On analyzing the results from the table we can see that the developed model tends to perform better even in unseen regions. Two observations made from this are: In the USA we can see that the earliness is a little higher than usual. On looking at the sequences using the company visualization tool we notice that this is due to the presence of large multi-lane roads in the USA. This increases the possibility of the vehicles being in the middle of the road and thus increasing the time to come to a confident prediction. Secondly, in France, we can see more false positives than usual. This can be attributed to the presence of certain scenarios that is unique to France. One such example is the presence of long dashed lane markers which can confuse the system if not trained using it as it is otherwise not present in the data.

It can be observed that the model tested on unseen countries performs better than the country-specific model for each of the countries except Germany. This phenomenon can be attributed to the fact that in the country-specific model, the data available is very less since we only include data from that specific country which is then split between train, opt, val, and test further reducing the data available for training. This reduces the model's learning capability. Another reason could be that in the case of country-specific models, there could be difficult scenarios that

are present in the test set which were not available in training since we specifically added difficult scenarios in the test set. Whereas, the unseen country models have a relatively large training data set. This allows the model to be trained on varied scenarios which include several tricky and complex cases from different countries. This learning is then applied to the unseen country which results in a higher accuracy.

6

Conclusion

6.1 Discussion

The data used in the project is sourced from different countries. This enriches the model with valuable information and helps the model learn better. This is evident when we look at section 5.7 where we can see that the overall performance of the General classifier model is better compared to the performance of the individual country-specific classifiers.

In the data used for training the model, each sequence starts after the hypothesis is turned on. This means that the data provided was truncated at the point where the hypotheses were initialized. Using this missing information could have improved the performance of the model or could have also adversely affected the earliness.

The data used is of variable length. The sequences were specifically left like this and not altered to be of equal length because we wanted the model to be able to handle real-time scenarios where the length will be varying.

In regards to the probabilistic classifier, all of them (XGBOOST, LightGBM, and Gradient Boosting) gave a good performance but as we have shown and discussed in section 5.2 we chose XGBOOST due to its superior performance and the advantages that it offers.

The modified trigger function 4.1 used in the developed system is derived from [46]. We decided to modify it since to get a better trade-off between accuracy and earliness the terms that proved to be more important were the difference in the first and second highest probabilities and the ratio of the elapsed timestep to the total length of the sequence. The term containing the highest probability was not as significant in ensuring accuracy. The simplified version proved to be better than the Mori et al (2019) trigger function as showcased in section 5.4.

The final system had 3 false positives and 23 late predictions. On analyzing the false positive cases, we found that due to some obstruction the left camera was blocked and the left lane marker was not visible in these sequences. Only the right primary and secondary lane markers were visible. Both of these visible lane markers were dashed. Since there was a possibility for both the contending hypotheses to have dashed right lane markers, the model makes a misprediction. The model quality values of the incorrect and correct hypothesis of one such false positive case are

shown in Figures 6.1, 6.2, and 6.3. Here, we can observe that the left lane geometry and the left lane type model qualities are not available due to the obstruction while the right lane geometry and right lane type are very similar contributing to the misclassification.

While analyzing the late predictions, it was found that similar to the false positives there were a few cases where one of the cameras was blocked causing missing information. This caused the system to wait for a longer time to obtain more clarity and then take a decision leading to a late prediction. Another scenario causing late prediction is the presence of long dashed lane markers which are sometimes recognized as solid and sometimes as dashed. Additionally, sometimes the lane marker type overlaid doesn't match with the HD map for all the active hypotheses causing the model to wait longer to get a higher confidence value to make a decision. In some cases the two contending hypotheses are present in the middle lanes and have similar model qualities for the left and right lane markers, thus resulting in a late prediction.

6.2 Limitations and Future work

6.2.1 Limitations

- Even though the developed system is fast and robust, it tends to take a longer time to arrive at a decision in certain cases where there are several lanes and the two contending hypotheses are at the center lanes of the road. In these cases, since the lane geometries and lane marker types are similar, the model requires more time to make a confident prediction.
- Based on the training data, the model tends to develop a bias on the prediction time causing it to predict on similar time steps for most of the sequences. This cannot be entirely considered as a limitation since in tricky cases when there are similar model qualities between the contending hypotheses (leading to close probabilities), the model tends to wait for more timesteps to take a reliable decision.

6.2.2 Future Work

- The above limitation of late prediction on center lanes can be solved by having new features that can provide information to distinguish between the contending hypotheses.
- Instead of using the model qualities (MQs), the various sensor data measurements can be directly used to create a model. This approach can be explored to see if it performs better and solves the limitations.

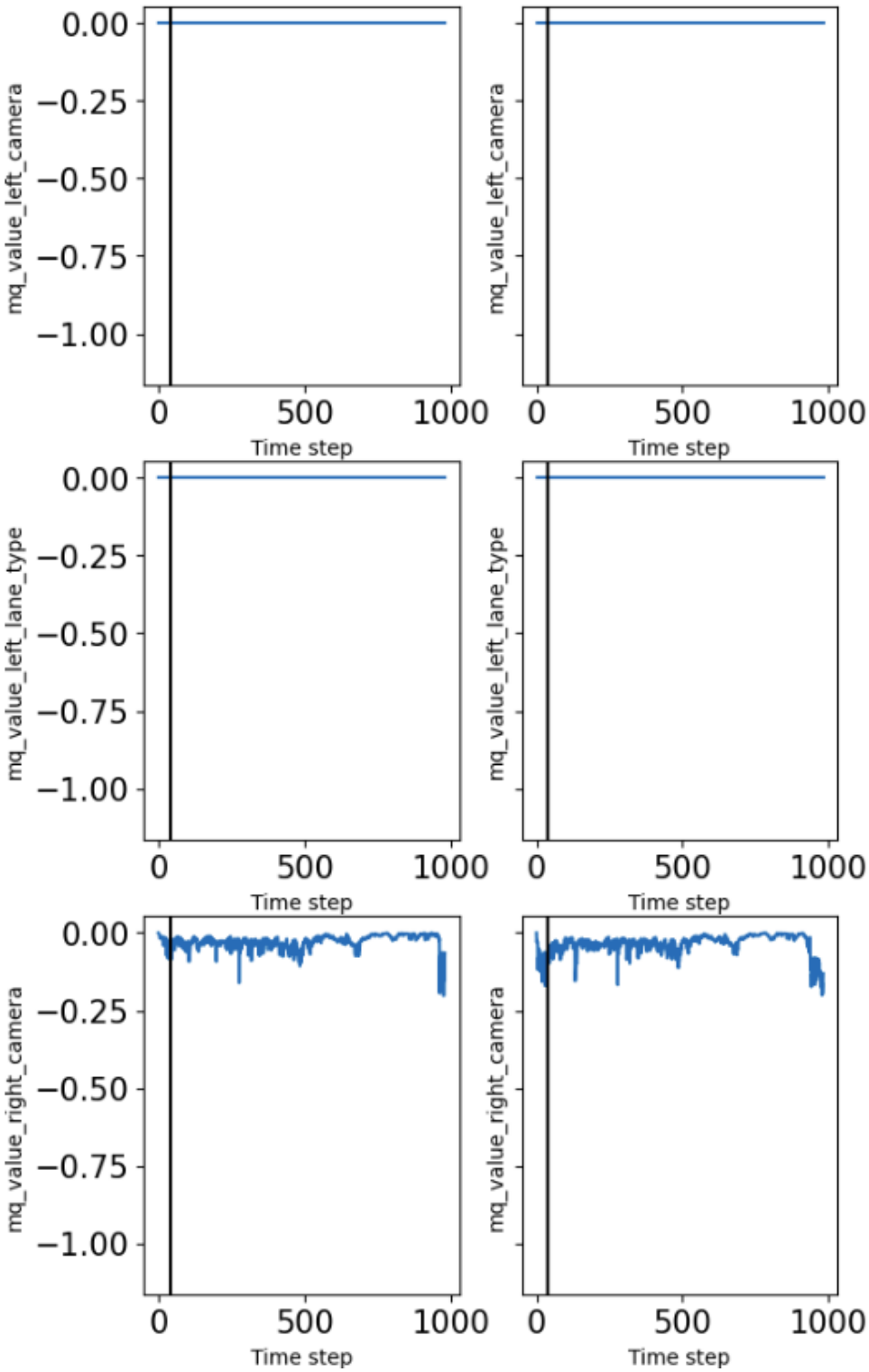

Figure 6.1: Model qualities of false positive case. Left column depicts the MQs of the incorrectly chosen hypothesis while the right column depicts the MQs of the correct hypothesis. The prediction time is depicted by a black line.

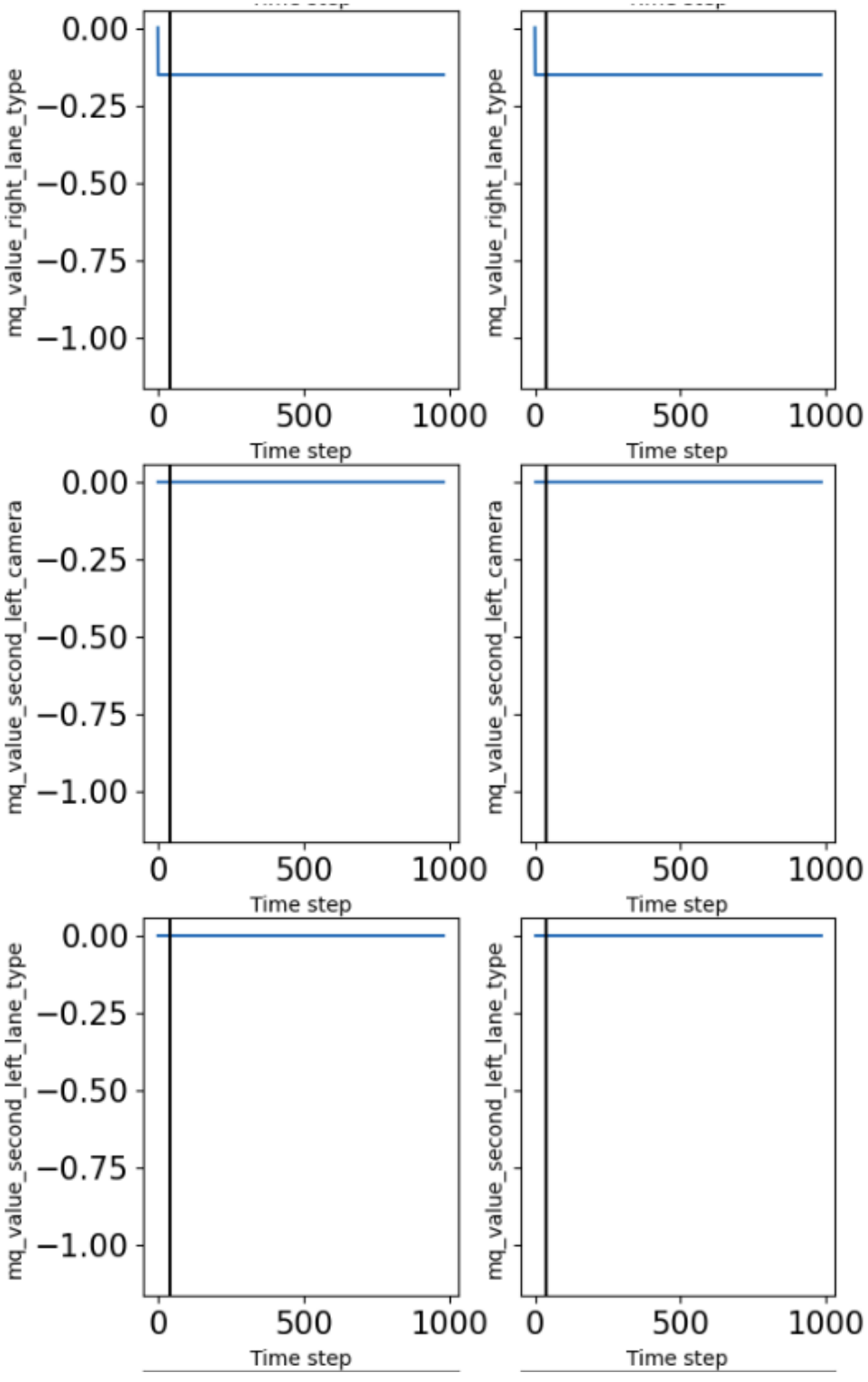

Figure 6.2: Model qualities of false positive case (cont.)

6.3 Conclusion

After performing all the experiments and analyzing the results, we conclude that the usage of a Hypothesis inference model using early time series classification for Lane

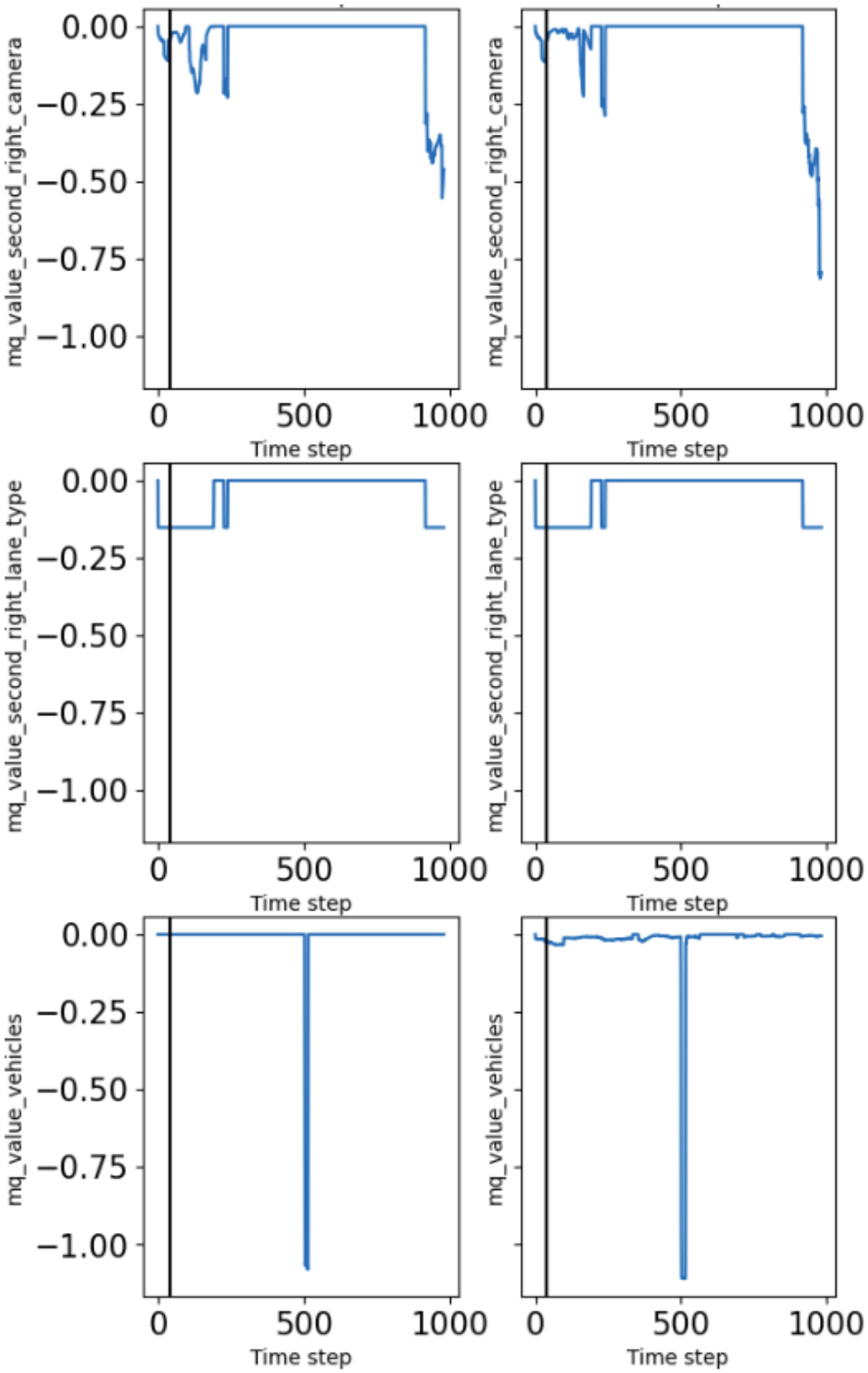

Figure 6.3: Model qualities of false positive case (cont.)

level localization is very beneficial. The developed system tends to achieve the goal of balancing accuracy and earliness while keeping the false positives minimal. The model achieves an accuracy of 99.053 % and an earliness of 0.109. The availability of

additional features and overall more data than the previous work proved to be crucial in reducing the number of false positives. The usage of a new and improved trigger function and optimizer along with a state-of-the-art classifier led to an increase in performance. The system is trained on data from different countries and generalizes well to new countries. This shows that the developed model is robust to different road scenarios.

Bibliography

[1] S. Singh and B. S. Saini, "Autonomous cars: Recent developments, challenges, and possible solutions," *IOP Conference Series: Materials Science and Engineering*, vol. 1022, no. 1, p. 012 028, Jan. 2021. DOI: 10.1088/1757-899X/1022/1/012028. [Online]. Available: https://dx.doi.org/10.1088/1757-899X/1022/1/012028.

[2] X. D. Scott Drew Pendleton Hans Andersen and M. H. Ang, "Perception, Planning, Control, and Coordination for Autonomous Vehicles," MDPI, 2017. [Online]. Available: https://doi.org/10.3390/machines5010006.

[3] O. Tengilimoglu, O. Carsten, and Z. Wadud, "Implications of automated vehicles for physical road environment: A comprehensive review," *Transportation Research Part E: Logistics and Transportation Review*, vol. 169, p. 102 989, 2023, ISSN: 1366-5545. DOI: https://doi.org/10.1016/j.tre.2022.102989. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S1366554522003660.

[4] S. Malik, M. A. Khan, H. El-Sayed, M. J. Khan, and O. Ullah, "How do autonomous vehicles decide?" *Sensors (Basel, Switzerland)*, vol. 23, 2022.

[5] S. D. Pendleton, H. Andersen, X. Du, *et al.*, "Perception, planning, control, and coordination for autonomous vehicles," *Machines*, vol. 5, no. 1, 2017, ISSN: 2075-1702. DOI: 10.3390/machines5010006. [Online]. Available: https://www.mdpi.com/2075-1702/5/1/6.

[6] M. N. D. Somphop Limsoonthrakul and M. Parnichkun, "Intelligent Vehicle Localization Using GPS, Compass, and Machine Vision," Klong Luang, Pathumthani, Thailand: IEEE, 2009, ISBN: 9781424438037. DOI: 10.1109/IROS.2009.5354042. [Online]. Available: https://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=5354042.

[7] J. Laconte, A. Kasmi, R. Aufrère, M. Vaidis, and R. Chapuis, "A survey of localization methods for autonomous vehicles in highway scenarios," *Sensors*, vol. 22, p. 247, Dec. 2021. DOI: 10.3390/s22010247.

[8] K. Kim, S. Cho, and W. Chung, "HD map update for autonomous driving with crowdsourced data," *IEEE Robotics and Automation Letters*, vol. 6, no. 2, pp. 1895–1901, Apr. 2021. DOI: 10.1109/lra.2021.3060406. [Online]. Available: https://doi.org/10.1109/lra.2021.3060406.

[9] K. Arras, J. Castellanos, and R. Siegwart, "Feature-based multi-hypothesis localization and tracking for mobile robots using geometric constraints.," vol. 2, Jan. 2002, pp. 1371–1377. DOI: 10.1109/ROBOT.2002.1014734.

[10] K. Arras, J. B, M. Schilff, C. Superior, and M. Luna, "Towards feature-based multi-hypothesis localization and tracking," Dec. 2001.
[11] *Zenseact - Towards zero faster*, https://zenseact.com/.
[12] *Machine Learning Meets Localization*, https://hdl.handle.net/20.500.12380/304888.
[13] S. Han and J. Wang, "Integrated gps/ins navigation system with dual-rate kalman filter," *GPS Solutions*, vol. 16, Jul. 2012. DOI: 10.1007/s10291-011-0240-x.
[14] H. Han, J. Wang, J. Wang, and A. Moraleda, "Reliable partial ambiguity resolution for single-frequency gps/bds and ins integration," *GPS Solutions*, vol. 21, Jan. 2017. DOI: 10.1007/s10291-016-0519-z.
[15] C. Qian, H. Liu, J. Tang, *et al.*, "An integrated gnss/ins/lidar-slam positioning method for highly accurate forest stem mapping," *Remote Sensing*, vol. 9, no. 1, 2017, ISSN: 2072-4292. DOI: 10.3390/rs9010003. [Online]. Available: https://www.mdpi.com/2072-4292/9/1/3.
[16] D. Shin, K.-m. Park, and M. Park, "High definition map-based localization using adas environment sensors for application to automated driving vehicles," *Applied Sciences*, vol. 10, no. 14, 2020, ISSN: 2076-3417. DOI: 10.3390/app10144924. [Online]. Available: https://www.mdpi.com/2076-3417/10/14/4924.
[17] H. Wang, C. Xue, Y. Zhou, F. Wen, and H. Zhang, "Visual semantic localization based on hd map for autonomous vehicles in urban scenarios," in *2021 IEEE International Conference on Robotics and Automation (ICRA)*, 2021, pp. 11 255–11 261. DOI: 10.1109/ICRA48506.2021.9561459.
[18] S. Zheng and J. Wang, "High definition map-based vehicle localization for highly automated driving: Geometric analysis," in *2017 International Conference on Localization and GNSS (ICL-GNSS)*, 2017, pp. 1–8. DOI: 10.1109/ICL-GNSS.2017.8376252.
[19] J. K. Suhr, J. Jang, D. Min, and H. G. Jung, "Sensor fusion-based low-cost vehicle localization system for complex urban environments," *IEEE Transactions on Intelligent Transportation Systems*, vol. 18, no. 5, pp. 1078–1086, 2017. DOI: 10.1109/TITS.2016.2595618.
[20] *Understanding localization in self-driving cars and its technology*, https://medium.com/self-driving-cars/understanding-localization-in-self-driving-cars-and-its-technology-ad3b96caa466.
[21] R. L. José-Luis Rullán-Lara Sergio Salazar, "Real-time localization of an uav using kalman filter and a wireless sensor network," *Journal of Intelligent Robotic Systems*, 2012. DOI: https://doi.org/10.1007/s10846-011-9599-8.
[22] R. E. Kálmán, "A new approach to linear filtering and prediction problems" transaction of the asme journal of basic," 1960.
[23] *An introduction to Kalman Filter*, https://towardsdatascience.com/an-intro-to-kalman-filters-for-autonomous-vehicles-f43dd2e2004b.
[24] P. Misra and Siddharth, "Machine learning and time series: Real world applications," in *2017 International Conference on Computing, Communication and Automation (ICCCA)*, 2017, pp. 389–394. DOI: 10.1109/CCAA.2017.8229832.

[25] A. Bagnall, J. Lines, A. Bostrom, J. Large, and E. Keogh, "The great time series classification bake off: A review and experimental evaluation of recent algorithmic advances," *Data Mining and Knowledge Discovery*, vol. 31, no. 3, pp. 606–660, May 2017, ISSN: 1573-756X. DOI: 10.1007/s10618-016-0483-9. [Online]. Available: https://doi.org/10.1007/s10618-016-0483-9.

[26] Y. Tong, J. Liu, L. Yu, *et al.*, "Technology investigation on time series classification and prediction," en, *PeerJ Comput Sci*, vol. 8, e982, May 2022.

[27] H. Ding, G. Trajcevski, P. Scheuermann, X. Wang, and E. Keogh, "Querying and mining of time series data: Experimental comparison of representations and distance measures," *Proc. VLDB Endow.*, vol. 1, no. 2, pp. 1542–1552, Aug. 2008, ISSN: 2150-8097. DOI: 10.14778/1454159.1454226. [Online]. Available: https://doi.org/10.14778/1454159.1454226.

[28] X. Xi, E. Keogh, C. Shelton, L. Wei, and C. A. Ratanamahatana, "Fast time series classification using numerosity reduction," in *Proceedings of the 23rd International Conference on Machine Learning*, ser. ICML '06, Pittsburgh, Pennsylvania, USA: Association for Computing Machinery, 2006, pp. 1033–1040, ISBN: 1595933832. DOI: 10.1145/1143844.1143974. [Online]. Available: https://doi.org/10.1145/1143844.1143974.

[29] L. Ye and E. Keogh, "Time series shapelets: A new primitive for data mining," in *Proceedings of the 15th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, ser. KDD '09, Paris, France: Association for Computing Machinery, 2009, pp. 947–956, ISBN: 9781605584959. DOI: 10.1145/1557019.1557122. [Online]. Available: https://doi.org/10.1145/1557019.1557122.

[30] T. Rakthanmanon and E. Keogh, "Fast shapelets: A scalable algorithm for discovering time series shapelets," in *Proceedings of the 2013 SIAM International Conference on Data Mining (SDM)*, pp. 668–676. DOI: 10.1137/1.9781611972832.74. eprint: https://epubs.siam.org/doi/pdf/10.1137/1.9781611972832.74. [Online]. Available: https://epubs.siam.org/doi/abs/10.1137/1.9781611972832.74.

[31] H. A. Dau, D. F. Silva, F. Petitjean, *et al.*, "Optimizing dynamic time warping's window width for time series data mining applications," *Data Mining and Knowledge Discovery*, vol. 32, no. 4, pp. 1074–1120, Jul. 2018, ISSN: 1573-756X. DOI: 10.1007/s10618-018-0565-y. [Online]. Available: https://doi.org/10.1007/s10618-018-0565-y.

[32] A. Dempster, F. Petitjean, and G. I. Webb, "ROCKET: exceptionally fast and accurate time series classification using random convolutional kernels," *CoRR*, vol. abs/1910.13051, 2019. arXiv: 1910.13051. [Online]. Available: http://arxiv.org/abs/1910.13051.

[33] L. Wei and E. Keogh, "Semi-supervised time series classification," in *Proceedings of the 12th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, ser. KDD '06, Philadelphia, PA, USA: Association for Computing Machinery, 2006, pp. 748–753, ISBN: 1595933395. DOI: 10.1145/1150402.1150498. [Online]. Available: https://doi.org/10.1145/1150402.1150498.

[34] Y. Chen, B. Hu, E. Keogh, and G. E. Batista, "Dtw-d: Time series semi-supervised learning from a single example," in *Proceedings of the 19th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, ser. KDD '13, Chicago, Illinois, USA: Association for Computing Machinery, 2013, pp. 383–391, ISBN: 9781450321747. DOI: 10.1145/2487575.2487633. [Online]. Available: https://doi.org/10.1145/2487575.2487633.

[35] K. Marussy and K. Buza, "Success: A new approach for semi-supervised classification of time-series," in *Artificial Intelligence and Soft Computing*, L. Rutkowski, M. Korytkowski, R. Scherer, R. Tadeusiewicz, L. A. Zadeh, and J. M. Zurada, Eds., Berlin, Heidelberg: Springer Berlin Heidelberg, 2013, pp. 437–447, ISBN: 978-3-642-38658-9.

[36] N. Begum, B. Hu, T. Rakthanmanon, and E. Keogh, "Towards a minimum description length based stopping criterion for semi-supervised time series classification," in *2013 IEEE 14th International Conference on Information Reuse Integration (IRI)*, 2013, pp. 333–340. DOI: 10.1109/IRI.2013.6642490.

[37] A. Gupta, H. P. Gupta, B. Biswas, and T. Dutta, "Approaches and applications of early classification of time series: A review," *CoRR*, vol. abs/2005.02595, 2020. arXiv: 2005.02595. [Online]. Available: https://arxiv.org/abs/2005.02595.

[38] J. J. Rodríguez, C. J. Alonso, and H. Boström, "Boosting interval based literals," *Intelligent Data Analysis*, vol. 5, pp. 245–262, 2001, 3, ISSN: 1571-4128. DOI: 10.3233/IDA-2001-5305. [Online]. Available: https://doi.org/10.3233/IDA-2001-5305.

[39] P. S. Zhengzheng Xing1 Jian Pei, "Early classification on time series," *Under consideration for publication in Knowledge and Information*,

[40] A. Bregon, M. Hurtado, J. Rodríguez, C. Alonso, B. Pulido, and Q. Moro-Sancho, "Early fault classification in dynamic systems using case-based reasoning," vol. 4177, Nov. 2005, pp. 211–220, ISBN: 978-3-540-45914-9. DOI: 10.1007/11881216_23.

[41] Z. Xing, J. Pei, and S. Y. Philip, "Early prediction on time series: A nearest neighbor approach.," in *IJCAI*, Citeseer, 2009, pp. 1297–1302.

[42] Z. z. Xing, J. Pei, P. Yu, and K. Wang, "Extracting interpretable features for early classification on time series," Apr. 2011, pp. 247–258. DOI: 10.1137/1.9781611972818.22.

[43] M. F. Ghalwash and Z. Obradovic, "Early classification of multivariate temporal observations by extraction of interpretable shapelets," *BMC Bioinformatics*, vol. 13, no. 1, p. 195, Aug. 2012, ISSN: 1471-2105. DOI: 10.1186/1471-2105-13-195. [Online]. Available: https://doi.org/10.1186/1471-2105-13-195.

[44] G. He, Y. Duan, R. Peng, X. Jing, T. Qian, and L. Wang, "Early classification on multivariate time series," *Neurocomputing*, vol. 149, pp. 777–787, 2015, ISSN: 0925-2312. DOI: https://doi.org/10.1016/j.neucom.2014.07.056. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S092523121401008X.

[45] U. Mori, A. Mendiburu, E. Keogh, and J. A. Lozano, "Reliable early classification of time series based on discriminating the classes over time," *Data*

Mining and Knowledge Discovery, vol. 31, no. 1, pp. 233–263, Jan. 2017, ISSN: 1573-756X. DOI: 10.1007/s10618-016-0462-1. [Online]. Available: https://doi.org/10.1007/s10618-016-0462-1.

[46] U. Mori, A. Mendiburu, I. Miranda, and J. Lozano, "Early classification of time series using multi-objective optimization techniques," *Information Sciences*, vol. 492, pp. 204–218, 2019, ISSN: 0020-0255. DOI: https://doi.org/10.1016/j.ins.2019.04.024. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0020025519303317.

[47] P. Schäfer and U. Leser, "Teaser: Early and accurate time series classification," *Data Mining and Knowledge Discovery*, vol. 34, no. 5, pp. 1336–1362, Sep. 2020, ISSN: 1573-756X. DOI: 10.1007/s10618-020-00690-z. [Online]. Available: https://doi.org/10.1007/s10618-020-00690-z.

[48] *XGBoost*, https://xgboost.readthedocs.io/en/stable/.

[49] *XGBoost : Everything you need to know*, https://neptune.ai/blog/xgboost-everything-you-need-to-know.

[50] *How XGBoost algorithm works*, https://pro.arcgis.com/en/pro-app/latest/tool-reference/geoai/how-xgboost-works.htm.

[51] S. Sharma and V. Kumar, "A comprehensive review on multi-objective optimization techniques: Past, present and future," *Archives of Computational Methods in Engineering*, vol. 29, no. 7, pp. 5605–5633, Nov. 2022, ISSN: 1886-1784. DOI: 10.1007/s11831-022-09778-9. [Online]. Available: https://doi.org/10.1007/s11831-022-09778-9.

[52] N. Zolpakar, S. singh lodhi, S. Pathak, and M. Sharma, "Application of multi-objective genetic algorithm (moga) optimization in machining processes," in Jun. 2019, pp. 1–15, ISBN: 978-3-030-19637-0. DOI: 10.1007/978-3-030-19638-7_8.

[53] K. Deb, A. Pratap, S. Agarwal, and T. Meyarivan, "A fast and elitist multi-objective genetic algorithm: Nsga-ii," *IEEE transactions on evolutionary computation*, vol. 6, no. 2, pp. 182–197, 2002.

[54] I. Yevseyeva, A. Guerreiro, M. Emmerich, and C. Fonseca, "A portfolio optimization approach to selection in multiobjective evolutionary algorithms," Sep. 2014, pp. 672–681, ISBN: 978-3-319-10761-5. DOI: 10.1007/978-3-319-10762-2_66.

[55] J. Bader and E. Zitzler, "Hype: An algorithm for fast hypervolume-based many-objective optimization," *Evolutionary computation*, vol. 19, no. 1, pp. 45–76, 2011.

[56] Q. Zhang and H. Li, "Moea/d: A multiobjective evolutionary algorithm based on decomposition," *IEEE Transactions on evolutionary computation*, vol. 11, no. 6, pp. 712–731, 2007.

[57] K. Deb and H. Jain, "An evolutionary many-objective optimization algorithm using reference-point-based nondominated sorting approach, part i: Solving problems with box constraints," *IEEE transactions on evolutionary computation*, vol. 18, no. 4, pp. 577–601, 2013.

[58] A. Panichella, "An adaptive evolutionary algorithm based on non-euclidean geometry for many-objective optimization," in *Proceedings of the Genetic and Evolutionary Computation Conference*, ser. GECCO '19, Prague, Czech Re-

public: Association for Computing Machinery, 2019, pp. 595–603, ISBN: 9781450361118. DOI: 10.1145/3321707.3321839. [Online]. Available: https://doi.org/10.1145/3321707.3321839.

[59] *Key:highway*, https://wiki.openstreetmap.org/wiki/Key:highway.

[60] *pymoo - Decomposition*, https://pymoo.org/misc/decomposition.html.

[61] A. P. Wierzbicki, "The use of reference objectives in multiobjective optimization," in *Multiple Criteria Decision Making Theory and Application*, G. Fandel and T. Gal, Eds., Berlin, Heidelberg: Springer Berlin Heidelberg, 1980, pp. 468–486, ISBN: 978-3-642-48782-8.

[62] V. Borisov, T. Leemann, K. SeSSler, J. Haug, M. Pawelczyk, and G. Kasneci, "Deep neural networks and tabular data: A survey," 2021. DOI: 10.48550/ARXIV.2110.01889. [Online]. Available: https://arxiv.org/abs/2110.01889.

[63] R. Shwartz-Ziv and A. Armon, *Tabular data: Deep learning is not all you need*, 2021. DOI: 10.48550/ARXIV.2106.03253. [Online]. Available: https://arxiv.org/abs/2106.03253.

[64] F. Pedregosa, G. Varoquaux, A. Gramfort, *et al.*, "Scikit-learn: Machine learning in Python," *Journal of Machine Learning Research*, vol. 12, pp. 2825–2830, 2011.

[65] *LightGBM*, https://lightgbm.readthedocs.io/en/v3.3.5/.