APPLICATION OF MACHINE LEARNING MODELS IN PUBLIC TRANSPORTATION IN THE CONTEXT OF IMBALANCED DATA

By

Juan Camilo Martínez Muñoz

Dissertation

Submitted to the Faculty of the

Graduate School of Vanderbilt University

in partial fulfillment of the requirements

for the degree of

MASTER OF SCIENCE

in

Environmental Engineering

May 12, 2023

Nashville, Tennessee

Approved:

Jonathan Gilligan, Ph.D.

Hiba Baroud, Ph.D.

To Nala with all my love ...

ACKNOWLEDGMENTS

So many people to thank. In particular, I want to thank Nala, my parents, and brother for their love and unconditional support.

Dr. Gilligan for his support, patience, and critical reviews. Without your mentorship, this accomplishment would not be possible. Also, Dr. Baroud, your class was instrumental in this project, and your feedback made it better.

Dr. Dubey and Ayan for giving me the chance to participate in the transit project. It was a great learning experience for Big Data and Machine Learning. Your feedback and support will always be appreciated.

Dr. Sanchez and the Civil and Environmental Engineering Department, thank you for your support since the beginning. Without your help, this would not be possible.

In general, all my friends and loved ones (in Nashville and Colombia). Your support has been important and treasured, particularly during the pandemic. Without your support, I would not be here.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

CHAPTER 1

Introduction: Public transportation and imbalanced learning

1.1 Introduction and Backgrounds

Public transit provides critical services that enable residents of a city, especially those without personal cars, to commute to work and access essential services Lao et al. (2016). As a result, public transportation is a crucial determinant of the quality of life for the urban and suburban population. Cities strive to maximize the coverage of transportation services under budgetary constraints. However, optimizing the spatial and temporal distribution of transit is a complex multi-dimensional stochastic optimization problem. Uncertainty in the problem domain arises from various factors; on the one hand, ridership patterns themselves are uncertain, and on the other, urban environments are highly dynamic. Further, catastrophic events like the novel coronavirus disease (COVID-19) pandemic and subsequent mitigation efforts have created powerful new challenges for public transit systems Tirachini (2020).

Higher population density and connectivity can also aid the spread of pandemics Olmo and Sanso-Navarro (2020); Medo (2020). One of the most effective measures for slowing down or stopping the spread of a contagious disease is *social distancing*, that is, reducing the number of times that people come into close contact with each other Brooks et al. (2021). To minimize contact between passengers and drivers, agencies switch to fare-free operations and block front-door boarding. To reduce contact between passengers, they limit passenger capacity. In some cases, the agencies are also decreasing the frequency of the fixed-route service as ridership has declined. While these ad-hoc changes are reducing the number of close contacts between passengers, they are also affecting people that have higher socio-economic vulnerabilities Brough et al. (2020). Indeed, the effects of COVID-19 on public transit ridership are not uniform across demographic, spatial, and temporal variations Hu and Chen (2021).

One of the pre-requisites for optimizing public transit is to accurately estimate ridership and occupancy demand. Historical occupancy data can be used to forecast expected demand in the future, which in turn can be used to optimize some utility function (e.g., total demand-weighted coverage or waiting time) that captures the requirements of the concerned transit agency. Estimating ridership demand also helps passengers make informed choices. However, the changes introduced to counter pandemic concerns have made these prediction tasks difficult. While some transit agencies have partnered with software developers to estimate ridership

patterns and inform commuters through smartphone applications Darsena et al. (2020); Couture et al. (2020), such approaches are largely ad-hoc and myopic. As a result, commuters end up planning their travel under considerable uncertainty.

Forecasting ridership is challenging. Indeed, as we show in later sections and chapters, even well-established statistical and algorithmic approaches to data-driven learning fail to estimate ridership demand at high spatial and temporal resolutions accurately. A significant challenge in predicting demand is the high propensity of zero counts in the data. For example, consider ridership patterns in Chattanooga, which has a population of about 180,000. In figure 1.1, we present hourly ridership data in fixed-line buses in Chattanooga grouped by month from 2019 to mid-2020. An important finding is the high volume of zero counts; even though we try to depict the historical data through its quartiles graphically, most box plots barely rise above zero. As Mukhopadhyay et al. (2020) points out, standard statistical models for count-based data (e.g., Poisson regression) fail to work well in such scenarios.

Prior work has explored several different aspects of forecasting transit ridership. For example, Karnberger and Antoniou (2020) provide an insight into the relationship between public transit ridership and Spatio-temporal influences from exogenous events. Zhou et al. (2017) explore the impact of daily weather condition changes on the usage of public transit. There has also been work that attempts to predict passenger occupancy on public transportation in the near future by using real-time information from smart cards Van Oort et al. (2015); Nuzzolo et al. (2013). This manuscript describes statistical and algorithmic models to forecast ridership based on automated passenger counting (APC) and transit data from the General Transit Feed Specification (GTFS), while considering and assessing the uncertainty of the input data (trend changes and zero counts) by exploring different probability distributions.

One approach to circumvent the high zero counts is to learn forecasting models over coarse spatial and temporal resolutions. Naturally, aggregating data across space and time reduces the number of zero occurrences. However, transit agencies often require fine-grained forecasts to optimize daily transit schedules, dispatch secondary vehicles in case of over-crowding, and inform riders about potential delays and occupancy. Instead, we focus on designing principled data-driven approaches to estimate ridership and occupancy at high spatial-temporal resolutions. Our method is based on combining zero-inflated models (a widely used statistical model for handling excess zero counts) with ensemble learning. We also explore how neural networks can be used in this context. We evaluate such methods along with well-known statistical models such as Poisson regression, negative binomial regression, and others. Our analysis is based on real-world transit data from

Nashville and Chattanooga, USA and our findings reveal that the choice of model depends to a large extent on the dependent variable under consideration and the proportion of zero counts in the data.

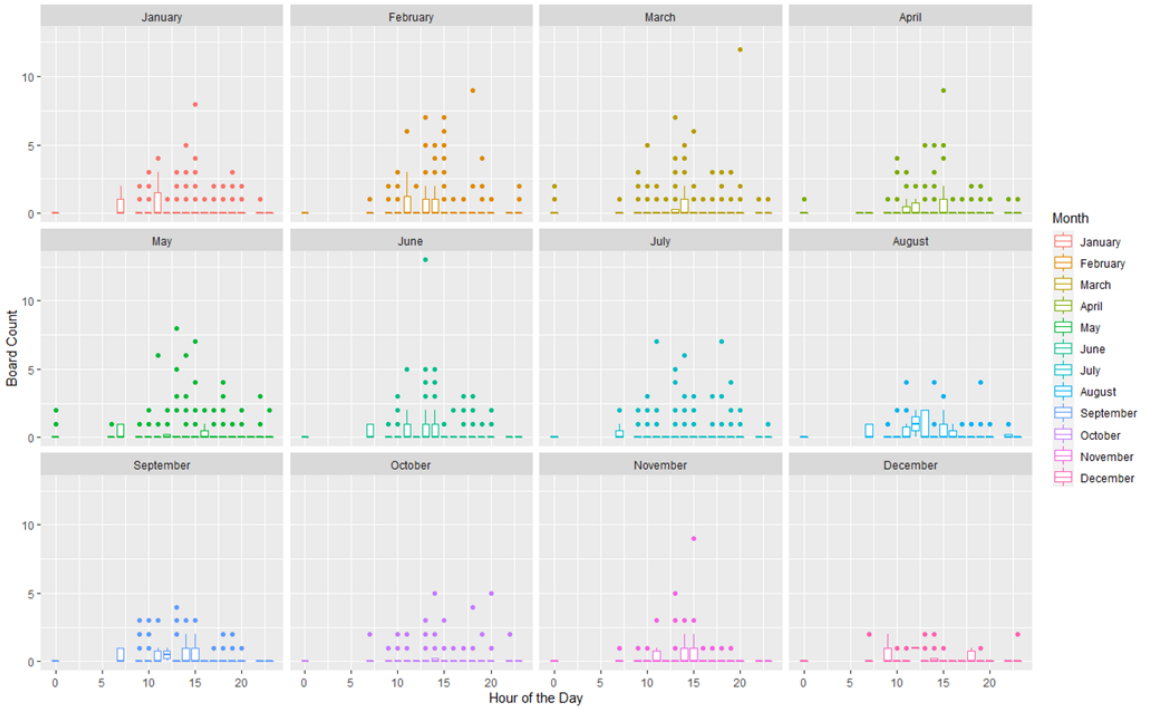

Figure 1.1: Board count box-plots grouped by month and hour of the day across all trips and bus stops. There is a high dispersion in data as can be seen by the small (almost invisible) bands of the box plots.

1.2 Challenges in Public Transportation Modeling

In developed countries, public transportation has faced challenges increasing its ridership or demand. In particular, the possibility to afford a car has been been one of the main reasons. The rates of car ownership have been increasing as incomes rise and cars become more affordable. The continuing decentralization of cities into suburban and exurban areas has generated land-use patterns and trips that are difficult for public transport systems to serve Buehler and Pucher (2012). But this trend is not uniform across developed countries, for example, Germans are five times more likely than Americans to use public transportation, which means that zero counts are less prevalent in Germany and there are computational and modeling challenges to model public transportation demand in the United States.

According to the International Association of Public Transport, the ridership in developed countries has increased by at least 18% since 2000. The United States still has a low demand, and has the lowest number of journeys per capita between all the participant countries. There is a historical relationship between poverty and public transportation Sanchez (2008) in The United States. Also, bus routes do not necessarily connect

residential and working locations Blackley (1990). Thus, it is important to explore new alternatives to improve (and optimize) public transportation and increase quality and demand. Also, it is crucial to take into account the data noise, which comes from instrument imprecision.

This manuscript does not address the causes related to low demand in public transportation in the United States. Instead, it proposes a new method to predict boarding counts (and maximum occupancy) in the presence of high proportion of zero counts. This is not a trivial task since standard modeling approaches such as Poisson regression use the concept of average to estimate its parameters and these averages will tend to zero as the proportion zero counts gets closer to one. Consequently, all the predictions from a model of these characteristics will be influenced by the high proportion of zero counts. Then, these predictions will tend to zero as well, which will not provide meaningful information about the boarding counts.

Considering this situation, transit authorities in Nashville and Chattanooga have been forced to provide ad-hoc solutions, relaying on the experience and expertise of transit agents, but not taking advantage of the recorded data.

1.3 Imbalanced Learning

Addressing the modeling and computational challenges related to the high proportion of zero counts is important to improve public transportation quality. The ability to accurately predict zero boardings (i.e., the number of passengers taking a particular at a particular location) and boarding counts is crucial for better and optimized bus dispatching.

In the context of Machine Learning, when the dependent variable does not have a similar class proportion, it is known as an *imbalanced learning* problem, that is, the learning process for data representation and information extraction with severe data distribution skews to develop effective decision boundaries to support the decision-making process. He and Ma (2013). The main negative effect with the imbalanced learning problem is the ability of imbalanced data to significantly compromise the performance of most standard learning and mining algorithms He and Ma (2013).

Our proposed solution creates new dependent variables that only has two classes: zero-counts and counts. That is, we train a classifier to predict zero-counts from counts. Then, we train a regression model to predict the boarding counts. Therefore, we require a classifiers or learners that with high predictive accuracy for both the minority and majority classes on the dataset. Nonetheless, in our dataset we found that the proportion of

zero counts is greater than 90%, and depending on the route and direction characteristics it can be as high as 97%. This means that the boarding counts represent, at best, only the 10% of the dataset.

There are multiple approaches to the imbalanced learning problem, and in this manuscript we focus on modifying the sampling method to provide a balanced distribution of classes. That is, we randomly oversample the underrepresented classes (i.e., boarding counts), while we randomly undersample the overrepresented class (zero counts).

In the next section, we present a literature review about related work and approaches. This part provides information about the main differences, in terms of advantages and disadvantages, between previous work and our approach. Then, we explain our problem formalization and link it with our research hypothesis.

CHAPTER 2

Related Work and Problem Formalization

2.1 Literature Review

In cities where public transportation demand is highly, predicting the number of passengers at different spatial and temporal aggregates have offered valuable insights for improving the service Horaždovskỳ et al. (2018); Wilbur et al. (2020). In contrast, cities that do not have a high a demand tend to use static itineraries that can be valid for months Ceder (1984). However, the increasing demand in public transportation around the world represents an opportunity to train predictive models to provide a better service. In this section, we present a literature review about related modeling approaches.

Ceder (1984) proposed a statistical method to calculate peak load factors to determine policy headways (or bus frequency dispatching), but this method aggregates boarding counts by days and only uses boarding count data. The main problem of this approach is that it is fundamentally deterministic and does not provide information about the hourly fluctuations of boarding counts and maximum occupancies, and does not integrate other variables that may be relevant such as temperature and precipitation.

Hofmann and O'Mahony (2005) explored the how adverse weather conditions impacted on urban bus performance measures in Ireland. They created a dataset using electronic fare collection for boarding counts and environmental variables such as Date, Hour, Precipitation, and Temperature. Then, the datasets were joined using the date and hour of both the passenger boarding and the meteorological data set. The authors calculated aggregated averages (e.g., 4pm - 6pm) for most of the meteorological variables, and transformed the precipitation variable from continuous to dichotomous (rain or no rain). While the incorporation of meteorological or environmental variables improved the accuracy of boarding counts, the proposed approach is not able to capture the hourly fluctuations of ridership, it can only predict changes in total daily ridership given some meteorological variables.

Similarly, Guo et al. (2007) and Stover and McCormack (2012) explored the relationship between ridership and environmental conditions in Chicago (Illinois) and Percy County (Washington), and these studies found that environmental variables have an important predictive power on ridership, which is promising given the differences in population sizes of these two locations.

Recently, Zhou et al. (2017) suggested that it is important to explore the impact of weather on public transit at fine-grained temporal and spatial scales to improve our understanding of the relationship between weather and travel behavior. Moreover, the authors found a strong statistical associations between intra-day variations in public transit ridership, both system-wide and at station level, and the changes in weather conditions.

Nuzzolo et al. (2013), Van Oort et al. (2015), and Horaždovskỳ et al. (2018) proposed to study passenger data from a Big Data perspective taking advantage of smart card systems, automated vehicle location, and instruments for Automatic Passenger Counting (APC). The studies highlighted the need to move from ad-hoc approaches for decision making to taking advantage of the large amount of information. The proposed method provided valuable information for planners since they were able to explore a significant number of scenarios, which improved their ability of real-time decision making.

The previous studies provided valuable insights about the data public transportation demand and occupancy in terms of the relevant variables, the importance of using fine-grained spatio-temporal models, and the large amount of information that can be gathered on a daily basis. However, these studies did not provide explicit and clear analysis about the relationship between the dependent variables (boarding counts or maximum occupancy) and their explanatory, nor suggested reasonable modeling approaches to predict the future values of these variables. In the next section, we present the problem formalization and give context to define our proposed dependent variables. Then, we present our research hypothesis and two modeling strategies.

2.2 Problem Formalization

Consider a transit agency that operates a set of buses$\mathscr{V}$. Each bus $v \in \mathscr{V}$ follows a fixed-line route $h \in \mathscr{H}$, where $\mathscr{H}$ is the set of all possible routes in operation. Let $\mathscr{S}$ represent the set of all bus stops and garages, with $s_i \in \mathscr{S}$ denoting a particular stop. Then, any directed graph connecting a subset of $\mathscr{S}$ is a possible bus route, and $\mathscr{H}$ represents the set of all possible directed graphs over $\mathscr{S}$. Note that two graphs traversing the same nodes in different directions represent different bus routes. The set of specific trips made daily by all vehicles $\mathscr{V}$ is denoted by $\mathscr{T}$. Trips are usually set and controlled by a master schedule created by the agency. Each trip $t \in \mathscr{T}$ involves a particular vehicle that follows a specific route $h \in \mathscr{H}$. For each trip, the arrival time $\mathbf{t}_s^{arrival}$ of the bus at a stop $s \in \mathscr{S}$ serving the trip is retrieved from the GTFS feed. We define the number of passengers boarding the bus in trip $t \in \mathscr{T}$ at stop s_i as $\gamma_t(s_i)$. As an example, 1.1 shows the registered board counts for the busiest route in Chattanooga. The number of passengers getting off at s_i during trip $t \in \mathscr{T}$ is denoted by $\alpha_t(s_i)$.

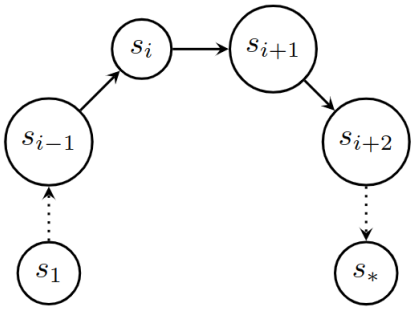

Figure 2.1: Routes represented by directed graphs. Each route and direction have a specific sequence of bus stops $\{s_i\}$.

Formally, we denote conditional distributions over board and alight counts by F_b and F_a respectively. Our goal is to learn $F_b(\gamma_t(s_i) \mid W)$ and $F_a(\alpha_t(s_i) \mid W)$, where W is a set of features that characterize conditions at location s_i at time $\mathbf{t}$. For example, W can include weather, time of day, and ridership data from previous trips. Our model starts from a predicted demand at the origin of the trip (we use the predicted board count at the origin as a proxy for this demand). As the bus follows its route, we sample from the distributions F_b and F_a to generate board and alight counts at every stop, thereby simulating an entire trip made by a vehicle. Note that F_b and F_a vary with time of day, bus stop, weather, occupancy of the bus, and other relevant determinants. In practice, the samples drawn from the distributions must follow practical constraints. For example, the total number of passengers that leave a bus cannot be more than the number of people on board.

We model the *occupancy* of a bus as a random integer L, which denotes the total number of passengers inside the bus at a given point in time. The occupancy on trip t at location s_i, denoted by $L_t(s_i)$, can be calculated from models of the boarding and alighting processes. We model occupancy as an autoregressive process:

$$L_t(s_i) = L_t(s_{i-1}) + \gamma_t(s_i) - \alpha_t(s_i) \tag{2.1}$$

If we group the trip data by date and hour of the day, we can define an expression for the maximum occupancy of trip t, denoted by $L_{t_{max}}$, using equation 2.1. Specifically,

$$L_{t_{max}} = \max\{L_t(s_i)\} \tag{2.2}$$

We aim to combine multiple data streams to estimate boarding counts at different stops and maximum occupancy on each trip. A wide range of temporal granularity can be used to learn such models. One approach is

to learn a model for each unique trip directly. However, each trip contributes relatively little data to learning; consequently, individual models can overfit historical data. On the other hand, learning a universal model for all trips (conditional on a set of features) ignores the information in nearby trips that are not explicitly modeled in the feature space (where nearby denotes proximity in abstract feature space). As a result, we choose a discretization that is mid-way – we group trips according to the hour of the day. *Consequently, our problem is reduced to estimating occupancy (and maximum occupancy) on stops and trips*.

2.3 Research Hypothesis

We point out that it is unnecessary to model board counts, alight counts, and occupancy separately. Note that it is sufficient to know any two of the three variables to estimate the third variable automatically. For example, suppose the initial occupancy at the start of the trip is known. In that case, modeling the board count and alight count at every stop automatically results in a model for estimating occupancy. We choose to model board counts and occupancy directly; alight counts can be calculated using the learned models over board counts and occupancy. We use a set of well-known statistical and algorithmic approaches to data-driven learning to estimate occupancy in trips. Statistical methods assume the existence of a stochastic data model that generates the data Breiman (2001b). Algorithmic strategies, on the other hand, focus on finding a function that takes as input the set of features W and predicts the concerned output (e.g., γ_t) Breiman (2001b). Also, we use a stratified sampling approach to subdivide the data into two parts: a train set to train the models and tune the model parameters, and a test set to compute their test RMSEs. We start by briefly describing the statistical models first and then explain our proposed algorithmic approach.

Statistical Models: We use five statistical models, namely the Poisson, negative binomial, zero-inflated Poisson, zero-inflated negative binomial, and hurdle (binomial and Poisson) models to estimate the distribution of board and occupancy counts. For the sake of brevity, we describe these well-known approaches to statistical modeling only briefly. The Poisson distribution is one of the most widely used approaches for modeling count data Menon and Lee (2017). Each event (a single passenger boarding a bus given a trip and a stop) is considered a result of an independent Bernoulli trial. As the number of trials increases and the probability of success decreases, the count of the number of successes (total passengers boarding or alighting) takes the form of a Poisson distribution. To model the set of features W, the regression model assumes that the logarithm of the expected value (mean) of the dependent random variable (e.g., board counts) can be modeled as a linear function.

A shortcoming of the Poisson model is the assumption that the variance and mean of the distribution are

the same. The negative binomial model (essentially a hierarchical Poisson model where the mean parameter follows a gamma distribution) has been shown to be more flexible Mukhopadhyay et al. (2020). As with many real-world datasets that model counts of events, ridership data consists of many zeros, i.e., during many trips and at many stops, no passengers board or get down from the vehicles. Zero-inflated models can handle excess zeros that standard count-based models cannot explain by considering a separate *state* in the underlying statistical process Mukhopadhyay et al. (2020). Essentially, two processes govern the generation of the observed data. The first process generates zero counts. The second process can be modeled by a count-based discrete distribution such as a Poisson distribution. The second random process generates counts for the events, some of which can be zeros. The negative binomial model can also be used as the count-based model. We use both the zero-inflated Poisson and the zero-inflated negative binomial model in our analysis.

We also use hurdle models to account for the excess zeros. Hurdle models are similar to zero-inflated models in principle but have an important difference. A hurdle model also has two different processes to model the random variable of interest. The first process is used to account for *all* the observed zero counts. The second process generates the non-zero counts only and is a distribution truncated at 0. The zero-generating process is assumed to be a *hurdle* that must be overcome to attain non-zero values. For example, as in the seminal work regarding hurdle models Cragg (1971), a truncated Gaussian distribution can be used to model the non-zero counts, and a probit model can be used for the zero counts.

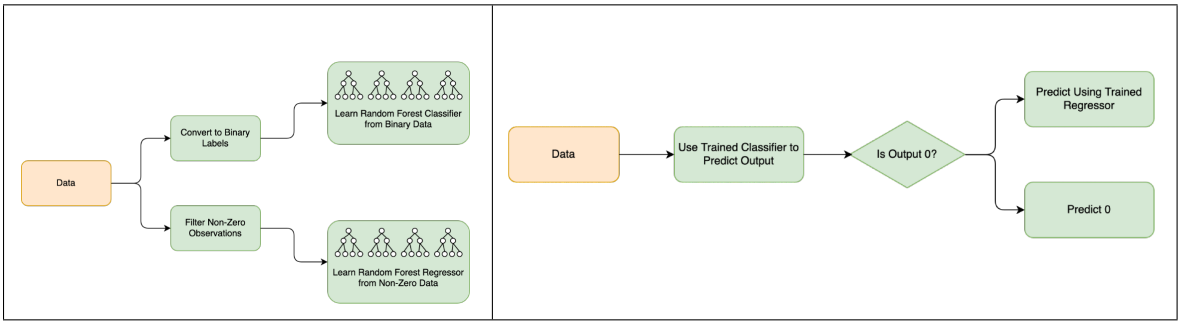

Figure 2.2: **(a)** Random fern classifier on the binary label (zero or not-zero). **(b)** The trained classifier is first used on an unseen data point. If the output is non-zero, the trained regressor is used to predict the final output.

Algorithmic Approaches: Unlike statistical approaches to modeling, algorithmic approaches[1] treat the data generating process as complex and partly unknown. Such approaches aim to find a function that can take w (a specific realization of the feature set W) as input and output the dependent variable of interest. The underlying assumption is that such a function would be a good predictor for any arbitrary $\hat{w}$ in the future (naturally,

[1] while such approaches are also known as machine learning-based approaches, we follow the terminology introduced by Breiman Breiman (2001a)

the belief is conditional on $\hat{w}$ and w being drawn from the same, although unknown, distribution). We use random forests as our baseline algorithmic approach to learning, which constructs an ensemble of decision trees at training time to fit the observed data Breiman (2001a). The model's output is calculated by taking the mean or average prediction of the individual trees in case of regression. In case of classification problems, a vote is taken using the outputs of all the learned trees. Random forests are known to work on a wide variety of data and prevent overfitting, unlike decision trees.

The performance of random forest regression on data that has a high frequency of zero values is unexplored, to the best of our knowledge. We propose zero-inflated or hierarchical random forest regression, which uses two models (both based on random forests) to account for the excess zeros. We show our approach in Figure 2.2. First, we learn a random forest (using the random ferns algorithm) classifier that categorizes an arbitrary w to one of two classes — a class denoting zero-count and another denoting non-zero count. From the perspective of hierarchical modeling, the classifier can be interpreted as a top-level process. Then, a random forest regressor (using the ranger algorithm) produces an output for data points classified to the non-zero class by the top-level process. To learn the zero-inflated random forest model, we first convert observed outcomes (e.g., board counts) to a binary variable by labeling all non-zero observations as 1. Then, we train a classifier on this transformed data. For training the regressor, we only use data with non-zero observations. Note that in principle, the regressor can still predict zeros for an arbitrary $\hat{w}$. However, our training mechanism segregates the two processes that generate zeros and non-zero observations.

Neural networks are computational models that use multiple processing layers to learn abstract representations of data LeCun et al. (2015). Neural networks contain an input layer which maps to the input data (w in our case), a set of hidden layers, and an output layer. Each layer consists of a set of computational nodes called neurons. In feed-forward networks, which were used in this analysis, neurons accept an intermediate computation from nodes in the previous layer, pass it through an activation function, and pass it on to the nodes in the next layer. Gradient based approaches can be used to optimize the network based on a predefined loss function. The neural network was designed to predict the board count at a particular stop as well as maximum occupancy for trips using a linear activation in the output layer and sigmoid activation in all the hidden layers.

CHAPTER 3

Methodology

This chapter covers a wide range of topics, it explains the main data sources, their relevant variables (and keys), and how all the datasets were merged to build one database that can be used for modeling purposes. Also, section 3.1 explains how the proposed data preprocessing can be replicated or applied to different cities with access to APC and GTFS datasets.

Moreover, we explianed how to transform and create new variables with the existing datasets that can improve the predictive power of the models. Also, it is assumed that the observed board and alight counts are independent, and thus, we can perform random k-fold cross-validation. Also, one of the main ideas of this research was to study the effect of lockdown (and its derived decrease in ridership) on modeling accuracy for both pre-lockdown and post-lockdown conditions.

3.1 Data

There are two sources of information in this study: General Transit Feed Specification (GTFS) and Automated Passenger Count (APC). GTFS allows public transit agencies to share and publish their transit data, such as bus stop locations and route patterns, in a structured format for software application purposes. This dataset tends to be accurate and most information is spatially indexed. Also, GTFS requires that the published data has to be valid for at least the next seven days, trying to keep a valid information system.

In contrast, APC data could be noisy and requires preprocessing and transformation. The data is generated from electronic devices installed on vehicles to register boarding and alighting data. The pattern in which the infrared light beam is interrupted by a passenger determines if the person is entering or exiting the vehicle. However, these devices are not completely accurate and can lead to negative occupancies. Therefore, most of the uncertainty comes from boarding and alighting counts. Besides that, the occupancy or number of passengers between two consecutive bus stops is not generated by counting the current number of passengers, and thus, this variable is calculated from the board and alight counts.

During the preprocessing stage the raw APC and GTFS data are cleaned. The first step is to look for duplicates, misspellings, and erroneous information, such as contradictory date or time values for a sequence of bus stops. Also, the date and time information is organized in a proper or standard format that can be easier

to manipulate.

The GTFS dataset has several tables related to the paths that buses have to cover:

- `routes`: Describe the routes of a given transit agency.
- `stops`: Spatial location of all the bus stops covered by a given transit agency.
- `shapes`: The spatial geometry of all the routes of the given transit agency.
- `trips`: It relates each trip of a transit agency to a unique route, direction, service, and shape id.

Figure 3.1 shows the most relevant variables of each dataset before preprocessing. Clearly, neither APC nor GTFS have all the required variables for modeling purposes. For instance, APC has all the information about boarding counts at a given bus stop on a given date of the year. However, it is not easy to derive spatial-temporal patterns of boarding counts since APC does not have further information about routes and their spatial shapes. Therefore, it is necessary to combine the APC and GTFS datasets based on `route_id`, `direction_id`, and `stop_id` to get a unified dataset (and schema) to add spatio-temporal information to the APC data.

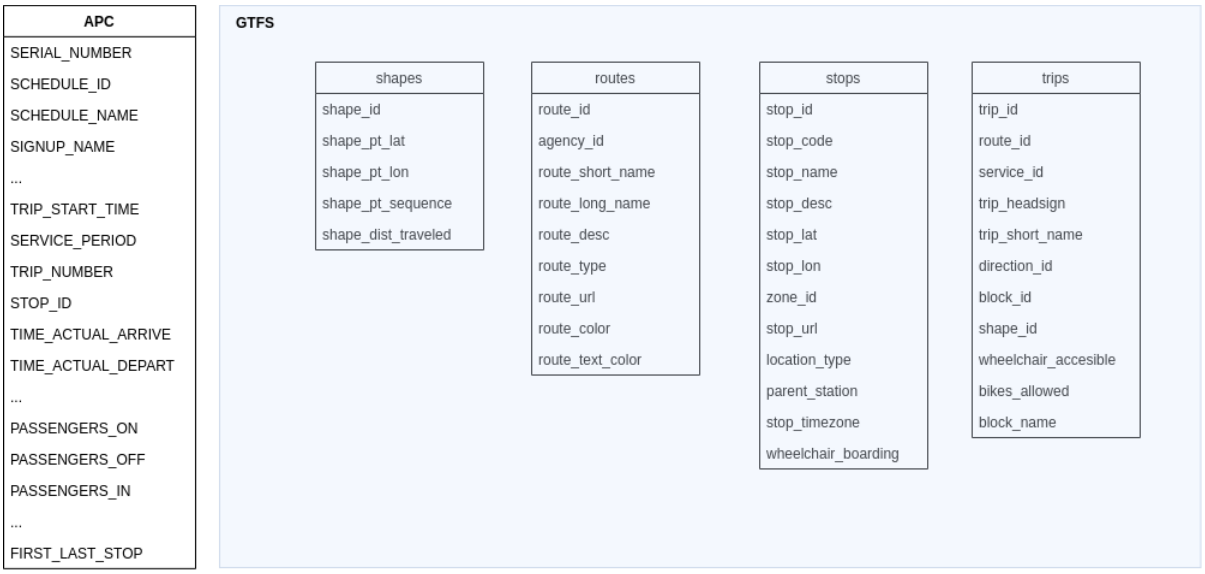

Figure 3.1: The APC and GTFS datasets. Each table or entity has a key that can be used for joins.

The weather dataset gathered data from multiple stations across Tennessee and its surrounding states. These stations measure different environmental variables every 1.5 seconds (40 measurements per minute). Figure 3.2 (left figure) shows the spatial distribution of the weather stations that were considered in this study. Also, it shows the selected weather stations based on proximity to Chattanooga:

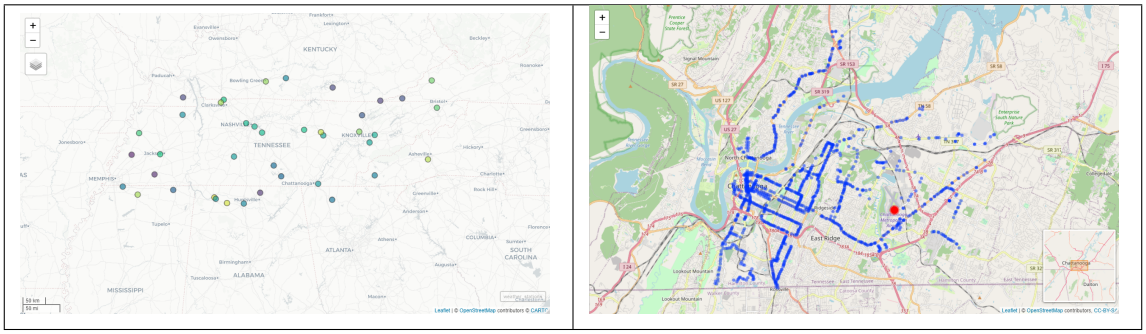

Figure 3.2: Left figure: Spatial distribution of the weather stations considered in this study. Right figure: bus stops in Chattanooga in blue and considered weather station in red.

Clearly, not all the weather stations were considered in this study given that the minimum distance between weather stations exceeded the maximum distance between bus stops. Therefore, we selected the closest weather station to each bus stop. For instance, figure 3.3 shows the temperature and precipitation patterns in Nashville from late 2019 to early 2022 from the closest weather station:

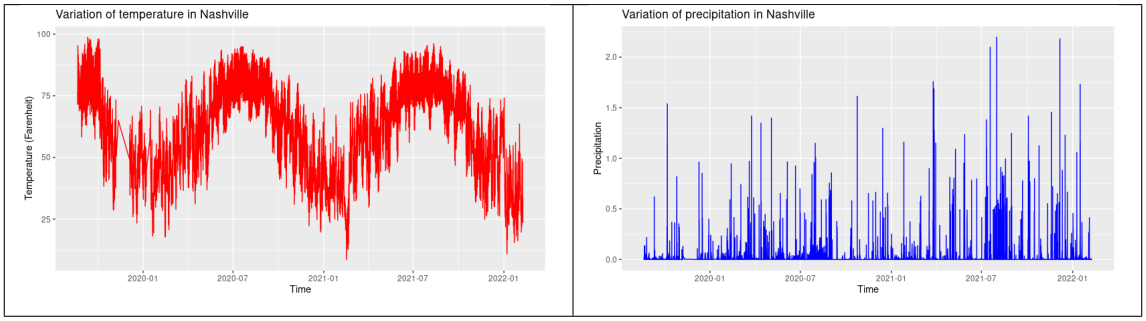

Figure 3.3: Variation of temperature and precipitation in Nashville, TN. These two environmental variables were considered as predictors in the models.

Before merging the weather data with the APC and GTFS dataset, we aggregated the temperature and precipitation by hour of the day to get hourly averages of these two variables for each date in the APC and GTFS dataset. Then, we merged the weather with the unified APC and GTFS dataset based on the arrival time hour of the buses to the bus stops. We collected weather data from Dark Sky and Weatherbit. In other words, we combined public transportation data and its spatio-temporal patterns with environmental variables such as temperature and precipitation.

Table 3.1 presents a schema for the processed data, in which it is possible to relate the variables or fields with their sources and some relevant description.

Once the datasets are cleaned, they are merged to create a final unified dataset that can be used for modeling

purposes and data queries. The following section presents a data schema of the processed data from APC and GTFS.

Field	Source	Description
trip_id	APC, GTFS	unique trip identifier
scheduled_arrival_time	APC, GTFS	time when vehicle was scheduled to arrive at stop hour:min:sec
stop_id	APC, GTFS	unique stop identifier
stop_sequence	GTFS	each trip has an ordered sequence of stops visited, this is the sequence number of stop: stop_id in trip: trip_id
stop_lat	GTFS	latitude of this pickup location (stop)
stop_lon	GTFS	longitude of this pickup location (stop)
route_id	GTFS	unique route identifier
direction_id	GTFS	direction of travel along this route. 0 is outbound, 1 is inbound
board_count	APC	number of passengers boarding at this stop
alight_count	APC	number of passengers exiting vehicle at this stop
occupancy	APC	number of passengers on vehicle after vehicle leaves this stop
direction_desc	APC	same as direction_id but in string format. Should be either OUTBOUND or INBOUND.
service_period	APC	either Weekday or Weekend
date	APC	date of vehicle arriving at this stop year-month-day
scheduled_datetime	APC, GTFS	date + time of vehicle's scheduled arrival at this stop year-month-day hour:min:sec
actual_arrival_datetime	APC	date + time of vehicle's actual arrival at this stop year-month-day hour:min:sec
trip_start_time	APC	represents the time at which this trip started. should be the same for all stops visited for a given trip
day_of_week	APC	the day of the week (0, 1, ... 6)
trip_date	APC	similar to trip_start_time, this is the date at which this trip started. Note that CARTA operates trips that cross over midnight, so trip_date can be different than date field
hour	APC	hour of the day (0, 1 ... 23)
Estimated_temp	Dark Sky	Aggregated mean temperature by hour of the day.
Estimated_precip	Dark Sky	Aggregated mean precipitation by hour of the day.

Table 3.1: Data schema for the processed dataset.

3.1.1 Data Preparation

As part of the preprocessing stage, some column names are changed. For instance, 'PASSENGERS_ON' is changed to 'board_count', 'PASSENGERS_IN' to 'occupancy', and 'LONGITUDE' to 'stop_lon', which could be more informative or less ambiguous. This process is important to maintain a unique schema. Also, the datasets are sorted by date and stored for further analysis.

Part of the main goal of this research was to assess the performance of the models before and afternoon lockdown conditions were imposed, pre- and post-lockdown, respectively. Imbalanced data or excess of zero counts in public transportation can be exacerbated if lockdown conditions are imposed. Therefore, we identified the dates on which the ridership dropped drastically due to COVID-19 restrictions to identify the dates that reflected these new conditions. Then, we split the data into pre-lockdown and post-lockdown datasets.

We consider data from 2019-01-02 to *mid 2021* from both Nashville and Chattanooga. According to Wilbur et al. (2020), lockdown restrictions affected the ridership in Nashville in Chattanooga on the following dates:

- Nashville:
  - Pre-lockdown data: `date < 2020-04-19`
  - Post-lockdown data: `date >= 2020-04-19`
- Chattanooga:
  - Pre-lockdown data: `date < 2020-03-05`
  - Post-lockdown data: `date >= 2020-03-05`

For every route, direction, and bus stop, we created a new variable called *surrounding board counts* or *surrounding alight counts* to take into account the number of boarding or alighting events that happened an hour before a targeted hour within a half-mile radius with respect to a given bus stop.

We defined two dependent variables that will be defined in the following sections of the document. Also, these sections explain how these variables are generated from the dataset.

3.1.1.1 Demand: Board Counts

Public transportation demand can be modeled by board counts. As explained in table 3.1, each boarding event has information of its date-time, route, direction, trip number, and bus stop. Also, there is information about precipitation and temperature during the boarding event. Therefore, the dataset has potential explanatory variables that can be used in machine learning models to predict board counts. However, the zero counts are the dominant proportion, and figure 3.4 shows that zero counts can be higher than the 90% of the observations:

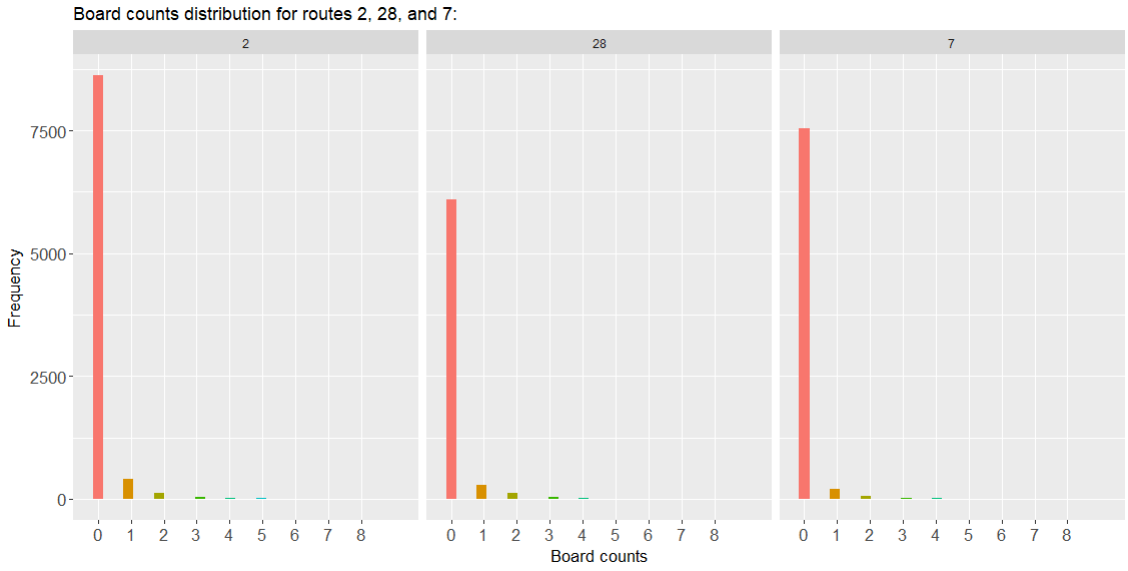

Figure 3.4: Board count distribution of routes 2, 28, and 7 from the Chattanooga public transportation system.

Imbalanced data is a clear challenge in machine learning He and Ma (2013). If this is not addressed, the models will have poor predictive performances. Therefore, the distributions of board counts per bus route, direction, and bus stop need to be assessed. Figure 3.5 compares the empirical and theoretical distributions (using Poisson distributions) of the board counts of routes 2, 28, and 7. The obtained rates are around 0.10 approximately, which reflects the influence of the proportion of zeros:

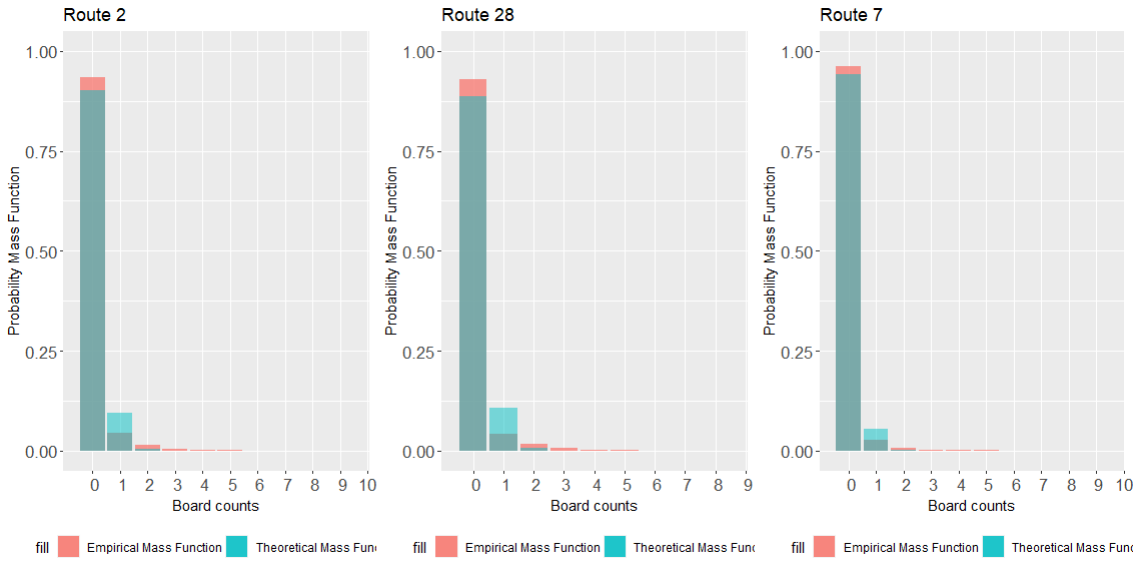

Figure 3.5: Comparison of the empirical and theoretial distributions of board counts.

The goodness of fit of these distributions can be assessed using a Kolmogorov-Smirnov test. The results indicate that as the proportion of zero counts increases, the test suggests that the data do not come from a

Poisson distribution. Moreover, the Negative Binomial distribution was considered in cases with overdispersion. However, hypothesis tests also suggest that the high proportion of zero counts affects the quality of goodness of fit.

It is also important to perform an exhaustive data exploration analysis to identify trends in the boarding counts. For instance, figure 3.6 shows the distribution of board counts as box plots across all the bus stops covered by route 2:

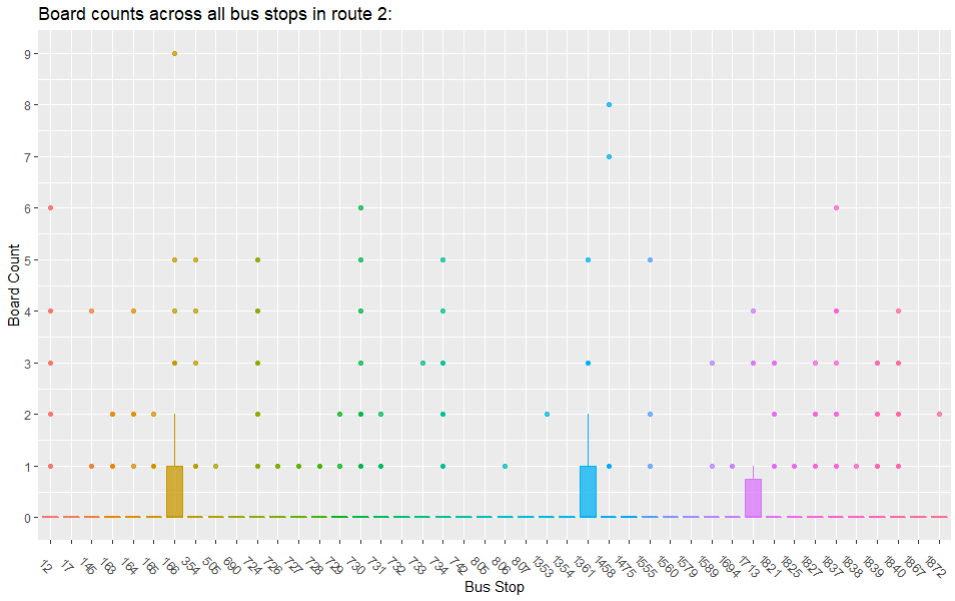

Figure 3.6: Board count across all the bus stops of route.

The overwhelming majority of the counts are regarded as outliers, and just in some specific cases, the third quartile indicates a boarding count of only one passenger. The rest of them are box plots whose first and third quartiles are equal to zero.

If the data exploration analysis is restricted to bus stop 166 from route 2, it can be seen that board counts have a pattern that might be governed by time, or hour of the day as shown by figure 3.7. The same analysis was repeated at multiple bus stops from different routes suggesting that aggregating board counts by hours (at the bus stop level) reflects the fluctuations of public transportation demand.

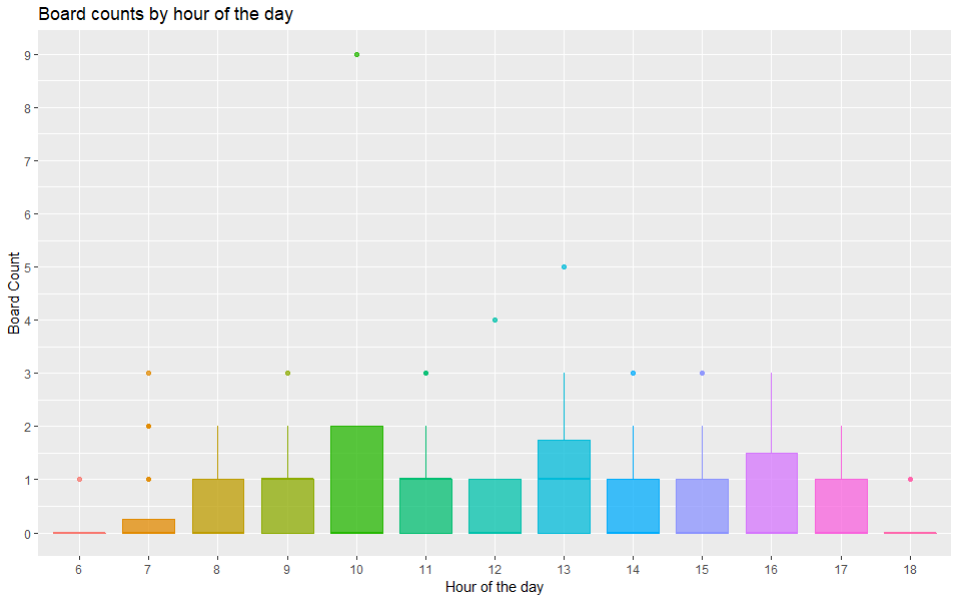

Figure 3.7: Board count by hour id the day of bus stop 166 from route 2.

Also, we performed a more detailed aggregation (every 15 minutes), and as anticipated, the proportion of counts decreased substantially. Furthermore, aggregating the board counts by time frames of the day (*e.g.*, morning, noon, afternoon, and evening) increased the proportion of counts but the information provided by the models was not useful for practical purposes, or decision-making for public transportation authorities.

3.1.1.2 Maximum Occupancy

One of the major challenges of public transportation during the pandemic was crowd avoidance or control as a mechanism to reduce possible social distancing violations at a bus or trip level. In this case, we consider trips as the units of analysis instead of bus stops (for the case of public transportation demand). Figure 3.8 shows the distribution of the maximum occupancies observed for all the trips from routes 2, 28, and 7:

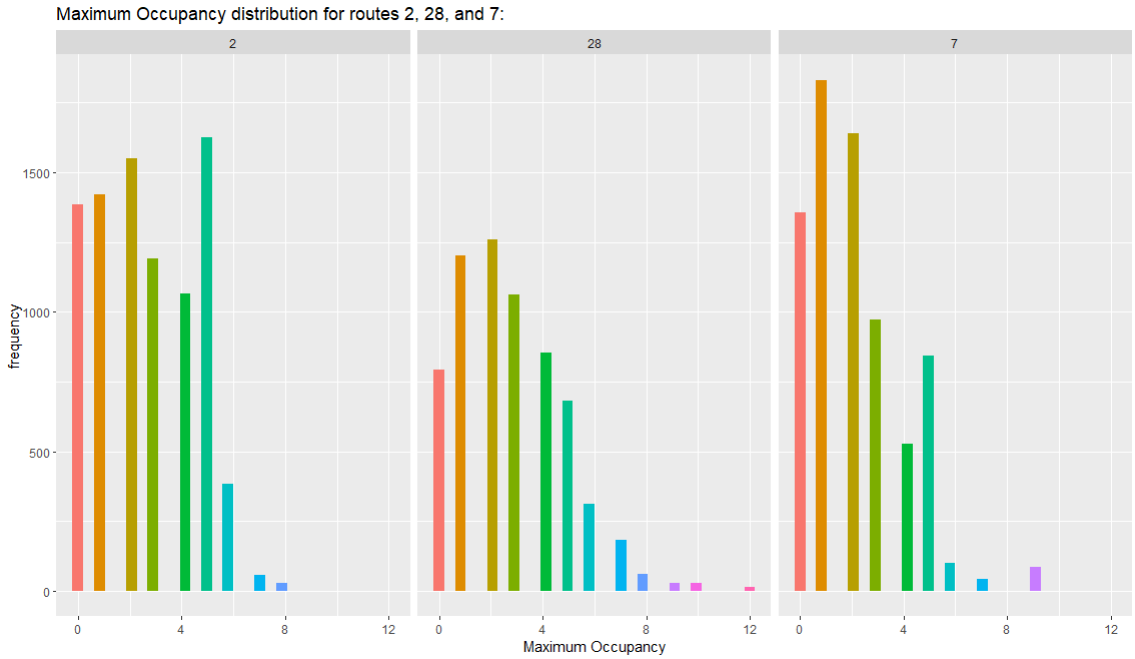

Figure 3.8: Maximum occupancy distribution for three routes of the Chattanooga public transportation system.

Clearly, the proportion of zeros, in this case representing the event of a trip with a maximum occupancy is zero, that is a trip that never has passengers goes to zero. In other words, the trips from these routes had at least one passenger during their commute time. For Chattanooga only, the number of zero counts is reduced and we only observe a 4.4% of zeros in maximum occupancy in trips in our data. Therefore, we can train a model f that allows us to predict maximum occupancies with the highest possible accuracy.

Maximum occupancy is particularly important since it gives meaningful information about the maximum number of passengers in a bus, which could lead to social distancing violations. Therefore, we constructed a new dependent variable called *maximum occupancy* (`max_occupancy`) that represented the maximum number of passengers on a trip.

The `trip_id` variable has information about its route and direction. Also, each trip has a working schedule defined by the GTFS dataset. To generate this variable we grouped the data by `date`, `trip_id`, and `hour` to get the maximum `occupancy` value across all the bus stops covered by a trip as presented by equation (2.2).

The main goal is to predict the maximum number of passengers on any given trip on any segment of its commute path. Also, we consider the same machine learning methods for both maximum occupancy and

demand to assess their performance under both situations.

3.1.2 Data Storage and Paths

We can increase the efficiency and performance of the computations by dividing and storing the dataset in different compartments. For instance, if we plan to train a set of machine learning models to predict the demand on route 1 from Chattanooga as seen in figure 3.9, it becomes clear that any information from route 4, 9, or any other route different than 1 is not relevant. Thereby, we can split the data by routes and direction values to optimize computational resources. Similarly, if we want to model the demand of route 1, we can decide to concentrate efforts on any of its directions, keeping order without losing information.

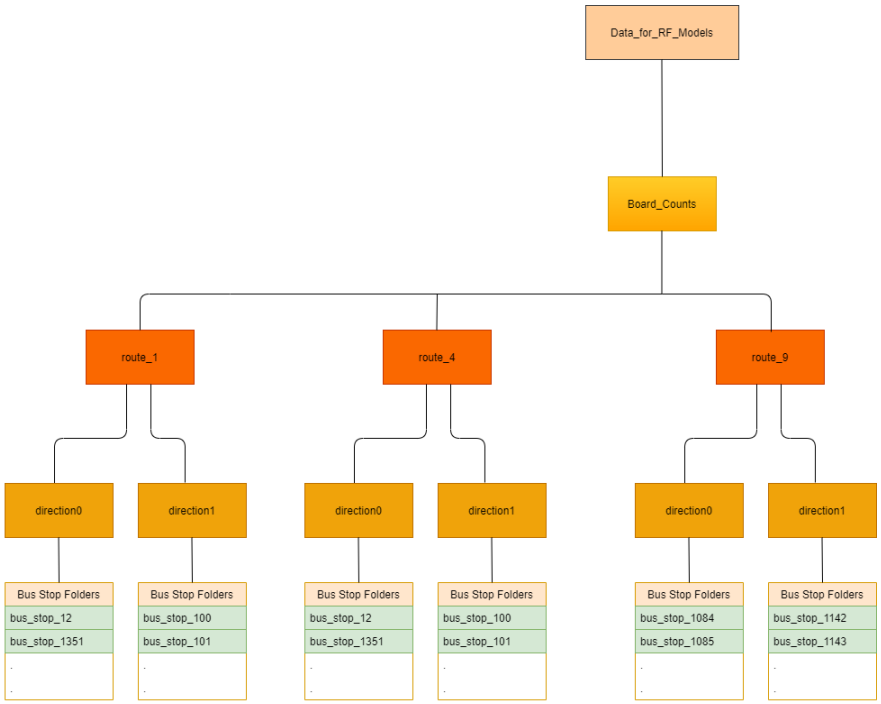

Figure 3.9: Hierarchical structure for the data based on transit agency, route, direction, and bus stop.

Also, we can think of routes and directions in a hierarchical structure: *a transit agency has routes, all routes have two directions, and all directions have a set of bus stops.* Therefore, we propose the same hierarchical structure to store all relevant data and machine learning models. For instance, an access path for the pre-lockdown test dataset, (`pre_lock_test_data.csv`), of bus route 1, direction 0, bus stop 12 is:

```
route_1/direction0/bus_stop_12/pre_lock_test_data.csv
```

3.2 Proposed Solution

We considered a wide range of methods that could cope with (a) count data, and (b) zero-inflated data. Considering the overwhelming evidence of zero-inflated counts, we wanted to assess their generalization error or relative performance using the *Test* Root Mean Square Error (RMSE) as the loss function:

$$L(y, \hat{f}(X)) = \underbrace{\sqrt{\frac{\sum_{i=1}^{N}(y - \hat{f}(X))}{N}}}_{RMSE_{Test}} \tag{3.1}$$

where N representes the number of sample in the test dataset; y and $\hat{f}(X)$ indicate the vectors of observations and predictions, respectively.

Moreover, we considered a set of benchmark methods (instead of one) since we were facing count data inflated in zero, and we generated a dataset for each combination of route, direction, and bus stop. Therefore, we needed a broad range of methods to cope with the particularities of each case. Moreover, previous studies reported that not modeling imbalanced data can lead to undesirable predictions Feng (2021). Therefore, we performed a stratified sampling approach to generate train and test sets based on the proportion of zeros and counts for each combination of route, direction, and bus stop (or trip). This process can be done in `R` using the `caret` package by Kuhn (2022) as follows:

```
index <- createDataPartition(y, p = 0.8, list = FALSE)
```

where `y` represents a binary dependent categorical variable that indicates if the observed outcome is a zero or a count. This process was repeated for each trip in any given route and direction combination, and also, for every bus stop for any given combination of route and direction. Also, the `index` variable would contain the row numbers of the training dataset.

The proposed solution was not a unique model, since the datasets had varying overdispersion and proportions of zero counts. Instead, we proposed to link two Random Forest models that could work in sequence to provide accurate results as explained in subsection 3.3.4.

We considered a set of explanatory variables to model the time dependency of public transportation and demand and maximum occupancy such as hour of the day, month, and service kind (weekday or weekend). Also, we considered hourly mean temperature and precipitation to take into account environmental information in the models and predictions as described by Stover and McCormack (2012). The combined information

of these two types of variables has been important to explain the decrease in ridership during extreme weather conditions Miao et al. (2019).

Besides that, we found that the aggregated average of monthly board counts is an important predictor of board counts. For instance, the monthly average of board counts in May of 2019 and 2020 are very different due to lockdown restrictions. Therefore, we considered this variable as an explanatory variable to model and predict board counts.

Similarly, the ridership of a given area during a given time frame can provide important insights into ridership patterns Montero-Lamas et al. (2022). Figure 3.10 shows bus stop 1391 in Chattanooga and its surrounding bus stops in purple. Therefore, we considered the surrounding board counts (board counts in nearby bus stops within a half-mile radius) in the previous hour. That is, we considered the aggregated mean board counts from surrounding bus stops at a given hour of the day to predict the board counts in the following hour of a bus stop of interest.

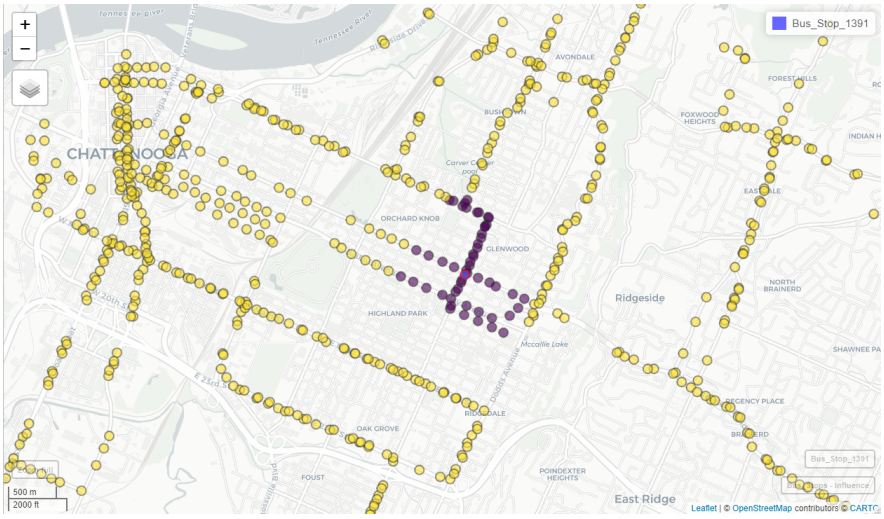

Figure 3.10: Bus stop 1391 in red and all the bus stops within a half-mile radius.

The maximum occupancy models only used hour of the day, mean temperature, and precipitation (aggregated by the hour of the day). In cases when the commute time of the trip was less than an hour, only mean temperature and precipitation were used.

3.3 Approaches to Modeling the Data

There are many possible ways to approach statistical (and machine learning) modeling of this data set. We examined a number of different kinds of models and compared them to determine which approach produced

the best predictions (or best generalization error). To avoid overfitting, we used k-fold cross-validation to compare the models and identify the one that performed the best.

The proposed modeling approaches were carefully reviewed based on the characteristics (empirical distribution and dispersion) of the dependent variables of the trips and each combination of route/direction/bus-stop in Nashville and Chattanooga. Poisson and Negative Binomial regression models are widely considered as based models for count data, and thus, we considered them as baseline models. Moreover, previous data exploration analysis showed that even for the routes with the highest demand, there was an excess of zeros as shown in figure 3.11, and the majority of bus stops recorded zero board counts as seen in figure 3.4. Therefore, we added Zero-Inflated and Hurdle regression models to the set of baseline models.

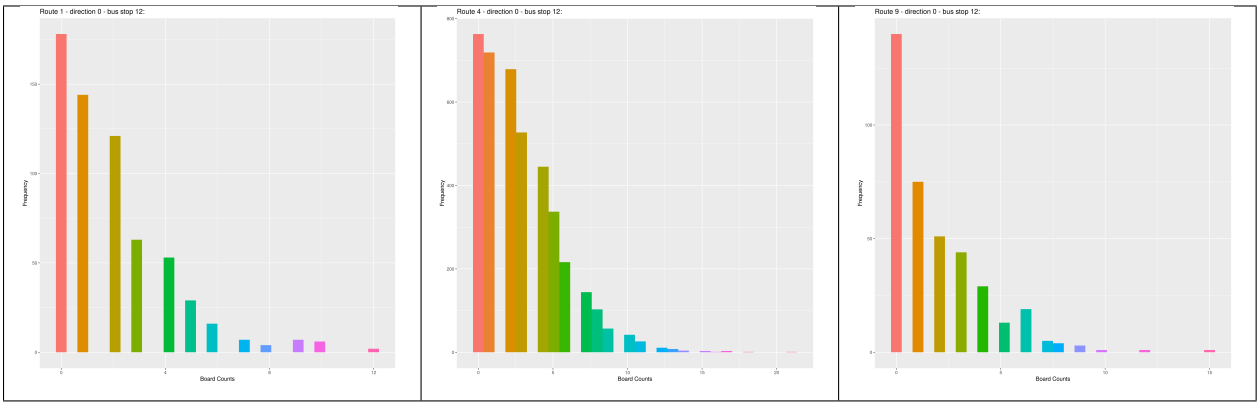

Figure 3.11: Board counts histograms of three different bus stops in Chattanooga.

Also, figure 3.11 represents the board counts of the three routes with the highest demand in Chattanooga, and thus, other bus stops had a higher proportion of zero counts, which could be beyond the capabilities of more traditional models like Poisson regression, and even Hurdle models.

The considered models were trained using a random search in repeated k-fold cross-validation with five folds and two repetitions as shown in the code below:

```
control <- trainControl(method = 'repeatedcv',
                        number = 5,
                        repeats = 2,
                        search = 'random')
```

There were training datsets that had less than 150 observations, and thus, dividing the data in the standard ten folds would lead to small datasets to determine hyperparameters. In contrast, five folds increase the number

of observations per fold, and the repetition of this process, balances the small number of folds.

The following subsections describe the main attributes of the models and how they are implemented in R. Generalized Linear Models (in subsection 3.3.1) are used as baseline (or benchmark) models. Hurdle and Zero-Inflated models are also included as baseline models. Then, subsection 3.3.4 describes the random forest algorithms: a *vanilla* version of the method using (ranger) Wright and Ziegler (2015), and zero-inflated version using Random Ferns by Kursa (2014) for classification that is then link to another ranger model for regression. Finally, a Neural Network with a pre-defined architecture.

3.3.1 Generalized Linear Models

These models are built on the basis that the dependent variable, y, and a set of regressors, w, by their conditional distribution, $y_i|w_i$ with a probability density function:

$$f(y;\, \lambda,\, \theta) = exp\left(\frac{y \cdot \lambda - b(y)}{\theta} + c(y,\, \theta)\right) \tag{3.2}$$

where λ is the canonical parameter whose value depends on the regressors, w, by a linear predictor, and θ is a dispersion parameter that needs to be tuned (or estimated using maximum likelihood). Also, the functions $b()$ and $c()$ are pre-defined probability distributions (*e. g.*, Binomial, Poisson, or Negative Binomial). Moreover, the conditional mean, $E[y_i|w_i] = \mu_i$, depends on the set of regressors w_i by the link function ($g()$):

$$g(\mu_i) = w_i^T \beta \tag{3.3}$$

The subindex i indicates that the observations are grouped by a parameter such us hour of the day. Therefore, y_i and μ_i indicate the board counts and mean board count during the i-th hour of the day, respectively.

Poisson regression models are a good choice for count data that follow a Poisson distribution. This approach models the intensity parameter, μ, with a set of covariates or explanatory variables such as the hour of the day and day of the week. The intensity or rate parameter can describe the number of events of interest per unit of time (*e.g.*, number of board counts per hour for a given combination route/direction/bus stop). If the relationship between μ and its covariates is *parametrically exact* and only involves exogenous covariates but no other source of stochastic variation, then, the model is known as a standard Poisson regression. However, if the relationship is stochastic and involves an underlying (and unknown) random process, then, the model is known as a mixed Poisson regression Cameron and Trivedi (2013).

If we aggregate the number of board counts by hour ($i = 0, 1, \ldots, 23$), the Poisson regression will establish a relationship between the number of board counts per hour, y_i (*i.e.*, board counts or maximum occupancy), with a set of linearly independent covariates (which are also aggregated by hour of the day) w, through a continuous function $\mu(y_i, \beta)$ such that (the expected value) $E[y_i|w_i] = \mu(y_i, \beta)$. The vector β represents a vector with the coefficients of the covariates, which are estimated by maximum likelihood. Also, each observation y_i given w_i follows a Poisson distribution (and Probability Mass Function):

$$f(y_i|w_i) = \frac{e^{-\mu_i}\mu_i^{y_i}}{y_i!} \tag{3.4}$$

Typically, $\mu_i > 0$ for hours with high public transportation demand, and $\mu_i \approx 0$ for hours with low public transportation demand. Also, the rate of board count per hour is parameterized as:

$$\mu_i = exp(w_i^T \beta) \tag{3.5}$$

This parametric relationship will ensure that $\mu > 0$. Figure 3.12 shows the distribution of board counts for three different combinations of route, direction, and bus stops for the city of Chattanooga. Their distribution suggests that Poisson property of $Var[y_i|w_i] = E[y_i|w_i]$ could be a good approximation for some cases in which the mean board count increases with their respective interquartile range:

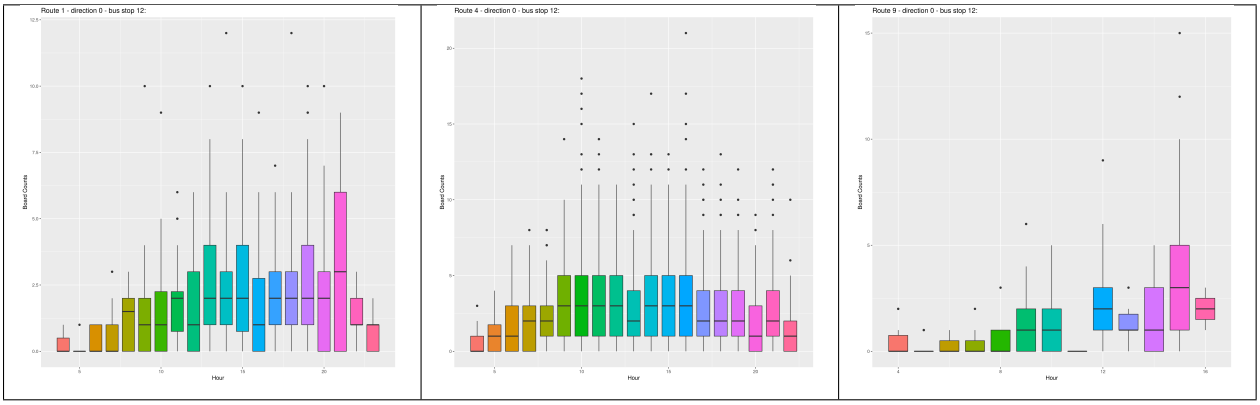

Figure 3.12: Board counts aggregated by hour of the day, route, direction, and bus stop in Chattanooga.

Moreover, figure 3.12 shows a significant overdispersion when the data is grouped by hour of the day. In some cases, the first quartile is zero due to the excess of zeros, which leads to an average of zero board counts per hour. High overdispersion and excess of zero counts are beyond the Poisson model assumptions, and thus, other approaches are more appropriate such as Negative Binomial or Hurdel models. For instance, the Neg-

ative Binomial model is built as a mixture of Gamma and Poisson distributions, which can be parameterized as a probability density function:

$$f(y;\, \mu,\, \theta) = \frac{\Gamma(y+\theta)}{\Gamma(\theta)\cdot y!} \cdot \frac{\mu^y \cdot \theta^\theta}{(\mu+\theta)^{y+\theta}} \tag{3.6}$$

where θ is shape parameter. If $\theta = 1$, then we get the geometric model. Both β and θ are estimated using maximum likelihood. The variance incorporates the shape parameter to cope with substantial overdispersion in the data:

$$Var(y_i|w_i) = \mu_i + \frac{\mu_i^2}{\theta} \tag{3.7}$$

Equation 3.7 is able to handle stronger overdispersion in the data with the additional term μ_i^2/θ.

There is a package in `R` called `stats` that can fit or train Poisson regression model by using the `glm()` function and specifying the parameter `family=poisson` R Core Team (2013). Similarly, the package `mass` can fit a Negative Binomial model using the `glm()` and setting the Negative Binomial distribution and a value for θ: `family=negative.binomial(theta = 2)` Venables and Ripley (2002).

3.3.2 Hurdle Models

Hurdle models used a left-truncated count component with a right-censored hurdle component Mullahy (1986). This model can cope with more zero observations than the Poisson or Negative Binomial models Zeileis et al. (2008); Feng (2021). This approach has two components, a truncated count component, such as Poisson to model (positive) counts, and a hurdle component that models zero or larger counts. For the latter, a binomial model is generally used Zeileis et al. (2008).

This modeling approach combines a probability density function for the counts, $f_{count}(y;\, w,\, \beta)$, which is left truncated at $y = 1$, and a right-censored (at $y = 1$) zero hurdle model, $f_{zero}(y;\, z,\, \gamma)$ (which is often a Binomial distribution), whose regressors are represented by z:

$$f_{hurdle}(y;\, w,\, z,\, \beta,\, \gamma) = \begin{cases} f_{zero}(0;\, z,\, \gamma), & if\, y = 0 \\ (1 - f_{zero}(0;\, z,\, y)) \cdot f_{count}(y;\, w,\, \beta)/(1 - f_{count}(0;\, w,\, \beta)), & if\, y > 0 \end{cases} \tag{3.8}$$

The parameters of the model (β and γ) are estimated using maximum likelihood as any other generalized

linear model. Also, the mean regression relationship is given by

$$log(\mu_i) = w_i^T \beta + log(1 - f_{zero}(0; z, y)) - log(1 - f_{count}(0; w, \beta)) \tag{3.9}$$

Hurdle models can have a different set of regressors for $f_{zero}(0; z, y)$ and $f_{count}(0; w, \beta)$ models. For instance, if the regressors for these two models are the same ($x_i = z_i$), a hypothesis test can be used to decide if the hurdle model is needed or not. The strength of this model is its capability to model the probability of observing a zero count using the zero hurdle model instead of using a simple shape parameter for overdispersion. In general, Poisson and Negative Binomial models assume that the observations (zeros and counts) are realizations of the same random process, whereas hurdle models assign a different model for zeros acknowledging that there are two underlying random processes controlling the observed zero and counts Feng (2021).

The `pscl` package by Jackman (2010) has the `hurdle()` function to train hurdle models. It requires the specification of certain parameters as follows:

```
hurdle(formula, data, subset, na.action, weights, offset,
       dist = "poisson", zero.dist = "binomial", link = "logit",
       control = hurdle.control(...),
       model = TRUE, y = TRUE, x = FALSE, ...)
```

This function has a distinctive input for the `formula` parameter, the regressors for the count and zero models have to be separated by a conditional symbol `|` as follows:

```
formula = y ~ w1 + w2 | z1 + z2
```

where `w1` and `w2` are the regressors for the count model. Similarly, `z1` and `z2` are the regressors for the zero model. Also, the default link function is "logit", but other functions are supported Jackman (2010).

3.3.3 Zero-Inflated Models (Generalized Linear Models Version)

Zero Inflated models use a mixture of models that combine a count component and a point mass at zero. A Poisson, Geometric, or Negative Binomial distribution can model the count distribution. The count distribution also models the zeros (or values in excess). Therefore, the overall zero-inflated probability density function, $f_{zeroinfl}(y; w, z, \beta, \gamma)$, is a mixture of a point mass at zero, $I_0(y)$, a count distribution $f_{count}(y; w, \beta)$, and the probability of observing a zero count is also modeled, and thus, augmented by $\pi = f_{zero}(0; z, \gamma)$ Zeileis et al. (2008).

$$f_{zeroinfl}(y; w, z, \beta, \gamma) = f_{zero}(0; z, \gamma) \cdot I_0(y) + (1 - f_{zero}(0; z, y)) \cdot f_{count}(y; w, \beta) \tag{3.10}$$

The corresponding regression equation for the mean is:

$$\mu_i = \pi_i \cdot 0 + (1 - \pi_i) \cdot exp(w_i^T \beta) \tag{3.11}$$

Similarly, the regressors for the zero and count components can be the same, but there is no restriction for their selection. Also, the default link function $g(\pi)$ (using the Binomial distribution) is the logit link function, but other functions can be used for this purpose. Moreover, this approach does not have a separate distribution for the zero counts, it simply uses a binomial distribution in the zero-inflation component Zeileis et al. (2008).

In general, Zero-Inflated and Hurdle models differ based on their conceptualization of the zeros and interpretation of model parameters Feng (2021). The Zero-Inflated model assumes that zero counts come from a mixture of two probability distributions, in which one always produces zero counts. These zero counts are also known as excessive zeros or *structural zeros*.

There is another important distinction between the Zero-Inflated and Hurdle models. The latter is more capable of handling zero-deflation Min and Agresti (2005). Therefore, the relative performance of these two approaches depends on the percentage of the zero-deflated data points in the data, and the possible differences in the data-generating processes between the structural zeros and sampling zeros Feng (2021). However, there is no clear threshold for the percentage of zeros to define a better model. Moreover, their relative performance also depends on the selected explanatory variables.

The `pscl` package also has the `zeroinfl()` function to train zero-inflated models. It requires the specification of certain parameters as follows:

```
zeroinfl(formula, data, subset, na.action, weights, offset,
         dist = "poisson", link = "logit",
         control = zeroinfl.control(...), model = TRUE,
         y = TRUE, x = FALSE, ...)
```

The main difference between the `hurlde()` and `zeroinfl()` functions is that the latter does not have the `zero.dist` argument because a Binomial model is always used to model the zero-inflated component Jackman (2010).

3.3.4 Random Forest

The original idea about trees was conceived by Leo Breiman Breiman et al. (1984), and then this idea was expanded to random forest by Breiman in Breiman (2001a). Tree-based methods use recursive splitting by stratifying the explanatory variable space into a number of regions. The algorithm uses Residual Sum of Squares (RSS) or Gini index to define the regions that minimize the differences between observations of a given region, and maximize the differences between observations from different regions Rhys (2020).

Once a tree is completely grown, it is possible to make predictions (for a given observation) by considering the mean (for the continuous region) or mode (for the discrete case) value in the region to which it *would* belong James et al. (2013). This algorithm has the ability to train an ensemble of decision trees using random subsets of features or explanatory variables to avoid correlation between other trees or learners. Then, it combines the information of multiple weak learners (trees) to generate a strong learner (random forest). Also, random forest methods have a *greedy* approach, since the node (or region) splits are not globally optimum, they only consider locally optimal splits, given that, it is computationally infeasible to consider every possible partition of the feature space.

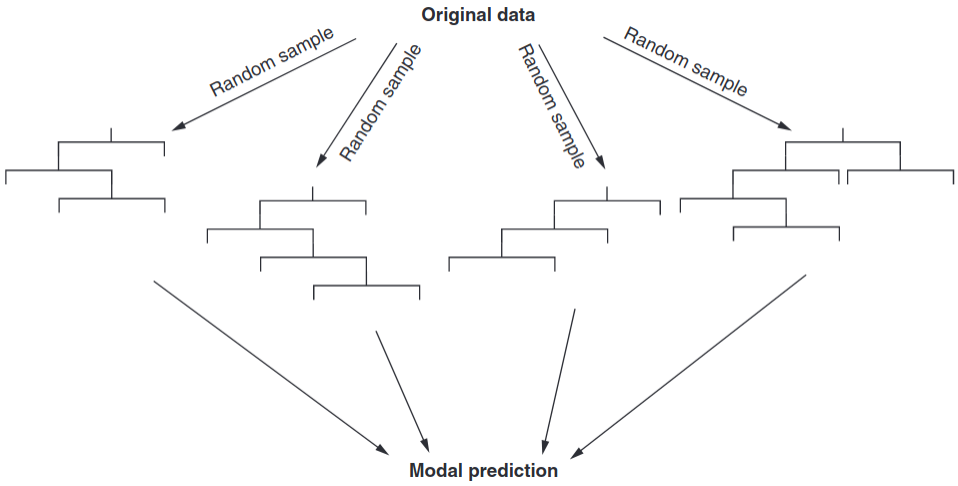

Figure 3.13: Ensemble of trees.

There are some problems with this approach since it cannot guarantee that a tree learns a globally optimum model. Also, the recursive partitioning can go on until it reaches maximum purity (by having one observation) in their leaves leading to overfitting. Moreover, growing trees can be computationally expensive in large datasets. Therefore, high-dimensionality represents a problem for this algorithm.

In 2015 M. Wright optimized the algorithm for high dimensional data in `R` in a software package called *RAN-*

dom forest GEneRator (ranger) Wright and Ziegler (2015). Most of the regressors are categorical variables that need to be transformed into dummy variables increasing the number of dimensions, which can lead to problems when tuning the hyperparameters. Therefore, we selected this algorithm for regression because it provides high throughput and accuracy.

The hyperparameters of the ranger algorithm are:

- Number of trees (submodels or weak learners).
- Sample size: Minimum number of observations (generated by random sampling) needed to grow a tree.
- Number of independent variables.
- `mtry`: Number of predictors considered for each split of the tree.
- Target node size: Number of observations in the internal nodes.

The `caret` package has a function called `train()` that can be used to train Random Forest models by simply specifying the desired algorithm in the parameter `method="ranger"`:

```
rf_reg_ranger <- train(y ~ .,
                       data = train_data,
                       method = 'ranger',
                       metric = 'RMSE',
                       tuneLength  = 20,
                       trControl = control)
```

Random Ferns is a powerful tree-based algorithm developed for classification purposes that performs an over-sampling of the under-represented class, that is, counts (or any integer greater than one) so that the number of elements of each class is equal on each bag Kursa (2014). This algorithm has the capability to identify patterns in the under-represented class that could lead to more accurate predictions. Therefore, this algorithm has the capability to detect patterns in the data that could lead to accurate predictions of zeros and counts.

This method uses k random selections of D features out of P possible features or explanatory variables, that is, $j_k \in \{X_1, X_2, \ldots, X_p\}$, $k = 1, \ldots, K$, to grow *ferns*:

$$Y_i^p = \underset{y}{arg\,max} \prod_k P\left(X_{i,j_k} | Y_i = y\right) \tag{3.12}$$

Equation 3.12 shows that the predictions of the ferns are combined in *naive* way, resembling the naive Bayes classifier. The option of taking subsets of the D features instead of the total number of features enables the method to represent more complex interactions and patterns in the data, which can lead to higher accuracy than in the purely naive case Ozuysal et al. (2007).

The conditional probabilities $P\left(X_{i,\,j_k}|Y_i = y\right)$ are estimated using the empirical probabilities calculated from a training dataset (or fold) $(X_{i,\,j}^{(t)},\, Y_i^{(t)})$ of size $n^{(t)} \times p$. This means, that they represent relative frequencies of class in each subspace of the feature space.

Moreover, each fern performs a partition of the feature space into regions corresponding to all possible combinations of values of attributes j_k. Since the attribute subsets j_k are generated randomly, the random ferns classifier is equivalent to a random subspace ensemble of K constrained decision trees with the ability to oversample the underrepresented class, and thus balancing the number of observations of each class on each bag.

This algorithm has two hyperparameters that need to be tuned: the number of ferns (which is equivalent to the number of trees) and the depth of each one Kursa (2014). These two hyperparameters have to be determined via cross-validation.

This algorithm can be implemented in `R` by using the `rFerns` package, which can also be used through the `caret` package as follows:

```
rf_random <- train(y_clf_train ~ .,
                   data = Board_train_clf,
                   method = 'rFerns',
                   metric = 'Accuracy',
                   tuneLength  = 20,
                   trControl = control)
```

Finally, these two algorithms can be used in sequence to first predict a zero or a count using random ferns, and then, ranger to predict the count number in case the predicted *class* is a count. Figure 3.14 shows a schematic representation of the proposed algorithm (solution) and how it can used sequentially:

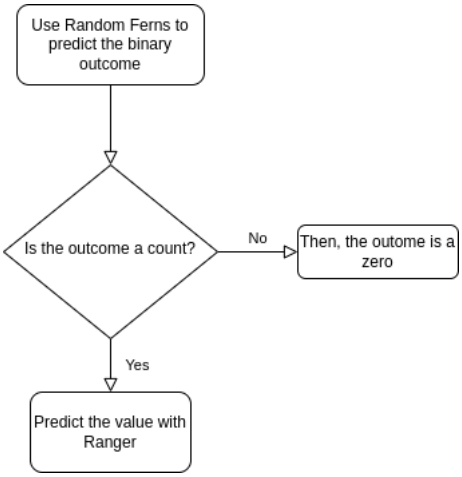

Figure 3.14: Algorithm for the proposed Zero-Inflated Random Forest model.

Also, the proposed solution can be automatized by creating a function in `R` to apply this method to any new set of observations. The appendix section (6) shows the complete code to run this function.

3.3.5 Artificial Neural Networks

Artificial Neural Networks are powerful methods that can take a wide variety of inputs $\{X_1, \ldots, X_p\}$ (also known as input layer) to use linear combinations of these independent variables to model the dependent variable. These linear combinations can be done in different ways depending on the architecture of the Network and its activation functions. They are highly flexible and can handle complex nonlinear patterns.

There can be multiple linear combinations of subsets of features, each one of these are called hidden units. This unit assigns weights to the considered features to represent the importance of the features in the linear combination. The set of hidden units that operate in parallel is called hidden layer. The number of hidden layers need to be estimated using cross-validation.

A small number of hidden units will decrease the flexibility of the model, in contrast, a large number of hidden units increases the flexibility of the model and its predictive power. However, setting a large number of number of hidden units can lead to overfitting Hastie et al. (2009).

During the forward propagation the method computes values by successively transforming the the input data (as features) through the layers, that is, the output of each layer is the input of the next layer layer. Figure 3.15 shows the acyclic graph structure of a neural network that uses forward propagation with two hidden layers with three hidden units per layer Ghatak (2019):

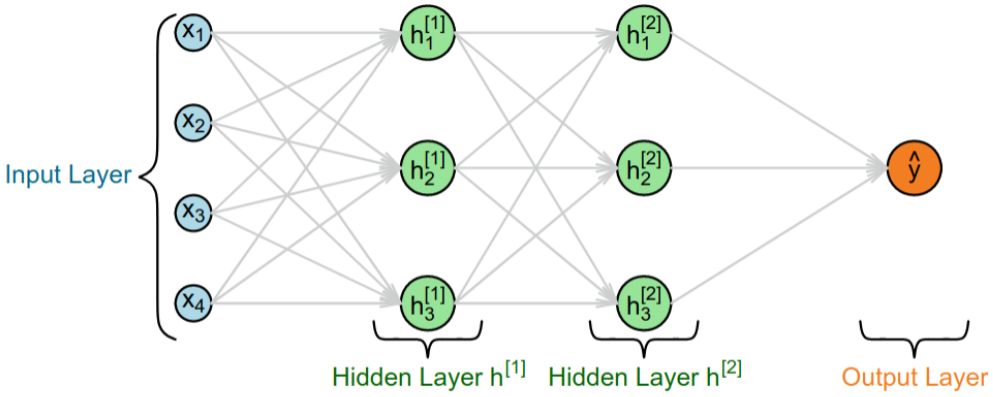

Figure 3.15: Representation of a two hidden layer artificial neural network Ghatak (2019).

Artificial neural networks can overfit the data easily, and thus require high signal-to-noise ratio to be effective. Also, the method is *overparametrized* and the optimization problems (to calculate the weights) are nonconvex. Therefore, it has some constraints with sample size and computer capabilities.

3.4 Model Selection

After training the eight models mentioned in the previous section we need to select the *best* model using an objective metric. The main purpose is to estimate the generalization performance (or error) of the models to determine their predictive power on independent test data. This metric will provide information about the *relative* performance of the models. In this section, we present the performance metric and how it was used to select the models.

There are many ways to compare model performances. The Akaike Information Criterion (AIC) measures the goodness of fit (accuracy) of the predictions and observations and then contrasts that with the complexity or number of parameters estimated by the model Akaike (1998). The preferred model is the one with the lowest AIC value, and more recent information criteria inherited this way of measuring the performance of the models. The main advantage of this approach is that it will provide information to select between models that can be accurate but are highly complex, and models that are less accurate but less complex (or easier to train). This is particularly important when hyperparameter tuning requires considerable computational power. Nonetheless, AIC is only valid when the models are trained using maximization of the log-likelihood function. Moreover, its performance decreases with the number of parameters (or dimensionality) of the models

leading to unsatisfactory results Bedrick and Tsai (1994).

The sample size, n, is another relevant factor that affects the performance of any criterion. For instance, the vector-corrected Kullback Information Criterion (KIC_{vc}) and AIC tend to give accurate information about model selection when n is small. In contrast, the multivariate Bayesian Information Criterion (MBIC) gives the best information when n is large Wu et al. (2013). However, these model selection approaches are designed for models that maximize the likelihood function, that is, they are built for parametric or regular models only. For instance, the maximum likelihood estimator diverges when applied to singular models (nonregular models), which can lead to overestimation of the generalization error.

Therefore, we consider cross-validation to estimate the generalization error of the models. The main advantage of this approach is that it can use any loss function and it can apply to models with different levels of complexity, ranging from generalized linear models to random forest methods. However, this approach gives information about the predictive power of the models and does not penalize the complexity or flexibility of the models.

K-Fold cross-validation is a popular approach to estimating the generalization error (and model hyperparameters or tuning). However, it requires repeated model fits and it can lead to problems when dealing with sparse data Gelman et al. (2014). Moreover, it is not clear if cross-validation estimates have an asymptotic behavior Watanabe and Opper (2010). In general, information criteria estimates can have unexpected behaviors when applied to singular models such as random forest or neural networks Watanabe (2001).

Figure 3.16 presents a schematic representation of a five-fold cross-validation. In this process, the training dataset is *randomly* divided into five folds using stratified sampling to keep similar proportions of zeros and counts across the folds:

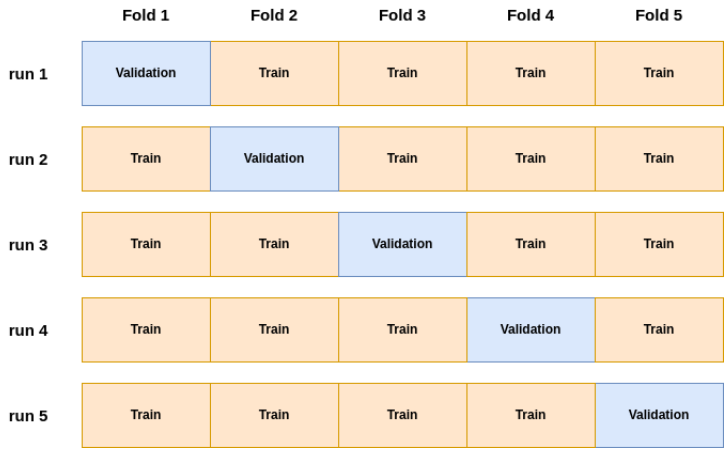

Figure 3.16: A five-fold ($K = 5$) cross-validation representation.

In the case of generalized linear models, which do not have hyperparameters to tune, we fit the model using folds 2 through 5. Also, we use fold 1 as an *independent* set to predict the outcomes of fold 1. Then, we calculate the RMSE for the predictions of the model on fold 1. This process is repeated iteratively until all the folds are used as *independent* folds.

Cross-Validation can also be used to tune hyperparameters. For instance, Random forest algorithms have hyperparameters to tune, and the number of tuning hyperparameters will define the hyperparameter space. Naturally, there is an infinite number of possible estimates for a given continuous (and discrete in some cases) hyperparameter, and thus, it is important to define optimum sampling plans to ensure that we are using representative samples of this hyperparameter space.

In this study, we consider a random search (*i.e*, using a uniform random distribution) of the hyperparameter space given that there was not any relevant information about the possible distribution of the tuned hyperparameters, θ. Also, we predefined the number of samples per hyperparameter to control the hyperparamter exploration considering computational time and resources as the main constraints. For instance, the ranger algorithm has five hyperparameters and if we predefined the sample size for the hyperparameters to 20, then there will be 20^5 samples points and the cross-validation method will have to test the performance of the model using 20^5 different combinations of hyperparameters. Despite the size of the sample size, there is no guarantee that this approach will find the true hyperparameter vector θ. This approach only gives an estimate or approximation of the true hyperparameters ($\hat{\theta}$).

Equation 3.13 shows how k-fold cross-validation is applied to a candidate model $\hat{f}$ considering a vector of

hyperparamter values θ using any loss function:

$$CV(\hat{f}, \theta) = \frac{1}{N} \sum_{i=1}^{N} L(y_i, \hat{f}^{-\kappa(i)}(x_i, \theta)) \tag{3.13}$$

The function $CV(\hat{f}, \theta)$ gives an estimate of the test error and information about $\hat{\theta}$ that minimizes it. The length or dimension of the θ will vary with the number of hyperparameters of the models.

CHAPTER 4

Results and Discussion

4.1 Demand: Board counts at the bus stop level

Setup and hyper-parameter tuning: Data collected through automated passenger counting devices can be noisy. For example, some trips in our data showed negative occupancy and some had unexpectedly high values. Buses operated by CARTA have a maximum capacity of 32 but at times, we observed occupancy close to 50. We assigned 0 to all negative occupancy values and replaced any value greater than the maximum capacity of a bus with its maximum capacity. We used a repeated (3 times) five-fold cross-validation with stratified sampling to tune hyper-parameters for our models.

Also, we tested a neural network model that had an input layer, two hidden layers, and one output neuron to predict board counts for the trip at the stop. In the input layer, there was one neuron for each predictor variable. The two dense hidden layers have 15, and 5 neurons, respectively. We use linear activation in the output layer and sigmoid activation in all the hidden layers. We optimize the model using the Adam optimizer with learning rate 0.001. For forecasting maximum occupancy, we modify the number of neurons in the dense hidden layers to 40 and 15 respectively. The parameters were chosen based on cross-validation. The neural networks were implemented using tensorflow-keras Abadi et al. (2015).

We use ranger, a fast implementation of random forests for high dimensional data Wright and Ziegler (2015) to train the random forest regressors. For classifiers based on random forests, we use random ferns Kursa (2014), that typically outperform other tree-based classifiers while learning on imbalanced data. Random ferns internally enforce the balance of *impact* made by each class by a process that resembles the standard procedure of oversampling under-represented classes so that the number of objects of each class stays similar Kursa (2014). For random forest models, we tuned the depth of the tree, minimum node size (minimum number of observations in the terminal nodes), the number of randomly drawn candidate variables out of which each split is selected when growing a tree, and the split rule (a categorical hyper-parameter that assesses splitting criteria in the nodes).

Null models were also considered as baseline models to assess the relative performance of the models. That is, we were able to compare specialized and flexible methods such as Hurdle and Random Forest models

with models without any predictors. We computed the mean board count and maximum occupancy from each training set. Then, we use these averages to determine the test RMSEs considering the board counts and maximum occupancies from the test sets.

4.2 Results

We start with presenting root mean squared errors in estimating board counts. We present aggregated results in the paper for the sake of brevity. Results on test data on pre-lockdown and post-lockdown periods are shown in Figures 4.1 and 4.2 respectively. We see that zero-inflated random forest models outperform other approaches, including neural networks in all cases. We also observe that in general, machine learning (or algorithmic) approaches to data-driven learning outperform statistical models.

Also, we observe that the zero-inflated random forest has the least variance in forecasts in the test set. Our results show that while random forest regression has been shown to be particularly powerful in learning complex forecasting models using heterogeneous data Fawagreh et al. (2014), the performance of such models can be compromised due to highly imbalanced or sparse data (where we define sparsity as the prevalence of zero counts). While Meller et al. Mellor et al. (2015) explored the effect of imbalanced data on ensemble learning methods such as random forests, we show that the combining powerful algorithmic regression models with ideas from the statistical modeling such as hierarchical learning can help improve accuracy.

We can see in figure 4.1 that our proposed approach, zero-inflated random forest modeling outperforms other approaches, including neural networks in all cases. We also observe that in general, machine learning (or algorithmic) approaches to data-driven learning outperform statistical models.

Null models do not necessarily underperform when compared with the considered models. The high proportion of zero counts in the data forces the central tendency statistics to be zero or close to zero. Besides that, the dependent variables have asymmetric distributions, and thus, the magnitude of their RMSE values reflects the difference between the observed board counts and the calculated mean (or mode) board counts. In particular, null models can outperform *full* or *complete* models in cases where there are few board counts and their value do not exceed one.

Figure 4.1 shows that the performance of null models for the routes with the highest demand can be similar or even better than Negative Binomial and Poisson models. Also, this figure shows that the performances of the Neural Network and null models are very similar in route 4 and direction 1.

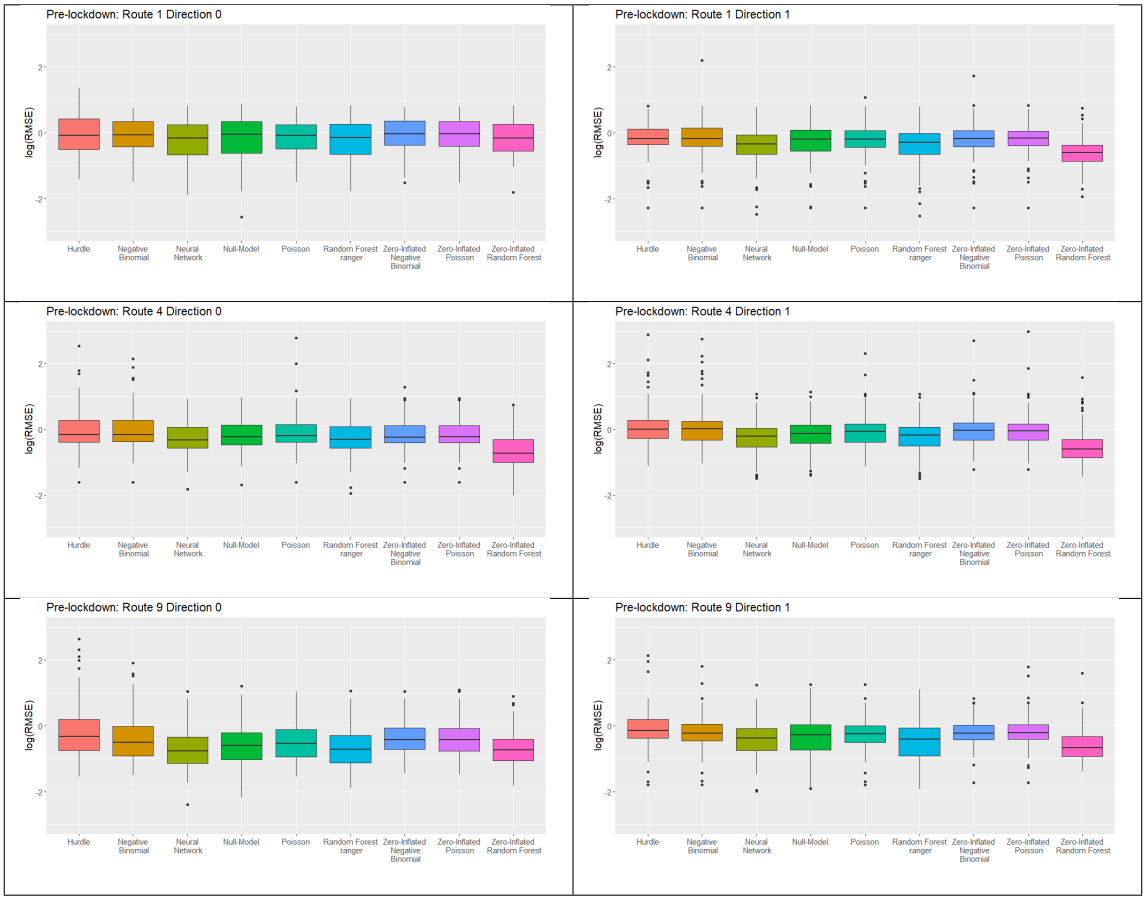

Figure 4.1: Aggregated root mean square errors on unseen data (test set) for bus stops grouped according to route and direction on pre-lockdown time period.

As with the results on pre-lockdown data shown in Figure 4.1, we see that our proposed approach, zero-inflated random forest modeling outperforms other approaches, including neural networks in all cases.

Likewise, we use the null model for comparison purposes. The proportion of zero counts is higher during post-lockdown conditions, and thus, the detection or prediction of counts is more challenging for all the models. Therefore, the test log-RMSEs of the null models tend to be lower. For instance, figure 4.2 shows that the median log-RMSE of the null model was lower than the proposed Zero-Inflated Random Forest model in route 9 and direction 0.

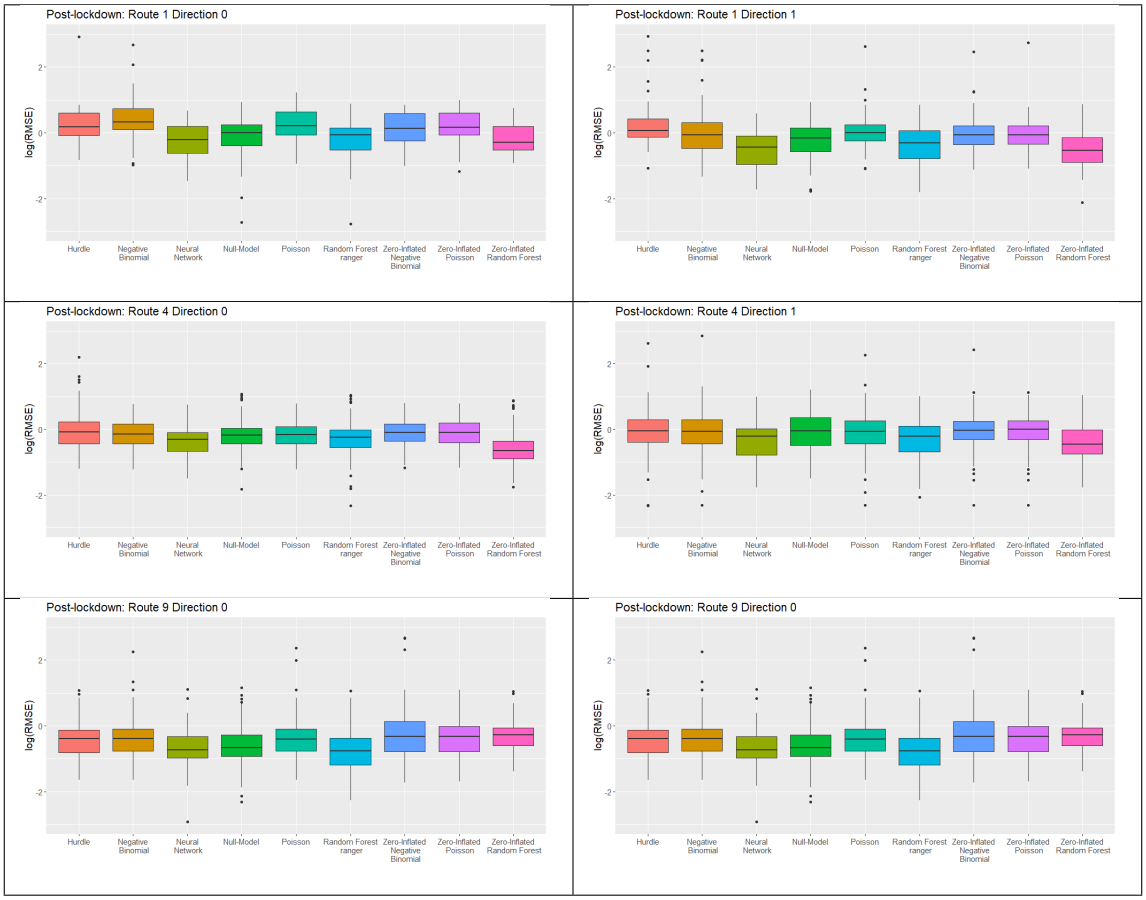

Figure 4.2: Aggregated root mean square errors on unseen demand data (test set) for bus stops grouped according to route and direction on post-lockdown time period.

Next, we present results on estimating maximum occupancy in trips. We present aggregated results on the entire dataset in Figure 6 (we present aggregated results for the sake of brevity). Our first observation is that statistical models perform significantly better on estimating maximum occupancy than the problem of estimating boarding counts. In fact, their performance is on par with ensemble learning. Also, neural networks perform considerably worse than all other models. We hypothesize that the poor performance is due to the lack of extensive training data. As we aggregate data from stops to calculate maximum occupancy, the volume of data shrinks. We also observe that standard random forest regression model generally outperforms its hierarchical (zero-inflated) counterpart.

4.3 Maximum Occupancy: Maximum load at a trip level

We summarize our key findings next. **1)** Our experimental analysis using real-world transit data reveals that standard statistical and algorithmic approaches to modeling are not accurate to estimate boarding events in public transit due to the extremely high concentration of zero counts in such data. **2)** We observe this be-

havior even with zero-inflated statistical models (e.g., zero-inflated Poisson regression), that are specifically designed to handle data which show an increased presence of zeros. We hypothesize that such models do not work on transit data since the presence of zeros is not only larger than usual but also dominating ($> 90\%$). **3)** We also find that zero-inflated random forests can successfully be used for such data. Our proposed approach first learns a classifier based on random forests to identify potential non-zero outputs and then uses a random forest regression model (ranger by Wright and Ziegler (2015)) based on random forests to predict counts greater than zeros. **4)** While modeling maximum occupancy in trips, we observe that zero-inflated models do not necessarily outperform standard approaches to forecasting because the number of zeros observed in the data is significantly lesser. Nonetheless, their overall performance is similar to other models such random forest and Poisson regression models. **5)** Based on our findings, we recommend that practitioners combine the zero-inflated modeling paradigm with other machine learning based approaches when they observe an extremely high volume of zero counts in data, but revert back to standard modeling paradigms when zero counts are lesser.

Figure 4.3 shows that the zero-inflated random forest approach does not perform as well as it does on estimating board counts, most likely due to the reduction in the number of zeros. Also, the performance of standard statistical models also improves with the reduction of zero counts.

We also measure the performance of the null models to gain more insights into the relative performance of the models. Figure 4.3 also shows the aggregated test RMSE values for routes 1, 4, and 9. These results suggest that the performance of the null models is similar to the statistical and algorithmic methods, and in most cases, the null models outperform the Neural Networks. Nonetheless, there is a significant overlap between the null models and the considered methods.

Clearly, algorithmic approaches such as Neural Networks and Random Forests are more flexible than null models. However, the quantity and quality of the explanatory variables always affect any model's performance. Therefore, these results suggest that the models need better explanatory variables to improve their relative performance Hao et al. (2022). That is, hour of the day, temperature, and precipitation may not be enough to explain the maximum occupancies across the trips.

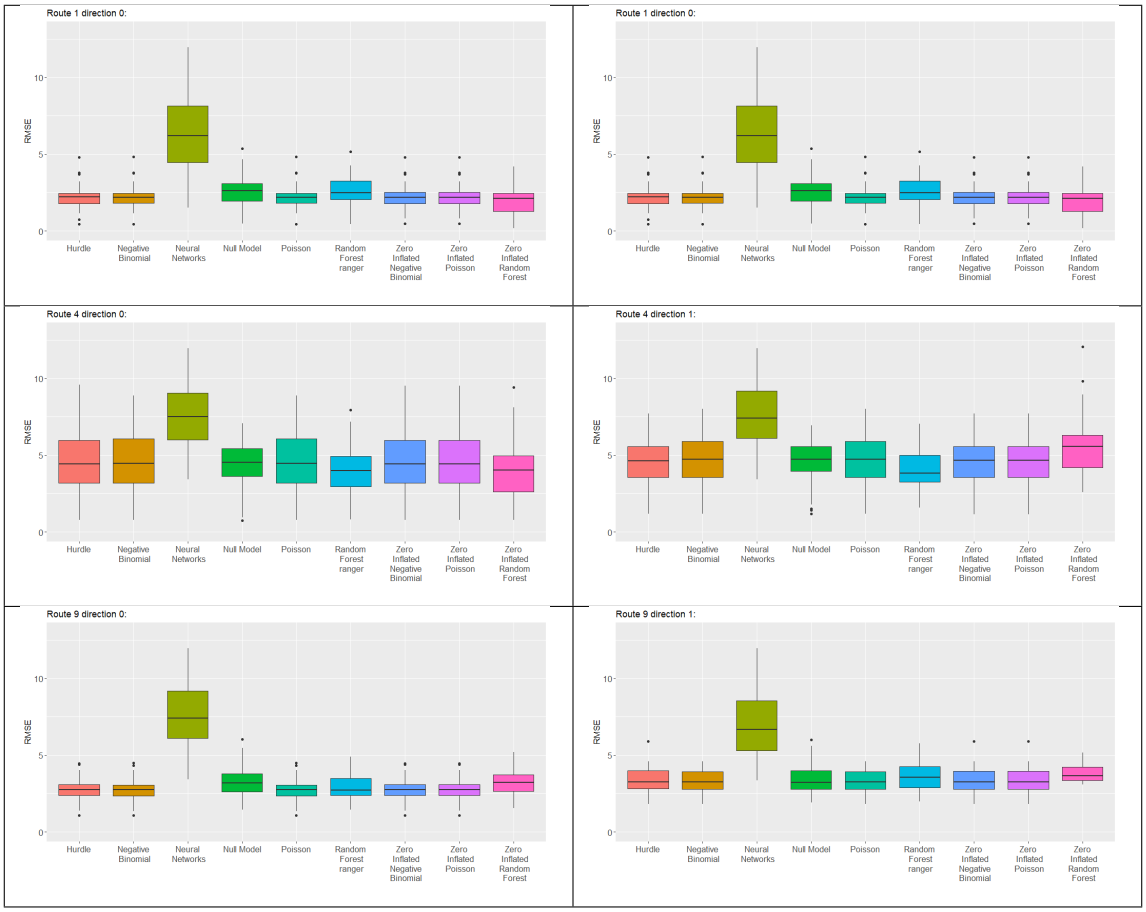

Figure 4.3: Performance of all the forecasting models on estimating maximum occupancy in trips on unseen (test) data.

CHAPTER 5

Conclusions

Estimating ridership patterns is imperative for planning and optimizing public transit. This exercise is particularly relevant when the world is dealing with a pandemic since social distancing norms must be strictly followed. However, estimating ridership patterns such as board counts and maximum occupancy during trips is not trivial. We show how standard approaches to statistical and algorithmic modeling can perform poorly on estimating board counts due to the high concentration of zeros in real-world data. We propose zero-inflated random forests, an approach that combines the hierarchical modeling paradigm of zero-inflated statistical models with the flexibility of ensemble learning. We show that the proposed approach outperforms state-of-the-art statistical and algorithmic approaches to data-driven modeling, including neural networks. We also observe that the proportion of zero counts in transit data is significantly reduced when maximum occupancy is modeled instead of board counts at individual stops. In such a scenario, we find that zero-inflated models perform (relatively) poorly in comparison with other approaches. The main disadvantage of the proposed approach is that it requires more computational time than the zero-inflated models. Our implementation is completely open-source for transportation engineers and urban planners to use.

CHAPTER 6

Appendix

The complete code is available on the following GitHub page:

- https://github.com/smarttransit-ai/transit-occupancy-analysis/tree/master/app/analysis

The following section simply describes the developed function written in `R` to train the proposed Zero-Inflated Random Forest model. There are multiple notebooks that describe the data preparation process (some of them written in Python) and model training.

The indentation of the following code has been changed to fit the margins of the page.

6.1 Proposed Solution: Code

```
RF_Ferns_and_Ranger <- function(rt, di, st, part){
    library(randomForest)
    library(mlbench)
    library(caret)
    library(e1071)
    library(dplyr)
    library(tidyr)
    library(readr)
    library(ranger)
    library(janitor)
    library(rFerns)
    library(ordinalForest)
    library(RRF)
    library(foreach)
    library(doParallel)

    path = paste0('data', '/', 'jmartinez', '/', 'Data_for_RF_Models',
                  '/', 'Board_Counts', '/',
                  paste('route', rt, sep = '_'), '/',
                  paste('direction', di, sep = ''), '/',
                  paste('bus_stop', st, sep = '_'), '/')
```

```
if(part == 'pre'){
    file_path_train = paste(path, 'pre_lock_train_data.csv', sep = '/')
    file_path_test = paste(path, 'pre_lock_test_data.csv', sep = '/')
    board_train = read_csv(file_path_train)
    board_test = read_csv(file_path_test)
    board_train$month = factor(board_train$month)
    board_train$service_kind = factor(board_train$service_kind)
    board_train$hour = factor(board_train$hour)
    board_test$month = factor(board_test$month)
    board_test$service_kind = factor(board_test$service_kind)
    board_test$hour = factor(board_test$hour)

else if(part == 'post'){
    file_path_train = paste(path, 'post_lock_train_data.csv', sep = '/')
    file_path_test = paste(path, 'post_lock_test_data.csv', sep = '/')
    board_train = read_csv(file_path_train)
    board_test = read_csv(file_path_test)
    board_train$month = factor(board_train$month)
    board_train$service_kind = factor(board_train$service_kind)
    board_train$hour = factor(board_train$hour)
    board_test$month = factor(board_test$month)
    board_test$service_kind = factor(board_test$service_kind)
    board_test$hour = factor(board_test$hour)
}
else{
    file_path_train = paste(path, 'train_data.csv', sep = '/')
    file_path_test = paste(path, 'test_data.csv', sep = '/')
    board_train = read_csv(file_path_train)
    board_test = read_csv(file_path_test)
    board_train$month = factor(board_train$month)
    board_train$service_kind = factor(board_train$service_kind)
    board_train$hour = factor(board_train$hour)
    board_test$month = factor(board_test$month)
    board_test$service_kind = factor(board_test$service_kind)
```

```
    board_test$hour = factor(board_test$hour)
}
train_month_levels = length(levels(board_train$month))
train_service_kind_levels = length(levels(board_train$service_kind))
train_hour_levels = length(levels(board_train$hour))

board_test = board_test %>%
    filter(hour %in% intersect(unique(board_test$hour),
           unique(board_train$hour)))

if(train_month_levels > 1){
    if(train_service_kind_levels > 1){
        if(train_hour_levels > 1){
            board_train = board_train
            board_test = board_test
        }
        else{
          board_train = board_train[,
          -which(names(board_train) == 'hour')]
          board_test = board_test[,
          -which(names(board_test) == 'hour')]
        }
    }
    else{
        if(train_hour_levels > 1){
          board_train = board_train[,
          -which(names(board_train) == 'service_kind')]
          board_test = board_test[,
          -which(names(board_test) == 'service_kind')]
        }
        else{
          board_train = board_train[,
          -which(names(board_train) == c('service_kind', 'hour'))]
          board_test = board_test[,
          -which(names(board_test) == c('service_kind', 'hour'))]
```

```
        }
    }
}
else{
    if(train_service_kind_levels > 1){
        if(train_hour_levels > 1){
            board_train = board_train[,
            -which(names(board_train) == 'month')]
            board_test = board_test[,
            -which(names(board_test) == 'month')]
        }
        else{
            board_train = board_train[,
            -which(names(board_train) %in% c('month', 'hour'))]
            board_test = board_test[,
            -which(names(board_test) %in% c('month', 'hour'))]
        }
    }
    else{
        if(train_hour_levels > 1){
            board_train = board_train[,
            -which(names(board_train) == c('month', 'service_kind'))]
            board_test = board_test[,
            -which(names(board_test) == c('month', 'service_kind'))]
        }
        else{
            board_train = board_train[,
            -which(names(board_train) == c('month',
            'service_kind', 'hour'))]
            board_test = board_test[,
            -which(names(board_test) == c('month',
            'service_kind', 'hour'))]
        }
    }
}
```

```
board_train = remove_empty(board_train,
              which = c('cols'), quiet = TRUE)
board_test = remove_empty(board_test,
             which = c('cols'), quiet = TRUE)

train_board_counts = unique(board_train$board_count)
test_board_counts = unique(board_test$board_count)

n_row_train = nrow(board_train)

if(n_row_train < 60){
    return('Insufficient data for analysis!')
}
else if(n_row_train >= 60){

    if(length(train_board_counts) > 1){
        y_clf_train = board_train$board_count
        y_clf_train = factor(if_else(y_clf_train == 0, 0, 1))

        y_clf_test = board_test$board_count
        y_clf_test = factor(if_else(y_clf_test == 0, 0, 1))

        Board_train_clf <- data.frame(cbind(y_clf_train,
                           board_train[, -c(1)]))
        Board_test_clf <- data.frame(cbind(y_clf_test,
                          board_test[, -c(1)]))
        #-------------------------------------------------
        # Training characteristics for model tuning:
        #-------------------------------------------------
        control <- trainControl(method = 'repeatedcv',
                                number = 5,
                                repeats = 2,
                                search = 'random')
        #-------------------------------------------------
```

```
# Classification using Random Ferns:
#-------------------------------------------------
set.seed(1)
rf_random <- train(y_clf_train ~ .,
                   data = Board_train_clf,
                   method = 'rFerns',
                   metric = 'Accuracy',
                   tuneLength  = 20,
                   trControl = control)

RF_Ferns <- print(rf_random)
rf_random_pred <- predict(rf_random,
                  newdata = Board_test_clf)

rf_random_conf_mat <- confusionMatrix(y_clf_test,
                      rf_random_pred)
rf_random_conf_mat <- data.frame(rf_random_conf_mat[4])
colnames(rf_random_conf_mat) <- c('Value')

# Index for regression data:
index_for_reg <- which(rf_random_pred == '1', arr.ind = T)
#-------------------------------------------------------
# Regression Model using Ranger:
set.seed(1)
rf_reg_ranger <- train(board_count ~ .,
                       data = (board_train %>% filter(board_count > 0)),
                       method = 'ranger',
                       metric = 'RMSE',
                       tuneLength  = 20,
                       trControl = control)

RF_Ranger <- print(rf_reg_ranger)
#-------------------------------------------------------
# Validation:
```

```
Board_Test_Val = board_test
nrow_test = (1:nrow(board_test))

Board_Test_Val$index = nrow_test
Board_test_reg = Board_Test_Val[index_for_reg, ]

rf_reg_ranger_pred <- predict(rf_reg_ranger,
                      newdata = Board_test_reg)
Board_test_reg$Ranger_Pred = rf_reg_ranger_pred

Board_Test_Val = left_join(Board_Test_Val,
                 Board_test_reg, by = 'index')

Board_Test_Val = Board_Test_Val %>%
mutate(RF_Pred = if_else(is.na(Ranger_Pred) == T,
                 0, Ranger_Pred))

board_test$RF_Pred = Board_Test_Val$RF_Pred

RF_test_RMSE = sqrt(mean((board_test$board_count -
                          board_test$RF_Pred)^{2}))

if(part == 'pre'){
    file_path_clf = paste(path, 'pre_lock_RF_Fern.txt', sep = '/')
    file_path_clf_conf_mat = paste(path,
                                   'pre_Conf_Mat_RF_Fern.csv',
                                   sep = '/')

    file_path_reg = paste(path, 'pre_lock_RF_Reg.txt', sep = '/')
    file_path_RF_Chart = paste(path, 'pre_RF_Chart.csv', sep = '/')

    final_clf_model = paste(path,
                      'Pre_Random_Ferns_model.rds')
    final_reg_model = paste(path,
                      'Pre_Random_Forest_RANGER_model.rds')
```

```
    write.table(RF_Ferns, file_path_clf)
    write.csv(rf_random_conf_mat, file_path_clf_conf_mat)

    write.table(RF_Ranger, file_path_reg)
    write.csv(board_test, file_path_RF_Chart)

    saveRDS(rf_random, final_clf_model)
    saveRDS(rf_reg_ranger, final_reg_model)

}
else if(part == 'post'){
    file_path_clf = paste(path,
                    'post_lock_RF_Fern.txt', sep = '/')
    file_path_clf_conf_mat = paste(path,
                                   'post_Conf_Mat_RF_Fern.csv',
                                   sep = '/')
    file_path_reg = paste(path,
                    'post_lock_RF_Reg.txt', sep = '/')
    file_path_RF_Chart = paste(path,
                         'post_RF_Chart.csv', sep = '/')
    final_clf_model = paste(path,
                      'Post_Random_Ferns_model.rds')
    final_reg_model = paste(path,
                      'Post_Random_Forest_RANGER_model.rds')
    write.table(RF_Ferns, file_path_clf)
    write.csv(rf_random_conf_mat, file_path_clf_conf_mat)
    write.table(RF_Ranger, file_path_reg)
    write.csv(board_test, file_path_RF_Chart)
    saveRDS(rf_random, final_clf_model)
    saveRDS(rf_reg_ranger, final_reg_model)
}
else{
    file_path_clf = paste(path, 'RF_Fern.txt', sep = '/')
    file_path_clf_conf_mat = paste(path, 'Conf_Mat_RF_Fern.csv',
```

```
                                                sep = '/')
                    file_path_reg = paste(path, 'RF_Reg.txt', sep = '/')
                    file_path_RF_Chart = paste(path, 'pre_RF_Chart.csv', sep = '/')
                    final_clf_model = paste(path, 'Random_Ferns_model.rds')
                    final_reg_model = paste(path, 'Random_Forest_RANGER_model.rds')
                    write.table(RF_Ferns, file_path_clf)
                    write.csv(rf_random_conf_mat, file_path_clf_conf_mat)
                    write.table(RF_Ranger, file_path_reg)
                    write.csv(board_test, file_path_RF_Chart)
                    saveRDS(rf_random, final_clf_model)
                    saveRDS(rf_reg_ranger, final_reg_model)
                }
                return('Done!')
            }
            else{
            return('This bus stop does not have variability in the response variable.')
            }
        }
       }
      ]
    }
```

References

Abadi, M., Agarwal, A., Barham, P., Brevdo, E., Chen, Z., Citro, C., Corrado, G. S., Davis, A., Dean, J., Devin, M., Ghemawat, S., Goodfellow, I., Harp, A., Irving, G., Isard, M., Jia, Y., Jozefowicz, R., Kaiser, L., Kudlur, M., Levenberg, J., Mané, D., Monga, R., Moore, S., Murray, D., Olah, C., Schuster, M., Shlens, J., Steiner, B., Sutskever, I., Talwar, K., Tucker, P., Vanhoucke, V., Vasudevan, V., Viégas, F., Vinyals, O., Warden, P., Wattenberg, M., Wicke, M., Yu, Y., and Zheng, X. (2015). TensorFlow: Large-scale machine learning on heterogeneous systems. Software available from tensorflow.org.

Akaike, H. (1998). Information theory and an extension of the maximum likelihood principle. *Selected papers of hirotugu akaike*, pages 199–213.

Bedrick, E. J. and Tsai, C.-L. (1994). Model selection for multivariate regression in small samples. *Biometrics*, pages 226–231.

Blackley, P. R. (1990). Spatial mismatch in urban labor markets: Evidence from large us metropolitan areas. *Social Science Quarterly*, 71(1):39.

Breiman, L. (2001a). Random forests. *Machine learning*, 45(1):5–32.

Breiman, L. (2001b). Statistical modeling: The two cultures (with comments and a rejoinder by the author). *Statistical science*, 16(3):199–231.

Breiman, L., Friedman, J. H., Olshen, R. A., and Stone, C. J. (1984). *Classification and regression trees*. Routledge.

Brooks, J. H. M., Tingay, R., and Varney, J. (2021). Social distancing and COVID-19: an unprecedented active transport public health opportunity. *British Journal of Sports Medicine*, 55(8):411–412.

Brough, R., Freedman, M., and Phillips, D. (2020). Understanding Socioeconomic Disparities in Travel Behavior during the COVID-19 Pandemic. *SSRN Electronic Journal*.

Buehler, R. and Pucher, J. (2012). Demand for public transport in germany and the usa: an analysis of rider characteristics. *Transport Reviews*, 32(5):541–567.

Cameron, A. C. and Trivedi, P. K. (2013). *Regression analysis of count data*, volume 53. Cambridge university press.

Ceder, A. (1984). Bus frequency determination using passenger count data. *Transportation Research Part A: General*, 18(5-6):439–453.

Couture, V., Dingel, J. I., Green, A., Handbury, J., and Williams, K. (2020). Measuring movement and social contact with smartphone data: a real-time application to covid-19. *NBER Working Paper*, (w27560).

Cragg, J. G. (1971). Some statistical models for limited dependent variables with application to the demand for durable goods. *Econometrica: Journal of the Econometric Society*, pages 829–844.

Darsena, D., Gelli, G., Iudice, I., and Verde, F. (2020). Safe and reliable public transportation systems (salutary) in the covid-19 pandemic. *arXiv preprint arXiv:2009.12619*.

Fawagreh, K., Gaber, M. M., and Elyan, E. (2014). Random forests: from early developments to recent advancements. *Systems Science & Control Engineering: An Open Access Journal*, 2(1):602–609.

Feng, C. X. (2021). A comparison of zero-inflated and hurdle models for modeling zero-inflated count data. *Journal of statistical distributions and applications*, 8(1):1–19.

Gelman, A., Hwang, J., and Vehtari, A. (2014). Understanding predictive information criteria for bayesian models. *Statistics and computing*, 24(6):997–1016.

Ghatak, A. (2019). *Deep learning with R*. Springer.

Guo, Z., Wilson, N. H., and Rahbee, A. (2007). Impact of weather on transit ridership in chicago, illinois. *Transportation Research Record*, 2034(1):3–10.

Hao, X., Hu, X., Liu, T., Wang, C., and Wang, L. (2022). Estimating urban pm2. 5 concentration: An analysis on the nonlinear effects of explanatory variables based on gradient boosted regression tree. *Urban Climate*, 44:101172.

Hastie, T., Tibshirani, R., and Friedman, J. (2009). *The elements of statistical learning: data mining, inference, and prediction*. Springer Science & Business Media.

He, H. and Ma, Y. (2013). *Imbalanced learning: foundations, algorithms, and applications*. Wiley-IEEE Press.

Hofmann, M. and O'Mahony, M. (2005). The impact of adverse weather conditions on urban bus performance measures. In *Proceedings. 2005 IEEE Intelligent Transportation Systems, 2005.*, pages 84–89. IEEE.

Horaždovskỳ, P., Novotnỳ, V., and Svítek, M. (2018). Data-driven management of dynamic public transport. In *2018 Smart City Symposium Prague (SCSP)*, pages 1–5. IEEE.

Hu, S. and Chen, P. (2021). Who left riding transit? Examining socioeconomic disparities in the impact of COVID-19 on ridership. *Transportation Research Part D: Transport and Environment*, 90:102654.

Jackman, S. (2010). pscl: Classes and methods for r. developed in the political science computational laboratory, stanford university. department of political science, stanford university, stanford, ca. r package version 1.03. 5. *http://www. pscl. stanford. edu/*.

James, G., Witten, D., Hastie, T., and Tibshirani, R. (2013). *An introduction to statistical learning*, volume 112. Springer.

Karnberger, S. and Antoniou, C. (2020). Network–wide prediction of public transportation ridership using spatio–temporal link–level information. *Journal of Transport Geography*, 82:102549.

Kuhn, M. (2022). *caret: Classification and Regression Training*. R package version 6.0-92.

Kursa, M. B. (2014). **rFerns** : An Implementation of the Random Ferns Method for General-Purpose Machine Learning. *Journal of Statistical Software*, 61(10).

Lao, X., Zhang, X., Shen, T., and Skitmore, M. (2016). Comparing china's city transportation and economic networks. *Cities*, 53:43–50.

LeCun, Y., Bengio, Y., and Hinton, G. (2015). Deep learning. *nature*, 521(7553):436–444.

Medo, M. (2020). Epidemic spreading on spatial networks with distance-dependent connectivity. *arXiv preprint arXiv:2003.13160*.

Mellor, A., Boukir, S., Haywood, A., and Jones, S. (2015). Exploring issues of training data imbalance and mislabelling on random forest performance for large area land cover classification using the ensemble margin. *ISPRS Journal of Photogrammetry and Remote Sensing*, 105:155–168.

Menon, A. K. and Lee, Y. (2017). Predicting short-term public transport demand via inhomogeneous poisson processes. In *Proceedings of the 2017 ACM on Conference on Information and Knowledge Management*, pages 2207–2210.

Miao, Q., Welch, E. W., and Sriraj, P. (2019). Extreme weather, public transport ridership and moderating effect of bus stop shelters. *Journal of Transport Geography*, 74:125–133.

Min, Y. and Agresti, A. (2005). Random effect models for repeated measures of zero-inflated count data. *Statistical modelling*, 5(1):1–19.

Montero-Lamas, Y., Orro, A., Novales, M., and Varela-García, F.-A. (2022). Analysis of the relationship between the characteristics of the areas of influence of bus stops and the decrease in ridership during covid-19 lockdowns. *Sustainability*, 14(7):4248.

Mukhopadhyay, A., Pettet, G., Vazirizade, S., Vorobeychik, Y., Kochenderfer, M., and Dubey, A. (2020). A review of emergency incident prediction, resource allocation and dispatch models. *arXiv preprint arXiv:2006.04200*.

Mullahy, J. (1986). Specification and testing of some modified count data models. *Journal of econometrics*, 33(3):341–365.

Nuzzolo, A., Crisalli, U., Rosati, L., and Ibeas, A. (2013). Stop: a short term transit occupancy prediction tool for aptis and real time transit management systems. In *16th International IEEE Conference on Intelligent Transportation Systems (ITSC 2013)*, pages 1894–1899. IEEE.

Olmo, J. and Sanso-Navarro, M. (2020). Modelling and forecasting the spread of COVID-19 in New York City. page 30.

Ozuysal, M., Fua, P., and Lepetit, V. (2007). Fast keypoint recognition in ten lines of code. In *2007 IEEE Conference on Computer Vision and Pattern Recognition*, pages 1–8. Ieee.

R Core Team (2013). *R: A Language and Environment for Statistical Computing*. R Foundation for Statistical Computing, Vienna, Austria. ISBN 3-900051-07-0.

Rhys, H. (2020). *Machine Learning with R, the tidyverse, and mlr*. Simon and Schuster.

Sanchez, T. W. (2008). Poverty, policy, and public transportation. *Transportation Research Part A: Policy and Practice*, 42(5):833–841.

Stover, V. W. and McCormack, E. D. (2012). The impact of weather on bus ridership in pierce county, washington. *Journal of Public Transportation*, 15(1):95–110.

Tirachini, A. (2020). COVID-19 and Public Transportation: Current Assessment, Prospects, and Research Needs. page 21.

Van Oort, N., Brands, T., and de Romph, E. (2015). Short term ridership prediction in public transport by processing smart card data. *Transportation Research Record*, 2535(1):105–111.

Venables, W. N. and Ripley, B. D. (2002). *Modern Applied Statistics with S*. Springer, New York, fourth edition. ISBN 0-387-95457-0.

Watanabe, S. (2001). Algebraic analysis for nonidentifiable learning machines. *Neural Computation*, 13(4):899–933.

Watanabe, S. and Opper, M. (2010). Asymptotic equivalence of bayes cross validation and widely applicable information criterion in singular learning theory. *Journal of machine learning research*, 11(12).

Wilbur, M., Ayman, A., Ouyang, A., Poon, V., Kabir, R., Vadali, A., Pugliese, P., Freudberg, D., Laszka, A., and Dubey, A. (2020). Impact of covid-19 on public transit accessibility and ridership. *Proceedings of the Annual Conference of Transportation Research Board*.

Wright, M. N. and Ziegler, A. (2015). ranger: A fast implementation of random forests for high dimensional data in c++ and r. *arXiv preprint arXiv:1508.04409*.

Wu, T.-J., Chen, P., and Yan, Y. (2013). The weighted average information criterion for multivariate regression model selection. *Signal Processing*, 93(1):49–55.

Zeileis, A., Kleiber, C., and Jackman, S. (2008). Regression models for count data in r. *Journal of statistical software*, 27(8):1–25.

Zhou, M., Wang, D., Li, Q., Yue, Y., Tu, W., and Cao, R. (2017). Impacts of weather on public transport ridership: Results from mining data from different sources. *Transportation research part C: emerging technologies*, 75:17–29.