UNIVERSIDAD DE INVESTIGACIÓN DE TECNOLOGÍA EXPERIMENTAL YACHAY

Escuela de Ciencias Matemáticas y Computacionales

TÍTULO: Activity pattern generation using machine learning techniques for transport demand models

Trabajo de integración curricular presentado como requisito para la obtención del título de Ingeniero en Tecnologías de la Información

Autor:

Ajala Ramos Santiago Alexander

Tutor:

Ph.D. - Armas Andrade Tito Rolando

Co-Tutor:

Ph.D. - Morales Navarrete Diego Fabian

Urcuquí, Noviembre - 2024

Autoría

Yo, **SANTIAGO ALEXANDER AJALA RAMOS**, con cédula de identidad 1003818497, declaro que las ideas, juicios, valoraciones, interpretaciones, consultas bibliográficas, definiciones y conceptualizaciones expuestas en el presente trabajo; así cómo, los procedimientos y herramientas utilizadas en la investigación, son de absoluta responsabilidad de el/la autor/a del trabajo de integración curricular. Así mismo, me acojo a los reglamentos internos de la Universidad de Investigación de Tecnología Experimental Yachay.

Urcuquí, Noviembre - 2024.

Santiago Alexander Ajala Ramos
CI: 1003818497

Autorización de publicación

Yo, **SANTIAGO ALEXANDER AJALA RAMOS**, con cédula de identidad 1003818497, cedo a la Universidad de Investigación de Tecnología Experimental Yachay, los derechos de publicación de la presente obra, sin que deba haber un reconocimiento económico por este concepto. Declaro además que el texto del presente trabajo de titulación no podrá ser cedido a ninguna empresa editorial para su publicación u otros fines, sin contar previamente con la autorización escrita de la Universidad.

Asimismo, autorizo a la Universidad que realice la digitalización y publicación de este trabajo de integración curricular en el repositorio virtual, de conformidad a lo dispuesto en el Art. 144 de la Ley Orgánica de Educación

Urcuquí, Noviembre - 2024.

Firmado electrónicamente por:
SANTIAGO ALEXANDER AJALA RAMOS

Santiago Alexander Ajala Ramos
CI: 1003818497

Dedication

I dedicate this work to all of my wonderful family members who have always looked after me, particularly my mother Angelita, who has inspired me since childhood. Without a doubt, she never abandoned me, even during difficult times, and despite her lack of knowledge of my thesis topic, she always encouraged me to persevere. I would also like to dedicate this work to my brother Wilson, who has always been there for me and never left me alone, despite our distance. Lastly, I dedicate this work to all those who provided their support in any way throughout this work.

Santiago Ajala Ramos

Acknowledgment

First and foremost, I want to thank God for providing me the life and knowledge to undertake this work.

I want to thank my wonderful advisor, PhD Rolando Armas, for his hard efforts to help me complete this task. I would also like to thank my co-advisor, PhD. Diego Morales, for contributing his knowledge to this work.

I would want to thank my university classmates for all the pleasure we had together. Especially my friend Saire Conejo, whom I met on the first day of college. We shared not only courses but also activities in our residence halls, which were filled with fun. Thank you very much Saire for all the shared moments.

Finally, I would like to thank Llactalab - Universidad de Cuenca who provided me with the dataset for this work.

Santiago Ajala Ramos

Resumen

La generación de patrones de actividad es un componente esencial en los sistemas de modelización de la demanda basados en la actividad, fundamentales para la planificación urbana y la gestión eficiente de los sistemas de transporte. Tradicionalmente, esta generación se ha realizado mediante técnicas convencionales; sin embargo, con el avance de la tecnología, las técnicas de aprendizaje automático se han utilizado cada vez más para tareas como la elección del medio de transporte y la predicción del flujo de tráfico. Este estudio se enfoca en la aplicación de técnicas de aprendizaje automático para la generación de patrones de actividad para modelos de demanda de transporte. Se realizarán cuatro tareas de clasificación: medio de transporte, motivo de desplazamiento, destino y hora de inicio del desplazamiento. Para dichas tareas, se cuenta con un dataset de la ciudad de Cuenca perteneciente a Ecuador de aproximadamente 3000 registros, el cual tuvo que ser preparado, filtrado y ajustado. Se utilizaron tres modelos de aprendizaje automático: Bosques Aleatorios, Árboles de Decisión y Redes Neuronales Artificiales, y se analizaron sus resultados para escoger el mejor para cada tarea de clasificación. Los resultados muestran que el mejor modelo para las cuatro tareas de clasificación fue el Bosque Aleatorio, en conjunto con la técnica de ajuste de hiperparámetros Búsqueda en Cuadrícula con Validación Cruzada. Finalmente, para dar utilidad a los mejores modelos, se construyó un dataset para una simulación utilizando la herramienta MATSim, logrando resultados favorables.

Palabras Clave:
Patrones de actividad, Aprendizaje automático, Árboles de decisión, Bosques aleatorios, Redes neuronales artificiales.

Abstract

Activity pattern generation is an essential component in activity-based demand modeling systems, which are fundamental for urban planning and efficient management of transportation systems. Traditionally, this generation has been performed using conventional techniques; however, with the advancement of technology, machine learning techniques have been increasingly used for tasks such as transportation mode choice and traffic flow prediction. This study focuses on the application of machine learning techniques for the generation of activity patterns for transportation demand models. Four classification tasks will be performed: means of transportation, reason for travel, destination and start time of travel. For these tasks, a dataset of the city of Cuenca belonging to Ecuador of approximately 3000 records had to be prepared, filtered and adjusted. Three machine learning models were used: Random Forest, Decision Tree and Artificial Neural Networks, and their results were analyzed to choose the best one for each classification task. The results show that the best model for the four classification tasks was the Random Forest, in conjunction with the Grid Search Cross Validation hyperparameter fitting technique. Finally, to give utility to the best models, a dataset was built for a simulation using the MATSim tool, achieving favorable results.

Keywords:
Activity pattern, Machine learning, Decision tree, Random forest, Artificial neural network

Contents

List of Tables

List of Figures

Chapter 1

Introduction

1.1 Background

Effective urban planning and transportation management require an understanding of human mobility patterns. Over the years, researchers have used a variety of approaches to analyze and model travel behavior, from conventional survey-based methods to more sophisticated data-driven methods. With the advancement of technology and the advent of machine learning (ML), new ways have been implemented that facilitate the identification of complex patterns of human activity and mobility [3].

Individual preferences, socioeconomic factors, and the urban environment affect the dynamics of human mobility over space and time [4]. Conventional transportation demand models, such as the four-step model, are generally based on aggregate data and assumptions. This makes them difficult to identify the heterogeneity and dynamics of travel behavior [4]. As a result, the use of machine learning (ML) techniques to improve the granularity and accuracy of these models is gaining popularity.

ML algorithms that can evaluate massive volumes of data and uncover underlying patterns include Decision Trees (DT), Random Forests (RF), and Neural Networks (NN) [5]. By building predictive models using a range of datasets covering human traits, travel qualities, and contextual elements, researchers can develop models that can estimate trip demand and comprehend activity patterns with more precise spatial and temporal precision [6].

According to recent research, ML-based techniques have proven effective in several transportation planning applications. For instance, travel mode choice analysis is crucial for understanding and forecasting travel demand in transportation planning and policy-making [7]. Because machine learning (ML)-based techniques like Random Forest have reduced processing costs and higher accuracy, they have demonstrated noticeably better performance in travel mode choice prediction [7].

Even with these advancements, implementing ML approaches into traditional transportation planning frameworks remains challenging. To ensure the moral and practical use of ML-based solutions in real-world scenarios, aspects such as data privacy, model interpretability, and scalability must be carefully considered [8]. By addressing these difficulties and leveraging machine learning skills, researchers can increase our understanding of hu-

man mobility patterns and contribute to the development of more efficient and sustainable urban transportation systems.

1.2 Problem statement

Urban planning and policy formulation greatly benefit from transportation demand models, as they guide decisions related to infrastructure development, traffic management, and environmental sustainability. However, the availability and quality of data pose a fundamental challenge in this field. These models rely on a wide range of data, including population and employment projections, land use patterns, travel surveys, traffic counts, public transit schedules, and network characteristics [9].

Furthermore, current models struggle to adapt to the dynamic nature of human behavior, particularly in response to shifts in socioeconomic conditions, technological advancements, and urbanization trends. Consequently, decision-makers are compelled to address transportation shortcomings and mitigate external negative consequences, such as congestion and pollution, without sufficient understanding of the underlying factors influencing travel behavior.

Machine learning techniques (ML) hold promise in enhancing the predictive capabilities of transportation demand models by leveraging large-scale datasets and advanced algorithms to uncover hidden patterns and relationships [10]. One of the primary challenges lies in developing ML models capable of accurately predicting individual activity patterns, including the timing, mode choice, purpose, and destination of trips.

1.3 Objectives

1.3.1 General Objective

This research introduces a methodology for training and evaluating machine learning techniques aimed at generating activity patterns through classification tasks suitable for integration into transportation demand models.

1.3.2 Specific Objectives

- Filter, prepare, and adjust the dataset to feed the different machine-learning models.
- Train and evaluate classification machine learning models such as Neural Networks, Random Forests, and Decision Trees for the classification of the mode of transportation, motive of travel, start time of displacements, and destination.
- Compare models in classification tasks and choose the best ones.
- Validate the best models by building a dataset to perform a transport simulation using a framework.

Chapter 2

Theoretical Framework

This chapter addresses the ideas required to comprehend this project, including an introduction and a detailed overview of the fundamentals of transportation demand models and activity-based travel demand models. Machine learning models, artificial neural networks, Random Forest classification, and Decision Tree classification are all discussed, as well as their structure and operation. Finally, the most common measures for evaluating the performance of machine learning models are discussed.

2.1 Transport Demand Models

Travel demand arises from thousands of individual travelers choosing their own routes, destinations, and times of departure [11]. Transportation models have been useful in projecting transportation demand and assessing the impact of plans and regulations [12]. They are essential tools in transportation planning and engineering, forecasting and assessing the demand for transportation services and infrastructure. These models leverage various data sources, methodologies, and assumptions to simulate the movement of people and goods within a transportation network. Planners use transportation models to learn about the behavior of transportation systems [12].

2.1.1 Basic Concepts

Transport demand models are designed to estimate the demand for transportation services based on various factors such as population demographics, economic conditions, land use patterns, and transportation infrastructure. Key concepts in transportation demand modeling are the following [11]:

- **Trip Generation:** Estimating the number of trips to and from specific zones or locations.
- **Trip Distribution:s** Determining the destinations of trips as well as the routes traveled between them.
- **Modal Split:** Allocating trips among different modes of transportation (e.g., car, public transit, walking, cycling).

- **Traffic Assignment:** Allocating trips to specific routes and estimating traffic flows on the transportation network.

2.1.2 Types of Models

There are many different kinds of transport demand models, from conventional trip-based models to sophisticated activity-based and agent-based models, each with its own approaches and uses [13]. A thorough examination of transportation networks is made possible by the variety of modeling techniques, which support various scales and levels of information. Comprehending the distinct categories of transport demand models is important in order to proficiently tackle the intricate predicaments of contemporary transportation planning [14]. Among the different types of models that exist, the following stand out:

- **Trip-based Model:** Also known as the four-step model, is the most widely used technique for modeling travel demand [15]. These models forecast travel patterns first, then disperse those patterns throughout an area, calculate the mode of transportation, and lastly allocate those patterns to the transportation network [16].

- **Activity-based Model:** These models concentrate on individuals' daily activities and travel behavior rather than just trips. [16]. These models, which emphasizes involvement in activities and focuses on sequences or patterns of behavior, can solve congestion management issues by looking at how people adjust their engagement in activities [17]. Since they do not have some of the drawbacks that come with four-step models, activity-based models are often hailed as being better than them [18].

- **Agent-base Model:** These models incorporate the agent's independent decision-making in addition to modeling and replicating mechanical movement [19]. Agent-based models imitate the actions and interactions of autonomous agents (individuals or groups) in order to determine their impact on the transportation system. They are effective in capturing complicated dynamics and emergent events [20].

2.2 Activity-based travel demand models

Because activity-based travel demand models (ABMs) explicitly focus on how people organize their activities in time and place, they are being utilized more and more to assist transport planners [21]. The goal of these demand models is to explain how people organize and arrange their daily activities, which have an impact on travel demand [22]. By capturing the daily activity patterns of individuals, ABMs provide a more detailed and realistic representation of travel behavior, leading to more accurate and reliable forecasts for transportation planning. Modules for activity generation, activity location choice, and mode choice are typically included in activity-based models [23].

2.2.1 Activity Generation Module

In this module, participants must forecast the tasks they will complete during the day by providing the what, when, and how long of each task. Activities may include job, school, shopping, leisure, and other personal or household tasks. [17].

2.2.2 Activity Location Choice Module

This module determines the locations where the generated activities will take place. It involves the spatial distribution of activities and the selection of specific destinations [24].

2.2.3 Mode Choice Module

The mode choice module is essential for modeling how individuals travel between activities by selecting the mode of transportation (e.g., car, public transit, walking, cycling) for each trip generated as part of their daily activity pattern [23].

Individuals' activity plans or schedules are the results of activity-based models. The activity generation module, which is the first one, is essential to creating a precise and realistic transport demand model [23].

2.3 Machine Learning

Machine learning (ML) is a branch of artificial intelligence (AI) that investigates algorithms and approaches for automating solutions to complicated problems that are difficult to solve using traditional programming methods [25]. In recent years, numerous researchers have made use of machine learning. Machine learning algorithms excel at managing enormous datasets, extracting useful insights and patterns that people would be unable to detect. The primary benefit of employing machine learning is that once an algorithm understands what to do with data, it can do its tasks automatically [26].

Machine learning is revolutionizing the transportation business by allowing more powerful tools for monitoring and forecasting traveler behavior, enhancing traffic management, and improving infrastructure. ML approaches can be used to estimate future travel demand based on previous data [27], forecast traffic conditions [28], optimize public transport timetables and routes by evaluating passenger demand [29], and for a variety of other purposes.

2.4 Neural Networks

From a biological standpoint, neural networks are inspired by the structure and function of the human brain, which consists of billions of interconnected neurons that process and send information via electrical signals [30]. Each neuron receives input, processes it, and sends outputs to other neurons, establishing intricate networks that enable advanced cognitive activities including as learning and memory [31]. Artificial Neural Networks, a subclass of machine learning, are computational models inspired by the structures and functions of the

human brain [26]. These artificial networks are made up of layers of nodes, or "neurons," each of which performs a basic mathematical function. During training, the connections between nodes are changed to decrease prediction errors, with weights analogous to synaptic connections in the brain [32]. It works on three levels. The input layer accepts input. The hidden layer processes the input. Finally, the output layer transmits the determined results [26].

2.4.1 Single-layer neural networks

Single-layer neural networks, also known as perceptrons, are the simplest form of artificial neural networks. The perceptron has no hidden layers, which implies it only has one input and one output layer [33]. Each input node is linked to an output node with a weight, and the output is typically decided by a step function that activates when the weighted total of inputs reaches a predetermined threshold [34]. A layer's perceptron can be viewed as a linear function that receives inputs, integrates them with weights and biases, and outputs the result. The Figure 2.1 shows the single-layer neural network model.

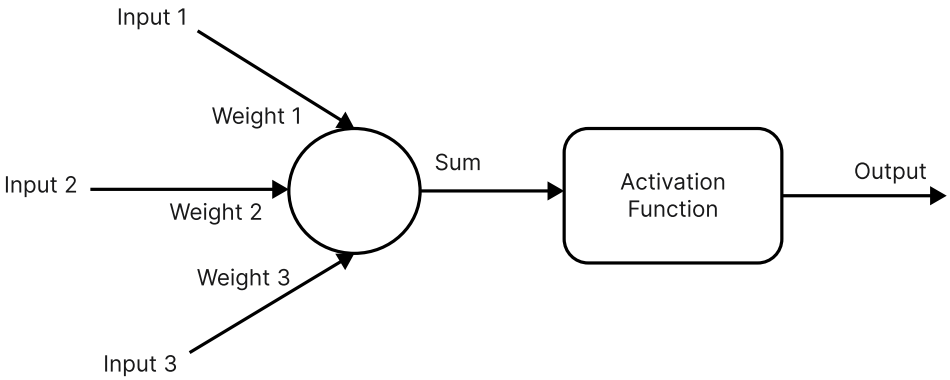

Figure 2.1: The single-layer neural network model.

2.4.2 Multi-layer neural networks

Multi-layer neural networks are regarded more advanced models because they enhance the capabilities of single-layer networks by integrating one or more hidden layers between the input and output layers. The additional hidden layers enable the network to understand and represent more complicated, nonlinear relationships in the data. Each layer comprises of nodes that use activation functions to introduce nonlinearity, allowing the network to record complex patterns [35]. Multilayer neural networks are trained using the back-propagation algorithm, which implies that the network modifies the weights of connections. This fact reduces the difference between expected and actual outputs by spreading the error gradient backward through the network [36].

The multi-layer neural network consists of three layers: input, hidden, and output. The input layer is made up of neurons that receive the initial data characteristics. In the hidden layer, neurons transform inputs using weighted operations and a nonlinear activation function. Finally, the output layer makes final predictions or classifications by

processing signals from the hidden layers and produces the network's output via weighted operations [37]. Figure 2.2 shows the multi-layer neural network model.

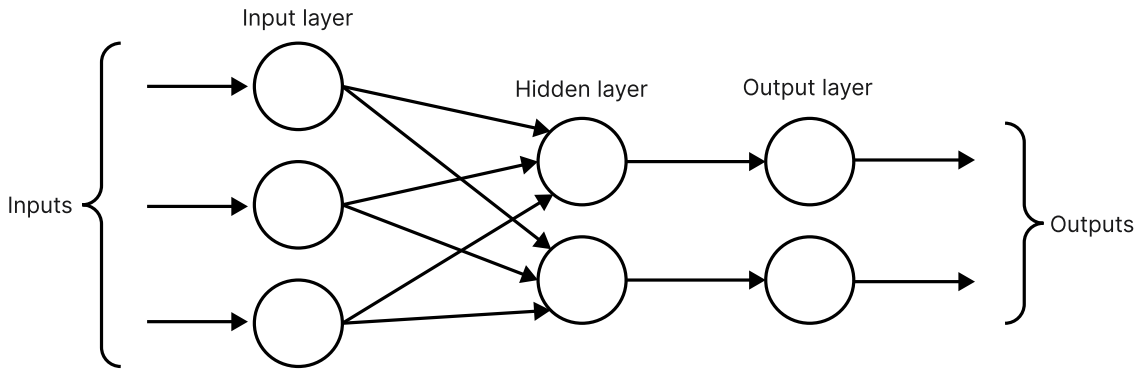

Figure 2.2: The multi-layer neural network model.

2.4.3 Optimizers

Optimizers in neural networks are algorithms that modify the weights of the network to reduce the loss function during training. They are critical to the training process because they determine how the model's parameters are changed in response to the gradients generated from the loss function [38]. Stochastic Gradient Descent (SGD), RMSprop, and Adam are examples of commonly used optimizers [39].

The Adam optimizer (Adaptive Moment Estimation) is a common optimization technique that combines two variants of stochastic gradient descent, AdaGrad and RMSprop. Adam computes adaptive learning rates for each parameter by calculating the gradient's first and second moments, allowing for faster and more efficient convergence [40].

2.5 Decision Trees

One of the most widely used techniques today not only for classification tasks, but also for regression tasks, are decision trees. A decision tree is a method that uses a tree structure where each path from the root to a leaf represents a sequence of data splits, culminating in a boolean result at the leaf node [2]. To put it simply, a decision tree is made up of decision nodes, which stand for decisions or attribute tests, branches, which reflect the results of these decisions, and leaf nodes, which stand for all potential outcomes or forecasts.

2.5.1 Structure of a Decision Tree

- **Root Node** Represent both the first decision to be made and the entire dataset. There are no incoming branches for it.
- **Internal Nodes** Represent judgments or attribute tests. There are one or more branches on each internal node.

- **Branches** Represent the result of a choice or examination that leads to a different node.
- **Leaf Nodes** Describe the ultimate judgment or forecast. At these nodes, no more splits take place.

Figure 2.3 represents the structure of a decision tree.

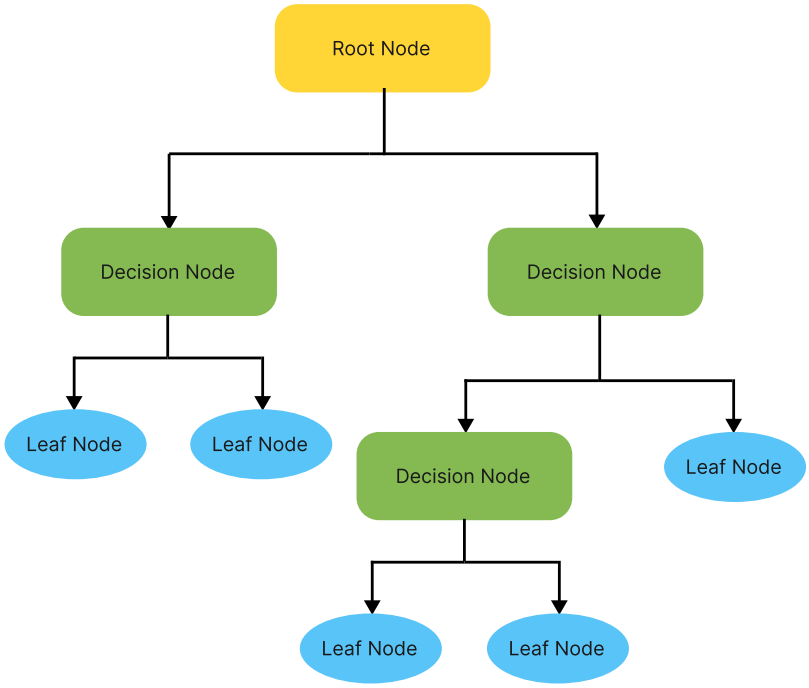

Figure 2.3: Decision Tree Structure [2].

2.5.2 Functionality of the Decision Tree

Using divide and conquer strategy, decision tree learning looks for the best split points inside the tree using a greedy search. The top-down, recursive nature of this splitting procedure is maintained until almost all records are assigned to certain class labels. To predict the class of a given dataset, the method commences at the root node of the tree. By comparing the value of the root attribute with the corresponding attribute in the actual dataset, it follows the appropriate branch to the subsequent node. This procedure is iteratively executed by the algorithm at each successive node, progressing along the tree by comparing the attribute value with those of the sub-nodes. This process persists until a leaf node is reached [41].

The principal difficulty in constructing a decision tree lies in determining the optimal value to use when splitting the node of the tree [41]. The method finds the best partition in the training set to solve this problem. The splitting criterion, often called the attribute selection method, is the metric used to identify the most useful attribute [42]. Two of the best splitting criterion are Gini Impurity and Entropy.

Gini Impurity

This measure quantifies the disorder or impurity within a group of elements. It assesses how often a randomly chosen element from the set would be incorrectly classified if its classification were assigned randomly according to the distribution of labels in the set [43]. The Equation 2.1 can be used to compute the Gini impurity.

$$\text{Gini} = 1 - \sum_{i=1}^{n} (p_i)^2 \tag{2.1}$$

Where n is the number of classes and p_i represents the likelihood or probability of an instance being assigned to a specific class.

Entropy

This measure, which usually ranges from 0 to 1, expresses the degree of uncertainty or impurity in a dataset. With 0 denoting complete certainty and higher values denoting greater uncertainty, a lower entropy value corresponds to less uncertainty [2]. The entropy is measured by Equation 2.2

$$\text{Entropy}(S) = -\sum p(x) \log_2 p(x) \tag{2.2}$$

Where S represents the dataset used to calculate entropy, X denotes the set of classes within the dataset, and the probability of data points in class X relative to the total number of data points in set S is denoted by $P(x)$ [44].

2.5.3 Decision Tree Algorithms

Using decision tree algorithms, one can get the best splits for distinct classes by deciding which qualities to evaluate at each node. Consistent splitting criteria are needed since every resulting partition at every branch strives for maximum purity [45]. Although decision tree algorithms come in a variety of forms, one of the most significant is the Classification And Regression Tree (CART).

CART, introduced by Breiman in 1984 [46], constructs classification trees through binary splitting of attributes. It utilizes the Gini index to select the optimal splitting attribute. CART also supports regression analysis through regression trees, which forecast a dependent variable based on predictor variables over a specified time period. CART accommodates both continuous and nominal attribute data and demonstrates moderate processing speed on average [47].

In CART, pruning of a complex tree structure occurs after splitting, starting from the leaves and progressing towards the root. To achieve the most effective decision tree in this algorithm, the tree is evaluated with randomly selected test data after each pruning step, aiming to determine the optimal tree structure [48].

2.6 Random Forest

Another machine learning model used for classification and regression tasks is random forest. In order for Random Forest to function, it builds several decision trees during training and outputs continuous values (regression) or discrete labels or specific categories (classification) for each tree [49]. In order to improve accuracy and control de overfitting the random forest combines the predictions of multiple decision tree. A random subset of the data and characteristics is used to train each tree in a random forest, introducing variation among the trees and assisting in the development of a strong model [50]. Random forest classification can be considered as an extension of the main random forest algorithm. It is necessary to understand key ideas and subprocesses in order to fully comprehend all of the mechanisms underlying the random forest approach.

2.6.1 Ensemble Learning

It is a technique where multiple models (weak learners) are combined to produce a singles robust model. Using this approach has the goal of preventing the selection of an inadequate model for a given problem. Rather, a more accurate solution can be achieved by integrating the results [51]. An example of ensemble learning is Random Forest, which combines several decision trees (weak learners) to create a reliable predictive model.

2.6.2 Bootstrapping

It is a probabilistic technique used to generate multiple random samples (called bootstrap samples) from the original data [52]. Bootstrapping is a technique used in Random Forest to generate numerous subsets of the initial training data. Since each subset is created by sampling the training data with replacement, certain data points may appear more than once in a subset while other data points might be left out.

2.6.3 Bootstrap Aggregation

It is also known as Bagging. Bootstrapping is used in this technique. Several iterations of a predictor are trained on several bootstrapped samples of the original dataset in bagging. Afterwards, a final forecast is formed by voting on each individual model's predictions in classification tasks [53].

When using a Random Forest classifier, numerous parameters can be changed to control its complexity and increase performance. Table 2.1 shows the parameters that can be changed, along with their definitions.

Table 2.1: Adjustable Parameters in Random Forest Classifier [1].

Parameter	**Definition**
Number of Trees (n_estimators)	It stands for the number of trees in the forest.
Maximum Depth of Trees (max_depth)	It stands for the depth of each tree.
Minimum Samples per Split (min_samples_split)	It stands for the bare minimum of samples needed to divide an internal node.
Minimum Samples per Leaf (min_samples_leaf)	It stands for the bare minimum of samples needed to be at a leaf node (A parent node without any children).
Maximum Features (max_features)	It stands for the greatest amount of features that were taken into account when dividing a node.
Bootstrap Sampling (bootstrap)	It represents if creating trees involves using bootstrap samples.
Criterion (criterion)	It stands for the function that gauges a split's quality.
Maximum Number of Leaf Nodes (max_leaf_nodes)	It restricts how many leaf nodes the trees can have.

2.7 Evaluation Metrics.

Currently, there are various evaluation metrics used in the field of machine learning. Evaluation techniques are used to comprehensively assess the effectiveness of machine learning models. An important aspect of evaluation metrics is their ability to differentiate between model results. The choice of evaluation metrics plays a fundamental role in the accurate assessment of machine learning systems [54].

2.7.1 Main Evaluation Metrics in Machine Learning Techniques

The following metrics are considered: accuracy, recall, precision, confusion matrix, and F1-score for the four classification tasks. These metrics are considered critical when working with machine learning models in classification tasks. Furthermore, a Kolmogorov-Smirnov (KS) test statistic was used, which has been applied in classification tasks in two approaches. The details of each metric are provided below.

Confusion Matrix

The confusion matrix is a crucial tool for assessing the performance of a classification model, applicable to both binary and multiclass problems. It is a two-dimensional matrix with rows showing the true labels and columns showing the anticipated labels by a classifier[55]. In binary classification, the confusion matrix is a 2x2 table representing four possible classification outcomes [56]:

- **True Positives (TP):** Instances correctly classified as positive.
- **False Positives (FP):** Instances incorrectly classified as positive. This condition is characterized as a Type 1 Error.
- **True Negatives (TN):** Instances correctly classified as negative.
- **False Negatives (FN):** Instances incorrectly classified as negative. This condition is characterized as a Type 2 Error and is equally harmful as a Type 1 Error.

The representation of a confusion matrix for a binary classification problem is outlined in table 2.2

Table 2.2: Confusion Matrix for Binary Classification.

		Predicted Class	
		Positive (1)	Negative (0)
Actual Class	Positive (1)	TP	FP
	Negative (0)	FN	TN

In the case of a confusion matrix for multiple classes, it resembles the binary-class matrix. In a multiclass classification problem, the confusion matrix is an NxN table (where N is the number of classes) that illustrates the relationship between actual classes and classes predicted by the model. The elements of the matrix are defined as follows:

- **True Positives (TP):** Instances correctly classified as belonging to class i.
- **False Positives (FP):** Instances incorrectly classified as belonging to class i (prediction of class i when the actual class is different from i).
- **True Negatives (TN):** Instances correctly classified as not belonging to class i.

- **False Negatives (FN):** Instances incorrectly classified as not belonging to class i (incorrect prediction of a class different from i).

We can obtain any metric for a class by calculating TP, TN, FP and FN for a respective class.

Accuracy

The accuracy (Equation 2.3) is determined by the proportion of correctly classified samples relative to the total number of samples in the evaluation dataset. Accuracy values range between 0 and 1, where a value of 1 signifies perfect prediction of all positive and negative samples, while 0 indicates accurate prediction of neither positive nor negative samples [57].

$$\text{Accuracy} = \frac{\#\text{ correctly classified samples}}{\#\text{ all samples}} = \frac{TP + TN}{TP + FP + TN + FN} \tag{2.3}$$

Recall

The recall (Equation 2.4) also referred to as sensitivity or True Positive Rate (TPR), quantifies the proportion of positive samples that are correctly classified. It is calculated as the ratio of correctly classified positive samples to the total number of samples that belong to the positive class [57].

$$\text{Recall} = \frac{\#\text{ true positive samples}}{\#\text{ samples classified positive}} = \frac{TP}{TP + FN} \tag{2.4}$$

Precision

Precision (Equation 2.5) represents the fraction of retrieved samples that are relevant, and it is calculated as the ratio of correctly classified samples to all samples predicted to belong to that class [57].

$$\text{Precision} = \frac{\#\text{ correct positive predictions}}{\#\text{ samples classified as positive}} = \frac{TP}{TP + FP} \tag{2.5}$$

F1-Score

The F1 score (Equation 2.6) is calculated as the harmonic mean of precision and recall, which implies that it mitigates the impact of extreme values in both metrics. This measure is asymmetric between classes, meaning it varies based on the definition of positive and negative classes [57].

$$\text{F1-Score} = 2 \times \frac{\text{precision} \times \text{recall}}{\text{precision} + \text{recall}} = \frac{2 \times TP}{2 \times TP + FP + FN} \tag{2.6}$$

Kolmogorov-Smirnov (KS) test

In straightforward terms, the KS statistic for the two-sample test can be defined as the maximum distance observed between the Cumulative Distribution Functions (CDFs) of

each sample [58]. The Cumulative Distribution Function (CDF) can be defined as a mathematical function that describes the accumulated probability of a random variable taking values less than or equal to a specified value. The KS test can be approached in two ways. The first approach is to determine whether the distributions of two samples adhere to the same distribution. Furthermore, it can serve as a metric in classification models to quantify the separation between the distributions of positive and negative classes, offering an alternative method for evaluating classifiers.

In the first approach, the two-sample KS test compares any two provided samples to see if they are from the same distribution. This measure computes the KS test statistic and its associated p-value. The KS test statistic computes the highest absolute difference between the cumulative distribution functions (CDFs) of two samples. The p-value is the likelihood of detecting a test statistic as extreme or more extreme under the null hypothesis that the samples come from the same distribution. If the p-value is less than a preset significance level (e.g., 0.05), the null hypothesis is rejected, implying that the samples come from different distributions [59].

The second strategy uses the KS test to examine the distribution of expected probability for the model's positive and negative classes. A high KS test statistic value combined with a low p-value indicates significant differences in probability distributions, implying that the model efficiently distinguishes across classes.

Calculating the KS statistic directly for multiclass classification jobs is not possible without first translating them into binary classification problems. This can be accomplished using the One vs Rest (OvR) and One vs One (OvO) methods. In our study, we chose to employ the OvR technique, in which each class is compared to all others concurrently. One class is classified as the "positive" class, while the others are called "negative" classes. This method is performed for each class in the dataset, with the results averaged to get a final assessment.

Chapter 3

State of the Art

This section will provide a detailed evaluation of the state of the art, examining studies and methodologies relevant to activity generating tasks. This analysis will contain a full explanation of the methodology used and the outcomes acquired from past research. Through this study, we hope to identify gaps in current understanding and opportunities for future research in the field of activity generation. In terms of activity generating tasks, we divide activity-based models into three categories: utility-based, rule-based, and machine learning (ML-based) [23].

3.1 Utility-based models

The utility-based models are built on the econometric premise that people choose a travel schedule that maximizes their overall utility or satisfaction [23]. One of the first utility models was developed by Bowman [60]. They proposed a model called DaySim.They expected that a basic activity pattern was established before deciding on more comprehensive activity and trip plans. DaySim depicted each individual agenda as a daily activity pattern incorporating tours. The primary tour was divided into three sections: a primary activity, a sub-tour type (only three for subsistence primary activity), and intermediate stops before and after the primary activity location.

Utility-based models can be seen as discrete choice models. Maximum likelihood estimators are used in this model to estimate utility function parameters across a fully enumerated choice set. However, this method is unsuitable for Activity Based Models (ABMs) due to the combinatorial nature of conceivable activities and their sequences, which are not fully observable. To address this, Pougala et al. [61] introduces a methodology for sampling a selected set of whole daily itineraries for individuals. The Metropolis-Hastings algorithm is used to efficiently explore the space of feasible schedules, generating both high and low probability options for reliable parameter estimate.

Nurul et al. [62] suggest a unique way to modeling activity generation using a utility-based paradigm. They presented an activity-based agenda formation paradigm. The concept focuses on the creation of activity agendas, which are a series of activities that people plan and carry out during the day. It takes into account a variety of factors that influence these selections, such as time limits, trip costs, and personal preferences. In addition, the

model includes other components, such as utility functions for various types of activities (e.g., work, leisure, and shopping), time and space limitations, and a probabilistic choice mechanism to mimic decision-making under uncertainty.

Nurul [63] introduces the Comprehensive Utility-Based System for Activity-Travel Scheduling Options Modelling (CUSTOM). This framework represents workers' daily activity scheduling decisions using the utility maximization principle. CUSTOM describes the complex process of scheduling activities and related travel modes. This framework offers specific utility functions for a variety of activities such as work, leisure, errands, and transportation modes. It combines these capabilities into a seamless framework that models how employees prioritize and sequence their tasks and travel throughout the day.

Västberg et al. [64] describes a dynamic discrete choice model (DDCM) for daily activity and trip planning. The proposed model makes travel decisions sequentially in time, beginning at home in the morning and finishing at home in the evening. At each choice step, the utility of an alternative is calculated by adding the action's one-stage utility and the predicted future utility in the attained state. The model creates comprehensive daily activity schedules with any number of excursions, each consisting of one of six activities, 1240 destinations, and four modes. The model is estimated using travel diaries, and simulation results show that it can reproduce time decisions, trip lengths, and the distribution of trip numbers within the sample.

3.2 Rule-based models

Rule-based models reflect individual activity decisions as heuristics or rules [18]. The findings of these models could be the consequence of incorporating observed distributions into activity generating modules. These distributions are typically created from trip survey data [23]. One of the first rule-based models was the Toronto Area Scheduling model for Household Agents (TASHA) proposed by Miller et al. [65, 66]. This model was created to simulate the activity scheduling and interaction process of household members. TASHA generated activities in the following order: activity frequency, start time, and duration. These activity features were derived using empirical probability distributions in a travel survey. The observed distributions for different activity types were developed by cross-classification of person, household, and schedule factors such as gender, age, occupation, job status, student status, the presence of children, and work project status.

Auld et al. [67] describe Agent-based Dynamic Activity Planning and Travel Scheduling (ADAPTS) model. The model is a framework intended to emulate people's everyday activity planning and trip scheduling procedures. The model stresses the dynamic and adaptive nature of these processes, allowing agents to modify their plans in response to changing circumstances and restrictions. The ADAPTS model is built around a thorough portrayal of activity planning procedures. These procedures include decisions about what activities to undertake, when, where, and with whom. The model's decision-making mechanisms are built on rule-based heuristics, which reflect the complexity and variety of human behavior.

Bellemans et al. [68] describe the creation and execution of the FEATHERS activity-based simulation platform. This approach aims to model and predict individual and household travel and activity patterns. The platform's goal is to provide extensive insights into the relationships between travel behavior, activity engagement, and environmental

effect. FEATHERS is an activity-based modeling approach to simulate individuals' daily activity patterns and travel decisions while taking into account numerous elements that influence activity participation, such as socio-demographic traits, time restrictions, and spatial-temporal interdependence.

3.3 Machine learning (ML-based) models

Machine learning is a practical method for automatically extracting rules from data. It may help to improve rule-based models by lowering the complexity of expert-designed components [23]. Hafezi et al. [69] develops a new comprehensive pattern recognition modeling framework utilizing machine learning approaches in the context of activity-based travel demand modelling. This approach uses activity data to generate clusters of similar daily activity patterns. The authors used a subtractive clustering approach to initialize the number of clusters and their centroids. Individuals with comparable activity patterns were recognized and grouped using the FCM (Fuzzy C-Means) clustering technique. Finally, the CART classifier method was used to investigate the interdependence of the qualities in each discovered cluster, as well as to correlate cluster membership with socio-demographic factors.

Hafezi et al. [70] presents a new modeling framework capable of simulating temporal information associated with a traveler's daily activity schedule for use in activity-based travel demand modeling. The suggested modeling framework is comprised of two phases. First, the Random Forest (RF) model predicts temporal information such as start time and activity duration for the set of activities on the agenda. Second, projected activities are put into a skeletal schedule using a heuristic decision rule-based technique and arranged according to two-tier limitations. The results demonstrate that the suggested model can assemble the traveler's schedule with an average accuracy of 81.62% across the 24-hour period.

Pineda-Jaramillo [5] investigated and discussed Machine Learning techniques utilized in transportation research, particularly for modeling travel mode choice. The author introduces Artificial Neural Networks (ANN), Decision Trees (DT), Support Vector Machines (SVM), and Cluster Analysis (CA) and compares them to the Multinomial Logit Model (MNL), which stands out as a discrete choice model employed in this sort of study. Finally, they conclude that Random Forest (variant of Decision Tree algorithms) is the best model for representing travel mode choice.

In the same way, Cheng et al. [7] proposed a robust random forest method to examine travel mode selections in order to assess prediction capabilities and model interpretability. The authors use household attributes, individual attributes, a constructed environment parameters, and travel information. The modes of transportation included in this study was walking, bicycle, E-motorcycle, public transportation (PT), and automobile. Finally, a comparison is done between the random forest approach (RF), support vector machine (SVM), adaptive boosting (AdaBoost), and multinomial logit (MNL), where the RF method obtained a general accuracy of approximately 85.6%, being the best among the others.

Zhao et al. [3] analyzed the primary distinctions in the development, evaluation, and behavioral interpretation of mode choice between logit models and machine-learning models.

In this investigation the authors compared the performance of two logit models (multinomial and mixed logit) and seven machine-learning classifiers, including Naive Bayes (NB), classification and regression trees (CART), boosting trees (BOOST), bagging trees (BAG), random forest (RF), support vector machine (SVM), and neural network (NN). Then, they focus on multinominal and mixed logit model and RF and NN model to extract behavioral insights and compare these findings. The data used for empirical evaluation came from a survey in which each participant was initially asked to estimate the trip attributes such as travel time, cost, and wait time for their home-to-work commute using each of the following modes: walking, biking, driving, and a new public transportation system. The results based on accuracy show that the RF model was the best performing model with an overall accuracy for all travel modes of 85.6%.

Chu et al. [71] introduced an advanced deep learning model named multi-scale convolutional long shortterm memory network (MultiConvLSTM) designed to enhance the prediction of travel demand and origin-destination (OD) flows. They tested the model on the New York taxi dataset, which includes pickup and dropoff times as well as pickup (origin) and dropoff (destination) locations for taxi services reported in New York between January 2009 and June 2015. The correctness of the model is assessed using conventional metrics such as root mean square error (RMSE), mean absolute error (MAE), and symmetric mean absolute percentage error (SMAPE). Finally, the model consistently produced low RMSE, MAE, and SMAPE values when compared to traditional statistical approaches and deep learning models.

Chapter 4

Methodology

4.1 Dataset Description

The dataset used for classification tasks originates from a survey conducted in the city of Cuenca, located in the Azuay province of Ecuador, and was provided by Llactalab-University of Cuenca [72]. The survey targeted approximately 5034 individuals, aiming to gather information on mobility patterns. It included inquiries into both personal information and details regarding participants' displacements. Within the survey, individuals reporting two displacements were included, with detailed information collected for each displacement. The survey is stored in an Excel file in which the rows correspond to individuals and the columns to the questions asked in the survey, which, for our study, will represent the characteristic variables.

Figure 4.1 depicts the diagram illustrating the methodology applied from survey processing to the final dataset utilized in the simulation. Steps 1, 2, and 3 are described in greater detail in Section 4.2, and steps 4, 5, 6, 7, 8 and 9 are explained in Section 4.3, steps 10, 11 and 12 are explained in Section 4.4 and finally the steps 13, 14, 15 and 16 are presented in the Section 4.5

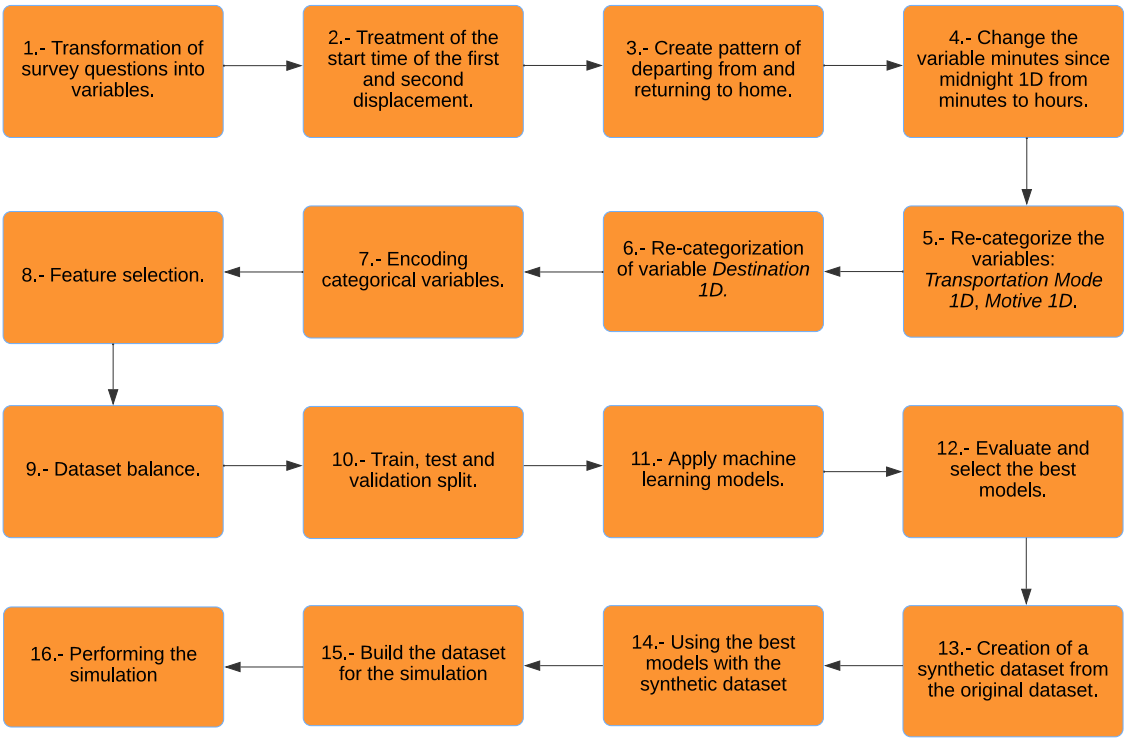

Figure 4.1: Diagram of the applied methodology where the suffix 1D denotes the first displacement.

4.2 Data Preprocessing

At the outset, the columns of the Excel file contained all survey questions. However, to focus on the information relevant to our objective, we selected only the pertinent questions, discarding those deemed unimportant. During data collection, it was observed that some individuals had a maximum of 5 displacements in the survey. Consequently, we decided to limit our sample to those with exactly two displacements, thus filtering the data. Furthermore, the survey included both complete addresses and their codifications. We opted to exclusively work with the address zoning for simplicity and consistency. During the first displacement, it was noted that some individuals did not depart from their homes, leading to their exclusion from the sample. Subsequently, individuals whose first displacement was related to returning home were also excluded. Finally, to streamline analysis, we renamed the Excel file columns, which initially represented survey questions, with names that now refer to variables in our dataset (Step 1). Table 4.1 illustrates how these variables are represented in our dataset, where the suffix 1D denotes the first displacement, and 2D denotes the second displacement.

Table 4.1: Transformation of Survey Questions into Variables.

Survey Question	**Variable**
Q1a.- Interviewee's Age	Age
Q1b.- Interviewee's Gender	Gender
Q1c.- Interviewee's Marital Status	Marital_Status
Q1d.- Occupation	Occupation
Q2si.- Number of Displacements Made	Displacements
Q3_11sib.- Origin Zoning	Origin_1D
Q3_12.- What was the motive for this displacement?	Motive_1D
Q3_13b.- Destination Zoning	Destination_1D
Q3_14.- What time did the interviewee leave the place of ORIGIN to go to this other place?	Departure_Time_1D
Q3_15_O1.- Please, indicate which mode(s) of transportation you used to go from one place to another for the motive of your displacement.	Transportation_Mode_1D
Q3_16.- How long, in MINUTES, did it take you to go from one place to another for the motive of your displacement?	Duration_1D
Q3_21b.- Origin Zoning	Origin_2D
Q3_22.- What was the motive for this 2nd displacement?	Motive_2D
Q3_23b.- Destination Zoning	Destination_2D
Q3_24.- What time did you leave the place to go to this other place?	Departure_Time_2D
Q3_25_O1.- Please, indicate which mode(s) of transportation you used to go from one place to another for the motive of your 2nd displacement.	Transportation_Mode_2D
Q3_26.- How long, in MINUTES, did it take you to go from one place to another for the motive of your 2nd displacement?	Duration_2D
Q4 Number of Vehicles in the Household?	#Household_Vehicles
Q5 Do you have a driver's license?	License
Q6 Do you have vehicle availability?	Vehicle

4.2.1 Treatment of the Start Time of the First and Second Displacement.

In the gathered information, it was observed that the variables Departure_Time_1D and Departure_Time_2D are formatted as HH:MM. To simplify and ensure consistency, these variables were converted to minutes since midnight (Step 2). To achieve this, Departure_Time_1D and Departure_Time_2D were partitioned into the following sub-variables: Hours_1D, Minutes_1D, Hours_2D, and Minutes_2D. Subsequently, variables named Minutes_since_midnight_1D and Minutes_since_midnight_2D were created. These variables encompass the sum of hours transformed into minutes (utilizing the sub-variables Hours_1D and Hours_2D), along with any additional minutes (utilizing the sub-variables Minutes_1D and Minutes_2D), thereby providing the total minutes since midnight.

4.2.2 Pattern of Departing from and Returning to Home.

The current dataset contains information on the two displacements executed by each individual. Our objective is to adhere to a pattern whereby individuals depart from their homes in the first displacement and return to their homes in the second displacement (Step 3). To achieve this, we have selected individuals whose motive for the second displacement is associated with returning home. In Figure 4.2, the motives related to the second displacement can be observed, confirming that all individuals returned home.

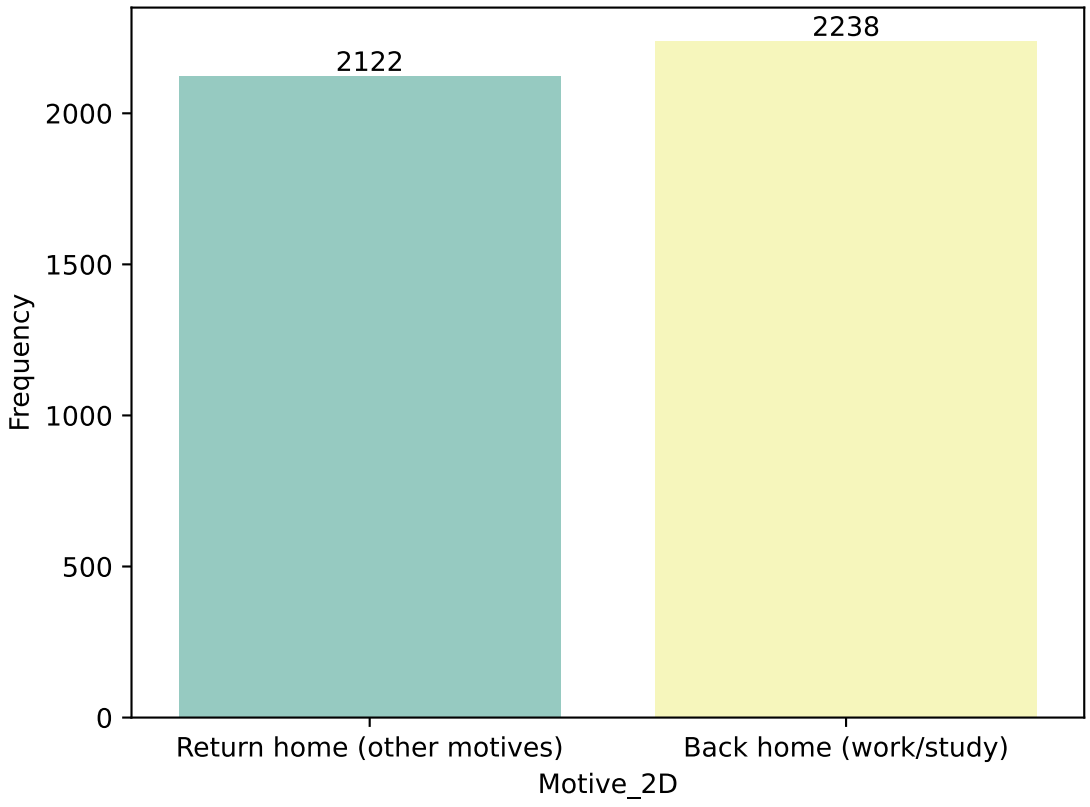

Figure 4.2: Motives of the second displacement

Furthermore, to comply with this pattern, the destination address of the first displacement must align with the origin address of the second displacement. Therefore, we have selected individuals who meet this condition. Additionally, we ensure that the destination of the second displacement corresponds to the origin of the first displacement. Finally, with these adjustments, we have successfully established the desired pattern (Figure 4.3) from the outset.

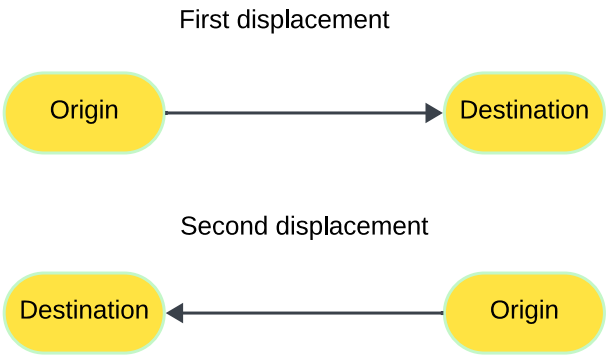

Figure 4.3: Established pattern of individuals with two displacements.

4.3 Preparing the dataset for the models.

The dataset with which we will work in the classification tasks with the objective of generating the activity patterns of the people will consist of the information of the people and the first displacement they made. Table 4.2 shows the 13 variables with which we intend to perform the tasks.

Table 4.2: Type of Variables.

Variable	**Type of Variable**
Age	Numeric
Gender	Categorical
Marital Status	Categorical
Occupation	Categorical
Origin_1D	Categorical
Motive_1D	Categorical
Destination_1D	Categorical
Transportation_Mode_1D	Categorical
Duration_1D	Numeric
#Household_Vehicles	Numeric
License	Categorical
Vehicle	Categorical
Minutes_since_midnight_1D	Numeric

The variable Minutes_from_midnight_1D is changed from minutes to hours, so now the variable will contain the hours from midnight (Step 4). The categories of the variable Transportation_Mode_1D are Urban bus, Car as a driver, Walking, Car as a passenger, Taxi, School bus, Motorcycle as a driver, Company bus, Other buses, Bicycle, Interurban bus, Motorcycle as a passenger, and Others. We re-categorize these categories as follows (Step 5):

- **Car:** Taxi, Car as a passenger, Car as a driver.
- **Bus:** Urban bus, Company bus, School bus, Other buses, Interurban bus
- **Motorcycle:** Motorcycle as a driver, Motorcycle as a passenger.
- **Bicycle:** Bicycle.
- **Others:** Others.

After the re-categorization of the variable Transportation_Mode_1D, we are left with only the mode of transport Car and Bus.

In the case of the variable Motive_1D the categories are Work, Studies/Formation, Shopping, Personal errands, Walking/Accompanying people/Visiting a family friend, Medical/Hospital, Leisure/Entertainment, Leisure/Entertainment, Non-leisure food, and Other. In the same case of the variable Transportation Mode, we re-categorize this variable as follows (Step 5):

- **Studies:** Studies/Formation
- **Personal:** Personal errands, Medical/Hospital, Walking/Accompanying people/Visiting a family friend
- **Others:** Leisure/Entertainment, Non-leisure food, Others.

Finally, with the above re-categorization, the variable Motive_1D went from having 9 categories to 5 categories like Work, Personal, Studies, Shopping and Others.

4.3.1 Re-categorization of variable Destination_1D.

As mentioned in Section 4.2, for the present study we will take into account the mobility micro-zoning of the city for both destination and origin. Figure 4.4 displays the micro-zoning contained in the Destination_1D variable. It can be observed that the variable in question comprises approximately 50 micro-zones. Therefore, we proceed to re-categorize the Destination_1D variable. Due to its significant nature and the complexity of its re-categorization, we opted to employ the K-means method, which is used for clustering samples (Step 6).

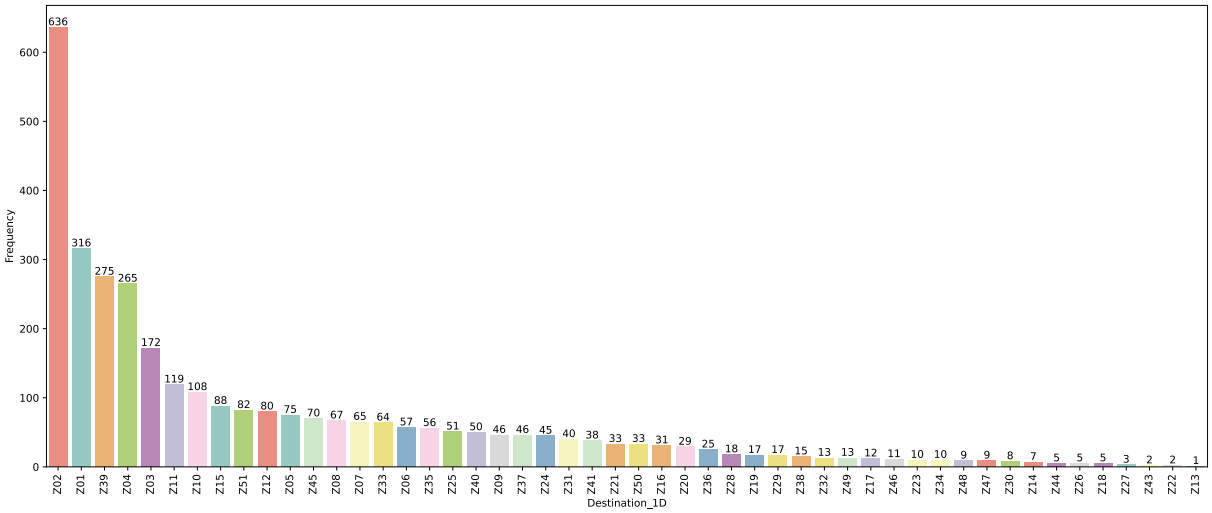

Figure 4.4: Frequency of the Destination_1D variable.

For this purpose, we have access to information regarding the centroids of the zones. From Figure 4.4, it is evident that there are 10 micro-zones with a frequency equal to or greater than 80. Consequently, we decided to choose a value of k=10 to perform the K-means model. Following the application of this algorithm, the micro-zones are grouped into 10 clusters denoted as Zones (Zone01 to Zone10). Finally, in the following list and Figure 4.5 we present how the clusters were ordered with the micro-zones that refer to the destinations of the individuals in the first displacement, leaving us with 10 categories of destinations.

- **Cluster 0(Zone01):** Z09, Z36, Z26, Z37.

- **Cluster 1(Zone02):** Z47, Z16, Z17, Z18, Z19, Z20, Z29, Z44.
- **Cluster 2(Zone03):** Z32, Z50, Z51
- **Cluster 3(Zone04):** Z12, Z10, Z11, Z46, Z41, Z35, Z40, Z31.
- **Cluster 4(Zone05):** Z48
- **Cluster 5(Zone06):** Z03, Z04, Z13, Z45, Z14, Z15, Z23.
- **Cluster 6(Zone07):** Z01, Z02, Z05, Z06, Z07, Z08.
- **Cluster 7(Zone08):** Z39, Z43, Z27, Z34, Z28, Z33.
- **Cluster 8(Zone09):** Z21, Z30, Z24, Z49.
- **Cluster 9(Zone10):** Z42, Z25, Z22, Z38.

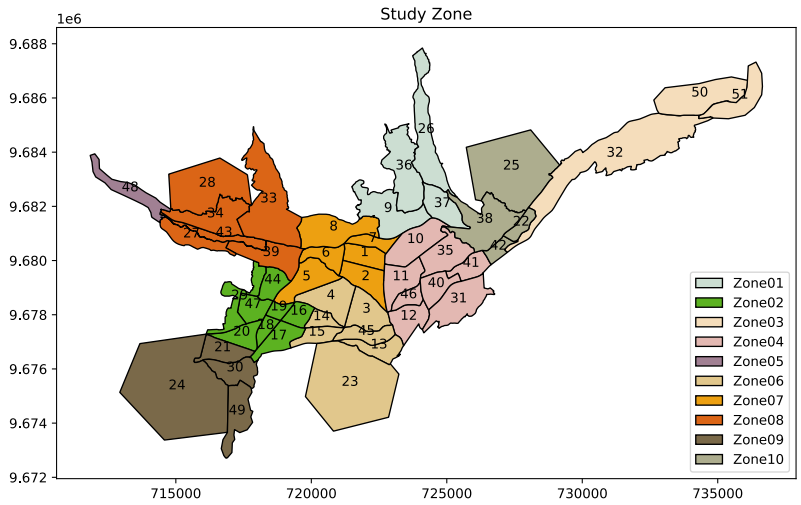

Figure 4.5: Clustering of the Destination variable.

4.3.2 Encoding Categorical Variables.

Because machine learning models are only able to perform their tasks using numerical values, it is necessary to take into account those variables that are categorical and treat them accordingly. In our dataset many variables are categorical, that is, non-numerical variables that acquire values from a limited number of classes or categories. The table 4.3 shows the categorical variables present in our dataset as well as the number of categories each one contains.

Table 4.3: Number of categories of variables categories.

Variable	**Number of categories**
Gender	2
Marital Status	6
Occupation	9
Origin_1D	42
Motive_1D	5
Destination_1D	10
Transportation_Mode_1D	2
License	3
Vehicle	3

Handling categorical variables can pose challenges for certain machine learning algorithms. Converting categorical variables into numerical data is essential to ensure the proper functioning of these algorithms. The method used to encode categorical variables significantly impacts the performance of various algorithms. Each feature's dataset may include one or more labels represented either in word or numeric format, making it easier for humans to interpret the data but requiring further processing to be understandable for computers [73].

Nowadays we can find a great variety of coding techniques for categorical variables that can be used. One-hot encoding and label encoding are the two methods most frequently used to encode categorical data [74]. In this research, given the abundance of categorical variables, we will employ the label coding method to transform them into numerical values in the different tasks that we have, thus allowing us to use them in machine learning models (Step 7).

Label encoding makes it easier to use numerical labels in a machine learning model and is an important step in data preprocessing for supervised learning methods. The Label Encoder generates a unique numerical value for each label in the dataset, replacing the original categorical labels with numerical representations. This technique assigns numbers ranging from 0 to $N - 1$ to each individual value in a category column [75].

4.3.3 Feature Selection.

During the development of a machine learning model, it is usual to come across datasets containing a large number of variables, some of which are important to the model-building process, while others are redundant or insignificant. Including these redundant or unnecessary features in the dataset can reduce the model's overall performance and accuracy [76]. As a result, it is critical to identify and choose the most relevant features from the data while excluding those that are irrelevant or of smaller relevance. This procedure, known

as feature selection in machine learning, is critical to improving model performance and efficacy [77].

One of the methods for feature selection is Mutual Information Feature Selection. Mutual information (MI) quantifies the amount of information one random variable holds about another random variable [78]. It is utilized to measure the reduction in entropy given the target value (Equation 4.1). MI between two random variables is a non-negative value that quantifies the dependency between them.

$$\text{MI}(\text{feature}; \text{target}) = \text{Entropy}(\text{feature}) - \text{Entropy}(\text{feature}|\text{target}) \tag{4.1}$$

A high MI value indicates a stronger association between the feature and the target, thus highlighting the feature's significance in model training. Conversely, a lower MI score, such as 0, suggests a weak relationship between the feature and the target.

In our study, we utilized a total of 13 variables within our dataset. With the aim of reducing the number of variables, as it is often considered that some variables may lack relevance for certain tasks, we opted to employ the Mutual Information Feature Selection method using the *sklearn.feature_selection* library for each of the tasks under investigation (Step 8). For each task, it was imperative to segregate the characteristic variables from the target variable within our dataset, followed by the subsequent feature selection process. Given that our study primarily deals with classification problems, we utilized the *mutual_info_classif* function from the aforementioned library.

Transportation Mode Classification

In Figure 4.6, we depict the characteristic variables alongside their significance concerning the target variable, which in this instance is the transportation mode variable. In this classification task, we will consider the six most significant variables, namely: #Household_Vehicle, Vehicle, License, Duration_1D, Destination_1D and Origin_1D. These variables are pertinent as we endeavor to classify the transportation mode individuals utilize, and thus, they exhibit a strong correlation with the task at hand.

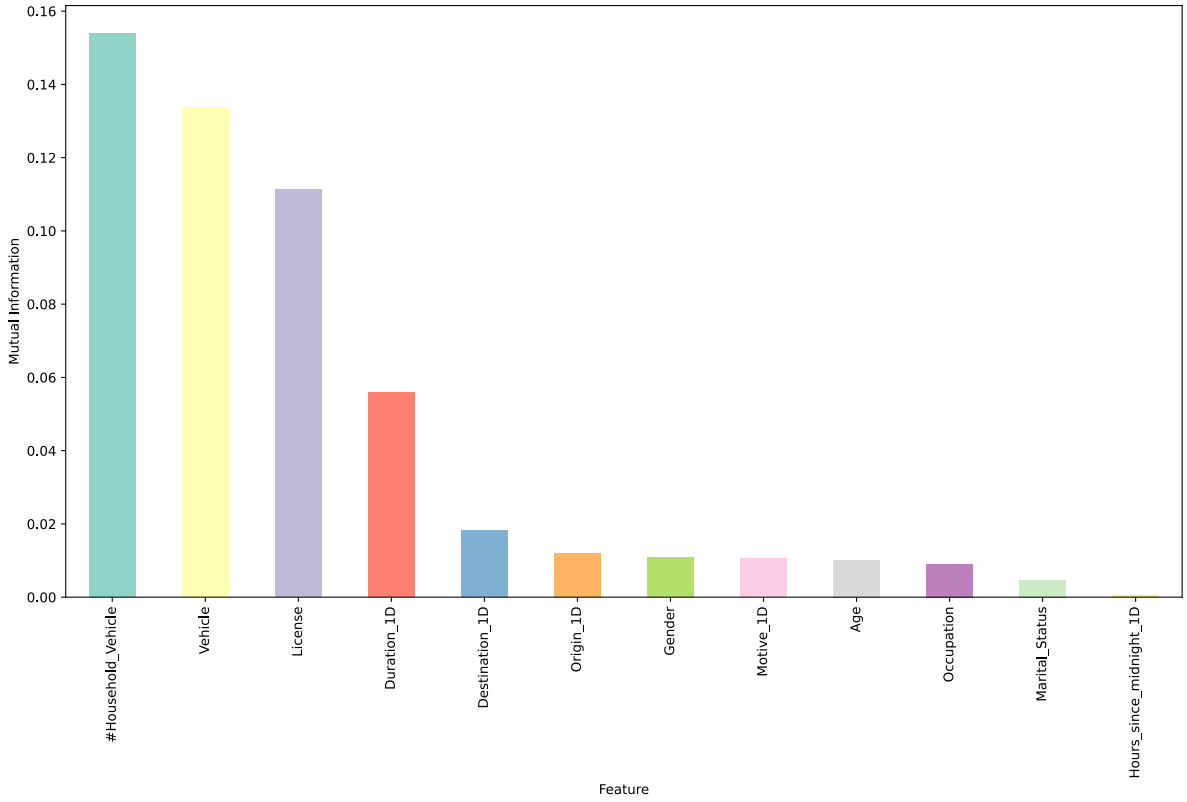

Figure 4.6: Mutual Feature Selection in Transportation Mode Classification.

Motive of Displacement Classification

In the context of classifying the motive of displacement, the six most significant variables identified through Mutual Information Feature Selection are as follows: Occupation, Age, Hours_since_midnight_1D, Marital_Status, Destination_1D, and License, as illustrated in Figure 4.7. Similarly, upon scrutinizing the outcomes of the feature selection method, it becomes evident that these variables are interconnected when endeavoring to classify individuals' motives for displacement.

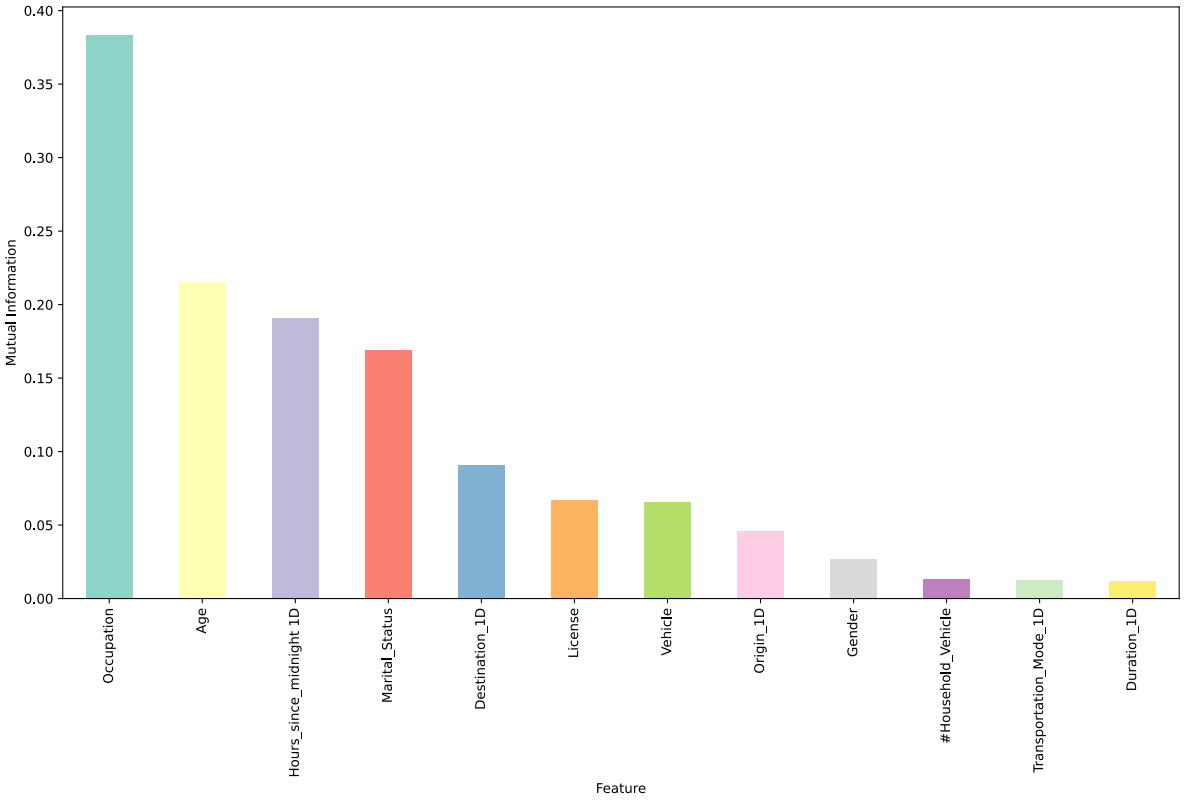

Figure 4.7: Mutual Feature Selection in Motive of Displacement Classification.

Destination Classification

Figure 4.8 illustrates the six most significant variables in the classification of the destination. These variables, namely Duration_1D, Motive_1D, Origin_1D, Hours_since_midnight_1D, Occupation, and Age, are crucial determinants in discerning individuals' destinations. Their prominence underscores their pivotal role in the classification process, as depicted in the graphical representation.

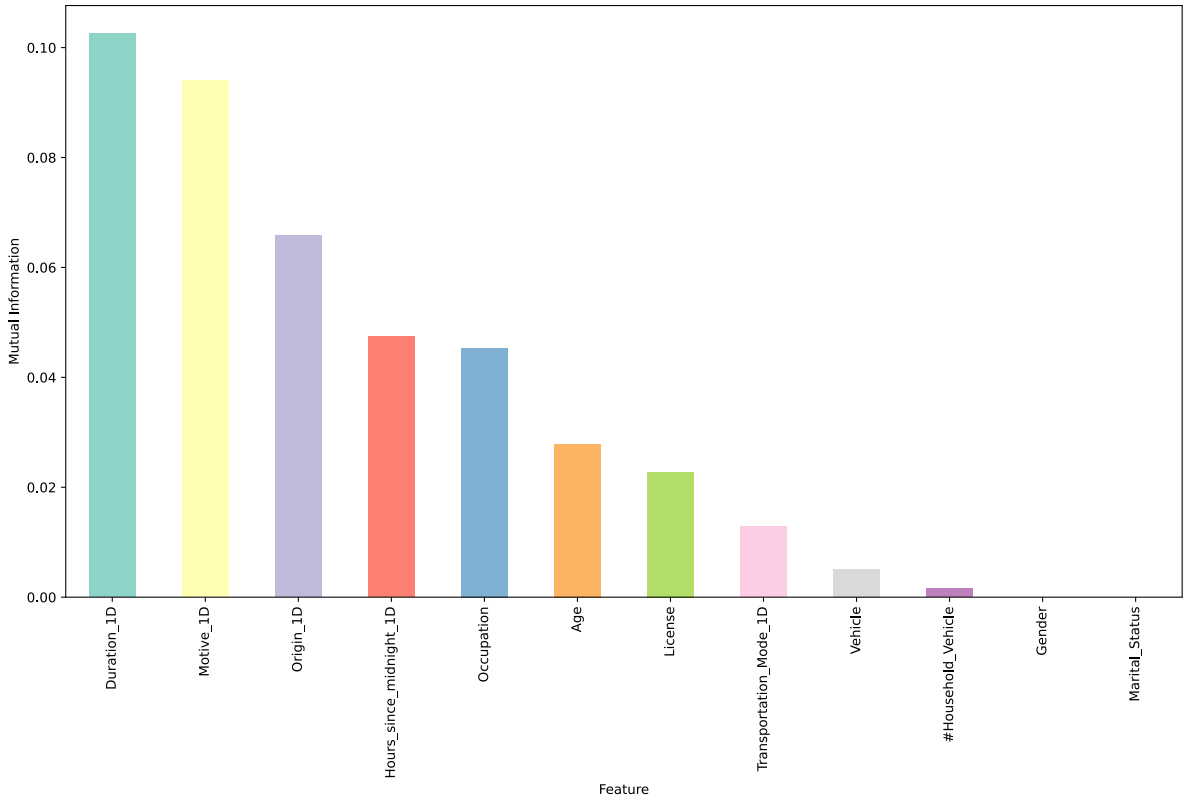

Figure 4.8: Mutual Feature Selection in Destination Classification.

Start Time of Displacement Classification

In the final task, regarding the classification of the start time of displacement, Figure 4.9 delineates the six most influential variables. These variables, namely Motive_1D, Occupation, Age, License, Destination_1D, and Vehicle, play pivotal roles in this classification task. These variables have a major role in this type of classification, especially the variable Motive_1D, since it has a very close relationship with the target variable. With respect to the other variables also great dependence with the target variable.

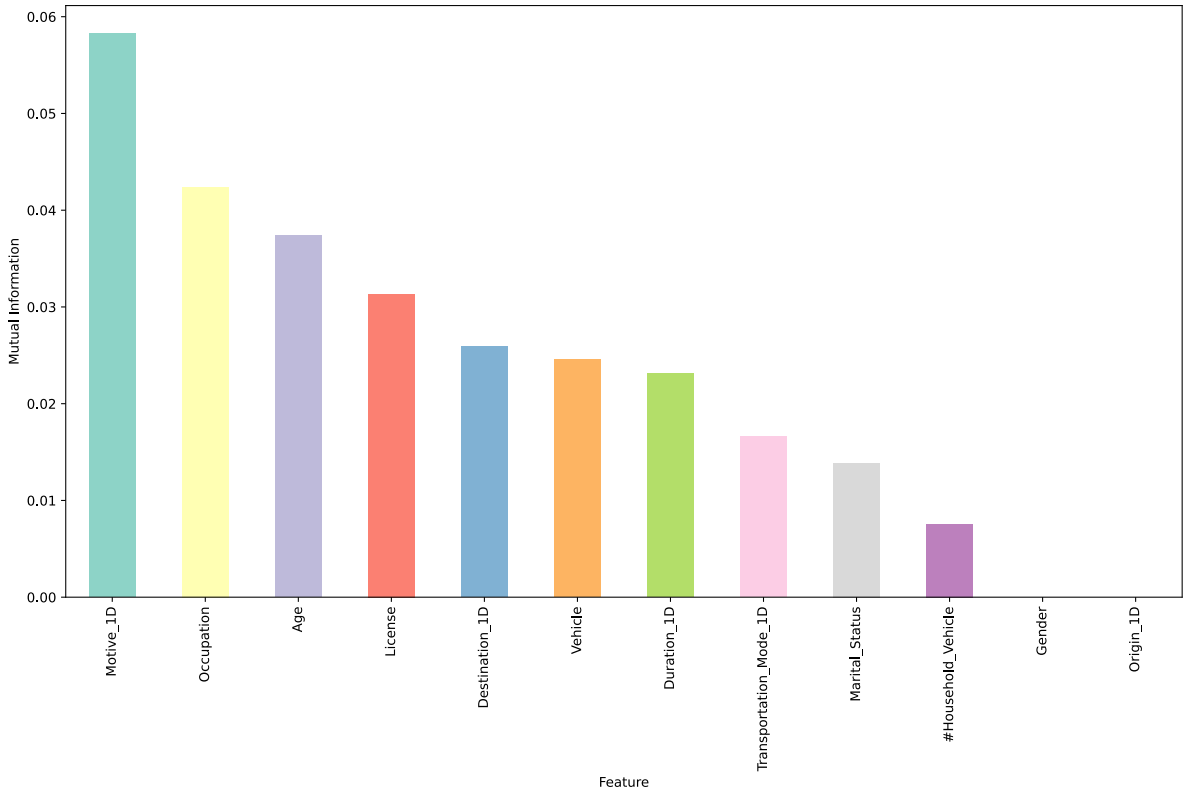

Figure 4.9: Mutual Feature Selection in Start Time of Displacement Classification.

4.3.4 Data balance.

Datasets sourced from real-world scenarios and utilized for classification problems often exhibit inherent imbalance. In such datasets, the instances belonging to certain classes vastly outnumber those of others, resulting in what is commonly referred to as class imbalance [79]. Class imbalance arises when there is a greater quantity of observations within one class compared to another, although the former class holds lesser significance than the latter [80]. Imbalanced datasets in machine learning represent a significant area of scientific concern that has garnered increased attention in recent times [81]. Addressing this issue appropriately is imperative, given its importance in ensuring robust model performance and reliable outcomes.

The most direct technique to remedy the class imbalance problem is to resample the original dataset [82]. Resampling techniques can be used to either undersample or oversample the dataset [83]. Undersampling is the process of reducing the number of majority target occurrences or samples [84]. Oversampling operates by generating new instances or duplicating existing examples from the minority class to augment its representation within the dataset [85]. One of the methods employing undersampling is the random undersampling technique, which involves randomly removing samples from the majority class until a balanced distribution between classes is achieved within the remaining dataset. Conversely, an oversampling technique known as the random oversampling technique involves augmenting the instances of the minority class through random replication of the existing samples within that class.

In our research, data balancing will be applied across all classification tasks, as significant class imbalance is evident within the target variables in certain tasks. Currently, the dataset comprises a total of 3254 records; however, the dataset size will be altered post-data balancing. Tables 4.4, 4.5, 4.6, and 4.7 display the frequency distributions of the categorical variable categories for the various classification tasks.

Table 4.4: Frequency of categories in Transportation Mode Classification.

Category	**Frequency**
Bus	1601
Car	1653

Table 4.5: Frequency of categories in Destination Classification.

Category	**Frequency**
Zone 01	122
Zone 02	125
Zone 03	128
Zone 04	502
Zone 05	9
Zone 06	613
Zone 07	1216
Zone 08	372
Zone 09	99
Zone 10	68

Table 4.6: Frequency of categories in Motive of Displacement Classification.

Category	**Frequency**
Work	1053
Personal	907
Studies	619
Shopping	573
Others	102

Table 4.7: Frequency of categories in Start Time of Displacement Classification.

Category	**Frequency**
0 to 4	6
4 to 8	835
8 to 12	1532
12 to 16	592
16 to 20	273
20 to 24	16

In most classification tasks, an imbalance in the categories of the target variable is often observed. An exception lies in the classification task of transportation mode, where the two categories are nearly balanced; however, we have chosen to undertake balancing nonetheless. We employ the random oversampling technique to rectify the imbalance in the categories of the target variables across our classification tasks (Step 9). This entails equalizing the minority categories to the majority categories in all classification tasks. We utilize the *RandomOverSampler* function from the *imblearn.over_sampling* library to achieve this. In addition, to guarantee reproducibility in the oversampling process, thus ensuring consistent results each time the code is run, we set a *random_state* parameter at 42. As a consequence of data balancing, the size of our dataset adjusts according to the classification task at hand. Table 4.8 summarizes the new dataset size for each classification task.

Table 4.8: Size of the dataset after applying random oversampling technique.

Task	**Size of dataset**
Transportation Mode Classification	3306
Motive of Displacement Classification	5265
Destination Classification	12160
Start Time of Displacement Classification	9192

4.4 Applied Machine Learning Technique

To address the stated objectives, several machine learning techniques will be applied, each designed to solve specific aspects of activity pattern generation in the context of urban transportation demand. It is important to mention that with the previously prepared datasets (balanced datasets), we can make use of machine learning models for the task such as the start time of displacement, transportation mode, motive of displacement, and destination classification.

Partitioning the dataset into training, validation, and testing datasets stands as one of the fundamental techniques in the development and evaluation of machine learning models. Solely the training samples are utilized to train the machine learning models, and once the training process is finalized with no further adjustments required, the testing samples are employed to evaluate the performance of the trained models under deployment conditions. To reduce model bias, the training step must be conducted without knowledge of the instances specified for testing. To divide our dataset into training, validation, and testing sets (Step 10), we utilize the *train_test_split* function of the *sklearn* library to divide it into training and test sets, using an 80% training and 20% testing ratio. The training set is then divided into two parts: a smaller training set (80% of the original) and a validation set (20% of the original). It employs the *stratify* option to maintain the distribution of classes inside each subset and a random seed (*random_state=42*) to ensure that the results are reproducible.

After partitioning our dataset into training, validation, and testing sets, we proceed with the utilization of three distinct machine learning models (Step 11), subsequently selecting the one demonstrating superior performance based on evaluation metric (accuracy). The optimal model is chosen for each classification task. It is noteworthy that identical model configurations are applied across all four classification tasks. A detailed description of each machine learning model used is provided below.

4.4.1 Random Forest Classifier

One of the machine learning models frequently utilized in tasks related to activity pattern generation is the Random Forest Classifier. Apart from its application in activity pattern generation tasks, Random Forest Classifiers have significantly influenced a wide array of classification tasks. Due to their inherent mechanism of constructing numerous distinct decision trees and subsequently amalgamating their outputs to yield more precise and reliable predictions, the Random Forest Classifier was selected as one of the models for our four classification tasks, aiming to facilitate the generation of individuals' activity patterns.

In pursuit of enhancing the model's performance, two popular techniques were employed for hyperparameter optimization: Grid Search and Random Search. Since both techniques utilize a parameter grid to determine the optimal parameters, said grids were defined accordingly. Tables 4.9 and 4.10 present the grids utilized in these techniques.

Table 4.9: Parameter grid for Grid Search.

Hyperparameter	**Values**
Criterion	Gini, Entropy
max_samples	None, 100,200
min_samples_leaf	3, 5, 6, 10, 20
n_estimators	40, 60, 100

Table 4.10: Parameter grid for Random Search.

Hyperparameter	**Values**
n_estimators	Random number between 50 and 500
max_depth	Random number between 1 and 100

It is noteworthy that ten-fold cross-validation was employed in both techniques to facilitate the hyperparameter search, aiming to achieve an optimal hyperparameter configuration.

For constructing the Random Forest Classifier model, we utilized the *sklearn.ensemble* module, which is part of the *scikit-learn* library, also known as *sklearn*. This library stands as one of the most widely used machine learning libraries in Python. Furthermore, for techniques such as Grid Search and Randomized Search with cross-validation, the *sklearn.model_selection* module was utilized, which is also a part of the *sklearn* library. Finally, after determining the optimal combination of hyperparameters using the two separate strategies, the model training procedure begins.

4.4.2 Decision Tree Classifier

Another learning model used in classification tasks is the Decision Tree Classifier. This model works by breaking down the dataset into smaller subsets based on key attributes. These partitions are carried out recursively so that each subset becomes more homogeneous in terms of the target variable being predicted. In our study, we have opted to incorporate the Decision Tree Classifier for our four classification tasks. With the aim of enhancing the performance of this model, we decided to conduct a search for the optimal parameters. The techniques employed were, similar to the Random Forest Classifier model, Grid Search and Random Search.

As previously stated, both Grid Search and Random Search strategies are based on parameter grids, which must be created before using these methods. However, when using a different model, these grids will be modified to suit the hyperparameter values unique to the new model. Tables 4.11 and 4.12 show parameter grids for the aforementioned approaches.

Table 4.11: Parameter grid for Grid Search in Decision Tree Classifier.

Hyperparameter	**Values**
Criterion	Gini, Entropy
max_depth	None, 100,200,300
min_samples_split	2, 5, 10
min_samples_leaf	3, 5, 6, 10, 20

Table 4.12: Parameter grid for random search in decision tree classifier.

Hyperparameter	**Values**
max_depth	Random number between 1 and 100
min_samples_split	Random number between 2 and 10
min_samples_leaf	Random number between 1 and 5

Both parameter search techniques were utilized with ten-fold cross-validation to facilitate the selection of the best hyperparameters and, furthermore, to mitigate overfitting, which is a common issue in machine learning models.

In order to construct the Decision Tree Classifier, we utilized the *sklearn.tree* module from the *sklearn* library. Additionally, for the implementation of Grid Search and Random Search techniques with cross-validation, we employed the *sklearn.model_selection* module, similar to the approach taken with the Random Forest Classifier. Subsequently, leveraging the outcomes derived from these techniques across the four classification tasks, we proceeded to train the models, facilitating the selection of the optimal model for each respective task.

4.4.3 Artificial Neural Network

The latest model utilized for the four classification tasks is the Artificial Neural Network (ANN). In this study, feedforward networks are employed due to the unidirectional flow of information, proceeding from the input layer through a series of hidden layers to the output layer. The neural network comprises multiple layers described as follows:

- **Input Layer:** A fully connected layer with 1000 neurons and ReLU activation function, receiving input data.
- **Dropout Layer 1:** A dropout layer that randomly deactivates 50% of the neurons during training to prevent overfitting.
- **Hidden Layer 1:** Another fully connected layer with 500 neurons and ReLU activation function.
- **Dropout Layer 2:** A dropout layer that randomly deactivates 50% of the neurons during training to prevent overfitting.
- **Hidden Layer 2:** A third fully connected layer with 50 neurons and ReLU activation function.
- **Output Layer:** The output layer tailored to the classification task, featuring neurons and an activation function specific to each task.

The input, hidden, and dropout layers have consistent configurations across all four classification tasks. However, the output layer differs in terms of both the number of neurons

and the activation function employed. Table 4.13 outlines the output layer configurations for each categorization task.

Table 4.13: Output layer configuration.

Task Classification	**Neurons**	**Activation Function**
Transportation Mode	1	Sigmoid
Motive of Displacement	5	Softmax
Destination	10	Softmax
Start Time of Displacement	6	Softmax

In the output layer, two activation functions are utilized. The softmax function is commonly employed in multiclass classification problems, assigning probabilities to multiple classes and ensuring that the sum of these probabilities equals one [86]. Conversely, the sigmoid function is utilized in binary classification problems, transforming the output into a value between 0 and 1 [87], interpreted as the probability of belonging to a specific class.

Regarding the configuration of the ANN for training, the Adam optimizer was utilized with a learning rate of 0.001 to optimize the network weights during training. The choice of loss function depends on the type of classification being performed. For binary classification (transport mode), *binary_crossentropy* is used, whereas for multiclass classification (motive, destination, and hour range), *sparse_categorical_crossentropy* is employed. The evaluation metric during training is accuracy, which enables monitoring of the model's performance as it trains on the data.

One of the primary challenges encountered in most classification problems is overfitting. Overfitting occurs when the neural network memorizes the training samples instead of learning generalizable patterns that can be applied to new classification data [88]. To mitigate the issue of overfitting, one method involves stopping training after a predetermined number of training epochs, which represent all rounds of iterations over the training dataset. To address this issue in our study, Early Stopping was implemented to halt training if the loss on the validation set (val_loss) ceases to decrease after a certain number of epochs (100 epochs), restoring the best weights to prevent overfitting.

Lastly, it is important to mention that numerical feature variables were standardized before being used in the ANN. Standardizing a dataset involves rescaling the distribution of values so that the mean of observed values is 0 and the standard deviation is 1. This was accomplished using the scikit-learn object *StandardScaler*. Additionally, the validation set created is used during the ANN training process. The ANN was trained during 500 epochs with the configuration described above.

4.4.4 Model Evaluation

In the evaluation of the models (Step 12), it is imperative to emphasize that the sole criterion for selecting the optimal machine learning model will be accuracy. The evaluation using this metric will be conducted on the three employed models: Random Forest, Decision

Tree, and Neural Network. Subsequently, after identifying the best-performing model for each task, we will evaluate it using the validation dataset, given the unavailability of a new dataset for the evaluation of the models with the best results. Each of the top-performing models will be assessed using the metrics described in Section 2.7.

The evaluation will consist, firstly, of examining the results of the confusion matrix, where the classification performance of the models will be observed. Subsequently, the performance of each class in each task will be assessed, culminating in the application of the KS statistical test in its two previously described approaches. Once the top-performing models have been evaluated, we will proceed to construct the dataset to be used for the simulation.

4.5 Simulation

Regarding the simulation we will use the MATSim tool (Multi-Agent Transport Simulation). MATSim (Multi-Agent Transport Simulation) serves as a robust instrument for simulating individual movements within transportation networks. MATSim, an activity-based, extensible, multi-agent simulation framework, is developed in Java. As an open-source project, it is accessible for download via the Internet. MATSim offers a comprehensive framework for executing large-scale, agent-based transport simulations. The framework is composed of various modules that can be integrated or utilized independently [89].

In a MATSim simulation, the number of iterations is variable. The daily activities of the population within the research region are utilized in each iteration. The individuals being modeled are referred to as agents. Typically, activity chains are generated using discrete choice modeling or by sampling from empirical data [90]. In our study, we will employ our most advanced artificial intelligence models to construct these activity chains.

4.5.1 Creation of a synthetic population and use the best models

In our study, the following variables will be required to conduct the simulation using MATSim: Origin, Destination, Start Time, End Time, Means of Transportation, and Motive of Displacement. The initial step will involve generating a synthetic population from the original dataset (Step 13). The creation of the synthetic dataset will be dependent on the distributions of each variable in the original dataset because our original dataset includes both numerical and categorical variables. It is significant to remember that the start time of the second displacement will be considered as the end time. As a result, the start time of the second displacement from the original dataset will also be taken into account when creating the synthetic population. We will create a synthetic population with 5000 records for our simulation.

Once the synthetic dataset is generated, we can apply our machine learning models (Step 14). For four of the variables necessary for the simulation (means of transport, motive, destination and start time), we will use machine learning models, since these are implemented for classification tasks and will allow us to obtain the mentioned variables. For this purpose, we use the synthetic dataset as well as the relevant variables for each of the classification tasks.

4.5.2 Creation of the dataset for simulation

Once we have used the models and obtained the results we can build the dataset for the simulation with the variables mentioned above (Step 15). At the moment, we do not have any developed machine learning models for the origin and end time variables. As a result, the synthetic dataset will be used to directly pick these variables. On the other hand, we will make use of models for the Means of Transportation and motive of displacement variables, as previously stated. No additional processing of model outcomes is required because the classification tasks used to generate these variables do not need class aggregate of the target variable.

A different situation arises when models are used to obtain the start time and destination variables. Since in the classification tasks the target variables were categorized (start time in 6 categories and destination in 10 categories), it is necessary to apply a special treatment, since the simulator lacks the capability to receive these variables in categories.

To mitigate this problem when obtaining results from the models, which are in the form of categories, it was necessary to make a selection based on probabilities to obtain a value that falls within these categories. These probabilities are based on the distribution (normalized) that each value has in the original dataset, to ensure that the choice of the value within the category is mostly representative of the original dataset.

We examine and filter out records where the end time is larger than the start time once the start time variable has run out of categories. This is due to the possibility that some cases with start times earlier than ends could have been generated during the probability-based selection procedure. Also, the variables start and end time will be converted to seconds, as required by the simulator. Since we will be using the simulator in its most basic configuration, we will filter the data set for the simulation, keeping only the records that used the car as a means of transportation.

4.5.3 Mobility Plan File and Configuration File

After preparing the dataset for simulation, we create the XML file containing the agents' mobility plans. The XML file comprises a list of people, and each person contains a list of plans, which in turn have a list of activities and legs. Figure 4.10 depicts an example of a mobility plan. In this example, we can see the person's ID as well as a section specifying if the plan would be executed by the simulation. The list of activities is then displayed, together with the type of activity and the time required to complete it. In addition, to characterize the location of an activity, a coordinate (x and y) is provided. Finally, we have a leg that defines how an agent intends to travel from one point to another; each leg requires a means of transportation.

```xml
<population>
  <person id="0">
    <plan selected="yes">
      <activity type="h0" x="720268.2651734542" y="9677481.594011476" end_time="15:00:00" />
      <leg mode="car" />
      <activity type="Shopping" x="727932.348516802" y="9682542.8692356" end_time="19:30:00" />
      <leg mode="car" />
      <activity type="h0" x="720268.2651734542" y="9677481.594011476" />
    </plan>
  </person>
```

Figure 4.10: Mobility Plan Example.

When we want to use the simulator, we must change the configuration file. Figure 4.11 depicts what the configuration file looks like. The red box depicts the network configuration module, which consists of a node and a link graph representing the study zone. The green box depicts the mobility plan file described above. The blue box specifies the configuration of the folder to which the simulation results will be stored, as well as the number of iterations to be done during the experiment. The yellow box is where the kind and length of each activity are set.

```xml
<config>
    <module name="network">
        <param name="inputNetworkFile" value="CuencaNet04.xml" />
    </module>

    <module name="plans">
        <param name="inputPlansFile" value="CuencaInitPlansRawSintetico.xml" />
    </module>

    <module name="controler">
        <param name="outputDirectory" value="output/test03" />
        <param name="firstIteration" value="0" />
        <param name="lastIteration" value="0" />
    </module>

    <module name="planCalcScore" >

        <parameterset type="activityParams" >
            <param name="activityType" value="h0" />
            <param name="typicalDuration" value="12:00:00" />
        </parameterset>

        <parameterset type="activityParams" >
            <param name="activityType" value="Study" />
            <param name="typicalDuration" value="12:00:00" />
        </parameterset>

        <parameterset type="activityParams" >
            <param name="activityType" value="Shopping" />
            <param name="typicalDuration" value="12:00:00" />
```

Figure 4.11: Configuration File Example.

Finally, with the mobility plans and configuration ready, we can perform the simulation using the MATSim tool (Step 16).

Chapter 5

Results and Discussion

5.1 Performance of Models in Specific Tasks

To validate the machine learning models, we assessed multiple models and selected the best one for each classification task. We utilized Python, Keras, and Scikit-learn environments to develop the three machine-learning models employed. Each of these models will be evaluated across the various classification tasks using the configurations and dataset described in the preceding section.

5.1.1 Transportation Mode Classification

For this type of task, the input variables that will be used in the three classification models are: #Household_Vehicle, Vehicle, License, Duration_1D, Destination_1D, and Origin_1D. The target variable is the mode of transportation used in the first displacement (Transportation_Mode_1D). In the case of the Random Forest Classifier and the Decision Tree Classifier, two different hyper-parameter tuning techniques were employed to optimize each model's performance. For the Random Forest Classifier, the best parameters identified using the Grid Search Cross-Validation (GSCV) technique were:

- **Criterion:** Gini
- **max_samples:** None
- **min_samples_leaf:** 3
- **n_estimators:** 40

while those identified using the Random Search Cross-Validation (RSCV) technique were

- **max_depth:** 71
- **n_estimators:** 349

Similarly, for the Decision Tree Classifier, the optimal parameters determined through Grid Search Cross Validation (GSCV) were:

- **Criterion:** Entropy
- **max_depth:** None
- **min_samples_leaf:** 20
- **min_samples_split:** 2

and those identified with Random Search Cross Validation (RSCV) were:

- **max_depth:** 13
- **min_samples_leaf:** 3
- **min_samples_split:** 7

As evident from Table 5.1, the top-performing model based in the test set for transportation classification was the Random Forest Classifier using the Grid Search Cross-Validation technique, achieving an accuracy of 0.8218. This accuracy outperforms other machine learning models, such as the Artificial Neural Network which reached an accuracy of 0.8006 and Decision Tree, with both hyper-parameter tuning approaches which reached a maximum accuracy of 0.8097. With respect to the hyperparameter fitting techniques, the technique that stood out was the Grid Search Cross-Validation technique, which in both the Random Forest and Decision Tree models obtained values of 0.8218 and 0.8097, respectively. Finally, the model that achieved the lowest performance was the Decision Tree using Random Search Cross-Validation technique, which barely achieved an accuracy of 0.7779.

Table 5.1: Performance based on accuracy.

Model	**Accuracy**
Random Forest Classifier using GSCV	**0.8218**
Random Forest Classifier using RSCV	0.8142
Decision Tree Classifier using GSCV	0.8097
Decision Tree Classifier using RSCV	0.7779
Artificial Neural Network	0.8006

Analysis on validation set

With the best model, Random Forest Classifier, utilizing the hyperparameters *criterion=Gini*, *max_samples=Nome*, *min_samples_leaf=3* and *n_estimators = 40*, we applied the model to the validation set. Figure 5.1 displays the confusion matrix of the model when making predictions using the validation set.

The confusion matrix illustrates that the model successfully identified the bus class in 202 instances (true positives) and the car class in 209 instances (true negatives). However,

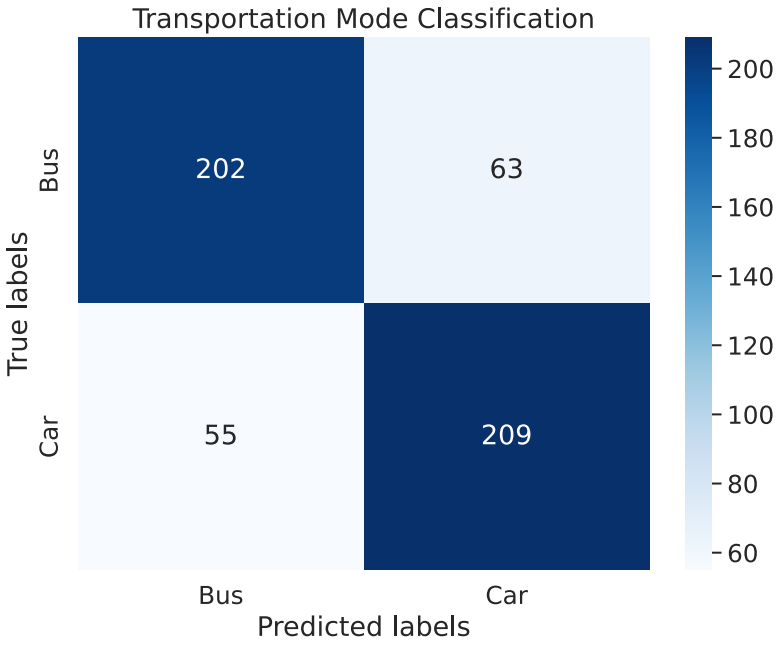

Figure 5.1: Confusion Matrix of Transportation Mode Classification.

the model also made errors by incorrectly labeling 63 car instances as buses (false positives) and 55 bus instances as cars (false negatives).

Table 5.2 shows the results of the metrics in both the validation and test sets. For the bus class, the model achieves somewhat poorer precision (0.7860) and F1-Score (0.7739) in the validation set than in the testing set (precision: 0.8403, F1-Score: 0.8168), but recall is pretty consistent in both sets. Similarly, the precision (0.7684) and F1-Score (0.7799) for the car class are comparable between the validation and testing sets, with the testing set outperforming the validation set marginally. On the other hand, the testing set has a little higher recall than the validation set (0.8489 vs. 0.7917).

The average precisions, recalls, and F1-Scores for all classes show that the model performed well in both the validation set (Average Precision: 0.7772, Average Recall: 0.7770, Average F1-Score: 0.7769) and the testing set (Average Precision: 0.8227, Average Recall: 0.8218, Average F1-Score: 0.8216). Finally, the model achieved an accuracy of 0.7769 in the validation set and 0.8218 in the testing set. This shows that the model has a strong ability to properly forecast the mode of transport, despite a slight difference in performance between the testing and validation data sets.

Table 5.2: Metrics in transportation mode classification.

Category	**Validation set**			**Testing set**		
	Precision	**Recall**	**F1-Score**	**Precision**	**Recall**	**F1-Score**
Bus	0.7860	0.7623	0.7739	0.8403	0.7946	0.8168
Car	0.7684	0.7917	0.7799	0.8052	0.8489	0.8265
Average	**0.7772**	**0.7770**	**0.7769**	**0.8227**	**0.8218**	**0.8216**
Accuracy	**0.7769**			**0.8218**		

The results of the Kolmogorov-Smirnov (KS) test indicate that the model performs well. In the first approach of the KS test, the value of the test statistic is 0.0151 and the p-value is 0.9999, since the p-value is not less than 0.05 (5% significance) thus we accept the null hypothesis and conclude that the distributions of the two samples of both the original values and the predicted values are the same. In the second approach, the test statistic value is 0.5539 and the p-value is 6.282×10^{-38}, indicating that the model in some part can distinguish between the two classes, which in this case are bus and car, based on the predicted values.

5.1.2 Motive of Displacement Classification

In this classification task, it is important to mention that the input variables are: Occupation, Age, Hours_since_midnight_1D, Marital_Status, Destination_1D, and License. In addition, the target variable is the motive of displacement in the first displacement (Motive_1D). When we used GSCV in the Random Forest Classifier we got the following best parameters:

- **Criterion:** Entropy
- **max_samples:** None
- **min_samples_leaf:** 3
- **n_estimators:** 100

On the other hand, we used RSCV and the best parameters were:

- **max_depth:** 95

- **n_estimators:** 292

In the case of Decision Tree Classifier, the optimal parameters when we used GSCV were:

- **Criterion:** Entropy
- **max_depth:** 300
- **min_samples_leaf:** 3
- **min_samples_split:** 10

and those identified with RSCV were:

- **max_depth:** 79
- **min_samples_leaf:** 1
- **min_samples_split:** 6

Based on the results in Table 5.3, we can infer that the Random Forest Classifier employing the RSCV approach has the highest accuracy, with a value of 0.7844. This model performs substantially better than the other models tested. The Random Forest with RSCV outperformed the Random Forest with GSCV, indicating that the randomized hyperparameter search was successful in identifying an ideal combination that resulted in higher accuracy. In the case of methods for hyperparameter fitting the Random Search Cross-Validation was the best technique since got the high accuracy in both methods Random Forest and Decision Tree classifier (0.7835 and 0.7094 respectively). In this classification experiment, tree-based models (Random Forest and Decision Tree) beat artificial neural networks. The Artificial Neural Network had the lowest accuracy (0.6752) of any of the models tested, which could indicate that in this situation, tree-based models are more effective at capturing the association between characteristics and motive of travel categories.

Table 5.3: Performance based on accuracy in motive of displacement classification.

Model	**Accuracy**
Random Forest Classifier using GSCV	0.7455
Random Forest Classifier using RSCV	**0.7844**
Decision Tree Classifier using GSCV	0.6895
Decision Tree Classifier using RSCV	0.7094
Artificial Neural Network	0.6752

Analysis on validation set

Taking the best model, Random Forest Classifier, utilizing the hyperparameters *max_depth=95* and *n_estimators = 292*, we applied the model to the validation set. Figure 5.2 displays the confusion matrix of the model in the motive of displacement classification.

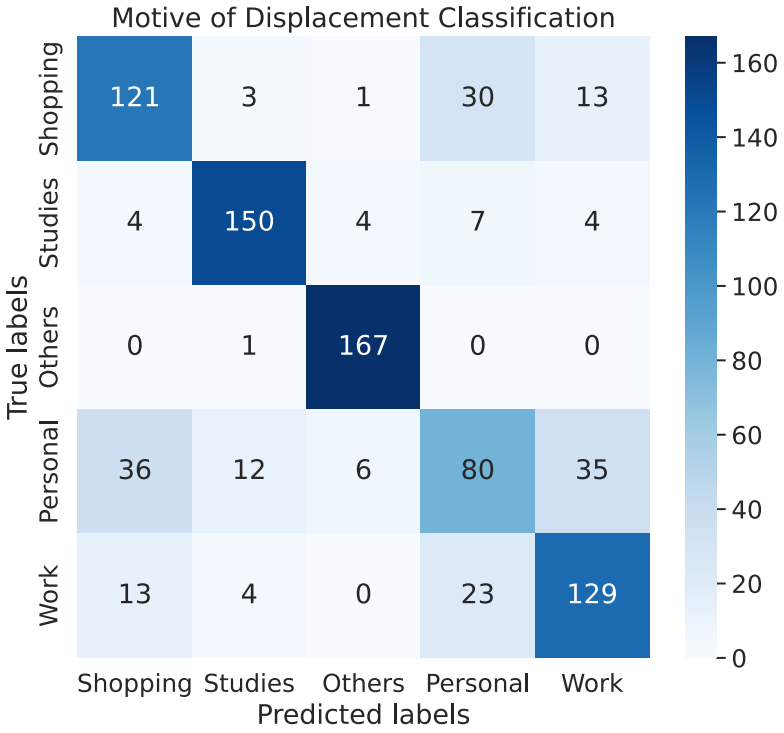

Figure 5.2: Confusion Matrix of Motive of Displacement Classification.

Based on the confusion matrix shown in Figure 5.2, the model correctly categorized 121 cases as ”Shopping”. However, there were several misclassifications, with three cases labeled as ”Studies,” one as ”Others,” 30 as ”Personal,” and 13 as ”Work”. Furthermore, the majority of occurrences labeled as ”Studies” were correctly predicted (150), but there were some inaccuracies, including 4 instances misclassified as ”Shopping”, 4 as ”Others”, 7 as ”Personal”, and 4 as ”Work”.

In the category ’Others’ the model correctly classified 167 instances and made only one incorrect classification. The model confused a instance that belonged to the ’Others’ class with the ’Studies’ class. This result was obtained by resolving the imbalance in this category using the Random Over Sampling technique, which produced more samples to balance this specific category. Although the majority of occurrences were correctly classified as ”Personal” (80), there were some inaccuracies, including 36 instances misclassified as ”Shopping”, 12 as ”Studies”, 6 as ”Others”, and 35 as ”Work”.

The majority of cases labeled ”Work” were properly classified (129). However, several inaccuracies were discovered, with 13 instances misclassified as ”Shopping”, 4 as ”Studies”, and 223 as ”Personal”. Overall, the model performs well in classifying the ”Other”, ”Studies”, and ”Work” categories, making a large number of right predictions. Nonetheless, more inaccuracies are visible in the ”Shopping” and ”Personal” categories, where some instances

were wrongly classified as other categories.

Table 5.4 displays the metrics in both the validation and test sets. The 'Shopping' class performed better in the test set, particularly in the recall and F1-score, which were 0.8341 and 0.7770, respectively. The validation set had the lowest performance in terms of precision, with a percentage of 0.6954. In the 'Studies' class, the metrics are high in both sets, with the test set showing a minor loss in accuracy but an increase in recall.

In the 'Others' class, however, all three metrics outperformed in the validation set, particularly accuracy, which earned a score of 0.9382. The 'Personal' class has the lowest metrics in both sets, showing that the model is having trouble accurately detecting personal excursions. In the 'Work' class, the precision metric is higher in the test set, while the recall and F1-score metrics perform better in the validation set.

The averages show similar values in both the validation and test sets which range between 0.7600 and 0.7844 with the test set values being the highest. Finally, the accutacy is better in the test set than in the validation set with a value of 0.7675.

Table 5.4: Metrics in motive of displacement classification.

Category	**Validation set**			**Testing set**		
	Precision	**Recall**	**F1-Score**	**Precision**	**Recall**	**F1-Score**
Shopping	0.6954	0.7202	0.7076	0.7273	0.8341	0.7770
Studies	0.8824	0.8876	0.8850	0.8527	0.9095	0.8802
Others	0.9382	0.9940	0.9653	0.9167	0.9905	0.9522
Personal	0.5714	0.4734	0.5178	0.6369	0.5403	0.5846
Work	0.7127	0.7633	0.7371	0.7556	0.6476	0.6974
Average	**0.7600**	**0.7677**	**0.7626**	**0.7778**	**0.7844**	**0.7783**
Accuracy	**0.7675**			**0.7844**		

Finally, considering the Kolmogorov-Smirnov (KS), the model exhibits commendable performance. In the first approach, the test statistic value is 0.0201, and the p-value is 0.9955. Since the p-value is not less than 0.05 (5% significance), we can accept the null hypothesis and conclude that both the original sample and the predicted sample share the same distribution. In the second approach, the average test statistic value is 0.7678, and

the average p-value is 3.9177×10^{-32}. Given the high test statistic value and very small p-value, we can conclude that the model effectively distinguishes between the 5 categories it classifies, based on the predicted class set.

5.1.3 Destination Classification

For this specific task, the input variables were: Duration_1D, Motive_1D, Origin_1D, Hours_since_midnight_1D, Occupation, and Age. The target variable is the destination of people on their first displacement (Destination_1D). When we used the GSCV technique in the Random Forest classifier the following best parameters were obtained:

- **Criterion:** Entropy
- **max_samples:** None
- **min_samples_leaf:** 3
- **n_estimators:** 100

on the other hand, when we used RSCV the next best parameters were obtained:

- **max_depth:** 39
- **n_estimators:** 457

Taking into account the Decision Tree Classifier, when we employed the GSCV technique we obtained the following optimal parameters:

- **Criterion:** Entropy
- **max_depth:** 200
- **min_samples_leaf:** 3
- **min_samples_split:** 5

and in the case when we used RSCV technique the optimal parameters were:

- **max_depth:** 23
- **min_samples_leaf:** 1
- **min_samples_split:** 5

The results presented in Table 5.5 show that the Random Forest Classifier models using RSCV and Decision Tree Classifier using RSCV obtained the best accuracies, with 0.8886 and 0.8623 respectively. These models outperformed the Decision Tree Classifier using GSCV, which achieved 0.8076 and Random Forest Classifier using GSCV, which achieved 0.8491. The Artificial Neural Network had an accuracy of 0.8215. In terms of performance, the Random Forest Classifier with RSCV stands out as the best model, outperforming the others in terms of accuracy. This suggests that the automatic hyperparameter tuning

capability provided by Randomized Search Cross Validation could be crucial in improving model accuracy in this particular case, allowing for better model performance in this classification task.

Table 5.5: Performance based on accuracy in destination classification.

Model	**Accuracy**
Random Forest Classifier using GSCV	0.8491
Random Forest Classifier using RSCV	**0.8886**
Decision Tree Classifier using GSCV	0.8076
Decision Tree Classifier using RSCV	0.8623
Artificial Neural Network	0.8215

Analysis on validation set

Now we have the best model for the destination classification task which is the Random Forest Classifier with the hyperparameters *max_depth=39* and *n_estimators = 457* we can use the validation set to see the performance of the model. In Figure 5.3 we can find the confusion matrix of the model for this specific classification task.

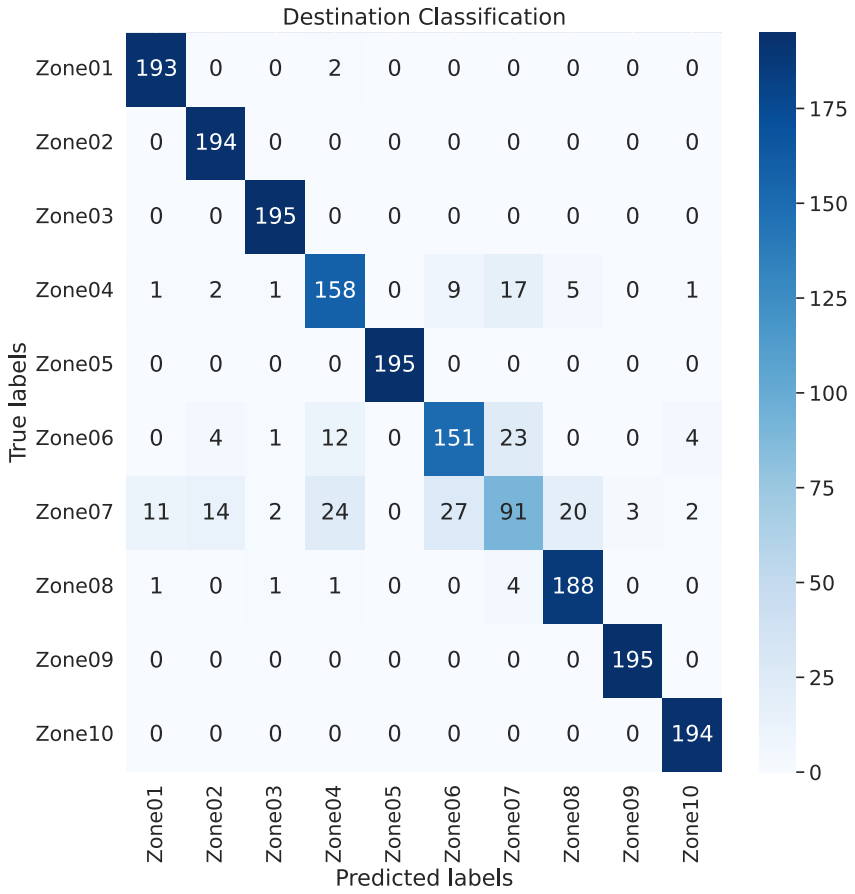

Figure 5.3: Confusion Matrix of Destination Classification.

In the confusion matrix presented in Figure 5.3, several important insights can be gleaned. Firstly, the majority of predictions align with the main diagonal, indicating high precision in classification for most categories (zones 1, 2, 3, 5, 8, 9, and 10). Zones 4, 6, and 7 display a notable number of incorrect predictions compared to other zones, evident from values off the main diagonal. This suggests these zones may pose greater challenges for accurate classification, likely due to similarities in travel patterns. Zone 8 shows a lower number of incorrect predictions compared to zones with a significant number of errors. The confusion matrix highlights the model's ability to distinguish among most categories while pinpointing areas where adjustments or enhancements to model precision may be warranted.

Based on the results presented in Table 5.6, it can be concluded that the model exhibits an overall accuracy of 0.9013 in the validation set and accuracy of 0.8886 in the test set, indicating strong predictive capability across most categories in the two sets. The precision rates are notably high across the majority of areas in the two sets, surpassing 90% for most categories, signifying a substantial proportion of correct positive predictions relative to all positive outcomes. However, we can see that the results in Zone 7 are very low compared to the others in both the validation set and the test set. The metric that obtained the lowest value was recall which obtained values of 0.4691 and 0.4631 for both the validation and test set respectively. This fact indicates that the model is not capturing many of the true positive examples. The recall values are high in most classes exceeding 0.90 in both

the validation and test sets with the exception of Zone 04 and Zone 06 where the values are between 0.80 and 0.70. The The F1-score remains consistently high in most areas, demonstrating a commendable balance between model correctness and completeness. In summary, the Random Forest Classifier with RandomizedSearchCV demonstrates a robust overall performance, with opportunities for enhancement primarily identified in the metrics for Zone 07.

Table 5.6: Metrics in destination classification.

Category	**Validation set**			**Testing set**		
	Precision	**Recall**	**F1-Score**	**Precision**	**Recall**	**F1-Score**
Zone 01	0.9369	0.9897	0.9626	0.9486	0.9877	0.9677
Zone 02	0.9065	1.0000	0.9510	0.9382	1.0000	0.9681
Zone 03	0.9750	1.0000	0.9873	0.9643	1.0000	0.9818
Zone 04	0.8020	0.8144	0.8082	0.7550	0.7705	0.7627
Zone 05	1.0000	1.0000	1.0000	0.9918	1.0000	0.9959
Zone 06	0.8075	0.7744	0.7906	0.7325	0.7325	0.7325
Zone 07	0.6741	0.4691	0.5532	0.6243	0.4631	0.5318
Zone 08	0.8826	0.9641	0.9216	0.8937	0.9342	0.9135
Zone 09	0.9848	1.0000	0.9924	0.9878	1.0000	0.9939
Zone 10	0.9652	1.0000	0.9823	0.9720	1.0000	0.9858
Average	**0.8935**	**0.9012**	**0.8949**	**0.8808**	**0.8888**	**0.8834**
Accuracy	**0.9013**			**0.8886**		

Finally, regarding the Kolmogorov-Smirnov (KS), in the first approach, the test statistic value is 0.0200 with an associated p-value of 0.8294. Since the p-value is not less than

0.05 (5% significance), we can accept the null hypothesis and conclude that both the sample containing the original data and the sample with the predicted data share the same distribution. In the second approach, the average test statistic value is 0.9433, and the average p-value is 9.2329×10^{-126}. With a high average test statistic value (close to 1) and a very small average p-value, we can affirm that the model effectively discriminates among all predicted categories corresponding to people's destinations.

5.1.4 Start Time of Displacement Classification

In this case, the input variables were: Motive_1D, Occupation, Age, License, Destination_1D, and Vehicle. The target variable is the start time (Hours_since_midnight_1D), which was pretreated before being used in the models. When we used the Random Forest Classifier with GSCV we got the following optimal parameters:

- **Criterion:** Gini
- **max_samples:** None
- **min_samples_leaf:** 3
- **n_estimators:** 100

when we used RSCV technique the optimal parameters were:

- **max_depth:** 91
- **n_estimators:** 252

On the contrary, when employing the Decision Tree Classifier in conjunction with GSCV technique, the resulting optimal parameters were identified as follows:

- **Criterion:** Gini
- **max_depth:** 100
- **min_samples_leaf:** 3
- **min_samples_split:** 5

and the optimal parameters using RSCV technique were:

- **max_depth:** 64
- **min_samples_leaf:** 1
- **min_samples_split:** 3

The results obtained (Table 5.7) show that the Random Forest Classifier with the RSCV technique was the best model in terms of accuracy, reaching a value of 0.8010. This model outperformed its similar model using the GSCV technique, which only achieved an accuracy of 0.7548. On the other hand, the Decision Tree Classifier that obtained the best accuracy using the RSCV technique reached a value of 0.7893. The Artificial Neural

Network had the lowest performance among the models tested, with an accuracy of 0.7194. Although neural networks are powerful for certain problems, in this specific case, they could be outperformed by decision tree-based models.

Table 5.7: Performance based on accuracy in start time of displacement classification

Model	**Accuracy**
Random Forest Classifier using GSCV	0.7548
Random Forest Classifier using RSCV	**0.8010**
Decision Tree Classifier using GSCV	0.7450
Decision Tree Classifier using RSCV	0.7983
Artificial Neural Network	0.7194

Analysis on validation set

The best model for this specific task is a Random Forest Classifier using the RSCV technique. With this model we can test the original unbalanced dataset. In Figure 5.4 we can find the confusion matrix of the model for this specific classification task.

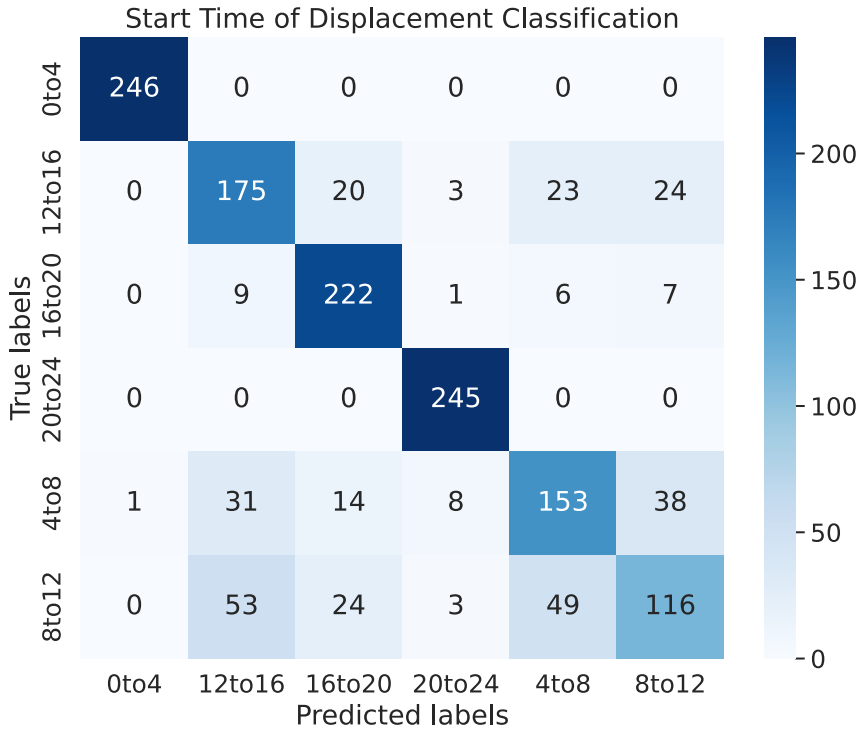

Figure 5.4: Confusion Matrix of Start Time of Displacement Classification.

Using the confusion matrix findings, we can see that the model obtained perfect classification accuracy for the time ranges 0 to 4 hours and 20 to 24 hours, with no misclas-

sifications. The model performed well in the 12 to 16 hour range, accurately identifying the majority of instances (175); but, it occasionally confused this category with the 16 to 20, 20 to 24, 4 to 8, and 8 to 12 categories. Similarly, for the 16 to 20 hour range, the model properly identified the majority of instances (222), but there were some errors, most notably mistaking this category with the 12 to 16 (9 instances) and 8 to 12 (7 instances) categories. There were classification errors between 4 and 8 hours, particularly occurrences that were wrongly classified as 12 to 16 and 8 to 12. Finally, in the 8 to 12 hour period, the model correctly identified 116 occurrences, despite mistakes in all other categories.

Based on the findings presented in Table 5.8, it is evident that the model achieved an accuracy of 0.7865 in the validation set and accuracy of 0.8010 in the validation set, indicating room for improvement. The precision varies across categories in the both sets, the higher values is observed in the class '0 to 4' in validation and test set where the values are 0.9960 and 0.9967 respectively. On the other hand lower values is observed in the class '8 to 12' whit values equal to 0.6270 and 0.6217 in the validation and test set respectively, suggesting that around 60% of positive predictions for this class were correct. Regarding recall, the highest values were achieved by categories 0 to 4 and 20 to 24, both scoring 1.00 in the two sets, demonstrating that the model accurately identified all instances within these classes. It is noteworthy that these categories were the most imbalanced, with additional records added during dataset balancing. Additionally, all categories exhibit an F1-Score exceeding 0.65, except for '8 to 12', which in the validation and test set displays a notably lower value (0.5395 and 0.5336 respectively).

Table 5.8: Metrics in start time of displacement classification.

Category	**Validation set**			**Testing set**		
	Precision	**Recall**	**F1-Score**	**Precision**	**Recall**	**F1-Score**
0 to 4	0.9960	1.0000	0.9980	0.9967	1.0000	0.9984
12 to 16	0.6530	0.7143	0.6823	0.7315	0.7745	0.7524
16 to 20	0.7929	0.9061	0.8457	0.8069	0.9121	0.8563
20 to 24	0.9423	1.0000	0.9703	0.9564	1.0000	0.9777
4 to 8	0.6623	0.6245	0.6429	0.6452	0.6515	0.6483
8 to 12	0.6270	0.4735	0.5395	0.6217	0.4673	0.5336
Average	**0.7789**	**0.7864**	**0.7798**	**0.7931**	**0.8009**	**0.7944**
Accuracy	**0.7865**			**0.8010**		

Finally, using the Kolmogorov-Smirnov (KS) in the first approach we obtained that the value of the test statistic was 0.0503 and the p-value was 0.0483. Because the p-value is less than 0.05 (5% significance) we reject the null hypothesis so it can be said that the distribution of the original data set and the predicted set are not equal. This may have occurred because in category 8 to 12 the model predicted fewer values (Figure 5.5). The original values for this category were 245 and the model could only predict 185 with a difference of 60. For this reason, we assume that the distribution of both the original and the predicted set do not have the same distribution.

On the other hand, in the second approach, we observe that the average value of the test statistic is 0.8092 and the average p-value is 6.2091×10^{-67}. This indicates that the model is capable of distinguishing between the 6-hour ranges present in this classification task during the generation of predictions, given the high average value of the test statistic and the very low average p-value.

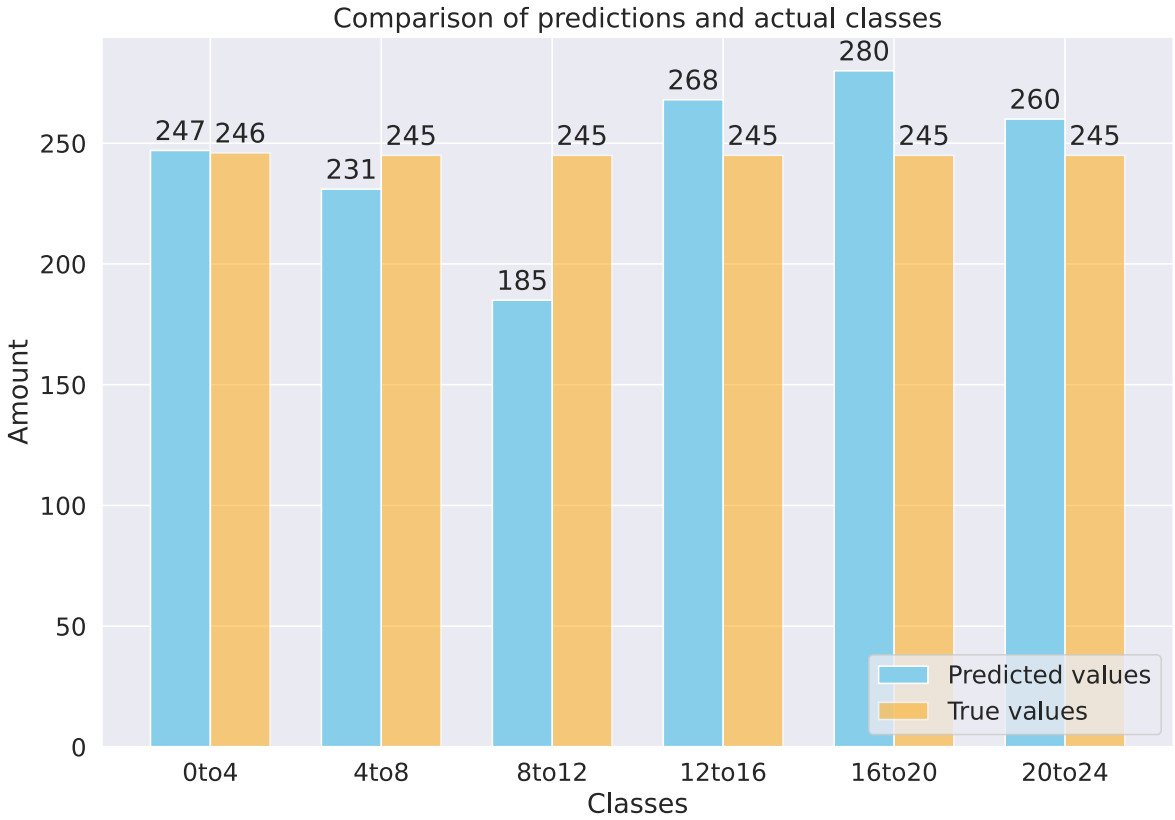

Figure 5.5: True values vs Predicted values.

5.1.5 Simulation

Original Dataset vs Predicted Dataset

In this section, we analyze the distributions of both the original and predicted datasets. The predicted dataset is the dataset utilized for the simulation, and specific variables within it were predicted using machine learning models. The distribution analysis concentrates

on the variables start time, transport mode, motive, and destination, as the origin and end time variables are derived directly from the synthetic dataset created from the original dataset. As a result, the variables end time and origin will be very similar to those found in the original dataset, making their study redundant.

Table 5.9: Distribution of the transportation mode.

Category	**Original Dataset**	**Predicted Dataset**	**Difference**
Bus	0.4920	0.5446	-0.0526
Car	0.5080	0.4554	0.05256

Table 5.9 displays the distributions of transportation modes in both the original and predicted datasets. The predicted dataset contains a higher fraction of 'Bus' than the original dataset. We can observe that the difference is around 5.26%. This shows that the machine learning model for this task tends to categorize more occurrences as 'Bus' than are present in the original dataset. Nevertheless, the predicted dataset has a lower proportion of 'Car' than the original dataset. Similarly, the difference is approximately 5.26%, showing that the model classifies less instances as 'Car' than the original data.

Finally, the discrepancies in both categories are equal but opposite, resulting in a total of zero. This signifies that the model is transferring a portion of the distribution from the 'Car' class to 'Bus'. Figure 5.6 shows the aforementioned.

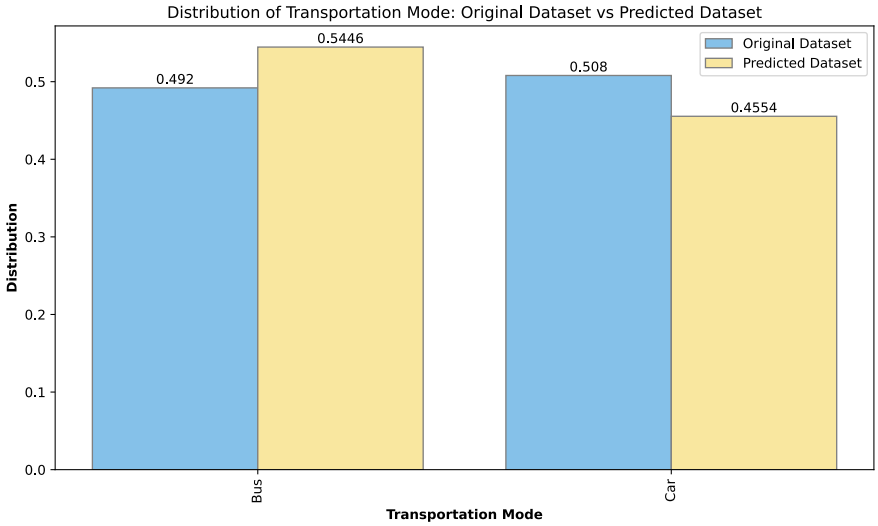

Figure 5.6: Distribution of Transportation Mode.

Table 5.10: Distribution of motive.

Category	**Original Dataset**	**Predicted Dataset**	**Difference**
Work	0.3236	0.3390	-0.0154
Personal	0.2787	0.3435	-0.0648
Study	0.1902	0.1104	0.0798
Shopping	0.1761	0.1792	-0.0031
Others	0.0313	0.0280	0.0033

The results of the examination of the distribution of categories between the original and predicted data sets, provided in Table 5.10, show reasonable accuracy in the majority of categories. The category "Work" has a slightly greater prediction than the original distribution, with a difference of -1.54%, suggesting high accuracy. "Shopping" also produces a forecast that is quite similar to the original distribution, with a minimum difference of -0.31%, exhibiting great accuracy. On the other side, the "Others" group has a slight underestimation of 0.33%, which is almost trivial.

However, there are some noticeable differences. The "Personal" category is greatly overstated in the prediction, with a -6.48% difference, indicating a tendency to overestimate this category. In contrast, the "Study" group is severely underrepresented, with a 7.98% differential. These findings suggest that, while the model performs well overall, specific changes are required to increase accuracy in the "Personal" and "Study" categories. Finally, the Figure 5.7 shows in a better way the result of distribution in the both datasets.

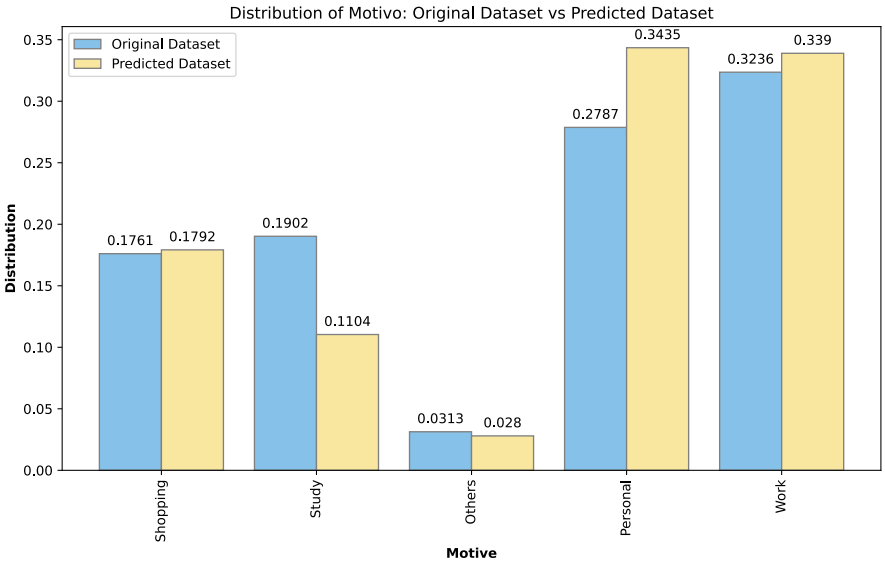

Figure 5.7: Distribution of Motive.

Table 5.11: Distribution of motive.

Category	**Original Dataset**	**Predicted Dataset**	**Difference**
Z01	0.0971	0.1142	-0.0171
Z11	0.0366	0.0343	0.0023
Z02	0.1955	0.2180	-0.0225
Z04	0.0814	0.0660	0.0154
Z03	0.0529	0.0388	0.0141
Z12	0.0246	0.0262	-0.0016
Z39	0.0845	0.0927	-0.0082
Z15	0.0270	0.0219	0.0051
Z51	0.0252	0.0315	-0.0063
Z10	0.0332	0.0272	0.0060

The distribution of destinations between the original and predicted datasets reveals some significant changes in numerous categories, as shown in Table 5.11. For example, categories Z01 and Z02 contain discrepancies of -0.0171 and -0.0225, indicating an underestimation in the actual data set when compared to the anticipated data set. This indicates that the prediction model overestimates the frequency of these motifs. Similarly, the Z39 and Z51 categories had slight underestimates of -0.0082 and -0.0063, respectively. These discrepancies, albeit little, might add up and have an impact on the prediction model's overall accuracy.

However, several categories show that the model predicts lower frequencies compared to the real data. For example, Z04 and Z03 show differences of 0.0154 and 0.0141, indicating that the model underestimates the frequency of these factors. The model has rather good accuracy in categories Z11, Z12, Z15, and Z10, as indicated by lower differences. Although the prediction model appears to function well in most categories, significant differences in some indicate need for improvement, particularly in categories Z01, Z02, Z04, and Z03. The Figure 5.8 displays the distribution of destination in the original and predicted dataset.

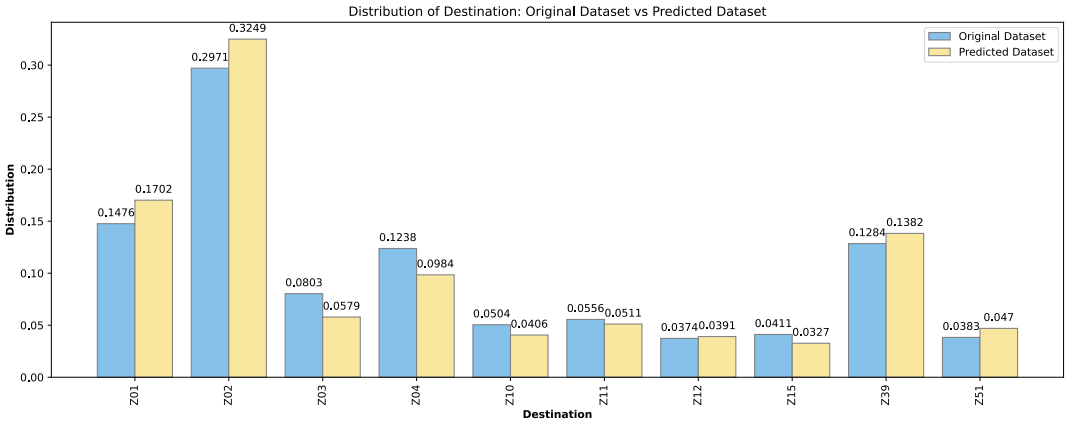

Figure 5.8: Distribution of Destination.

A comparison of the distributions in the original and predicted datasets reveals some notable differences. The histogram of the original data, shown in blue, looks to have a wider and possibly more uniform distribution than the histogram of the predicted data, displayed in red. Including KDE (Kernel Density Estimate) curves in both histograms provides a more refined perspective of the distributions in the two sets. For example, at 5 a.m., we can see that the original and predicted dataset distributions are identical. Figure 5.9 shows that the prediction model overestimates the frequency of the start time between 8 a.m. and the original data.

From 8 a.m. to shortly after 10 a.m., we can see that the distributions in both datasets are identical. Finally, around 12 p.m., the projected set contains fewer records than the original dataset, resulting in a considerable reduction in the anticipated dataset's distribution. This trend continues for the rest of the time.

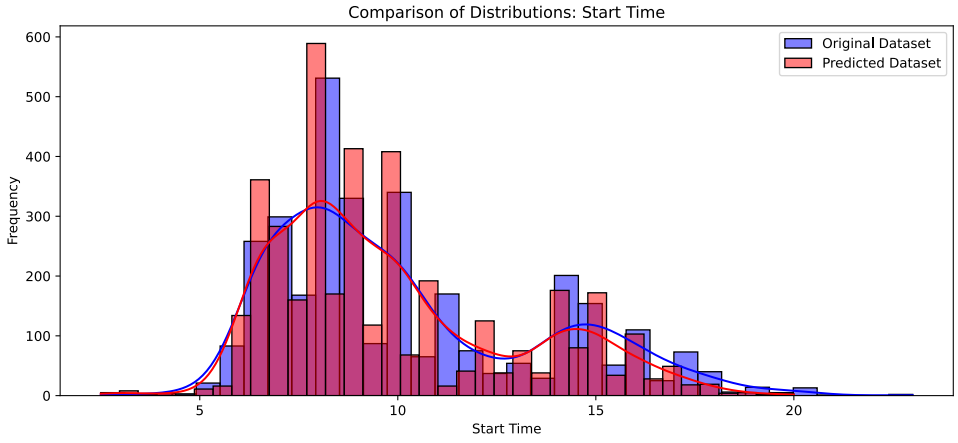

Figure 5.9: Distribution of Start Time.

Simulation result

Once the simulation dataset, which is our predicted dataset, has been analyzed, we proceed to execute the simulation first filtering the dataset for records that contain cars as a mode of transportation, as described in the methodology. When the simulation completed, several files were created in the output directory. The most straightforward approach to visualize MATSim results is to utilize VIA [91]. Then you can drag and drop files into this section (such as network.xml or events.xml.gz). Finally, the visualizer understands our data, and we can instruct it on how to visualize it. Figure 5.10 shows the trajectory of the vehicles in the simulation at approximately 7:30 and 8:00 am.

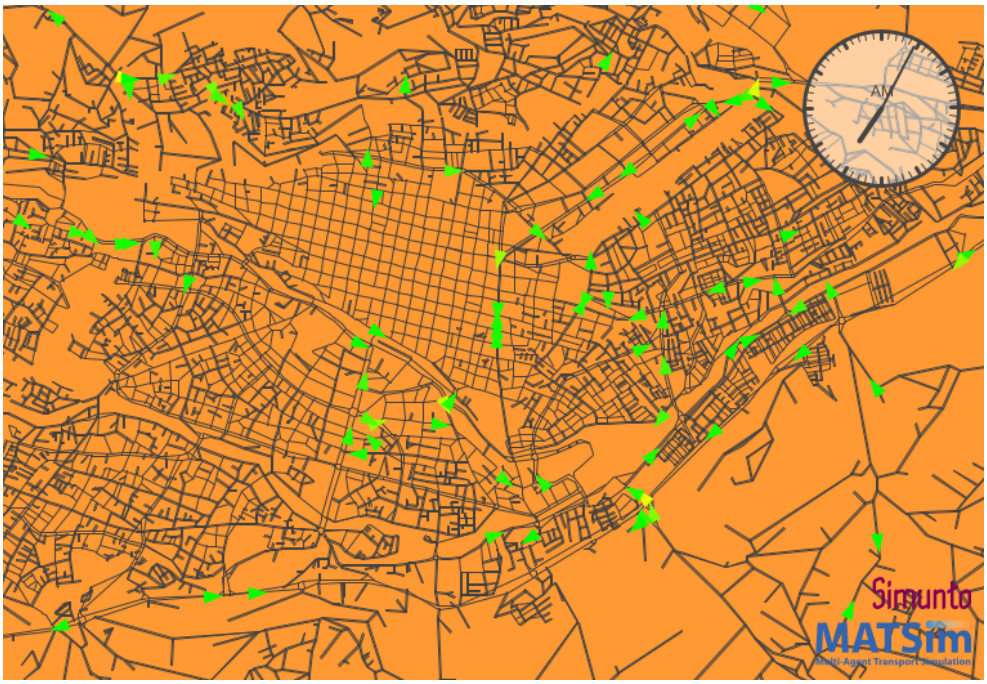

Figure 5.10: Trajectory of simulation vehicles.

Figure 5.11 represents the histogram of the distribution of vehicle movement during the simulation, with red vehicle leaving, green vehicle on route, and blue vehicle arriving. As can be observed, the red frequencies closely mirror the distribution of the actual dataset's start time, indicating that the predicted dataset built using machine learning models is good. However, it is significant to note that the peak at 8 a.m. in the histogram most closely resembles the distribution of the original dataset's start time; this period shows when people leave home more frequently to undertake their respective activities. In terms of the other hours, those before 8 a.m. reflect the distribution in the original dataset since they tend to grow, as indicated in the histogram; however, those after 8 a.m. do not resemble the distribution of the original dataset.

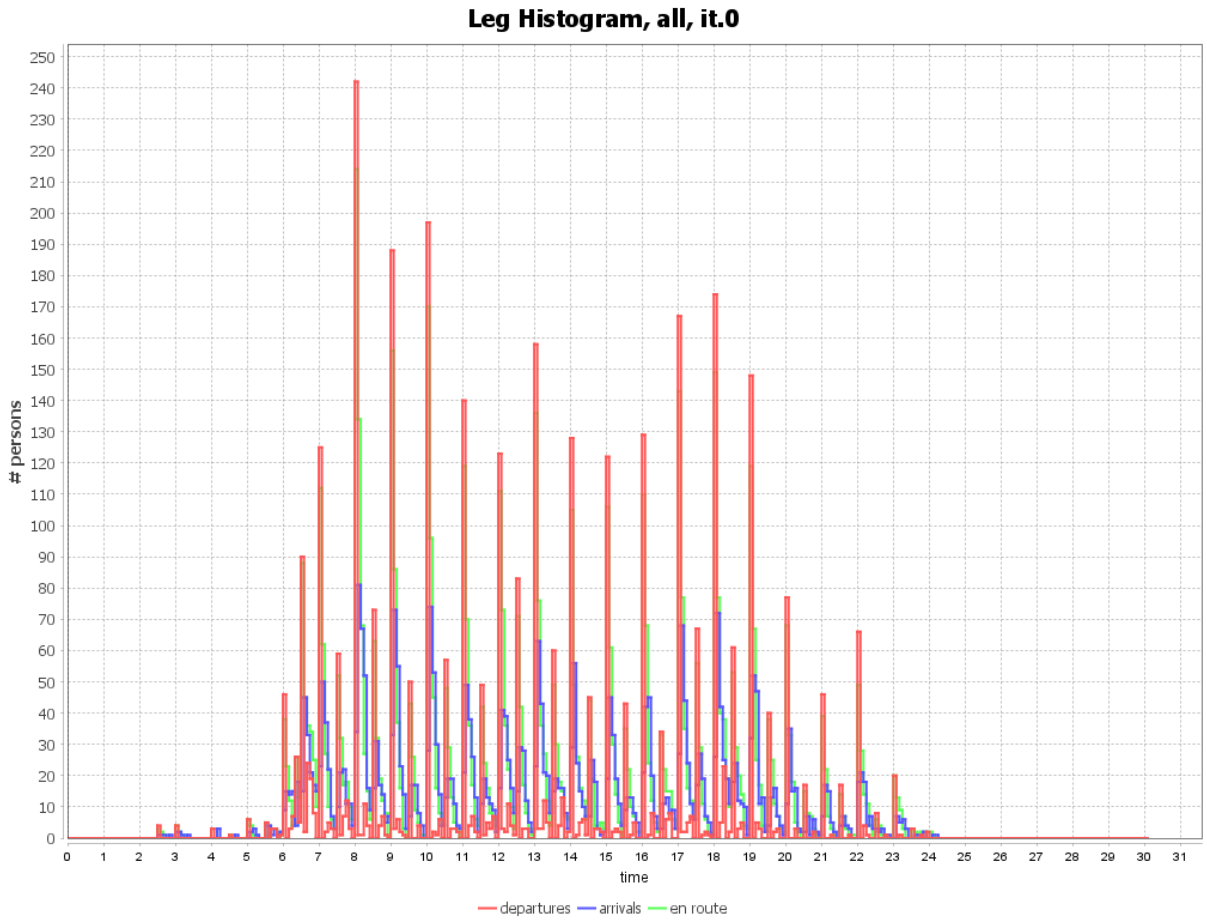

Figure 5.11: Histogram of the distribution of the simulation movement.

Chapter 6

Conclusions

The present work focused on a general objective: the generation of activity patterns by means of machine learning techniques to be used in transportation demand models. The fulfillment of specific objectives, the results obtained and their respective analysis provide useful information to support this objective.

We get a mobility survey dataset [72] from Cuenca city. Having an initial dataset that did not have the conditions to be used in the machine learning models, the first step was to prepare, filter, and adjust the dataset before being used in the machine learning models. The dataset went through a certain number of steps described in the methodology so that our final dataset has the necessary conditions to be used with the machine learning models.

The analysis of the literature revealed a trend toward the use of machine learning models for the generation of activity patterns, since at the beginning only traditional models were used, but with time they presented certain limitations. With the advance of technology, more modern models were used, such as the automatic learning models, among which the Random Forest and Decision Tree models stand out. As a result of this analysis, it was decided to use these models and also include a third model which is the Artificial Neural Networks to see the performance of this model in this type of task.

Four classification tasks were applied, which were performed for the following variables: means of transportation, motive of displacement, destination, and start time of the displacement. To improve the performance of the Radom Forest and Decision Tree models, hyperparameter adjustment techniques such as Grid Search Cross-Validation and Random Search Cross-Validation were used. In addition, when presenting a considerable amount of variables for each type of classification, the feature selection method was used to keep only the most important variables.

In the case of the transportation classification task, all three machine learning models were adequately employed. In this task, the categories of the target variable were mostly balanced which led to very relevant results. The model that stood out the most in this task was the Random Forest using the GSCV technique, however when analyzing the model using the validation set it performed below the performance with the test set. However, the Smirnov Kolmogorov test in its two approaches confirms that the model performed well in this task.

In the case of the displacement motive classification task, the dataset was unbalanced because the Radom Over Sampling technique had to be applied to achieve a balance in

all classes of the target variable. After this, the three machine learning models were applied. In this case, the model that stood out was the Radom Forest, but unlike the previous classification task, here the best hyperparameter adjustment technique was RSCV. When comparing the validation and test set, we were able to observe that the average of the metrics was similar, being in an interval between 0.7600 and 0.7844. Similarly, the Smirnov Kolmogorov test indicated that the model can to some extent distinguish between the 5 classes during classification and also this test statistic helps to conclude that the distribution of the original sample and the predicted sample are the same.

In the task of destination classification, it was observed that the dataset was very unbalanced, so after balancing the dataset, its size was increased, reaching a size of 12160 records. In addition, in this task, it was necessary to use the K-Means algorithm to group the destinations into 10 clusters based on their centroid, since this variable had about 50 classes. When applying the three machine learning models, the best one was Random Forest using the RSCV. In this task at the moment of the comparison with the validation and test set, it was obtained that the best model had a better performance in the validation set reaching an accuracy of 90%. Similarly, the test statistic supports the conclusion that the best model can distinguish between the 10 categories and that the original and predicted samples are equal.

Concerning the task of classifying the start time, it was necessary to group the target variable by intervals of hours. However, when grouping by intervals, these intervals were unbalanced, so, as in the previous classification tasks, the Random Over Sampling technique was used to balance the dataset. After applying machine learning models, the one that stood out the most was the Random Forest in the same way using the RSCV technique. When comparing the results between the validation and test sets, we can conclude that the values of the metrics do not have much variation in both sets. These values are in the range between 0.7789 and 0.8009. The test statistic indicates that the model can distinguish between the existing classes, but the distribution of the original and predicted samples are not the same, suggesting that the model can be improved.

In general terms comparing the models of the four classification tasks, we can conclude that the model that stood out the most was Random Forest which takes advantage of the nature of its operation to stand out among the other models. On the other hand, concerning the hyperparameter fitting techniques, the one that performed the best was Random Search Cross-Validation.

Finally, to give an application to the models, we built a dataset that will be used to perform a simulation using the MATSim tool. The best models for each classification task were used to build the dataset. When analyzing the distributions between the original dataset and the simulation dataset (predicted dataset) in three of the predicted variables the differences between the original and predicted distributions are minimal except for the classification task at the start time. For example, in this case, the predicted distribution contains significant peaks or valleys that are not reflected in the original data, which means that the model is not sufficiently capturing some patterns in the actual data set. Discrepancies between distributions, particularly those with a higher frequency than the other, may suggest that the model needs to be changed or additional features added to increase accuracy. Finally, the observed differences suggest that the prediction model might be improved to better resemble the original data set's start time distribution. The simulation results show that, despite the fact that the model for start time prediction does

not capture the start time ranges adequately, the results are not significantly different from the original dataset.

6.1 Future works

With a limited data set, machine learning models could be used to generate new and realistic data from a training data set such as Generative Antagonistic Networks (GANs). On the other hand, by having an unbalanced dataset for the 4 classification tasks, different data balancing techniques could be explored to obtain more consistent results. Finally, conduct the same study but in another area of Ecuador where the information could be accessed.

Bibliography

[1] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg *et al.*, "Scikit-learn: Machine learning in python," *the Journal of machine Learning research*, vol. 12, pp. 2825–2830, 2011.

[2] B. Charbuty and A. Abdulazeez, "Classification based on decision tree algorithm for machine learning," *Journal of Applied Science and Technology Trends*, vol. 2, no. 01, pp. 20–28, 2021.

[3] X. Zhao, X. Yan, A. Yu, and P. Van Hentenryck, "Prediction and behavioral analysis of travel mode choice: A comparison of machine learning and logit models," *Travel behaviour and society*, vol. 20, pp. 22–35, 2020.

[4] C. Song, Z. Qu, N. Blumm, and A.-L. Barabási, "Limits of predictability in human mobility," *Science*, vol. 327, no. 5968, pp. 1018–1021, 2010.

[5] J. D. Pineda-Jaramillo, "A review of machine learning (ml) algorithms used for modeling travel mode choice," *Dyna*, vol. 86, no. 211, pp. 32–41, 2019.

[6] Y. Zheng, "Trajectory data mining: an overview," *ACM Transactions on Intelligent Systems and Technology (TIST)*, vol. 6, no. 3, pp. 1–41, 2015.

[7] L. Cheng, X. Chen, J. De Vos, X. Lai, and F. Witlox, "Applying a random forest method approach to model travel mode choice behavior," *Travel behaviour and society*, vol. 14, pp. 1–10, 2019.

[8] R. Santhiya and C. GeethaPriya, "Machine learning techniques for intelligent transportation systems-an overview," in *2021 12th International Conference on Computing Communication and Networking Technologies (ICCCNT)*. IEEE, 2021, pp. 1–7.

[9] E. J. Miller, "Travel demand models, the next generation: Boldly going where no-one has gone before," in *Mapping the Travel Behavior Genome*. Elsevier, 2020, pp. 29–46.

[10] E. Miller, "Modeling the demand for new transportation services and technologies," *Transportation Research Record: Journal of the Transportation Research Board*, vol. 2658, pp. 1–7, 01 2017.

[11] W. Heyns and S. Van Jaarsveld, "Transportation modelling in practice: connecting basic theory to practice," *Transportation, Land Use and Integration: Perspectives for Developing Countries, WIT Transactions on State of the Art in Science and Engineering*, vol. 100, pp. 3–27, 2017.

[12] D. Wang and T. Cheng, "A spatio-temporal data model for activity-based transport demand modelling," *International Journal of Geographical Information Science*, vol. 15, no. 6, pp. 561–585, 2001.

[13] J. de Dios Ortúzar and L. G. Willumsen, *Modelling transport*. John wiley & sons, 2024.

[14] L. Adenaw and Q. Bachmeier, "Generating activity-based mobility plans from trip-based models and mobility surveys," *Applied Sciences*, vol. 12, no. 17, p. 8456, 2022.

[15] R. Moeckel, N. Kuehnel, C. Llorca, A. T. Moreno, and H. Rayaprolu, "Agent-based simulation to improve policy sensitivity of trip-based models," *Journal of Advanced Transportation*, vol. 2020, pp. 1–13, 2020.

[16] V. L. Bernardin Jr, T. Mansfield, B. Swanson, H. Sadrsadat, and S. Bindra, "Scenario modeling of autonomous vehicles with trip-based models," *Transportation Research Record*, vol. 2673, no. 10, pp. 261–270, 2019.

[17] C. R. Bhat and F. S. Koppelman, "Activity-based modeling of travel demand," in *Handbook of transportation Science*. Springer, 1999, pp. 35–61.

[18] S. Rasouli and H. Timmermans, "Activity-based models of travel demand: promises, progress and prospects," *International Journal of Urban Sciences*, vol. 18, no. 1, pp. 31–60, 2014.

[19] J. Huang, Y. Cui, L. Zhang, W. Tong, Y. Shi, Z. Liu *et al.*, "An overview of agent-based models for transport simulation and analysis," *Journal of Advanced Transportation*, vol. 2022, 2022.

[20] A. L. Bazzan and F. Klügl, "A review on agent-based technology for traffic and transportation," *The Knowledge Engineering Review*, vol. 29, no. 3, pp. 375–403, 2014.

[21] S. Algers, J. Eliasson, and L.-G. Mattsson, "Activity-based model development to support transport planning in the stockholm region," 2001.

[22] K. W. Axhausen and T. Gärling, "Activity-based approaches to travel analysis: conceptual frameworks, models, and research problems," *Transport reviews*, vol. 12, no. 4, pp. 323–341, 1992.

[23] D. T. Phan and H. L. Vu, "A novel activity pattern generation incorporating deep learning for transport demand models," *arXiv preprint arXiv:2104.02278*, 2021.

[24] J. L. Bowman and M. E. Ben-Akiva, "Activity-based disaggregate travel demand model system with activity schedules," *Transportation research part a: policy and practice*, vol. 35, no. 1, pp. 1–28, 2001.

[25] G. Rebala, A. Ravi, and S. Churiwala, *An introduction to machine learning*. Springer, 2019.

[26] B. Mahesh, "Machine learning algorithms-a review," *International Journal of Science and Research (IJSR).[Internet]*, vol. 9, no. 1, pp. 381–386, 2020.

[27] L. Zhu, F. R. Yu, Y. Wang, B. Ning, and T. Tang, "Big data analytics in intelligent transportation systems: A survey," *IEEE Transactions on Intelligent Transportation Systems*, vol. 20, no. 1, pp. 383–398, 2018.

[28] Y. Lv, Y. Duan, W. Kang, Z. Li, and F.-Y. Wang, "Traffic flow prediction with big data: A deep learning approach," *Ieee transactions on intelligent transportation systems*, vol. 16, no. 2, pp. 865–873, 2014.

[29] X. Ma, Z. Tao, Y. Wang, H. Yu, and Y. Wang, "Long short-term memory neural network for traffic speed prediction using remote microwave sensor data," *Transportation Research Part C: Emerging Technologies*, vol. 54, pp. 187–197, 2015.

[30] M. Bear, B. Connors, and M. Paradiso, *Neuroscience: Exploring the Brain*. Wolters Kluwer, 2016. [Online]. Available: https://books.google.com.ec/books?id=vVz4oAEACAAJ

[31] T. Macpherson, A. Churchland, T. Sejnowski, J. DiCarlo, Y. Kamitani, H. Takahashi, and T. Hikida, "Natural and artificial intelligence: A brief introduction to the interplay between ai and neuroscience research," *Neural Networks*, vol. 144, pp. 603–613, 2021.

[32] O. I. Abiodun, A. Jantan, A. E. Omolara, K. V. Dada, N. A. Mohamed, and H. Arshad, "State-of-the-art in artificial neural network applications: A survey," *Heliyon*, vol. 4, no. 11, 2018.

[33] J. Singh and R. Banerjee, "A study on single and multi-layer perceptron neural network," in *2019 3rd International Conference on Computing Methodologies and Communication (ICCMC)*. IEEE, 2019, pp. 35–40.

[34] A. Thakur and A. Konde, "Fundamentals of neural networks," *International Journal for Research in Applied Science and Engineering Technology*, vol. 9, no. VIII, pp. 407–426, 2021.

[35] A. A. Heidari, H. Faris, S. Mirjalili, I. Aljarah, and M. Mafarja, "Ant lion optimizer: theory, literature review, and application in multi-layer perceptron neural networks," *Nature-Inspired Optimizers: Theories, Literature Reviews and Applications*, pp. 23–46, 2020.

[36] K. Han and Y. Wang, "A review of artificial neural network techniques for environmental issues prediction," *Journal of Thermal Analysis and Calorimetry*, vol. 145, no. 4, pp. 2191–2207, 2021.

[37] M. Uzair and N. Jamil, "Effects of hidden layers on the efficiency of neural networks," in *2020 IEEE 23rd international multitopic conference (INMIC)*. IEEE, 2020, pp. 1–6.

[38] D. Choi, C. J. Shallue, Z. Nado, J. Lee, C. J. Maddison, and G. E. Dahl, "On empirical comparisons of optimizers for deep learning," *arXiv preprint arXiv:1910.05446*, 2019.

[39] S. Sun, Z. Cao, H. Zhu, and J. Zhao, "A survey of optimization methods from a machine learning perspective," *IEEE transactions on cybernetics*, vol. 50, no. 8, pp. 3668–3681, 2019.

[40] S. Bock and M. Weiß, "A proof of local convergence for the adam optimizer," in *2019 international joint conference on neural networks (IJCNN)*. IEEE, 2019, pp. 1–8.

[41] K. Rai, M. S. Devi, and A. Guleria, "Decision tree based algorithm for intrusion detection," *International Journal of Advanced Networking and Applications*, vol. 7, no. 4, p. 2828, 2016.

[42] Priyanka and D. Kumar, "Decision tree classifier: a detailed survey," *International Journal of Information and Decision Sciences*, vol. 12, no. 3, pp. 246–269, 2020.

[43] S. Lee, C. Lee, K. G. Mun, and D. Kim, "Decision tree algorithm considering distances between classes," *IEEE Access*, vol. 10, pp. 69 750–69 756, 2022.

[44] M. Somvanshi, P. Chavan, S. Tambade, and S. Shinde, "A review of machine learning techniques using decision tree and support vector machine," in *2016 international conference on computing communication control and automation (ICCUBEA)*. IEEE, 2016, pp. 1–7.

[45] H. H. Patel and P. Prajapati, "Study and analysis of decision tree based classification algorithms," *International Journal of Computer Sciences and Engineering*, vol. 6, no. 10, pp. 74–78, 2018.

[46] L. Breiman, *Classification and regression trees*. Routledge, 2017.

[47] H. Sharma, S. Kumar *et al.*, "A survey on decision tree algorithms of classification in data mining," *International Journal of Science and Research (IJSR)*, vol. 5, no. 4, pp. 2094–2097, 2016.

[48] Z. Çetinkaya and F. Horasan, "Decision trees in large data sets," *International Journal of Engineering Research and Development*, vol. 13, no. 1, pp. 140–151, 2021.

[49] M. Schonlau and R. Y. Zou, "The random forest algorithm for statistical learning," *The Stata Journal*, vol. 20, no. 1, pp. 3–29, 2020.

[50] A. Parmar, R. Katariya, and V. Patel, "A review on random forest: An ensemble classifier," in *International conference on intelligent data communication technologies and internet of things (ICICI) 2018*. Springer, 2019, pp. 758–763.

[51] X. Dong, Z. Yu, W. Cao, Y. Shi, and Q. Ma, "A survey on ensemble learning," *Frontiers of Computer Science*, vol. 14, pp. 241–258, 2020.

[52] S. F. Mokhtar, Z. M. Yusof, and H. Sapiri, "Confidence intervals by bootstrapping approach: a significance review," *Malaysian Journal of Fundamental and Applied Sciences*, vol. 19, no. 1, pp. 30–42, 2023.

[53] Z. P. Brodeur, J. D. Herman, and S. Steinschneider, "Bootstrap aggregation and cross-validation methods to reduce overfitting in reservoir control policy search," *Water Resources Research*, vol. 56, no. 8, p. e2020WR027184, 2020.

[54] J. Zhou, A. H. Gandomi, F. Chen, and A. Holzinger, "Evaluating the quality of machine learning explanations: A survey on methods and metrics," *Electronics*, vol. 10, no. 5, p. 593, 2021.

[55] M. Heydarian, T. E. Doyle, and R. Samavi, "Mlcm: Multi-label confusion matrix," *IEEE Access*, vol. 10, pp. 19 083–19 095, 2022.

[56] Ž. Vujović *et al.*, "Classification model evaluation metrics," *International Journal of Advanced Computer Science and Applications*, vol. 12, no. 6, pp. 599–606, 2021.

[57] S. A. Hicks, I. Strümke, V. Thambawita, M. Hammou, M. A. Riegler, P. Halvorsen, and S. Parasa, "On evaluation metrics for medical applications of artificial intelligence," *Scientific reports*, vol. 12, no. 1, p. 5979, 2022.

[58] V. Sadhanala, Y.-X. Wang, A. Ramdas, and R. J. Tibshirani, "A higher-order kolmogorov-smirnov test," in *The 22nd International Conference on Artificial Intelligence and Statistics*. PMLR, 2019, pp. 2621–2630.

[59] J. R. Lanzante, "Testing for differences between two distributions in the presence of serial correlation using the kolmogorov–smirnov and kuiper's tests," *Journal Name*, 2021.

[60] J. L. Bowman, "The day activity schedule approach to travel demand analysis," Ph.D. dissertation, Massachusetts Institute of Technology, 1998.

[61] J. Pougala, T. Hillel, and M. Bierlaire, "Choice set generation for activity-based models," 2021.

[62] K. M. Nurul Habib and E. J. Miller, "Modelling activity generation: a utility-based model for activity-agenda formation," *Transportmetrica*, vol. 5, no. 1, pp. 3–23, 2009.

[63] K. Nurul Habib, "A comprehensive utility-based system of activity-travel scheduling options modelling (custom) for worker's daily activity scheduling processes," *Transportmetrica A: Transport Science*, vol. 14, no. 4, pp. 292–315, 2018.

[64] O. B. Västberg, A. Karlström, D. Jonsson, and M. Sundberg, "A dynamic discrete choice activity-based travel demand model," *Transportation science*, vol. 54, no. 1, pp. 21–41, 2020.

[65] E. J. Miller and M. J. Roorda, "Prototype model of household activity-travel scheduling," *Transportation Research Record*, vol. 1831, no. 1, pp. 114–121, 2003.

[66] E. J. Miller, J. Vaughan, D. King, and M. Austin, "Implementation of a "next generation" activity-based travel demand model: the toronto case," in *Presentation at the Travel Demand Modelling and Traffic Simulation Session of the 2015 Conference of the Transportation Association of Canada*, 2015.

[67] J. Auld and A. K. Mohammadian, "Activity planning processes in the agent-based dynamic activity planning and travel scheduling (adapts) model," *Transportation Research Part A: Policy and Practice*, vol. 46, no. 8, pp. 1386–1403, 2012.

[68] T. Bellemans, B. Kochan, D. Janssens, G. Wets, T. Arentze, and H. Timmermans, "Implementation framework and development trajectory of feathers activity-based simulation platform," *Transportation research record*, vol. 2175, no. 1, pp. 111–119, 2010.

[69] M. H. Hafezi, L. Liu, and H. Millward, "A time-use activity-pattern recognition model for activity-based travel demand modeling," *Transportation*, vol. 46, pp. 1369–1394, 2019.

[70] M. H. Hafezi, N. S. Daisy, H. Millward, and L. Liu, "Ensemble learning activity scheduler for activity based travel demand models," *Transportation Research Part C: Emerging Technologies*, vol. 123, p. 102972, 2021.

[71] K.-F. Chu, A. Y. Lam, and V. O. Li, "Deep multi-scale convolutional lstm network for travel demand and origin-destination predictions," *IEEE Transactions on Intelligent Transportation Systems*, vol. 21, no. 8, pp. 3219–3232, 2019.

[72] LlactaLAB, "Gad municipal de cuenca (2016) encuesta de movilidad de hogares," 2024, accessed: 2024-08-02. [Online]. Available: https://llactalab.ucuenca.edu.ec/

[73] M. Al Duhayyim, S. Abbas, A. Al Hejaili, N. Kryvinska, A. Almadhor, and U. G. Mohammad, "An ensemble machine learning technique for stroke prognosis." *Computer Systems Science & Engineering*, vol. 47, no. 1, 2023.

[74] Y. Ma and Z. Zhang, "Travel mode choice prediction using deep neural networks with entity embeddings," *IEEE Access*, vol. 8, pp. 64 959–64 970, 2020.

[75] V. G. Raju, K. P. Lakshmi, V. M. Jain, A. Kalidindi, and V. Padma, "Study the influence of normalization/transformation process on the accuracy of supervised classification," in *2020 Third International Conference on Smart Systems and Inventive Technology (ICSSIT)*. IEEE, 2020, pp. 729–735.

[76] B. Remeseiro and V. Bolon-Canedo, "A review of feature selection methods in medical applications," *Computers in biology and medicine*, vol. 112, p. 103375, 2019.

[77] I. Guyon, S. Gunn, M. Nikravesh, and L. A. Zadeh, *Feature Extraction: Foundations and Applications (Studies in Fuzziness and Soft Computing)*. Berlin, Heidelberg: Springer-Verlag, 2006.

[78] H. Zhou, X. Wang, and R. Zhu, "Feature selection based on mutual information with correlation coefficient," *Applied Intelligence*, vol. 52, no. 5, pp. 5457–5474, 2022.

[79] S. Tyagi and S. Mittal, "Sampling approaches for imbalanced data classification problem in machine learning," in *Proceedings of ICRIC 2019: Recent innovations in computing*. Springer, 2020, pp. 209–221.

[80] H. Liu, M. Zhou, and Q. Liu, "An embedded feature selection method for imbalanced data classification," *IEEE/CAA Journal of Automatica Sinica*, vol. 6, no. 3, pp. 703–715, 2019.

[81] F. Provost, "Machine learning from imbalanced data sets 101," in *Proceedings of the AAAI'2000 workshop on imbalanced data sets*, vol. 68, no. 2000. AAAI Press, 2000, pp. 1–3.

[82] Y. Yang, G. Ma *et al.*, "Ensemble-based active learning for class imbalance problem," *Journal of Biomedical Science and Engineering*, vol. 3, no. 10, p. 1022, 2010.

[83] R. Mohammed, J. Rawashdeh, and M. Abdullah, "Machine learning with oversampling and undersampling techniques: overview study and experimental results," in *2020 11th international conference on information and communication systems (ICICS)*. IEEE, 2020, pp. 243–248.

[84] H. He and Y. Ma, "Imbalanced learning: foundations, algorithms, and applications," 2013.

[85] T. Wongvorachan, S. He, and O. Bulut, "A comparison of undersampling, oversampling, and smote methods for dealing with imbalanced classification in educational data mining," *Information*, vol. 14, no. 1, p. 54, 2023.

[86] V. Shatravin, D. Shashev, and S. Shidlovskiy, "Implementation of the softmax activation for reconfigurable neural network hardware accelerators," *Applied Sciences*, vol. 13, no. 23, p. 12784, 2023.

[87] S. R. Dubey, S. K. Singh, and B. B. Chaudhuri, "Activation functions in deep learning: A comprehensive survey and benchmark," *Neurocomputing*, vol. 503, pp. 92–108, 2022.

[88] X. Ying, "An overview of overfitting and its solutions," in *Journal of physics: Conference series*, vol. 1168. IOP Publishing, 2019, p. 022022.

[89] M. Community, "Matsim: Multi-agent transport simulation," https://www.matsim.org, 2024, accessed: 30-06-2024.

[90] A. Horni, K. Nagel, and K. W. Axhausen, "Introducing matsim," in *The multi-agent transport simulation MATSim*. Ubiquity Press, 2016, pp. 3–7.

[91] Simunto, "Via: Visualization and analysis for matsim," 2024, accessed: 2024-08-02. [Online]. Available: https://www.simunto.com/via/