Navigating Machine Learning for Travel Behavior Analysis: A Comprehensive Guide to Inferring Trip Purposes Using Transit Survey and Automatic Fare Collection Data Fusion

Kwangho Baek[a], Alireza Khani[a,*]

[a] *University of Minnesota, Minneapolis, MN 55455, United States*

ARTICLE INFO

Article history:
Preprint Submitted to Elsevier on 27 Sep 2025

Keywords:
Machine Learning
Travel Behavior Analysis
Trip Purpose Inference
Data Fusion
Hyperparameter Tuning
Feature Engineering
Automatic Fare Collection (AFC)

ABSTRACT

Applying machine learning (ML) to travel behavior analysis can be challenging due to the extensive model selection, feature engineering, and hyperparameter tuning involved. To address these complexities, this study provides a comprehensive, step-by-step guideline, illustrated through a trip purpose inference application that utilizes transit onboard survey (OBS) and automatic fare collection (AFC) data fusion. Two primary stages of experiments are conducted: first, systematically comparing different feature engineering techniques—including positional encoding and categorical variable encoding methods—across five popular ML models (random forest, XGBoost, CatBoost, neural networks, and support vector machines); and second, examining the impacts of detailed hyperparameter tuning using an automated optimization framework. Results demonstrate that while appropriate feature engineering significantly enhances model performance and stability, hyperparameter tuning remains crucial to achieving optimal predictive accuracy. An adjusted application of positional encoding in a tabular form effectively captures spatial information. In contrast, advanced categorical encoding methods show limited benefit due to the low cardinality of the categorical variables used. Finally, a case study employing the best-performing model validates the feasibility of OBS-AFC fusion, enabling continuous, spatiotemporally detailed tracking of changes in transit travel behavior, particularly highlighting shifts in commute patterns during and after the COVID-19 pandemic. While the proposed methodology demonstrates robustness, practical applications require consideration of dataset limitations, including feature completeness and temporal proximity between surveys and AFC data collection periods.

1. Introduction

In an era of rapid technological change and evolving mobility patterns, the landscape of transit travel behavior is undergoing significant transformation. The rise of emerging mobility options, such as micromobility and ride-hailing, has introduced new dynamics that either complement or compete with traditional transit systems (Baek et al., 2021; Kong et al., 2020). Moreover, increased flexibility in work arrangements following the COVID-19 pandemic has resulted in persistent shifts in travel patterns (Balbontin et al., 2024; Barrero et al., 2021).

Despite these, transit planning methodologies remain largely reliant on infrequent, survey-based data collection methods. Frequently referred to as an "active" travel behavior dataset, surveys are rich in behavioral and demographic information, but are typically conducted only a few times a decade, limiting their usefulness in capturing evolving travel patterns. In contrast, "passive" data sources, such as Automated Fare Collection (AFC) or smart card data, offer continuous, high-volume observations, but often lack key behavioral variables such as trip purpose.

To complement the limitations of the two types of datasets, recent studies have explored various data fusion strategies that transfer knowledge from active datasets to passive ones. These methods aim to "learn" patterns from survey-based data by developing models that can infer or label unobserved variables in passive data records. While early approaches used rule-based or statistical inference models (Alsger et al., 2018; Faroqi and Mesbah, 2021; Hossain and Habib, 2021), recent work has adopted supervised machine learning (ML) classifiers (Kim et al., 2021; Zhu, 2020), semi-supervised learning (Gao et al., 2024) and generative models (Kim et al., 2022) for improved accuracy.

Building on this line of work, we develop a methodology for fusing transit On-Board Surveys (OBS) and AFC data using supervised ML models trained to infer trip purpose—a critical planning variable missing from AFC data. Our primary contribution lies in the rigorous examination of technical modeling decisions in travel behavior research that, despite their crucial role, are not always thoroughly discussed: the impact of *feature engineering* (the process of transforming and encoding input variables to improve model performance) and *hyperparameter tuning* (the task of optimizing model-specific settings that are not learned during training) on model performance or prediction accuracy. This work is not limited to the specific OBS-AFC fusion task we demonstrate; instead, it offers practical and theoretical guidance relevant to a wide range of ML-based travel behavior analyses whose datasets involve transportation-specific features, namely the times and locations associated with trip origins and destinations.

To establish this guideline rigorously, we conducted a systematic, large-scale two-stage experiment with 1,014 model optimization studies, each with 50 modeling trials, to determine which feature engineering strategies are optimal for which model. They are then followed by 10 in-depth model-specific studies, each with 110 modeling trials, which collectively sum to 51,800 model fittings for an exhaustive comparison. Our findings from the above comprehensive and controlled experiment reveal that widely held performance rankings—or the comparative advantages often suggested in other ML-application papers in travel behavior analysis—can be overturned. This is especially true when such claims are not evaluated under carefully controlled feature engineering and hyperparameter tuning conditions. In other words, typical statements like "Model X is better than Model Y as its accuracy is Z percentage points higher" can easily be rebutted if not accompanied by the full revelation of such steps applied to the study. That is, even within the same study context, applying different strategies—either pre-model or within-model—can lead to entirely different conclusions.

In addition, our proposed approach is readily transferable. OBS is one of the most standardized survey instruments in transit planning (Memarian et al., 2012; Schaller, 2005), and AFC systems have been widely adopted in urban transit networks. Although OBS and AFC data differ substantially in structure and available variables, we introduce a *feature extraction* framework utilizing existing algorithms to harmonize and align their attributes. We provide detailed guidelines before discussing ML models for adapting this process.

* Corresponding author
E-mail address: akhani@umn.edu (A. Khani).

Finally, we demonstrated our OBS–AFC fusion framework using data from the Minneapolis-St. Paul metropolitan area (the Twin Cities). The results show that data fusion enables continuous, localized analysis of travel behavior that would not be possible using OBS data alone.

In essence, this paper addresses the research question: How can we effectively employ machine learning classification models for travel behavior analysis? Demonstrated with an example of augmenting AFC data for adaptive understanding of transit travel behavior, we focus on:

- Data preparation and feature extraction: methods to extract, augment, and align features in OBS and AFC to enable data fusion.
- Feature engineering: comparative evaluation of preprocessing methods to transform raw features, especially categorical and spatiotemporal variables, into formats optimal for ML models.
- Model configuration: comparative evaluation of ML models and hyperparameter tuning to achieve higher model performance.
- Case study: application of survey-trained ML models to AFC for transit commute behavior analysis during and after the pandemic.

The remainder of this paper is organized as follows. **Section 2** reviews the literature on trip purpose inference and ML applications in travel behavior analysis, followed by **Section 3** presenting methods for extracting and aligning features from survey and AFC datasets to enable data fusion. In **Section 4**, we detail the ML models and feature engineering techniques, linked to **Section 5** that describes the design of our comparative experiments, whose results are discussed in **Section 6**. Finally, **Section 7** demonstrates the selected model in a case study of survey-AFC fusion.

2. Literature Review

Trip purpose is a fundamental determinant of travel behavior, influencing decisions related to choosing travel destination, travel mode, and departure time. As articulated in conventional transportation planning, travel is considered a derived demand (Stopher and Meyburg, 1975), undertaken to fulfill specific activities such as work, shopping, or leisure. The utility derived from these activities must outweigh the associated travel costs—be it time, money, or inconvenience—for a trip to occur. Consequently, individuals are more inclined to undertake mandatory trips, like commuting to work or catching a flight, even at higher travel costs, whereas discretionary trips are more susceptible to being deferred or canceled in response to minor cost increases.

Understanding and accurately imputing trip purpose for travel behavior analysis is crucial, as it enhances the precision of travel demand models and informs effective transportation planning and policymaking (Ortúzar and Willumsen, 2011). This aligns with a growing body of methodological work that leverages trip purposes to explain better or predict various travel behaviors. For example, Zailani et al. (2016) applied the theory of planned behavior to examine individuals' intentions to use public transit for different trip purposes. Their findings demonstrated that incorporating normative data, which is often overlooked in traditional models, significantly enhances the understanding of travel intention for each trip purpose. Moreover, with the advent of new technologies, the use of trip purpose information in travel behavior analysis has expanded into emerging mobility services, such as bike sharing (Wang et al., 2024a), ride sourcing (Aghaabbasi et al., 2020), and mobility-as-a-service systems (Duan et al., 2022)—highlighting the continued importance of tracking trip purposes as the transportation landscape evolves.

In parallel, as new passive or automated data sources emerge, researchers are increasingly exploring their potential to recover or infer trip purpose, thereby enriching behavioral insights. For instance, the adoption of personal vehicle- or taxi-mounted global positioning system (GPS) devices opened new opportunities to infer trip purpose from high-resolution travel trajectory data (Bohte and Maat, 2009; Luo et al., 2021). This approach has since been extended to smartphone-embedded GPS-based travel surveys, some of which even link trip trajectories with the demographic attributes of the device owner (Montini et al., 2014; Xiao et al., 2016).

In the context of transit systems, inferring trip purpose from AFC datasets has also emerged as a significant research topic. Some efforts primarily relied on rule-based or statistical approaches (e.g., distance-based clustering) that mainly utilized the spatiotemporal context of AFC-recorded trips (Alsger et al., 2018; Aminpour and Saidi, 2025; Devillaine et al., 2012; Faroqi and Mesbah, 2021; Lee and Hickman, 2014). Meanwhile, with the advancements of computing power, AI applications, such as feedforward neural networks (Sari Aslam et al., 2021) and random forest (Kim et al., 2021) models have gained attention due to the convenience they offer through automated and standardized fitting procedures (i.e., no need to devise rules). These approaches are particularly appealing compared to non-ML methods, which often require substantial context-specific customization due to the variability in transit environments across cities, as well as the improved prediction fidelity.

However, directly inferring trip purposes from passive datasets like GPS and AFC remains challenging, as these data sources do not inherently include purpose-related information. As a result, the above studies were typically limited to applying ML models at a reduced scale, only on selected subsets of observations that were either linked to auxiliary datasets or self-reported trip purposes recorded from recruited volunteers during the study period. In contrast, active datasets, such as travel diary surveys, almost always include trip purpose as a key question, which has enabled more extensive ML applications in this domain. For example, Cui et al. (2018) employed Bayesian neural networks to train a trip purpose inference model using household travel survey data supplemented with social media location data. Ermagun et al. (2017) compared nested logit and random forest models for trip purpose inference, evaluating cases trained on household travel survey data, point-of-interest (POI) data from an online map service, and a combination of both. Lu and Zhang (2015) used decision trees to infer trip purpose for long-distance travel, exploring scenarios with varying levels of data availability and examining how consolidating trip purpose categories (e.g., merging "Social visit" and "Leisure" into a single "Pleasure" category—an example of a basic feature engineering technique) affects prediction accuracy.

Other studies applying ML models in activity-travel behavior analysis are extensively reviewed by Koushik et al. (2020). According to their review, decision tree–based models (including XGBoost and random forest), followed by neural networks and support vector machines, are the most used ML techniques for data mining and behavioral inference tasks. They found that the most popular topic at the time they surveyed was ML applications for mode choice problems, which naturally involve discrete outcomes. For example, Wang and Ross (2018) compared XGBoost model with the multinomial logit model for a mode choice study and found that ML models achieved higher accuracy, emphasizing the importance of proper model tuning to ensure it. However, concerns about poor interpretability and potential overfitting of ML models remained.

For other discrete variable inference/prediction, studies like Allahviranloo and Recker (2013) and Zhu (2020) developed ML models from household travel surveys to infer activity patterns and types, respectively. In a practical application context, Pérez et al. (2020) developed a classification model to predict trip cancellations and no-shows in paratransit systems, enabling adaptive resource allocations. The classification ML models have also been designed to assist with data preprocessing. For example, Shalit et al. (2022) developed a model to infer missing boarding stops in AFC data, demonstrating how an ML model can effectively predict ordinal variables (i.e., bus stops as a sequence). For continuous outcomes, several studies have employed ML regression models. Sana et al. (2018) used neural networks and support vector machines to estimate origin-destination demand, comparing results to traditional methods like ordinary least squares regression. Similar comparative approaches were used by Miah et al. (2022) for estimating bike traffic and by Yin et al. (2024) for forecasting non-motorized travel flows. For more comprehensive studies applying ML in transportation research, we direct interested readers to Wang et al. (2024b). In their systematic review and large-scale empirical benchmark, they compared hundreds of ML and discrete choice models across diverse *contextual* factors, including choice categories, sample sizes, and data sources. This large-scale, context-focused effort provides a crucial complement to our study's subsequent focus on the *internal* operational rigor of fitting ML models.

Despite the growing application of ML in transportation studies, few papers have thoroughly discussed the role of feature engineering and model tuning. Paredes et al. (2017), in the context of car ownership modeling, found that ML models benefited from incorporating a larger number of input features, whereas a mere increase in the number of features deteriorated the performance of discrete choice models. In contrast, Wang et al. (2019), demonstrated that performing feature selection and eliminating redundant variables improved overall prediction accuracy in a trip chaining context. Together, these findings suggest that although ML models are generally more capable of handling high-dimensional input spaces than traditional econometric models, including too many features can lead to performance degradation ("garbage-in, garbage-out").

Meanwhile, numerous studies emphasize the critical role of spatiotemporal features in travel behavior studies, as they have consistently been proven to be important in enhancing prediction accuracy (Alsger et al., 2018; Cui et al., 2018; Ermagun et al., 2017; Luo et al., 2021). Moreover, augmenting raw data with derived features, such as extracting Euclidean distance from coordinates or activity duration from timestamps (Devillaine et al., 2012; Gao et al., 2024; Lee and Hickman, 2014; Wang et al., 2024a; Yin et al., 2024), along with transforming these spatiotemporal features into high-dimensional representations (Kim et al., 2022), have been proven beneficial. These insights are often supported by variable importance analyses, which show that time and location variables possess significantly higher importance than other variables. For example, decision tree–based models provide native measures of feature importance (Kim et al., 2021; Montini et al., 2014), whereas custom scoring based on cross-validation can be developed (Duan et al., 2022) for specific tasks, and model-agnostic methods such as Shapley Additive Explanations (Lundberg and Lee, 2017) can quantify the marginal influence of individual variables on ML-based inference (Yin et al., 2024).

Despite the increasing use of ML models in travel behavior analysis reviewed in this section, few papers have thoroughly discussed or systematically investigated the foundational design decisions related to specific choices for feature engineering and hyperparameter tuning. As summarized in **Table 1**, the studies frequently leverage feature extraction (addition) and selection (removal) techniques to identify an optimal subset of variables. However, feature engineering methods focused on transforming variables for optimal ML compatibility are less consistently applied or reported. Likewise, the systematic process of hyperparameter tuning largely remains peripheral to the main methodological discussion in literature. Many studies either rely on 1) default parameters or employ 2) resource-intensive techniques like Grid Search or 3) K-fold cross-validation that require manual post-validation adjustments/selections, while the detailed impact of these configurations is often omitted entirely.

Table 1. Summary of cited AI/ML travel behavior studies: documented practices in Feature controls and hyperparameter tuning

Publication	Transportation Topic	(Primary) AI/ML Model*	Feature Extraction/Selection (S) or Feature Engineering (E)**	Hyperparameter Tuning
Allahviranloo & Recker (2013)	Daily activity pattern recognition	SVM	Use of different sets of explanatory variables (S)	Differential evolution algorithm
Cui et al. (2018)	Forecasting current and next trip purpose	BN, RF, SVM, KNN	Elastic Net: LASSO/Ridge blending method (S)	Grid search
Duan et al. (2022)	Predicting Mobility as a Service adoption/use	NN	Not specified	Not specified
Ermagun et al. (2017)	Real-time prediction of trip purposes	RF	Data transformation informed by factor analysis (**E**)	K-fold cross-validation
Faroqi & Mesbah (2021)	Inferring trip purpose	Agglomerative Hierarchical Clustering	Jaccard similarity index (**E**)	Comparison of five different clustering distances
Gao et al. (2024)	Activity type detection	Teacher–Student, NN, XG, RF, SVM	Not specified	Grid search, Self-confidence threshold
Kim et al. (2021)	Estimating trip purposes	RF	Smart card/GIS fusion (S) Stratified sampling (**E**)	Grid search, 4-fold cross-validation
Kim et al. (2022)	Imputing qualitative attributes	Conditional GAN, Conditional VAE, BN	Positional encoding (**E**), Embedding (**E**)	Grid search, Hill-climbing algorithm
Montini et al. (2014)	Trip Purpose Identification	RF	Activity clustering (S), Feature importance analysis (S)	Not specified
Perez et al. (2020)	Predicting Trip Cancellations	RF, GB, XG, LR, DT, NN, NB	Recursive feature elimination (S)	Not specified
Sana et al. (2018)	OD demand estimation	NN, SVM, Lasso/Ridge regression, RF, KNN	Not Specified	Grid search
Sari Aslam et al. (2018)	Predicting public transport trip purposes	NN, RF, SVM, LR, NB	Activity-POIs consolidation (S), Random sampling (**E**)	Grid search
Shalit et al. (2022)	Imputing missing boarding stops	XG, RF, LR	Stepwise selection with SHAP values (S), Embedding (**E**)	Not specified
Wang et al. (2019)	Trip chain extraction	DT, NN, SVM	Sliding-window distance (S), F-Score-based feature-selection (S)	Not specified
Wang & Ross (2018)	Travel mode choice	XG	Rule-based (S)	K-fold cross-validation
Yin et al. (2024)	Examining active travel behavior	RF, GB, DT	Calculation of various distances (S)	10-fold cross-validation

* Acronyms: SVM–Support Vector Machine; NN–Neural Networks (including deep, artificial, feed-forward, fully-connected NNs, and multi-layer perception); LR–Logistic Regression; NB–Naïve Bayes; KNN–K-Nearest Neighbors; RF–Random Forest; GAN–Generative Adversarial Network; VAE–Variational AutoEncoder; BN–Bayesian Network; GB–Gradient Boosting; XG–Extreme Gradient boosting; DT–Decision Tree

** Excluded simple continuous to categorical variable discretization, categorical levels consolidation/reclassification, and one-hot encoding

This gap is especially notable given the unique characteristic of travel behavior studies, where key inputs such as origin–destination coordinates and timestamps are inherently spatiotemporal and paired, as mainstream ML research rarely addresses these unique aspects. Instead, most methodological work in broader ML community concentrates on tabular data (Shwartz-Ziv and Armon, 2022), image data (Xu et al., 2021), multimodal inputs (Rahate et al., 2022), and occasionally extends to map-based data (Klemmer et al., 2023). Moreover, neglecting to align feature representations with a model's architecture or to tune its parameters adequately can lead to overly simplistic assessments of one model's performance compared to another. To address these shortcomings, the remaining parts of this study focus on a systematic evaluation framework that emphasizes methodological rigor in the setup and design of ML models for trip purpose inference. In addition, given the expected large-scale deployment requirement on the automated smart card data domain (with over millions of records), more recent and advanced structures were not

included in our experiment. For example, Transformers (including attention mechanism) were not since they are less scalable for high-throughput classification tasks on structured data.

3. Feature Extraction for Survey-AFC Datasets Alignment

Travel surveys or active transit data face notable limitations, including potential inaccuracies stemming from respondents' reliance on memory during data collection (Richardson et al., 1995). Household travel surveys, a type of active transportation data, aim to capture information on all modes of transportation; yet, they often lack detailed data on transit trip legs, such as which routes were taken and the number of transfers made. Conversely, transit on-board surveys involve interviewers directly engaging with passengers during their transit journeys, providing more accurate and comprehensive data on transit-specific trip details. On the other hand, while abundant in continuous observations, passive or automated transit data often lack qualitative features, such as demographic attributes, due to privacy concerns. For instance, essential information for travel behavior analysis, like gender, income, or age, may be either systematically absent or intentionally masked when transaction data is released. Moreover, passive datasets may not always provide complete information, as many AFC systems are not designed to store alighting information, necessitating the use of additional inference methods to accurately analyze transit travel behavior at the path- or person-level. This section describes the necessary steps to prepare the two datasets for fusion, named *feature extraction*, as a precursor to ML inference and the proposed data fusion.

3.1. Feature Extraction Overview

For our proposed data fusion framework based on ML-inferred trip purposes, the primary dataset required is actively collected travel behavior data, such as transit on-board surveys and household travel surveys, which contain rich features that ML models can be trained on. The secondary dataset consists of passively collected AFC data, which can be transformed into a comparable format to align with the preprocessed or feature-extracted active survey data. Since each dataset contains different types of features, even when they indicate the same individual's single trip, as illustrated in **Figure 1**, the first task is to employ external inference algorithms on those datasets to extract information in a shared format.

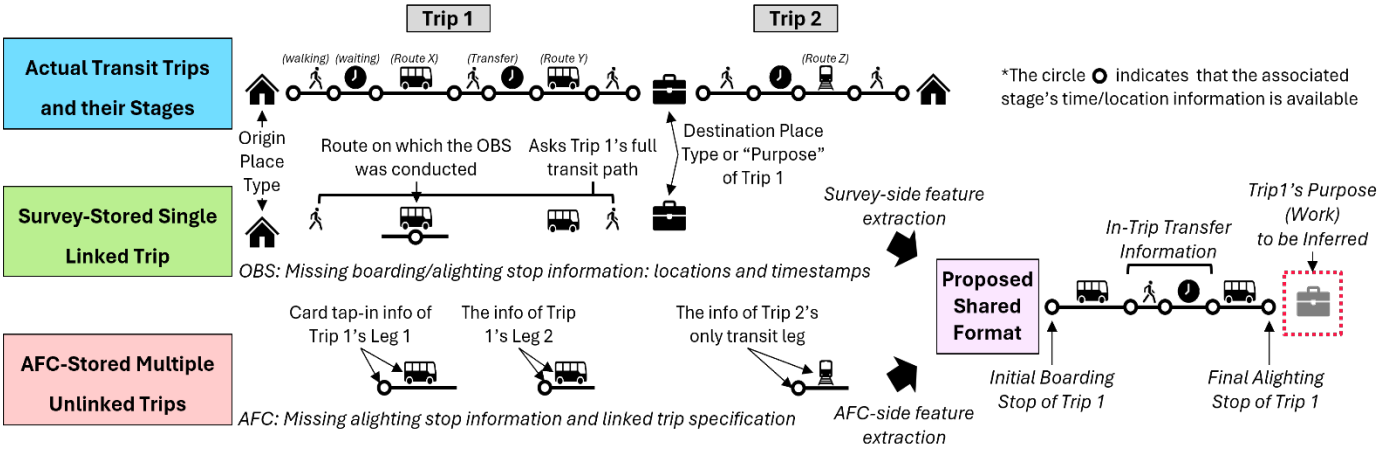


Figure 1 Example of a travel diary observed differently by OBS and AFC, and their proposed alignment to the shared format

For example, surveys typically store trip information as *linked trip*s or full transit *path* (i.e., when a trip consists of multiple transit routes with transfers, surveys will record the routes used in sequence, such as *Home→Bike→Route 1→Route 2→Walk→Work*, along with details on the origin/destination locations, and the survey timestamp on a specific route if it is an OBS. In contrast, AFC data records are typically incomplete *unlinked trip*s where either boarding or alighting information is missing, requiring an external method to identify alighting stops and which sequential AFC taps from each fare card correspond to a single linked trip. Even after relevant AFC features are extracted with an existing algorithm, it is essential to note that precise origin and destination locations cannot be determined from AFC domain; only the initial boarding (near the trip origin) and final alighting (near the trip destination) stops can be identified, unlike the survey observations where those are explicitly asked for. Conversely, detailed boarding and alighting information is typically missing from surveys, so we must extract these stops using an alternative algorithm. To implement these identification algorithms, both for the survey and AFC datasets, auxiliary data containing transit operation details is required. At a minimum, this data must include each route's predefined itinerary and the stop times for each vehicle trip. Increasingly, such datasets are available in formats like General Transit Feed Specification (GTFS) or Automatic Vehicle Location (AVL).

3.2. Feature Extraction with Schedule-Based Transit Assignment on Transit On-board Survey (OBS) Data

The survey dataset used in our experiment and case study for the proposed fusion framework is the Minnesota Metropolitan Council's Transit On-Board Survey (Metropolitan Council, 2023), covering the Twin Cities Metropolitan Area in Minnesota, USA. The survey was conducted in 2022 and collected 25,463 responses, excluding a dozen weeks during the harsh winter, but including weekend trips.

As discussed earlier, recovering the initial boarding and alighting stops for each survey observation is crucial. This can be achieved using *transit assignment* algorithms, which were originally designed to identify optimal transit paths between a given origin, destination, and transit network. Unlike traffic assignment models, transit assignment accounts for multimodal access (e.g., walking, biking, park-and-ride) and transfers between transit services. In these models, travel time is weighted to reflect passengers' preferences for multiple options, with various stages of the journey, such as waiting time, in-vehicle time, and transfer penalties, summed to form the path's generalized cost or (dis)utility (Train, 2009) that is used to assess which path is optimal. These concepts are illustrated in **Figure 2**.

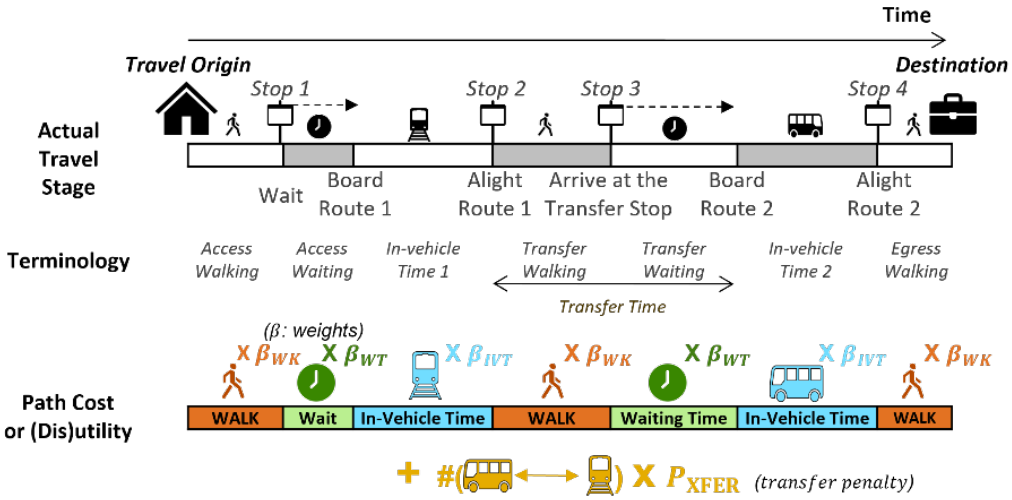


Figure 2 Terminology and utility-based generalized path cost used in transit assignment

For transit systems with frequent services, where headways are generally less than 15 minutes, frequency-based transit assignment algorithms are widely used. These assume random passenger arrivals, with waiting time calculated as half the route's design headway. This approach allows for multiple routes to serve the same segments, with intermediate transfers (the "strategic" behavior) potentially reducing total travel time (Nguyen and Pallottino, 1988). On the other hand, schedule-based assignments are more suitable for low- or mixed-frequency routes, where passengers are assumed to rely on published timetables to plan their trips and avoid excessive waiting times. These models account for peak and off-peak differences in headway and are particularly useful for transit networks with limited services (Tong and Richardson, 1984). With the advent of standardized transit schedules in GTFS format, recent advancements in schedule-based algorithms have been notable. For example, Khani et al. (2015) developed a set of transit assignment algorithms based on GTFS schedules, suitable for planning purposes. Given the availability of date and time information in the survey data, incorporating these features into schedule-based transit assignment algorithms yields greater accuracy in recovering path details than frequency-based ones. However, they require more computational resources due to their dynamic nature.

For our study, we applied on our OBS datasets the Schedule-Based transit Shortest Path algorithm with Trip Elimination (SBSP-TE) developed by Tomhave and Khani (2022). This algorithm constructs a time-dependent transit network using GTFS schedule data and customizes it for each survey response by adding auxiliary links (e.g., walking, biking) from the surveyed origin or destination points to nearby transit nodes within criteria that modelers set. The algorithm employs a label-correcting procedure to determine the shortest paths, incorporating penalties for waiting, walking, and transfers. Unlike typical transit assignment algorithms that return only a single optimal path, this algorithm iteratively eliminates previously identified transit routes from the network and reruns the search to find the second-optimal, third-optimal, and subsequent alternatives. This property is especially valuable for our application, as it increases the likelihood of identifying survey-matching paths and thus yields a larger set of usable data for the model training and data fusion.

The feature extraction results for our OBS identified 20,950 transit paths that exactly matched the self-reported paths from the survey respondents. It is important to note, however, that not all survey responses fed into the SBSP-TE yielded a complete transit path match. This is due to several factors: 1) inherent limitations (missing exact timestamps for each stage) and potential measurement errors (e.g., geocoding surveyed origin/destination addresses) in the OBS survey data; 2) behavioral assumptions embedded in the SBSP-TE algorithm to maintain computational efficiency, such as capping walking distances for transit access and egress; 3) reliance on open-source pedestrian and cycling networks (Boeing, 2017), which may contain inaccuracies; and 4) discrepancies between actual transit operations and the published GTFS schedules. For further details on the SBSP-TE algorithm and its limitations, readers are referred to Tomhave and Khani (2022). The specific parameter settings and adaptations made for our feature extraction process are documented in Baek and Khani (2024).

3.3. Feature Extraction with Trip Chaining on Automated Fare Collection (AFC) Data

The AFC data used in this study is derived from fare card transactions or passenger taps on the fare collector device on buses or at train stations, representing a substantial portion of transit ridership in the Twin Cities Metropolitan Area. The dataset covers the period from January 1, 2018, to August 28, 2023, excluding some days between April and July 2020, when fare collection was suspended during the early phase of the COVID-19 pandemic. In total, the dataset contains 148,816,875 unlinked trip records, averaging 72,167 transactions per day. As noted earlier, however, our raw AFC data lacks important features regarding alighting stops and trip connections, which are crucial for travel behavior analysis and data fusion. To align this data with the shared feature format illustrated in **Figure 1**, a *trip chaining* algorithm must be applied.

Trip chaining, as formally defined by Trépanier et al. (2007), involves integrating multiple unlinked trips to reconstruct linked trip trajectories. Nassir et al. (2015) emphasized the need for context-specific adaptations of trip chaining algorithms, noting that AFC datasets vary widely in structure and coverage: common challenges include missing boarding/alighting taps, closed-system configurations that do not require tapping between transfers, and transfer identification. Robinson et al. (2014) further classified AFC data issues into software, hardware, user, and system design aspects, highlighting "bad" behaviors like premature tap-outs, which are considered in their trip chaining algorithm development. Similarly, Luo et al. (2018) developed preprocessing techniques that leverage AVL and GTFS data to mitigate those challenges in AFC datasets.

Building on these insights, we evaluated the structure and limitations of the AFC dataset available in the Twin Cities area. **Figure 3** outlines both the raw AFC attributes and the technically recoverable features through trip chaining. For each AFC record, we can observe systematically stored route and direction information at the time of boarding, but the alighting details are missing. Our primary objective is to determine the most plausible alighting stop and distinguish between transfer taps and initial-boarding taps. After reviewing available methods, we found that the algorithm proposed by Kumar et al. (2018), which utilizes GTFS and was developed for the same dataset as ours, is best suited to the required structure; therefore, we adopted it.

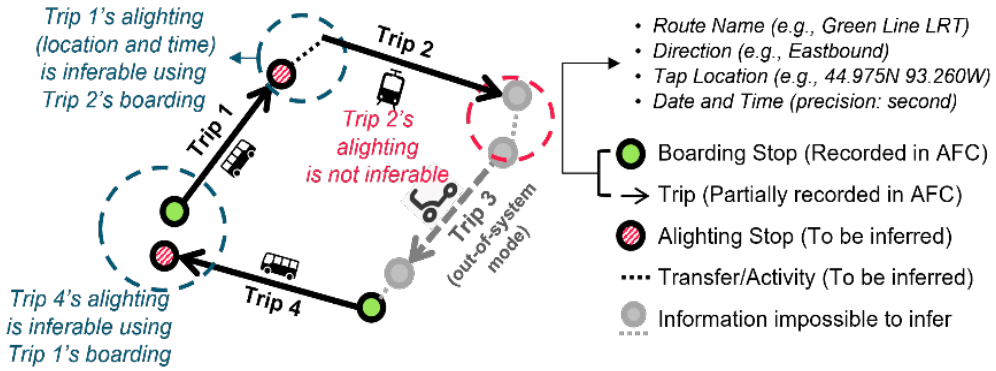


Figure 3 Information recorded by AFC and the trip details that can be inferred through trip chaining

Applying the above trip chaining algorithm to our AFC dataset resulted in 73,058,936 linked trips, with 14,475,138 transfers identified. While the raw dataset counts unlinked taps and the output dataset reflects linked trips, some data loss occurred during this feature extraction. This loss can be attributed to multiple factors: AFC-side errors such as missing boarding stops or route data; GTFS-side discrepancies between scheduled and actual operations (e.g., due to congestion or unexpected events); and for both datasets, the launch of new major BRT services introduced instability during transitional periods. Additional data quality issues stem from passenger behavior—for example, multiple taps on the same station, single-use cardholders, out-of-transit system trips in-between (e.g., Trip 3 in **Figure 3**), or initial boarding/transfer made without tapping—which can be either intentional (i.e., fare evasion) or by mistake. Finally, algorithmic constraints also contribute to data loss. Loosening chaining tolerances (e.g., allowing a greater time difference between AFC and GTFS timestamps) can increase match rates but reduce confidence in inferred paths. To maintain reliability, our implementation balances trip coverage with output quality by tuning these thresholds conservatively, which is also reported in Baek and Khani (2024).

3.4. Additional Feature Extractions and Aligning the Features of the Two Augmented Datasets

Once feature extractions are completed for both OBS and AFC datasets, the next step is to align them to a common format for the data fusion. First, both datasets are likely to include information about trip dates. Typically, AFC data provides the exact date, while OBS data usually includes at least some indication, such as the day of the week or the month. Next, the first boarding and last alighting stops of the linked trips need to be identified. For AFC data, information for the last alighting stops is a derivative of the trip chaining process. Likewise, although OBS data often contains only addresses for the origin and destination, a matching path identified from transit assignment algorithms can provide this information. Departure and arrival times for each linked trip can also be similarly extracted as part of these processes for both types of datasets. In some cases, additional information, such as transit card type (e.g., regular, student, day pass) or fare (e.g., full fare, discount, free), is available in the survey and simultaneously encoded in the AFC system, thereby making them matchable. This information can be helpful for trip purpose inference (Devillaine et al., 2012; Lee and Hickman, 2014).

Once the first boarding and alighting locations are inferred (hereinafter referred to as "origin" and "destination," unless otherwise specified), additional features can be derived. The Euclidean distance between the origin and destination stops can be calculated using the Haversine formula, which provides a spherical approximation of Earth's surface based on geocoded latitude and longitude. Network distances along the routes, even if approximate in some cases, can be extracted from GTFS data using the stop visit sequence of each identified transit path. Additionally, a binary variable can be created to indicate whether a trip included at least one specific transit route. For instance, in our case study context covering the Twin Cities region, we can flag trips that include a BRT/LRT leg, as these routes have unique characteristics, such as out-of-vehicle fare collection, frequent headways, and exclusive right-of-way, properties that enhance transit trip experience, which can influence the residents' travel behavior (Berrebi et al., 2022). Finally, the number of transfers for each linked trip can also be extracted as an additional feature.

After aligning the features and extracting additional information from our datasets, we identified three categorical-like variables—namely, season, day of the week, and number of transfers, although the last one is inherently integer—are shared by both datasets (OBS, AFC). The reason for categorizing the number of transfers is that, in most cases, the addition of a transfer is not proportionally affecting travel behavior, as the perception between a direct path (0 transfers) and a single-transfer path can differ from that of single-transfer and two-transfer paths. These three variables initially contain 4 seasons, 5 day-of-week information (Monday/Tuesday-Thursday/Friday/Saturday/Sunday), and 6 (0 to 5 transfers) levels, respectively. However, for broader explorations, we created an additional dataset that consolidated the levels of the three categorical variables into binary variables. Specifically, months in this new dataset were grouped as "summer" or "other" to reflect school breaks and vacations, and day of the week as "weekday" and "weekend," and the number of transfers as direct and transfer-including paths. This consolidation approach affects how ML algorithms treat the categorical variables and how they are encoded for input, which is discussed in **Section 4**. The extracted features serve as predictors (independent variables) for inferring trip purpose.

Our dependent variable, trip purpose, or, in ML terminology, referred to as either *target or label*, can be derived from a survey question inquiring about destination place types. While the trip purpose in conventional four-step transportation models is derived from both origin and destination, like "home-based work" (Ortúzar and Willumsen, 2011), we should consider that both origin and destination types are missing from the AFC data, so jointly inferring two place types simultaneously can be challenging. Therefore, we used the survey's destination place type.

In our OBS, the destination place type, in its raw form, had 11 levels, including minor categories such as attending a sports event or going to the airport as a passenger. We consolidated these into four categories: *Home*, *Work*, *Shopping*, and *Others*. This consolidation acknowledges the importance of analyzing work or commute trips for transit planning (Aminpour and Saidi, 2025), high applicability of AFC-detected home locations (Zou et al., 2018), and reflects the increased frequency of shopping trips post-COVID-19 (Balbontin et al., 2024; Barrero et al., 2021). Other purposes that we consolidated, while they can be important for specific analyses, are typically treated as peripheral: studies consolidated school trips as work trips (Aghaabbasi et al., 2020; Hossain and Habib, 2021; Xiao et al., 2016; Zailani et al., 2016), and some pointed out that having more minor levels causes deterioration of model accuracy (Faroqi and Mesbah, 2021; Lu and Zhang, 2015; Wang and Ross, 2018).

The features extracted from the OBS with the encoded trip purpose label are summarized in **Table 2**.

Table 2. Summary of features and trip purpose label statistics for the 2022 OBS, following transit assignment and feature extraction

Non-binary Categorical Features				Everything Else		
Name	**Level**	**Count**	**Proportion**	**Name**	**Mean**	**Std.**
Season	Spring (Apr-May)	3,945	19.50%	Origin latitude (decimal degrees)	44.959	0.0567
	Summer (Jun-Aug)	6,253	30.91%	Origin longitude (decimal degrees)	-93.225	0.0823
	Fall (Sep-Nov)	9,252	45.72%	Destination latitude (decimal degrees)	44.959	0.0555
	Winter (Dec-Mar)	793	3.870%	Destination longitude (decimal degrees)	-93.223	0.0802
Day of week	Monday	3,745	18.51%	Departure time (minutes past midnight)	762.90	248.98
	Tuesday-Thursday	14,571	72.02%	Arrival time (minutes past midnight)	794.49	250.06
	Friday	705	3.484%	Travel duration: arrival-departure (minutes)	31.592	1.0808
	Saturday	660	3.262%	Number of transfers[a]	0.2885	0.5244
	Sunday	552	2.728%	Euclidean distance (miles)	4.4768	3.4924
Season (consolidated)	Summer/Others	Refactored from the above full levels		Network distance (miles)	4.9180	4.0241
Day of week (consolidated)	Weekday/Weekend			BRT/LRT leg inclusion (Yes = 1)	0.4522	0.4977
Label: trip purpose	Home: 6,429 (31.8%); Work: 5,211 (25.8%); Shopping: 3,534 (17.5%); Others: 5,059 (24.9%)					
Number of observations	20,233					

[a] This is treated as a six-level categorical variable (levels: 0, 1, 2, 3, 4, 5) or a binary variable (levels: direct, transfer-including) for ML models input

4. Theory

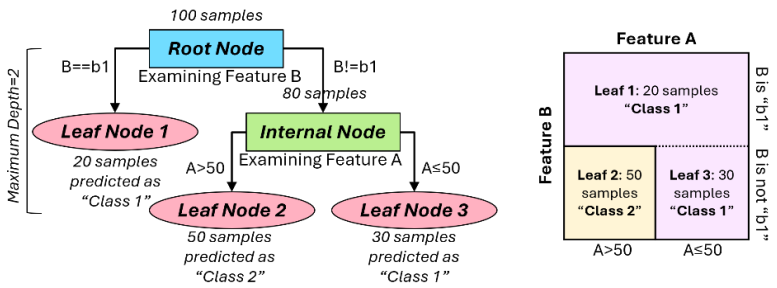
This section presents the theoretical foundations for our ML classification framework and its practical applications. We begin with the general principles of supervised classification, then introduce the five specific models used in this study, and finally describe the feature engineering techniques. These components form the basis of the model optimization and comparative experiments in **Section 5**, whose goal is to predict trip purpose (destination place type) from the shared features of preprocessed OBS and AFC datasets (**Table 2**) and compare the results with various settings. Because the trip purpose is directly observed only in the OBS, we train the models on that data and then apply them to the AFC dataset to enable continuous transit-behavior analysis. This transfer demonstrates how ML-driven inference can support adaptive transit planning, which we illustrate in the case study of **Section 7**.

4.1. Supervised Classification and Machine Learning

Machine learning is the technology of developing algorithms that can emulate human intelligence (El Naqa and Murphy, 2015). Supervised learning refers to the predictions based on the training sample of previously solved cases, where the joint values of all the variables are known (Hastie et al., 2009). Classification is a task that involves predicting the correct label or category for a given input data. Therefore, developing a supervised ML for classification (an "ML classifier") is formally defined as optimizing a function $f: \mathbb{R}^D \mapsto \mathcal{C}_i \in \mathbb{C}$ where D is the input data $\mathbf{x}$'s feature dimension and $\mathbb{C} = \{\mathcal{C}_1, \dots, \mathcal{C}_K\}$ is a set of possible classes or (target) label levels to be predicted, which has K distinct value. It is assumed that there is a fixed but unknown distribution on $\mathbb{R}^D \times \mathbb{C}$, where the dataset $\mathbb{D} = \{(\mathbf{x}_1, y_1), \dots, (\mathbf{x}_N, y_N)\}$ of N samples with $D + 1$ dimension is given from which we find the optimal classifier f that minimizes the loss $L(f) = P(f(\mathbf{x}) \neq y); y \in \mathbb{C}$ across N.

Several ML models are well-suited for supervised classification tasks. Among them, random forests, support vector machines, and neural networks are the most popularly used in activity-travel behavior research (Koushik et al., 2020). In addition, a comparative analysis of tabular data models has shown that Extreme Gradient Boosting (XGBoost) often outperforms both basic and advanced neural network architectures (Shwartz-Ziv and Armon, 2022). Another ML classifier gaining popularity is CatBoost, known for its automated handling of categorical features and comparatively user-friendly model tuning, although its application on travel behavior analysis is still scarce (Baumgarte et al., 2022). It is worth noting that, given the expected large-scale deployment requirement on the automated smart card data domain (with over millions of records), more recent yet less scalable architectures, for example, transformers, are not included in our experiment.

This study employs the aforementioned five ML models as our classifier and scrutinize their characteristics and implications on travel behavior analysis. To understand three ensemble ML classifiers (i.e., random forests, XGBoost, and CatBoost) out of five, however, it is essential to grasp the decision tree classifier, a base learner in these methods (Breiman, 1984). A decision tree begins at the *root node*, containing the whole dataset observations, and then employs a recursive partition of the data space. It consists of *internal nodes*, where the data is split based on a feature value, and *leaf nodes*, which are terminal points that assign a class label from which no further splitting occurs (**Figure 4**).



(a) Decision tree structure and terminology (b) How data space has been split

Figure 4 Overview of decision tree classifier

The tree "grows" at the root and internal nodes until stopping criteria are met (e.g., maximum depth or minimum samples per node), where the algorithm selects the feature and threshold that minimize impurity, commonly measured by Gini impurity computed by **Equation (1)**.

$$Gini\ impurity = 1 - \sum_{i=1}^{K} p_i^2 \quad (1)$$

Here, p_i is the proportion of observations belonging to class i in the current node. A Gini impurity of 0 indicates a pure node (only one class present). With the above settings, the algorithm at each split selects the feature and threshold that minimize the weighted average Gini impurity of the resulting child nodes. The impurity change after the split is named *gain*: $G = Gini_{parent} - (w_l Gini_{left} + w_r Gini_{right})$, where weights w (left/right) are the child node sample counts relative to the parent. The process continues until all nodes are pure or the gain becomes minimal.

4.1.1. Bagging Decision Tree—Random Forest (RF)

Random forest (RF) is a robust ensemble learning technique that combines predictions from multiple decision trees to enhance accuracy and leverage collective intelligence (Breiman, 2001). Each decision tree in the ensemble is constructed from a random subset of observations (rows) sampled with replacement—a method known as bootstrap aggregating, or *bagging*—and a random subset of features (columns). This inherent randomness ensures each tree focuses on distinct data facets. Individual trees from the sub-datasets generate predictions independently, and the most frequent prediction determines the final classification (majority voting). Here, RF's aggregating predictions across multiple trees mitigate overfitting (excessive adaptation to training data, which hinders generalization) and enhance robustness.

Key hyperparameters of RF, following the input argument notations of the Python package *scikit-learn* (Pedregosa et al., 2011) include:

- $n\_estimator$: the number of decision trees to grow for collective decision making or voting.
- max_depth: maximum allowed depth of each estimator or decision tree (longest path length from root to leaf).
- $min\_samples\_split$: minimum number of samples required to split an internal node (if the node is small, do not grow further).
- $min\_samples\_leaf$: minimum number of samples required to form a leaf node.
- $max\_features$: number of features (columns) randomly considered at each split; either set to $\sqrt{D}$ ("sqrt") or $\log_2 D$ ("log2").
- $max\_samples$: number of samples (rows) randomly considered at each split.

Random forests also provide measures of feature importance by summing each feature's contribution to the reduction in Gini impurity across all splits. The final prediction is based on majority voting among trees, which can be expressed as class probabilities (e.g., Home: 60%, Work: 20%, Shopping: 10%, Others: 10%). These probability estimates are valid for downstream applications, such as combining outputs from multiple models in an ensemble to arrive at a more robust inference.

4.1.2. Gradient Boosting Decision Tree—XGBoost (XG)

Being the most popular classifier in the Gradient Boosting Decision Tree (GBDT) family, XGBoost (Chen and Guestrin, 2016) builds an ensemble of shallow decision trees via iterative gradient boosting. At each iteration, XG adds a new tree trained to correct the residual errors (negative gradients) of the current model. Intuitively, leaves that yield larger error reductions receive larger weights, so their patterns carry more influence into the next tree. This structure steers the ensemble toward addressing the hard-to-predict regions. Although less interpretable than RF since the final probability is computed from a SoftMax-applied output score, unlike RF's majority vote, XG often achieves higher accuracy, particularly on imbalanced label distributions, and scales efficiently to huge datasets thanks to built-in regularization and parallelization.

Formally, for a binary classification problem ($\mathbf{x} \mapsto y \in \{0,1\}$), let $\hat{y}_n^{(t-1)}$ be the XG prediction for sample n after $t-1$ trees generated, and f_t be the new tree added at iteration t (note: the *base score* for defining $\hat{y}_n^{(0)}$ is set by the model). The objective to minimize is:

$$L^{(t)} = \sum_{n=1}^{N} l\left(y_n, \hat{y}_n^{(t-1)} + \eta f_t(\mathbf{x}_n)\right) + \Omega(f_t) \tag{2}$$

$$\Omega(f_t) = \gamma T + \frac{1}{2}\lambda\|\mathbf{w}\|_2^2, \text{and } \hat{y}_n^{(t-1)} + \eta f_t(\mathbf{x}_n) = \hat{y}_n^t \tag{3}$$

Here, the most popularly used per-sample loss l is derived from the cross-entropy loss formulation ($l(y, \hat{y}) = -[y\log p + (1-y)\log(1-p)]$, where $p = \text{sigmoid}(\hat{y})$), η the learning rate, and the added regularization term $\Omega(f_t)$ penalizes the incumbent tree's complexity: T the number of leaves and $\mathbf{w}$ the vector of leaf weights, where the optimal weights are obtained by minimizing the following objective:

$$\mathbf{w}^* = -(diag(\mathbf{h}) + \lambda\mathbf{I})^{-1}\mathbf{g} \tag{4}$$

where $\mathbf{I}$ is the identity matrix, $\mathbf{g}$ and $\mathbf{h}$ are the vectors of summed first- and second-order gradients over each leaf. That is, an element of $\mathbf{g}$ corresponding to leaf j can be expressed as $g_j = \sum_{n \in j} \partial l(y_n, \hat{y}_n)/\partial \hat{y}_n$. This regularization of tree complexity and weight magnitude helps prevent overfitting.

Key hyperparameters of XG, following the argument notation of the Python package *xgboost* (Chen and Guestrin, 2016) include:

- $n\_estimators$: the number of boosting rounds (i.e., the total number of trees iteratively employed).
- max_depth: maximum allowed depth of each decision tree (longest path length from root to leaf).
- $colsample\_bytree$: fraction (ranging from 0 to 1) of features (columns) randomly used to grow each tree.
- $learning\_rate$: step size shrinkage applied to each tree's contribution; η in **Equation (2)**.
- reg_lambda: L2-norm penalty on leaf weights $\mathbf{w}$ to reduce overfitting; λ in **Equation (3)**.
- $gamma$: minimum loss reduction required to make a further split; γ in **Equation (3)**: a split is only made if $-\Delta\mathcal{L} \geq \gamma$.

4.1.3. Specialized Gradient Boosting Decision Tree—CatBoost (CB)

CatBoost, standing for "Categorical Boosting," is another GBDT algorithm that, like XGBoost, builds trees iteratively to correct preceding errors. However, CB is specifically engineered to handle categorical features more effectively and efficiently and to reduce overfitting (Prokhorenkova et al., 2018). CB's key distinctions from XG include the following three aspects.

First, to prevent the *data leakage* inherent in standard gradient boosting (where gradients for each sample are computed using its own target label), CB uses random permutations of the training data—it first shuffles the input dataset rows and assigns a new ID for each row. Then, gradients for a given sample are calculated using only earlier samples in that order, i.e., the reordered n'-th sample's gradients are computed from the first to $(n'-1)$-th observations. This ensures no direct use of its own target value ($y_{n'}$) during tree construction. Second, CB comprises *symmetric* trees in which all nodes at the same depth split on the same feature and threshold, with an injection of randomness when growing trees to ensure ensemble diversity. The symmetric structure both regularizes the model and accelerates training and inference through more efficient computations. Finally, rather than requiring other encoding methods that ML classifiers require at the preprocessing step, CB implements its own *ordered target encoding*. Detailed explanations for the encoding methods are discussed in **Section 4.3**.

Together, these innovations make CB especially effective and user-friendly for datasets with high-cardinality categorical variables, as well as mitigating potential data leakage to prevent overfitting.

Key hyperparameters of CB, following the argument notation of the Python package *catboost* (Dorogush et al., 2018) include:

- $iterations$: maximum number of boosting rounds; note: unlike XG, CB can early stop the training if the loss reduction is small.
- $learning\_rate$: step size shrinkage applied to each tree's contribution; equivalent to XG's $learning\_rate$.
- $depth$: a fixed depth of each decision tree; note: unlike XG, the trees are grown symmetrically, resulting in 2^{depth} final leaves.
- $l2\_leaf\_reg$: the L2 regularization coefficient on leaf weights; equivalent to XG's reg_lambda.
- $random\_strength$: denoted r, controls the amount of noise added when splitting nodes; i.e., a uniform random term $\varepsilon_o \sim U(-1,1)$ drawn for node o are multiplied by r and scaled to be added in the gain calculation, so $G_o' = G_o + r\varepsilon_o\sigma_G$ is used instead of G_o.
- $auto\_class\_weights$: strategy for weighting classes; when set to "Balanced": weights inversely proportional to class frequencies, when set to "SqrtBalanced": weights inversely proportional to the square root of class frequencies.

4.1.4. Feed-forward Neural Networks (NN)

Artificial neural networks represent a class of machine learning models that are inspired by the way the human brain operates (Bishop, 2006). These networks consist of layers of interconnected neurons that process input data and learn to make predictions. NN excels at learning complex feature representations from raw data, automatically discovering relevant patterns and hierarchical relationships among features. By increasing the number of layers, the structure known as deep neural networks, NN can effectively model non-linear decision boundaries and adapt to various data distributions without relying on predefined assumptions.

The feed-forward, fully-connected deep neural network denoted $\Lambda(\cdot)$ comprising S hidden layers are defined as **Equation (5)**, where $a(\cdot)$ denotes the vector output activation function, typically set to the rectified linear unit (ReLU). $\mathbf{W}_s$ denotes the weight matrix between $(s-1)$-th and s-th layers, and $\mathbf{b}_s$ the bias vector for the s-th hidden layer. The weights and biases are optimized with the formulation in **Equation (6)** using a specific optimizer, where $\hat{y}_{ni}$ indicates the estimated probability of sample n being class i, the loss function $\mathcal{L}(\cdot)$ normally set to cross-entropy loss $-\mathbb{E}[\mathbf{y}^T \log \hat{\mathbf{y}}]$, and $\|\cdot\|_F$ the Frobenius norm used to regularize weights $\mathbf{W}$ with the model hyperparameter λ.

$$\Lambda(\mathbf{x}_n; \mathbf{W}, \mathbf{b}) = \hat{\mathbf{y}}_n = [\hat{y}_{n1}, \hat{y}_{n2}, \hat{y}_{n3}, \dots, \hat{y}_{nK}] = \text{softmax}\left(a\left(\mathbf{W}_{S+1}^T a(\mathbf{W}_S^T \cdot \dots \cdot a(\mathbf{W}_1^T \mathbf{X} + \mathbf{b}_1) + \mathbf{b}_S)\right) + \mathbf{b}_{S+1}\right) \quad (5)$$

$$Min\ L\colon \mathcal{L}(\hat{\mathbf{y}}_n, \mathbf{y}_n) + \Omega(\mathbf{W}) = -\frac{1}{N}\sum_{n=1}^{N}\sum_{i=1}^{K} y_{ni} \log(\hat{y}_{ni}) + \lambda \sum_{s=1}^{S+1} \|\mathbf{W}_s\|_F^2 \quad (6)$$

In modern practice, stochastic gradient descent algorithms like Adam Optimizer (Kingma and Ba, 2015) has become popular due to its scalability and faster convergence. It has driven a shift toward *weight decay* as the preferred form instead of the λ regularization in **Equation (6)**, as the decay is found to be more effective than λ in the modern optimizers (Loshchilov and Hutter, 2019). Intuitively, rather than adding a penalty term directly into the loss function, weight decay applies a small, consistent shrinkage to each weight at every update step.

Key hyperparameters of NN, some from the Python package Pytorch (Paszke et al., 2019) and the others custom-defined include:

- n_layers: the number of hidden layers in the network, or S in **Equation (5)**.
- $n\_units\_layer\_\{s\}$: the number of artificial neurons in the s-th hidden layer.
- $nEpoch$: the number of training (forward-backward paths) iterations in weight and bias optimization.
- $lrate$: step size for each parameter update.
- $dropout\_prob$: fraction of neurons randomly dropped ("zeroed") on each training step (a regularization technique).
- $weight\_decay$: strength of applying weight decay regularization (0: no decay, ~1: weights collapse toward zero).

4.1.5. Support Vector Machine (SVM)

Support Vector Machines (Schölkopf and Smola, 2002) identify an optimal decision boundary by constructing a hyperplane in feature space that maximizes class separation. While this yields clear, well-defined margins, interpreting SVM outputs can be challenging. Unlike the other models discussed, whose predictions are probabilities (RF) or probability-convertible scores (XG, CB, NN; before applying the SoftMax function), SVMs produce decision function margins. These margins do not directly correspond to probabilities unless explicitly calibrated. Consequently, when combining SVM outputs with other model predictions (i.e., "ensemble fusion"), special care must be taken.

SVM is formally defined for the linear and binary classification case first, and then can be extended to multiclass and non-linear cases with some modifications. For the simplest case ($y_i \in \{+1, -1\}$) where a decision is made depending on the output ($f(x_n)$) sign, the decision function or hyperplane is defined as $f(\mathbf{x}) = \mathbf{w}^T\mathbf{x} + b = 0$. SVM optimizes by choosing $\mathbf{w}$ that maximizes the class separation. More formally, the SVM training minimizes the average hinge loss plus an L2-norm penalty on the weights:

$$\min_{\mathbf{w},\mathrm{b}} \frac{1}{N}\sum_{n=1}^{N} \max\big(0, 1 - y_n f(\mathbf{x}_n)\big) + \lambda\|\mathbf{w}\|_2^2 \quad (7)$$

For multiclass or non-binary classification problems, SVM employs one-vs-one strategies, where separate binary support vectors are trained for each pair of classes and a voting scheme is used over their decision margins to make the final class prediction.

Moreover, SVM effectively handles non-linear data by leveraging the kernel trick, which is crucial for controlling computational cost. This technique allows the model to operate in a high-dimensional feature space without explicitly performing the mapping. Instead, it requires only the computation of inner products between input vectors $\mathbf{x}$ and $\mathbf{x}'$ or a pair of observations. Commonly used kernels, denoted κ, include:

- Linear kernel ("linear"): $\kappa_L(\mathbf{x}, \mathbf{x}') = \mathbf{x} \cdot \mathbf{x}'$
- Polynomial kernel of degree d ("poly"): $\kappa_P^d(\mathbf{x}, \mathbf{x}') = (\gamma\mathbf{x} \cdot \mathbf{x}' + c_0)^d$
- Radial basis kernel ("rbf"): $\kappa_R(\mathbf{x}, \mathbf{x}') = \exp\left(-\gamma\|\mathbf{x} - \mathbf{x}'\|_2^2\right)$

These kernels are used during the training process with dual optimization, which depends only on inner products between all pairs of training examples, denoted (μ, ν). For binary cases, SVM maximizes the objective **Equation (7)** subject to linear constraints on a Lagrange multiplier α_μ adopted, associated with the margin constraint for the μ^{th} training sample:

$$\max_{\alpha} \sum_{\mu} \alpha_\mu - \frac{1}{2}\sum_{\mu,\nu} \alpha_\mu \alpha_\nu y_\mu y_\nu\, \kappa(x_\mu, x_\nu) \quad (8)$$

Here, no explicit feature mapping is computed, meaning that rather than optimizing the weight vector $\mathbf{w}$ directly, the dual optimization solves the quadratic program in **Equation (8)** using the kernel trick.

Key hyperparameters of SVM, following the argument notation of the Python package *scikit-learn* (Pedregosa et al., 2011) include:

- $kernel$: kernel type for the dual optimization; common choices are "linear," "poly" (polynomial), and "rbf" (radial basis).
- C: regularization parameter (inverse of λ in **Equation (7)**); higher C means less regularization.
- $gamma$: kernel coefficient for non-linear kernels; higher values imply tighter, more localized decision regions.
- $coef0$: added term (c_0) in the polynomial kernel; it balances the influence of higher-order versus lower-order feature interactions.
- $degree$: degree d of the polynomial kernel; $d \geq 2$.
- $class\_weight$: class weighting scheme for imbalanced datasets; setting it applies weights inversely proportional to class frequencies, giving more emphasis to minority classes.

4.2. Feature Engineering for Continuous and Spatiotemporal Features

Continuous features, such as the Euclidean distance in our dataset, are typically standardized (zero mean, unit variance) before being fed into ML models by applying the mean and variance calculated solely from the training data. Without this step, variables on larger scales can dominate gradient-based optimization. In addition, many hyperparameters are tuned to operate on inputs roughly within the standardized range: standardized inputs mostly lie between -3 and 3, with even extreme outliers rarely exceeding an absolute value of 10. Tree-based models, however, apply an additional discretization step beyond standardization. This step, named *binning*, is built into each model; increasing its resolution usually yields diminishing accuracy improvements while also increasing computational cost. Therefore, the primary pre-processing for continuous variables in tree-based models remains standardization, enabling the models' built-in discretization to handle further binning as needed.

However, spatiotemporal features, a special type of continuous variable, require careful handling because their raw numeric forms—minutes past midnight for time, or latitude/longitude (or UTM coordinates) for location—fail to capture cyclic patterns, true spatial relationships, and temporal/spatial clusters. For instance, representing 1 AM as 60 minutes and 11:00 PM as 1,380 minutes makes them appear far apart on a linear scale and can even imply a 23-fold proportional relationship. On the other hand, raw latitude/longitude pairs overlook either point's neighborhood-specific context and/or the relationship between the origin and destination. Although some ML models can theoretically learn such discretization and complex relationships via extensive tuning, explicitly engineering features to add meaningful information often enhances ML models' performance with little downside. The following subsections present these spatiotemporal encoding strategies, focusing on paired origin–destination times and locations, which form the backbone of travel behavior analysis and trip-purpose inference. Crucially, these methods are applied as additions to the raw features (minutes past midnight, latitude/longitude) to supplement the original values, ensuring that no original information is lost during this process and thereby preventing so-called "destructive" transformation.

4.2.1. Cyclical Encoding (for temporal features)

Cyclical encoding can be applied to time features; for our dataset, departure and arrival times are measured in minutes past midnight. A cyclical encoder tailored to minute-represented times transforms an input $t \in [0,1440)$ minutes into a two-dimensional vector via sine and cosine functions with a wavelength of 1,440, yielding outputs in $[-1,1]$ for both sine and cosine dimensions (**Equation** (9)).

$$CyclicalEncoder(t) = \left(\cos\left(\frac{2\pi t}{1440}\right), \sin\left(\frac{2\pi t}{1440}\right)\right) \tag{9}$$

Because this maps each minute of the day to a unique point on the unit circle, it preserves the cyclic nature of time—consecutive minutes produce nearby vectors, and midnight (0) and 23:59 (1439) become adjacent rather than far apart. For example, it encodes midnight to (1,0), noon or $t = 720$ to $(-1,0)$, and 11:59 PM to $(0.9999, -0.0043)$. In the encoded space, the distance between 0 and 1,439 is small, which correctly reflects their temporal adjacency, while both remain distant from noon.

4.2.2. Polar Coordinates (for spatial features)

Any point (x, y) in Cartesian coordinates can be expressed in polar form (r, θ), where r is the Euclidean distance from the origin, and θ is the angle between the positive x-axis and the line segment joining the origin to (x, y). Inverting the trigonometric relations $x = r\cos\theta$ and $y = r\sin\theta$ gives **Equation (10)**, where $\text{atan2}(y, x)$ denotes the 2-argument arctangent first introduced in FORTRAN (International Business Machines, 1961) that uses the signs of both x and y to unambiguously determine the angle θ across all four quadrants. By convention, θ may be mapped to either $[0,2\pi)$ or $(-\pi, \pi]$ range; but for ML inputs alongside other standardized continuous features, the range $(-\pi, \pi]$ is preferable and the latter also aligns with the atan2 definition.

$$PolarEncoder(x, y) = (r, \theta) = \left(\sqrt{x^2 + y^2}, \text{atan2}(y, x)\right) \tag{10}$$

In urban economics and travel-behavior analysis, it is well established that cities tend to develop radially around a central business district, as described by *Monocentric Land-Rent Theory* and the *Spatial Patterns of Housing* (O'Sullivan, 2019). Therefore, encoding geographic locations in polar coordinates supplies ML models with features that naturally capture the radial structure of cities and people's movement.

For our study, we set the origin at the midpoint of Minneapolis and St. Paul city halls (44.9589, -93.2242), representing the twin cores of the Twin Cities metropolitan area. We convert each latitude/longitude pair into miles from this origin, apply the polar encoder, standardize r for model input, and retain θ in $(-\pi, \pi]$ without further scaling.

4.2.3. Elliptic Coordinates (for spatial features)

Similar to the polar coordinate system, a point (x, y) in Cartesian coordinates can be expressed in confocal elliptic coordinates (u, v), which parameterize families of ellipses and hyperbolae sharing the same foci at $(\pm a, 0)$, where x and y can be represented by $a\cosh u\cos v$ and $a\sinh u\sin v$, respectively, by which the following **Equations (11)-(12)** can be derived.

$$\frac{x^2}{a^2\cosh^2 u} + \frac{y^2}{a^2\sinh^2 u} = \cos^2 v + \sin^2 v = 1 \tag{11}$$

$$\frac{x^2}{a^2 cos^2 v} - \frac{y^2}{a^2 sin^2 v} = \cosh^2 u - \sinh^2 u = 1 \tag{12}$$

The two equations represent an ellipse (sum of focal distances constant) and hyperbolae (difference of focal distances constant), respectively, where $u \in \mathbb{R}^+$ and $v \in (-\pi, \pi]$. An intuitive understanding of those two parameters is as follows. The $u = c$ level curve is an ellipse whose major-axis length equals $2a\cosh u$, which is also the sum of distances to foci for any point on the ellipse. For a fixed u, varying v moves a point around that ellipse, but not by central angle as in polar coordinates. Instead, v corresponds to the family of the orthogonal hyperbolae, for the

point on which the difference of its distances to the two foci is $2a|\cos v|$. Because all points on the line segment between $(-a, 0)$ and $(a, 0)$ are mapped to $(0, v)$ with arbitrary v, the Cartesian to elliptic map is not one-to-one. In practice, however, real-world coordinates almost never lie exactly on the focal axis, and only for small u one should exercise caution since tiny changes in (x, y) can produce large swings in v.

Solving **Equations (11)-(12)** for (u, v) gives the $EllipticEncoder$ defined in **Equation (13)**.

$$EllipticEncoder(x, y) = (u, v) = \left(\operatorname{arccosh}\frac{\sqrt{x^2 + y^2}}{a}, \operatorname{atan2}(y, x)\right) \tag{13}$$

In metropolitan regions with two principal cores, such as the Twin Cities, the elliptic coordinates naturally capture both the sum of distances to the foci (via u) and their difference (via v). For our application, we fix the foci at the Minneapolis City Hall (44.9773, -93.2654) and St. Paul City Hall (44.9443, -93.0940), whose inter-focus distance is $2a = 8.7$ miles. We first convert each latitude/longitude pair into miles from the origin, rotate the coordinates to align x-axis with the inter-focus segment, apply the elliptic encoder, standardize u for model input, and retain v in $(-\pi, \pi]$ without further scaling.

Figure 5 illustrates our OBS-recorded destination (last alighting stop) locations and associated trip purpose (place type) color-coded, in raw latitude/longitude coordinates, polar encoded coordinates, and elliptic encoded coordinates, respectively.

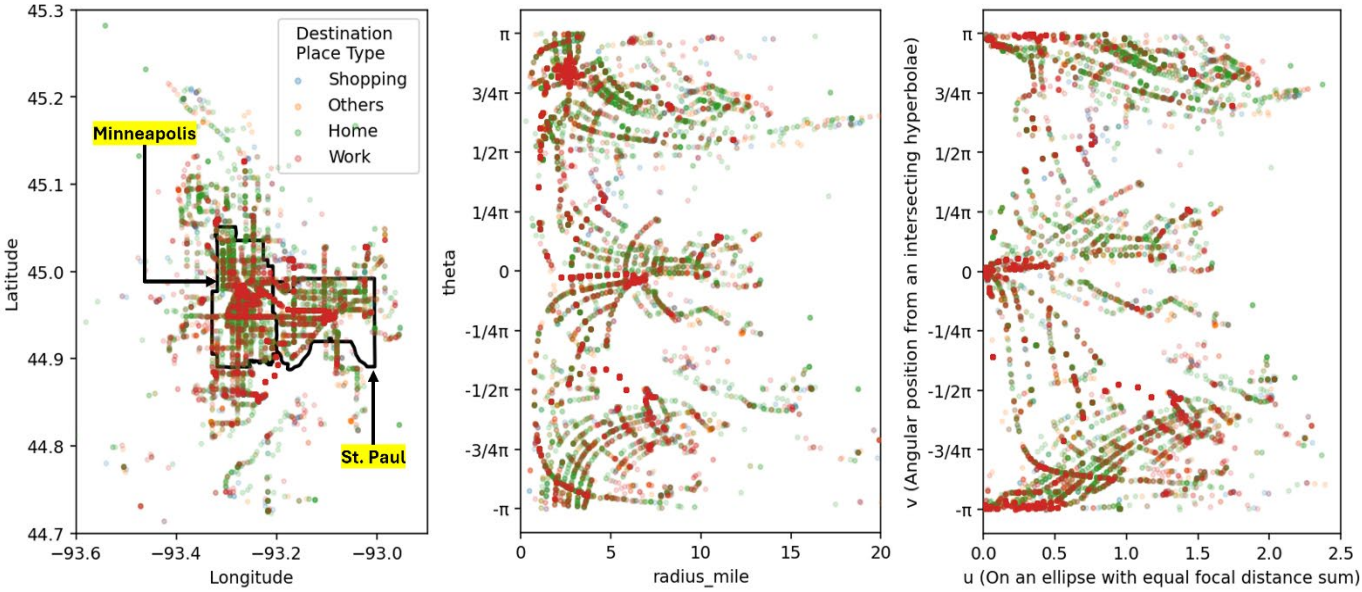


Figure 5 OBS-extracted transit paths' final alighting stop location with color-coded destination place type by different transformations

4.2.4. Zone Assignment (for spatial features)

A continuous geographic field can be discretized into a finite set of spatial zones, allowing each origin or destination point to be represented as a categorical "zone" feature. The most common zone scheme in transportation planning is the Traffic Analysis Zone (TAZ), which typically aligns with administrative boundaries, allowing for the easy aggregation of sociodemographic data and statistics. However, for transit boarding and alighting, TAZs can introduce noise: bus stops or rail exits on opposite sides of a roadway (where many TAZ borders are drawn on) often fall into different TAZs, even though they serve the same trip origin or destination.

Meanwhile, some agencies define transit-specific zones. In the Twin Cities, the Metropolitan Council's *Transit Market Areas* partition the region into polygons based on proximity to the downtown cores, transit-demand density, and route coverage (Metropolitan Council, 2015). When available, adopting such specialized zones can improve trip-purpose inference and other downstream tasks, especially if a model can handle high-cardinality categorical inputs (for example, CatBoost) or when an appropriate feature encoding is adopted.

In our application, we begin with the seven original Transit Market Area definitions, then overlay city boundaries for Minneapolis, St. Paul, and county boundaries for the seven suburban areas. We then introduced four additional "core" zones—Downtowns of Minneapolis, St. Paul, the University of Minnesota campus, and the MSP Airport/Mall of America complex—yielding a total of 31 polygons (see **Appendix A** for full definitions and zone map). Finally, we assigned each trip's origin and destination to one of these 31 zones for use as categorical input features.

4.2.5. Positional Encoding (for both temporal and spatial features)

Positional encoding (Vaswani et al., 2017) was initially developed to inject positional information into fixed-dimension word-embedded vectors within the attention mechanism of transformer models, for modern large language AI models. More recently, it has been applied to encode cyclical data such as time (Grądzki and Wójcik, 2024) to improve accuracy by helping AI models to learn both fine-grained ("block") and broader temporal patterns (Foumani et al., 2024), as well as spatial relationships (Li et al., 2021). Adapting positional encoding to geographical coordinates requires some customization (Mai et al., 2020), and when done correctly, it can outperform Cartesian representations of locations (Xu et al., 2021) and is especially helpful to capture non-Manhattan structures like transportation network (Klemmer et al., 2023).

Formally, for a nonnegative integer sequence x, positional encoding produces a vector of length d_{out} whose components alternate between sine and cosine functions' outputs via **Equations (14)-(15)**. Here, each even-indexed dimension $2i$ uses a sine wave, and each odd-indexed dimension $2i + 1$ uses a cosine wave. The wavelength of these sinusoids grows exponentially with the index, controlled by a base constant δ.

$$PE(x, d = 2i) = \sin\left(\frac{x}{\delta^{2i/d_{out}}}\right) \tag{14}$$

$$PE(x, d = 2i + 1) = \cos\left(\frac{x}{\delta^{2i/d_{out}}}\right) \tag{15}$$

The base constant δ is commonly set to 10,000 in attention-transformer applications in its original context—so that higher dimensions encode longer wavelengths or to tune its role to find long-distance relationships between inputs (i.e., the relationship between the first and last words/sentences of a chat-based AI's user input), while lower dimensions focus on localized context. For example, the highest even dimension simplifies **Equation (14)** to $\sin x/\delta$, which has a wavelength $2\pi\delta$, whereas the output values located in lower even dimensions are mapped through a sine function whose wavelength is smaller than $2\pi\delta$, repeating values more frequently as x progresses.

To reiterate, the intuitive benefit of positional encoding is that it provides each scalar input (time or either coordinate of location) with a multi-scale sinusoidal representation: short wavelengths capture localized patterns, and long wavelengths capture global context. In the trip-related data settings, encoding origin and destination times and locations in this way can help models learn various scales of time/space blocks themselves and also help understand relationships between the two places and times.

When adapting positional encoding outside of attention models, however, δ and d_{out} should be tuned considering the expected input range $x \in \mathbf{X}$. This is to preserve two key properties of positional encoding, which are *distance-aware* and *value-isotropic* (Foumani et al., 2024), where the former states "inputs that are close in their original domain remain close in the encoded space," and the latter "no single dimension or frequency band dominates, ensuring a balanced representation across all encoded components." For example, setting δ too low yields wavelengths too short to cover the full range $\mathbf{X}$ breaking distance awareness even for the higher-indexed dimensions, whereas setting δ excessively high pushes wavelengths into the global scale even for the lower-index dimensions, obscuring local patterns. Likewise, if d_{out} is too small, the encoded outputs become anisotropic, unevenly stretching some regions of $\mathbf{X}$ while compressing others with non-diverse representations.

Therefore, we treat both δ and d_{out} as pre-model hyperparameters in our experiments (**Section 5**). For δ, we ensure that the longest wavelength $2\pi\delta$ spans the maximum input range for both time and space. Specifically, for time inputs ranging from 0 to 1440 minutes, we set $\delta = 250$, allowing unique values to be represented up to $x \approx 1570$ at the highest dimension. For geographic coordinates, longitudes in our study range from –94.1531 to –92.8413 and latitudes from 44.6881 to 45.5526. At four-decimal precision, this corresponds near our study area to approximately 26 ft (8 m) in East-West longitude axis and 36 ft (11 m) in North-South latitude axis—tolerances acceptable for travel behavior analysis. Therefore, we multiplied the coordinates by 10,000, rounded to convert them to integers, then shifted so that its minimum becomes zero and its maximum 13,118. This range of numbers can be covered by positional encoding with $\delta = 2500$ while preserving their properties. Additionally, we experimented with various d_{out} settings to ensure sufficient frequency bands for capturing both localized and broad-scale spatiotemporal relationships, while avoiding excessive d_{out} to avoid the well-known ML limitation, the "curse of dimensionality."

4.3. Feature Engineering for Categorical Features

Because categorical features are not inherently numeric, machine learning models cannot process them directly without specialized encodings. This subsection reviews two classes of encoding techniques for categorical variables. First, target-agnostic methods independently represent each category as specific features or values. Second, target-based encodings transform categories based on their relationship to the target variable, which includes the method exclusively used in CatBoost ML model.

4.3.1. One-hot Encoding (OHE)

One-hot encoding (OHE), known as dummy variables in econometrics, is a target-agnostic method that represents each category level as an independent binary feature. For a variable with J levels, OHE creates either J or $J-1$ indicator columns. In full form, each vector has a 1 in the position corresponding to its level and 0 elsewhere; alternatively, dropping one level (usually the last) avoids perfect multicollinearity—a concern in linear models but generally handled internally by tree- or gradient-based ML models. Because OHE does not use any target information, it is free from data leakage and is fast to compute. However, rare categories can still lead to overfitting if models lack sufficient regularization, and high-cardinality variables can create a large number of sparse columns, increasing memory requirements and model fitting runtime.

4.3.2. Label Encoding (LE)

Label encoding assigns each category a unique integer ID from 1 to J (or 0 to $J-1$). This approach is computationally efficient and keeps the feature count unchanged. However, the numeric codes impose an arbitrary ordering (unless the variable is ordinal) and distance between levels, which can mislead many ML models. As a result, label encoding is most appropriate for transforming the target labels into multiclass classification—where the integers simply index the classes, simplifying $\mathbb{C} = \{\mathcal{C}_1, \dots, \mathcal{C}_K\}$ into i-indices—rather than for encoding input features.

4.3.3. Weighted Target Encoding (WTE)

The other three encoding methods, including the weighted target encoding (WTE), are all based on target encoding, also known as the mean encoding method. Target encoding replaces each category level v_j with the empirical distribution of the target within that level. In a K-class classification, the target encoding computes for each class i **Equation (16)** to generate K encoded columns for v:

$$TargetEncoder(v_j, i) = \frac{\left|\{n | x_{nv} = v_j \ \wedge \ y_n = i\}\right|}{\left|\{n | x_{nv} = v_j\}\right|} = \frac{N_{v_j}^i}{N_{v_j}} \tag{16}$$

Here, $N_{v_j}^i$ is the count of samples in class i with v's value v_j, and N_{v_j} the total count for level v_j for variable v. This encoding inflates the i-th component for any category v_j in which class i is particularly prevalent, enabling the model to capture the strength of association between each category level and each target class. However, the original target encoding is prone to overfitting: rare categories with few observations can produce extreme averages by chance, causing the model to memorize noise.

To address this limitation, weighted target encoding (WTE) applies Bayesian smoothing to shrink level-specific estimates toward the overall class prior $P_i = \Pr(y = i)$, reducing variance for categories with few observations. Introducing a smoothing parameter m that controls the influence of the prior, WTE works with **Equation (17)** for given v_j and i.

$$WTE(v_j, i) = \frac{N_{v_j}^i + mP_i}{N_{v_j} + m} \tag{17}$$

If m is set to 5 (a customary value), the prior P_i contributes the equivalent of five virtual observations. In practice, this means we need at least $m + 1 = 6$ actual samples before the empirical class frequency $N_{v_j}^i / N_{v_j}$ outweighs the prior; otherwise, the encoded value remains closer to P_i. By blending global and level-specific information, WTE mitigates the risk that rare categories produce extreme output values by chance. However, WTE still suffers from target leakage because it uses each sample's own target to compute its encoding, which can lead to overly optimistic performance at training.

4.3.4. k-Fold Target Encoding (kTE)

This encoding method extends basic target encoding or **Equation (16)** to mitigate data leakage by computing each sample's encoding using only other observations. Specifically, the dataset $\mathbb{D}$ is first partitioned into k folds. For all points in the k'-th fold, their target-based statistics are estimated from the remaining $k-1$ (i.e., out-of-fold means). As a result, no sample's own target value contributes to its encoding.

In practice, 5-fold target encoding (5TE) and leave-one-out target encoding (LOOTE, when $k = N$) are most commonly used.

4.3.5. Ordered Target Encoding (OTE)

CatBoost (CB) model, as described in **Section 4.1.3**, employs the ordered target encoding to eliminate target leakage entirely. First, it randomly permutes the dataset and assigns a new index to each row. Then, to encode the n'-th instance with category v_j, CB uses only the subset of preceding observations $\{1,2,3,\dots,n'-1\}$ that share v_j (i.e., data rows being used for n'-th entry is less than $n'-1$). It finally applies the smoothed target-mean formula for WTE in **Equation (16)** with smoothing parameter m set to 1, where prior P_i still denotes the global class frequency for i. Because each sample's own label is excluded from its encoding, this method prevents any leakage of target information.

Table 3 illustrates these encoding methods on a toy dataset with $K = 2$ binary target classes ($i \in \{1,2\}$) and a single categorical feature v of $J = 3$ levels ($x_{nv} \in \{v_1, v_2, v_3\}$) over $N = 10$ observations. We compare one-hot encoding, weighted target encoding with $m = 2$, and 5-fold target encoding, and ordered target encoding (assuming the reordered dataset sorted by has the same sequence or $n = n'$), excluding the label encoding only used for encoding the target $C_i \to i$ and will not be used in the following sections.

Table 3 Comparison of different categorical features encoding methods with a toy dataset

Raw data			**OHE**			**WTE ($m = 2$)**		**5TE**			**OTE**		
n	v	i	v_1	v_2	v_3	$v^{i=1}$	$v^{i=2}$	**Fold**	$v^{i=1}$	$v^{i=2}$	n'	$v^{i=1}$	$v^{i=2}$
1	v_1	1	1	0	0	0.40	0.60	1	0.00	1.00	1	0.30	0.70
2	v_2	2	0	1	0	0.23	0.77	1	0.25	0.75	2	0.30	0.70
3	v_2	2	0	1	0	0.23	0.77	2	0.25	0.75	3	0.15	0.85
4	v_3	2	0	0	1	0.32	0.68	2	0.50	0.50	4	0.30	0.70
5	v_3	2	0	0	1	0.32	0.68	3	0.50	0.50	5	0.15	0.85
6	v_2	1	0	1	0	0.23	0.77	3	0.00	1.00	6	0.10	0.90
7	v_3	1	0	0	1	0.32	0.68	4	0.00	1.00	7	0.10	0.90
8	v_2	2	0	1	0	0.23	0.77	4	0.25	0.75	8	0.33	0.67
9	v_2	2	0	1	0	0.23	0.77	5	0.25	0.75	9	0.26	0.74
10	v_1	2	1	0	0	0.40	0.60	5	1.00	0.00	10	0.65	0.35

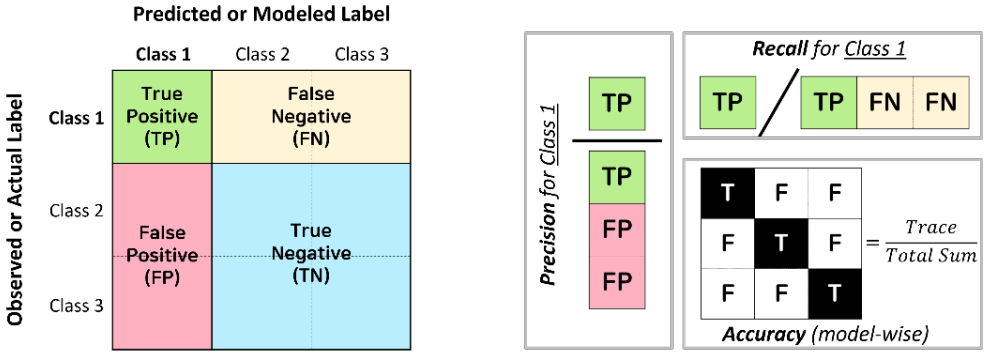
Although the number of observations $N = 10$ in the demonstration is too small to comprehensively show all instances of the six possible $v \times i$ combinations, the following intuitions can be drawn for each encoding method.

- **WTE**: It successfully smooths category means toward the global prior, so a rare level's encodings, for example, $v_1 \to [0.4, 0.6]$ (for $n = 1, 10$) remain reasonably close to $[P_1, P_2] = [0.3, 0.7]$; if it were the standard target encoding applied, v_1 would have been mapped to $[0.5, 0.5]$ instead. However, it still uses each sample's own label to derive the values.
- **5TE**: It eliminates direct leakage by using out-of-fold counts; in fold 1 of rows 1 and 2, statistics from rows 3-10 are used to encode the values. However, without smoothing, rare levels in a fold can go to exactly 0 or 1, leading to high variance and unstable results.
- **OTE**: It also avoids leakage and at the same time provides smoothened estimates initially, then gradually adapting as n' proceeds. However, it is computationally more complex to implement, and early rows rely heavily on the global prior.

5. Experiment Design

This section lays out our experimental design to identify, across combinations of feature-engineering strategies, ML models, and hyperparameter settings, those configurations that are 1) most accurate in terms of prediction, 2) most stable when the feature engineering choices vary, and 3) most robust to changes in hyperparameters for a given feature engineering applied. Insights from this analysis can directly inform the practical deployment of trip purpose inference and survey-AFC data fusion for continuous and adaptive travel behavior analysis.

We begin by defining how to evaluate the prediction accuracy of classification models under these varied settings. A confusion matrix, a contingency table with actual classes on one axis and predicted classes on the other, is both a model tuning aid and a performance summary (**Figure 6**). It is a practice to create each row to signify instances belonging to an actual class, while each column represents instances predicted to belong to a specific class. Consequently, each cell encapsulates *True Positive*, *True Negative*, *False Positive*, or *False Negative* from a specific target class's point of view. The counts of these four categories facilitate the computation of various model evaluation metrics.



(a) Confusion matrix (counts) with respect to **Class 1** (b) Class/model evaluation metrics

Figure 6 Example of a confusion matrix and performance evaluation metrics for classification models

The metrics of *Precision* and *Recall* can be calculated for each class. Precision is the fraction of correctly predicted instances among all instances predicted as that class (column-wise fraction), while recall is the fraction of correctly predicted instances among all actually belonging to that class (row-wise fraction). The *F1 Score*, the harmonic mean of precision and recall (i.e., $2 \times Precision \times Recall/(Precision + Recall)$), captures the balance between these two, and *Accuracy* measures the proportion of all correctly predicted labels (i.e., diagonal cells) across the entire matrix, or the trace divided by sum. All metrics range from 0 to 1, reflecting the model's predictive adequacy both per class and overall.

In practice, we measure model performance on a *test dataset*, which is a disjoint subset separate from the *training dataset* used to fit the models. This separation is necessary because even with regularization, models tend to overfit the training data and then struggle to generalize to unseen examples. Evaluating model performance on a separate test set helps ensure that conclusions about model effectiveness are based on truly unseen data, providing a more objective assessment of generalizability. A common approach to implement this is an *80/20 split*, indicating 80% of the data is used for training and 20% for testing. Among the various metrics introduced in **Figure 6**, when applied to the test dataset, accuracy is the most widely used for comparing models and configurations. However, if the modeling goal focuses on a particular level of the target, it is equally valid to use a specialized metric like a specific label's F1 score or to collapse unimportant labels' prediction results. For our experiment, we define an adjusted version of accuracy that evaluates how well the model predicts the "Home" and "Work" labels in the test dataset while combining the less critical (in terms of our data fusion context to identify commutes) "Shop" and "Others" labels into a single category. From the following subsections onward, the term *accuracy* refers to this adjusted accuracy applied to the test dataset.

5.1. Experiment Structure: Two-Staged Studies with Multiple Model Fitting Trials

Each ML model has diverse possible hyperparameter configurations, as already described in **Section 4.1**. Unlike a model's internal parameters (for example, weights and biases in neural networks), hyperparameters must be chosen before training. Proper tuning of these values is crucial because they govern the model's learning dynamics, generalization ability, and overall performance (Arnold et al., 2024). Since no universal analytic method exists for finding the optimal combination, researchers follow a practical, case-by-case workflow: begin with default settings, incorporate domain expertise, and then refine the settings through cross-validation (i.e., using the intermediate "validation" accuracy).

In parallel, the feature engineering strategies described in **Section 4.2** (continuous variables) and **Section 4.3** (categorical variables) introduce another layer of variation pre-model. We apply each method to two versions of our dataset (**Table 2**) to explore how these preprocessing choices affect both validation accuracy and the variance of the accuracy in ML-based trip purpose classification. Specifically, we design a two-stage experiment to systematically assess these factors. The first stage, named *feature engineering comparison*, generates dataset variants processed with each feature engineering pipeline for every model; these configurations are summarized in **Table 4**.

Table 4 Settings of experiment stage 1: investigating variable consolidation and feature engineering techniques for each of the five ML models

Techniques Set	Technique	Note	Techniques Set	Technique	Note
Day/Season consolidation	No consolidation	2 variables, 9 levels	**Temporal feature encoding**	Raw	Integer (0-1440)
	Tight consolidation	2 variables, 4 levels		Raw + cyclical encoding	Wavelengths set to 1440
Spatial feature encoding (4+9 settings)	Raw	Latitude/longitude		Raw + positional encoding	$d_{out}^{t} = 4$, $\delta^{t} = 250$
	Raw + polar	$\theta \in \{-\pi, \pi\}$	**Categorical feature encoding**	One-hot encoding (OHE)	Each level as column
	Raw + elliptic	$v \in \{-\pi, \pi\}$		Weighted target encoding (WTE)	$m = 5$
	Raw + Zone-based	31 zones		k-fold target encoding (kTE)	$k = 5$
	Raw + positional encoding	9 d_{out}^{s}, δ^{s} variations		Ordered target encoding (OTE)	Applies only to CB

The possible combinations of data preprocessing and feature engineering yield 1,014 distinct experiments. CB, using only ordered target encoding for categorical features, contributes 78 (2*13*1*3) of these runs, while each of the other four models (RF, XG, NN, SVM) is evaluated under all three categorical-encoding schemes (OHE, WTE, 5TE), producing 234 (2*13*3*3) experiments per model. For spatial features, positional encoding is parameterized by $d_{out}^{s} \in \{8,16,24\}$ and $\delta^{s} \in \{2500, 5000, 10000\}$, which are selected based on discussions in **Section 4.2.5**, yielding nine configurations whenever spatial positional encoding is applied. Although we initially explored a broader range, only these nine settings maintained acceptable performance; the rest led to significant degradation. For temporal features with the similar reason, we fixed the encoder at $d_{out}^{t} = 4$ and $\delta^{t} = 250$.

The results of the first stage are summarized in **Section 6** using the techniques outlined in **Table 4**. These results are expected to identify which preprocessing methods work best for each model and, conversely, which models exhibit the greatest variability under a given pipeline.

The second experimental stage, also described in **Section 6**, is named *hyperparameter tuning comparison*, and is conducted independently for each of the five models. Here, each model is trained under its most and least favorable pipelines from Stage 1. This probes the impact of individual hyperparameters, revealing which parameters most strongly affect performance, and which combinations yield the best and worst outcomes.

5.2. Experiment Tool: Automated Hyperparameter Tuning with Optuna

In both experimental stages, it is necessary to conduct multiple model-fitting trials by sampling hyperparameter settings from broad, predefined ranges that encompass common values in the ML literature. In an ideal setting when all hyperparameters are categorical, such as RF's $max\_features$ or SVM's $kernel$, an exhaustive grid search over their Cartesian product is feasible. However, many critical hyperparameters accept integer or continuous values, some of which must be sampled on a log scale (e.g., learning rates). To navigate these mixed search spaces efficiently, we employ automated hyperparameter tuning tools that iteratively propose new configurations based on intermediate performance and trial history.

Automated tuning is essential for both comparing models across the varied datasets generated in Stage 1 and for fine-grained hyperparameter exploration in Stage 2. Early Bayesian optimization frameworks like *Hyperopt* (Bergstra et al., 2015) paved the way for this approach. More recently, *Optuna* (Akiba et al., 2019) has become widely adopted due to its fast convergence to a customizable objective, built-in early stopping triggers for unpromising trials, and a simple Python API. We use Optuna in our experiments to ensure a systematic, reproducible, and efficient search over the hyperparameter space of each model.

Optuna's key technical advantages lie in its sampling and pruning algorithms. Sampling, the process of selecting which hyperparameter combination to evaluate at each trial, can be done via independent methods—such as Tree-structured Parzen Estimator (TPE)—or via strategies that exploit inter-parameter correlations—such as Covariance Matrix Adaptation Evolution Strategy (CMA-ES). Optuna's support for mixing TPE and CMA-ES often outperforms other automated tuning frameworks; in their comparative tests (Akiba et al., 2019), Optuna's was defeated by only one method (GPyOpt), but at the same tim,e the competitor demanded substantially more computational time. On the other hand, pruning

stops underperforming trials early to save time and resources. Optuna employs scalable, parallelizable Asynchronous Successive Halving (ASHA) to aggressively terminate poor trials, cutting overall computation while still aggregating robust performance estimates.

In our experiments, we configure Optuna to tune hyperparameters to maximize each model's test dataset-measured accuracy (hereinafter, interchangeably used as "objective" depending on context). Full details of the hyperparameter ranges explored for all five models are described in **Appendix B**.

6. Experiment Results

6.1. Results of Experiment Stage 1: Effects of feature engineering on fine-tuned models

Table 5 reports, for each ML model, two key marginal[1] averages for each feature engineering technique:

- (Average) **Top 5 mean**: the average across studies, where each study's value represents the average of the top five objective values out of 50 Optuna trials; i.e., the average of averages is reported in the table.
- (Average) **Gap**: the average across studies, each from the difference between the top 5 mean and the Study mean across all 50 trials.

Recall that in Stage 1, we ran 50 trials per feature-engineering pipeline ("Study") as outlined in **Table 4** with automated hyperparameter tuning applied using Optuna. Due to the notorious property of ML that necessitates multiple tuning trials, even cursory practitioners/modelers typically perform at least 5-10 tuning runs to fit a model. Therefore, summarizing the top five of 50 provides a realistic estimate of the high-end performance they might achieve. We average the top five rather than reporting a single best trial because the true optimal configuration is unknown. This means that an outlier best trial could simply reflect luck, whereas the average of the top five smooths out randomness and highlights practical numbers. Yet, it is worth noting that Optuna's guided sampling tends to concentrate strong trials in promising regions, so these top 5 means will generally exceed the "best" value of what one would obtain via a non-exhaustive grid search or manual tuning from a limited number of trials.

By contrast, each study's average objective value from all 50 trials, approximates the typical performance (or "decent" performance as it benefits from Optuna's efficient search as well; see **Figure 11** for more detail) under a given pipeline. The gap between the top 5 mean and the study mean quantifies the stability of each feature engineering employed: a larger gap indicates that only a narrow set of hyperparameters achieves high performance (i.e., lower robustness), whereas a smaller gap signals the consistency of the expected results.

Table 5 Marginal effects of feature engineering technique by model: averages of the top 5 objective mean and gap (top 5 mean – study mean)

	RF		**XG**		**CB**		**NN**		**SVM**		**Aggregated**	
	Top 5	**Gap**	**Top 5**	**Gap**	**Top 5**	**Gap**	**Top 5**	**Gap**	**Top 5**	**Gap**	**Top 5**	**Gap**
Feature engineering set 1: Variable level consolidation												
Raw levels	**0.629**[a]	0.013	0.645	0.010	0.643	0.012	0.615	0.016	**0.621**	0.067	**0.629**	0.025
Consolidated levels	0.628	0.014	0.645	0.010	0.643	0.012	0.615	0.015	0.619	0.071	0.628	0.026
Feature engineering set 2: Spatial feature encoding												
Raw coordinates	0.616	0.013	0.639	0.011	0.634	0.012	0.595	0.019	0.594	0.050	0.613	0.023
Polar	0.629	0.013	0.644	0.010	0.642	0.013	0.606	0.018	0.607	0.055	0.623	0.023
Elliptic	0.628	0.012	0.646	0.010	0.639	0.011	0.608	0.020	0.606	0.054	0.623	0.023
Zone	0.625	0.012	0.642	0.010	0.636	0.010	0.610	0.014	0.611	0.054	0.623	0.022
Positional encoding [b]	**0.631**	0.014	**0.648**	0.011	**0.648**	0.014	**0.626**	0.014	**0.636**	0.080	**0.636**	0.028
Feature engineering set 3: Temporal feature encoding												
Raw timestamps	**0.632**	0.014	**0.646**	0.010	**0.645**	0.013	0.619	0.016	**0.625**	0.069	**0.632**	0.026
Cyclical	0.624	0.012	**0.646**	0.010	0.644	0.012	0.619	0.015	0.623	0.068	0.629	0.025
Positional encoding	0.628	0.014	0.643	0.009	0.640	0.012	0.608	0.016	0.612	0.070	0.624	0.026
Feature engineering set 4: Categorical feature encoding [c]												
One-hot encoding	**0.629**	0.014	**0.646**	0.010	-	-	**0.616**	0.015	**0.621**	0.067	**0.628**	0.026
Weighted target enc.	**0.629**	0.013	0.645	0.010	-	-	0.615	0.016	0.619	0.069	0.627	0.027
5-fold target enc.	0.627	0.013	0.644	0.010	-	-	0.615	0.016	0.620	0.071	0.626	0.028
Model-wise statistics												
Average	**0.628**	**0.013**	**0.645**	**0.010**	**0.643**	**0.012**	**0.615**	**0.011**	**0.620**	**0.069**	**0.628**	**0.026**
Standard deviation	**0.002**	**0.006**	**0.001**	**0.003**	**0.002**	**0.005**	**0.007**	**0.016**	**0.016**	**0.015**	**0.015**	**0.025**

[a] Bolded the highest top 5 mean and underlined the largest gap (did not if all are tied until the third decimal) for each feature engineering set for each model
[b] Aggregated the values from studies with the highest top 5 means for each model across studies with different δ and d_{out} settings
[c] Ordinal target encoding is only applied to CatBoost (CB); thus the "Aggregated" column indicates the values from the four (RF, XG, NN, SVM) models

The results can be interpreted as follows:

- **Overall results:** XG demonstrated the best average performance (0.615), while SVM exhibited the largest performance gap (0.069). When investigated with the aggregated statistics (the last two columns), the top 5 means did not vary significantly across different feature engineering techniques for variable level consolidation and categorical feature engineering.
- **Variable level consolidation**: Consolidating unique categorical levels from multinomial to binary for *Season* and *Day of week* variables had a minimal impact, whereas applying the consolidation seems to negatively affect accuracy.
- **Spatial feature encoding**: Studies that applied positional encoding outperformed all other methods for all models. It is followed by elliptic and polar transformation for tree-based models (RF, XG, CB), whereas zonal representation has greater advantages in non-tree based NN, SVM models than coordinate transformation. Notably, the gaps are also the largest with positional encoding except for NN where the largest gap was found in elliptic transformation.

[1] Here, "marginal" means that for each row reported, we average the results over all studies using the very feature engineering technique, effectively integrating out the others in the set the used technique belongs to. For example, among the 234 RF studies, the values shown in *Temporal feature encoding-Cyclical* are the mean metrics from the 78 studies that applied cyclical encoding while varying techniques in the other feature engineering sets (i.e., RF studies with all possible combinations of variable level consolidation, spatial encoding, and categorical encoding techniques.

- **Temporal feature encoding**: No significant benefits were observed from applying advanced methods to timestamps, as they degrade the accuracy compared to the raw timestamps-only studies. A plausible explanation is that, unlike spatial features where origin and destination can inherit different properties (e.g., residential vs. commercial), departure and arrival times generally fall within the same time block (e.g., morning commute). In such cases, ML models may not require additional temporal information.
- **Categorical feature encoding**: Advanced encoding methods showed no improvement over OHE. This aligns with a finding from a comparative study suggesting that for low-cardinality variables (e.g., fewer than 10 unique levels applied to our data with 4-level *Season* and 5-level *Day of week*), OHE outperforms more complex feature engineering methods (Pargent et al., 2022).

Overall, the selection of feature engineering techniques does not seem to have a significant impact except for converting spatial features, and in some cases, keeping the raw format of the dataset without applying feature engineering performed the best. However, it should be noted that this conclusion is drawn from fine-tuned[2] hyperparameters for each study, where the severe degrading effects of poor hyperparameter tuning related to different combinations of feature engineering techniques are explored in **Section 6.2**.

In addition to the aggregated results (or marginal application) shown in **Table 5**, some feature engineering techniques are worth investigating in more detail, as certain contexts may reveal synergies between techniques from different feature engineering sets. A good example is the use of non-OHE techniques in categorical feature engineering, particularly when origin/destination locations are encoded as zone representations. In the basic OHE form, each origin and destination locations are expanded up to 31 data columns, respectively, whereas other encoding methods require only four data columns each (corresponding to the number of target level trip purposes). **Figure 7** compares the categorical encoding methods using datasets generated with locations represented as 31-level zone memberships (i.e., up to 62 columns added). This yields 18 studies for each ML model excluding CB, where it only utilizes ordered target encoding, and thus, for CB, 6 studies.

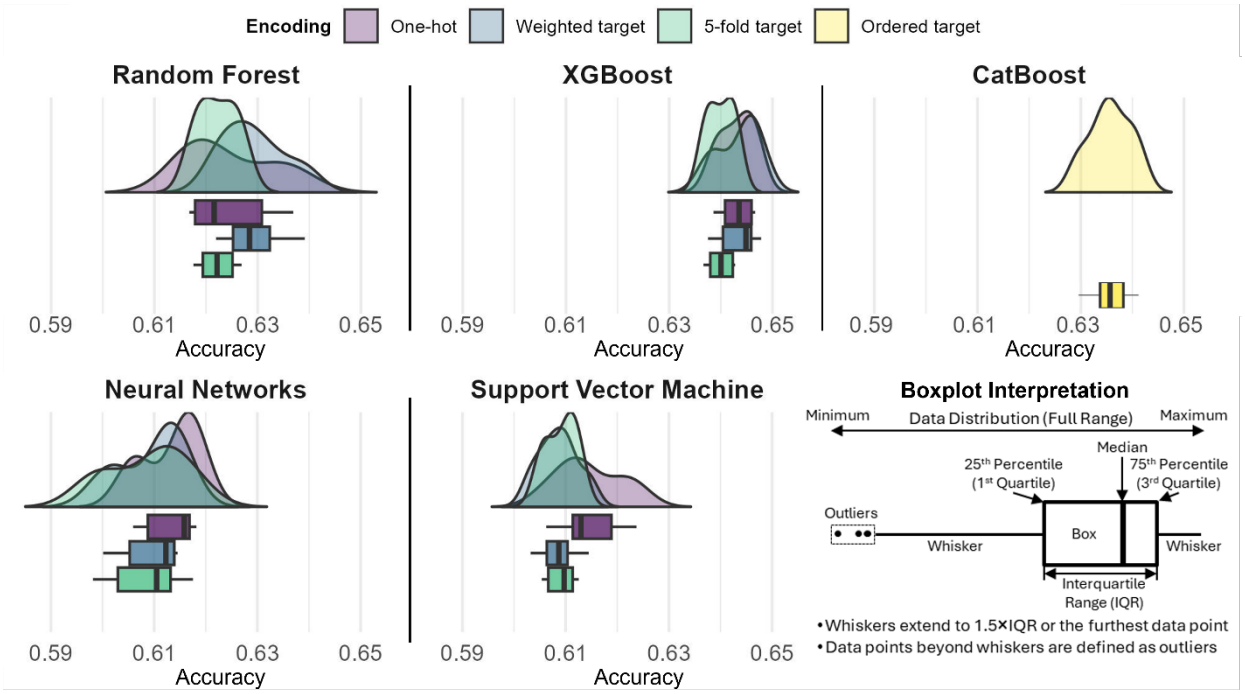


Figure 7 Top 5 mean range by categorical feature engineering technique by model from 78 studies with zonal representation of locations

The box plots in the first panel indicate that for RF, OHE yields the lowest median (the vertical band in each box), and its 3rd quartile value (the rightmost end of the box) is lower than that of WTE, where all three encodings exhibit relatively wide interquartile ranges. In XG, each encoding produces a compact interquartile range, indicating consistent performance regardless of the categorical feature engineering technique chosen, while WTE delivers the highest median and the single best study. CatBoost also maintains a tight interquartile range for its built-in encoding technique, OTE. By contrast, NN and SVM behave differently: OHE yields the best performance for both models across all quartiles, although 5FE's highest data point stretches beyond that of OHE in NN. In those models, applying advanced encodings generally degrades the accuracy. Therefore, it can be concluded that only the weighted target encoding adds little value for RF and XG, where 5FE can rarely bring a pleasant surprise for NN, even for a dataset with a moderate level of cardinality stemming from the zone representation.

Meanwhile, the representation of location requires additional investigation, as previous studies have pointed out that locations are among the most important predictors in trip purpose inference and travel behavior analysis (Cui et al., 2018; Gao et al., 2024; Luo et al., 2021). The ranges of values in **Table 5** also show spatial engineering yielded the most diverse performance among all technique sets investigated. To compare the different spatial encoding methods, we created **Figure 8**, which is a visualization of **Table 5** zooming into the spatial feature encoding techniques, including the distributions and medians of the top 5 means as box plots, whereas **Table 5** only shows the mean of the top 5 means.

The figure reinforces several insights from **Table 5**. First, feeding more information into an ML model via advanced spatial encodings never hurts compared to using raw latitude/longitude coordinates, supported by the observations that for each model, the lowermost located purple boxes' median value always falls below the others except in one case from CB's zone assignment option. Next, in all five models, positional encoding achieved its highest 1st, 3rd quartile, and median values for accuracies, which almost dominate the other techniques, whereas we can find some competitive ranges from XG and RF. In particular, the minimum accuracy achieved from positional encoding is better than any other values from the other techniques applied for SVM and NN. Furthermore, positional encoding's 1st quartiles from CB are even higher than the highest accuracies from other techniques. Generally, polar and elliptic transformations of coordinates followed the positional encoding, sometimes significantly better than zone assignment (in CB, RF), whereas for the other models those rankings and overlaps of distributions vary.

[2] It is evident that the top 5 means come from at least the top 10% of the best-performing hyperparameter combinations for each study, and even the study mean used to derive the gap possesses biased performance towards higher-performance area compared to the median mean one can achieve (as depicted in **Figure 11**).

Overall, advanced spatial feature engineering techniques tend to boost performance, and even the least effective other than raw coordinates, the zone assignment, is worth trying. If only one feature engineering technique for spatial data can be tested, positional encoding is most likely to deliver the largest gains in both median and maximum performance while keeping variance (interquartile ranges illustrated as the width of each box) moderately low across cases. However, positional encoding introduces two tuning parameters δ and d_{out}, so we created **Figure 9** to explore their optimal settings from studies that used the technique (702 total).

As discussed in **Section 4.2.5**, these parameters are simpler to tune than most model-specific hyperparameters but still require context-driven calibration. The base parameter δ defines the longest sinusoidal wavelength and should be at least as large as the maximum range of the scaled inputs (i.e., $2\pi\delta > |\mathbf{X}|$). The output-dimension parameter d_{out} sets the number of frequency bands: higher values represent the space with more multi-scale bands, but excessively large d_{out} can overwhelm the model and hurt performance.

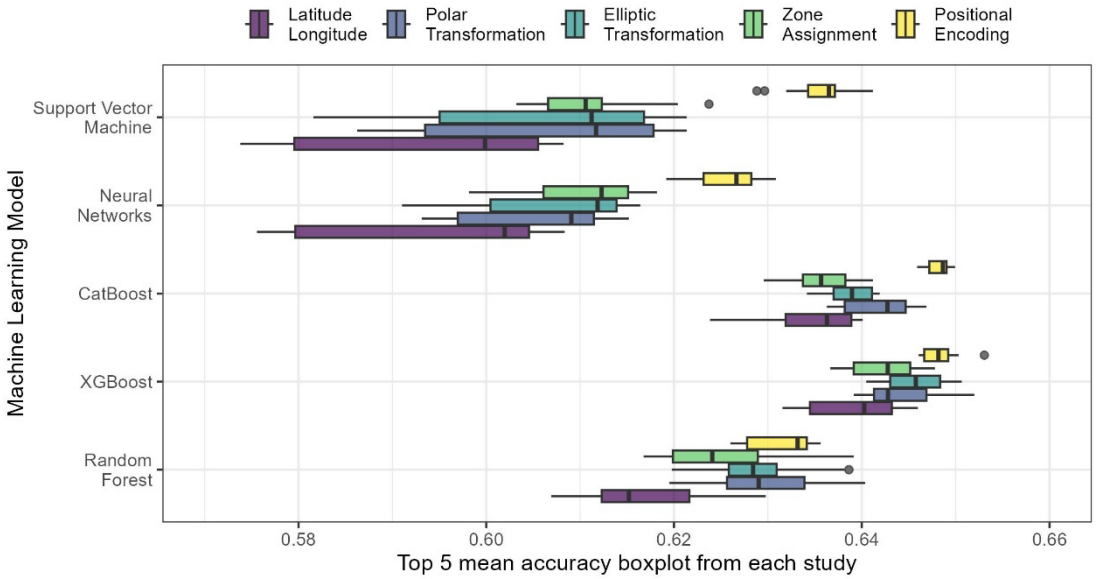


Figure 8 Comparison of the effects of spatial feature engineering techniques applied to different ML models

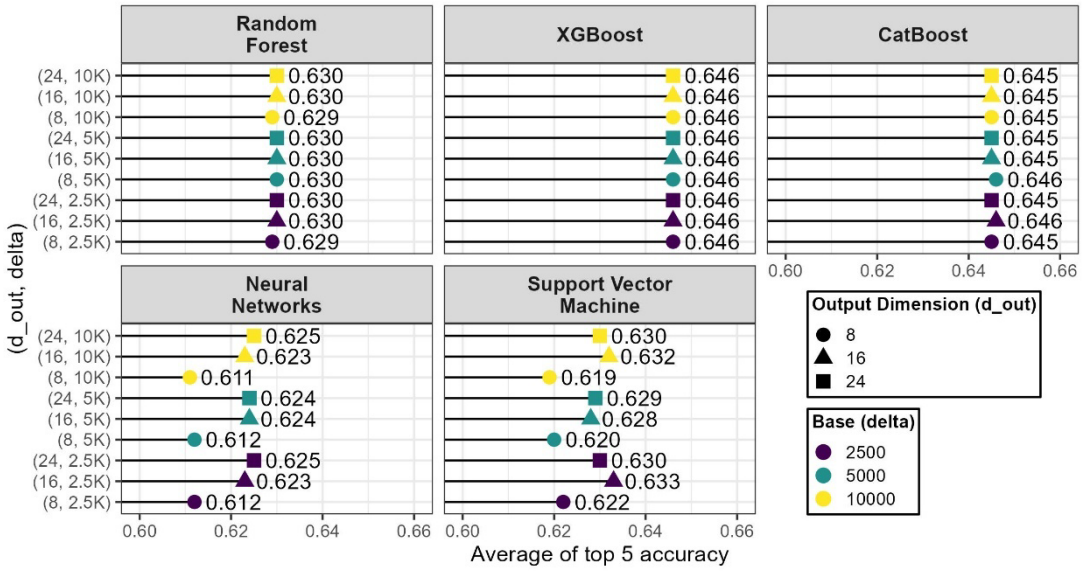


Figure 9 Effects of the positional encoding parameters on average model performance by model

The average performance values show little overall variation across the tested (d_{out}, δ) combinations, almost steady in the tree-based methods in the first row of panels in the figure. A clear pattern emerges for the other two NN and SVM models: d_{out} (denoted by marker shape) has a stronger influence on accuracy than δ (denoted by color). For every δ value for the two models, their lowest accuracy is achieved at the smallest[3] $d_{out} = 8$, whereas SVM showed the best values at the intermediate setting $d_{out} = 16$ for two out of three δ values, outperforming both 8 and 24. In contrast, even for the NN and SVM models, δ exhibits no significant effect across models. This likely reflects the fact that even our smallest $\delta = 2500$ yields wavelengths long enough to span the full range of scaled coordinates, and with adequate d_{out} frequency bands, the sinusoidal encodings capture the necessary spatial scales regardless of the different δ options represented.

Finally, **Table 6** displays, for each model, the feature engineering pipelines yielding the highest and lowest average accuracy, along with the corresponding numbers of input features generated. Apart from the consistent advantage of positional encoding for spatial features, no single pipeline and/or individual technique is universally superior or inferior. Therefore, practitioners should evaluate multiple preprocessing strategies and rigorously test each combination to identify the optimal pipeline for their specific model and context.

[3] Our lowest reported d_{out} is 8, yet preliminary tests with $d_{out} = 4$ degraded performance across all models with poorer representation of locations. A setting of $d_{out} = 2$ is thus rejected, and odd values are not recommended because positional encoding alternates sine and cosine in pairs, requiring an even dimension.

Table 6 Best and worst feature engineering pipelines for each ML model in terms of top 5 mean accuracy

Model	Pipeline	Top 5 mean performance	Variable level	Spatial encoding	Temporal encoding	Categorical encoding	Number of features
Random Forest	Best	0.640	Raw	Polar transformation	Raw	Weighted target	26
	Worst	0.607	Raw	Raw	Cyclical encoding	5-fold target	26
XGBoost	Best	0.653	Raw	Positional encoding (16, 10K)[a]	Cyclical encoding	One-hot	92
	Worst	0.632	Consolidated	Raw	Positional encoding	Weighted target	21
CatBoost	Best	0.650	Raw	Positional encoding (16, 10K)	Raw	-	77
	Worst	0.624	Consolidated	Raw	Positional encoding	-	21
Neural Networks	Best	0.631	Raw	Positional encoding (24, 10K)	Raw	5-fold target	118
	Worst	0.576	Consolidated	Raw	Positional encoding	5-fold target	21
Support Vector Machine	Best	0.641	Raw	Positional encoding (24, 10K)	Cyclical encoding	One-hot	128
	Worst	0.574	Raw	Raw	Positional encoding	5-fold target	30

[a] Denotes (d_{out}, δ)

6.2. Results of Experiment Stage 2: How Hyperparameter Tuning Matters

In Stage 2, we conduct in-depth analyses of each ML model under its "Best" and "Worst" feature-engineering pipelines identified in Stage 1 (**Table 6**). For each pipeline, we run 110 Optuna trials. We first present cross-model summaries, then turn to detailed, model-specific investigations to 1) rank hyperparameters by their impact on performance—thereby identifying any redundant or low-impact parameters; 2) examine the hyperparameter combinations used in the top ten trials per each model; and 3) explore interactions among key regularization parameters. All models use the same hyperparameter search spaces as in Stage 1 (see **Appendix B**).

Figure 10 compares the distribution of performance for the best- and worst-performing feature engineering pipelines for each model. To ensure fairness, we excluded the top five and bottom five trials in each study, so each boxplot reflects the results of the central 100 trials across 110 Optuna runs. Because Optuna's tuning is not random like grid search but a result from the advanced sampling and pruning, in each box, the left edge (the 1st quartile) represents the typical performance achieved from a moderate amount of hyperparameter tuning, while the right edge (the 3rd quartile) shows what diligent tuning can deliver.

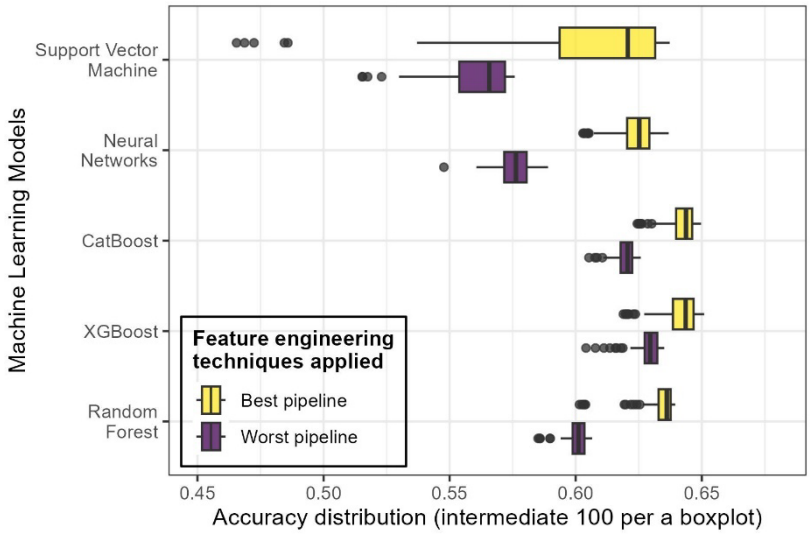


Figure 10 Accuracy distribution with "Best" and "Worst" feature engineering pipelines applied to each ML model

Comparing the edges of boxplots for the worst and best pipelines by each model, we see that even a well-tuned hyperparameter set under the worst pipeline can surpass a poorly tuned set under the best pipeline (an overlap of best-worst pipelines observed in all models except NN). This finding underscores that both feature engineering and hyperparameter tuning are crucial for building a high-performing model. The effect of tuning is directly visible in the width of the boxes and the length of the whiskers, which can span 5–10 percentage points. Moreover, each pipeline for each model has diverse interquartile ranges, which again highlights the importance of a modeler's effort to retrieve the best-performing model.

Figure 11 illustrates, for each model's best feature engineering pipeline, the evolution of the maximum objective value over Optuna's trials under its pruning and sampling strategies. The red curve, tracing the incumbent best value, is at most times concave and lies above the straight-line segments that would connect any two incumbent best values directly. Model-wise, the three tree-based models converge rapidly to their ultimate peak, while RF and SVM exhibit more low-accuracy outliers at intermediate trials, with SVM showing the greatest variability.

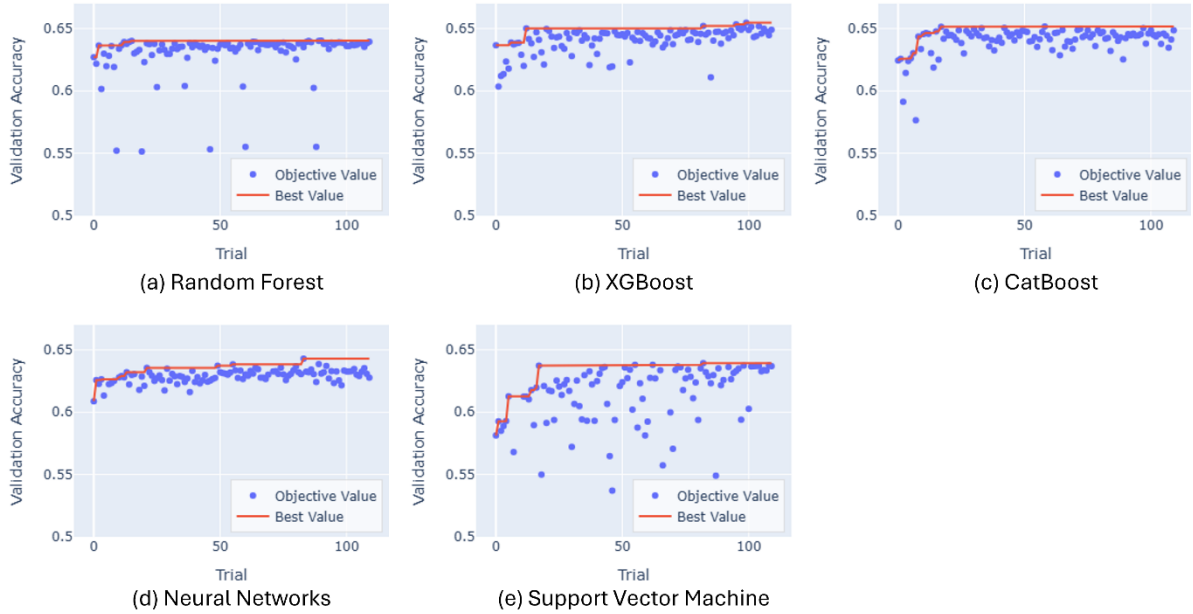


Figure 11 Hyperparameter tuning trials for each model

This result reinforces our use of the "Top 5 mean" in Stage 1 (**Section 6.1**) as a realistic estimate of the highest performance achievable in practice: it focuses on the most successful trials for a given feature engineering pipeline, which occupy a small, high-performance region of the search space. Similarly, the "Gap" metric of Stage 1 may understate the true spread as even a few trials highly optimize the models. Thus, in **Figure 10**, we can safely say even the 1st quartile already represents a level of accuracy that a practitioner could attain with moderate tuning.

On the other hand, a selection of feature engineering and ML models are, in practice, highly related to the running time to fit or train. **Figure 12** shows the benchmarks of the studies for the best pipeline for each ML model, categorized by whether CPU or GPU is used. All experiments were performed on the same device (laptop). For CPU benchmarks, we conducted the studies under Windows 11 with Python 3.12 on a 13th Gen Intel® Core™ i7-1360P, limiting thread usage to 12 of the 16 logical cores. GPU experiments were conducted in the Windows Subsystem for Linux (WSL), as required by the RAPIDS framework (RAPIDS Development Team, 2023), which enables model fitting of RF and SVM with GPU. A single NVIDIA GeForce RTX 3050 Laptop (4 GB) with CUDA driver version 12.4 was utilized for the GPU studies.

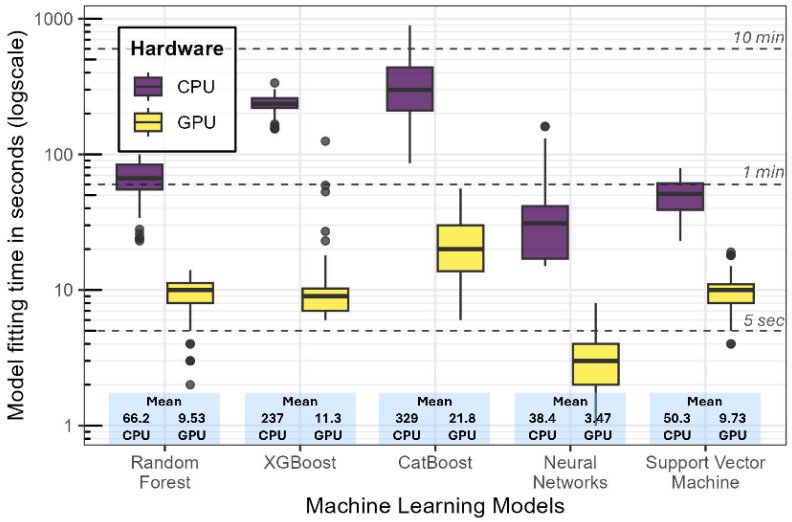


Figure 12 Benchmark of each model's fitting time distribution by hardware

Finally, **Table 7** details model evaluation metrics for the best trial (in terms of accuracy) for the best feature engineering pipeline applied study for each model, including precisions, recalls, and F1 Scores of the two most appearing labels in our OBS, "Home" and "Work," which are also the most important trip purposes for analyzing commute trip behaviors as we demonstrate with the data fusion in **Section 7**.

Table 7 Evaluation metrics from the best performing hyperparameter combinations for each ML model's single best trial

Model	"Home" Label			"Work" Label			Accuracy	GPU Training Time (sec)
	Precision	Recall	F1 Score	Precision	Recall	F1 Score		
Random Forest	0.618	**0.806**	0.700	0.631	**0.691**	**0.660**	0.640	13
XGBoost	0.653	0.773	**0.708**	0.635	0.671	0.652	**0.655**	10
CatBoost	**0.669**[a]	0.713	0.690	**0.650**	0.642	0.646	0.652	14
Neural Networks	0.644	0.750	0.693	0.652	0.629	0.641	0.643	**8**
Support Vector Machine	0.658	0.698	0.678	0.632	0.664	0.648	0.639	10

[a] Bolded the best value for each evaluation metric

It is worth noting that each row in the table highlights the model with the highest overall accuracy, not the model with the best score in any single column (for example, "Home" precision). For instance, SVM sometimes achieves very high precision or recall (over 0.95) for one class but at the expense of extremely low values (around 0.2) for other metrics. Those imbalances drive down its F1 score and overall accuracy, reflecting overfitting rather than genuine performance gains.

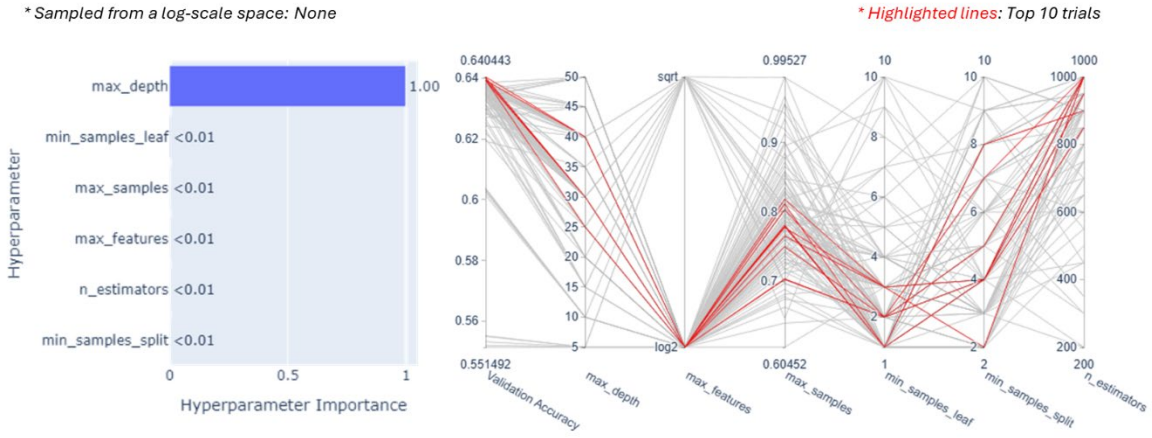
The results suggest, in our study context and settings after optimal feature engineering and thorough hyperparameter tuning, XGBoost achieved the highest accuracy, closely followed by CatBoost. Neural networks and SVM recorded the lowest accuracies, though their performance remained competitively strong. Moreover, when training on GPU hardware, all models exhibit practical training times—even when conducting 100 or more tuning trials.

Before presenting the model-specific results, we briefly recap how Optuna derives hyperparameter importance via functional ANOVA (fANOVA) by Hutter et al. (2014). Given an Optuna study, it first fits a surrogate random forest regressor to the trial data, treating each hyperparameter X_h and the objective value $f(X)$ as the target or dependent variable. Then, fANOVA decomposes the total variance of the objective into contributions attributable to each hyperparameter, yielding the importance I_h as in **Equation (18)**.

$$I_h = \frac{\mathrm{Var}_{X_h}[\mathbb{E}[f(X)|X_h]]}{\mathrm{Var}[f(X)]} \tag{18}$$

Here, the numerator measures how much the conditional expectation of f varies when X_h is changed, and the denominator normalizes these contributions, so they sum to one or $\sum_h I_h = 1$. However, the limitation of the importance I_h should be acknowledged. Although it allows one to compare hyperparameter importance within a single study, the scores are not directly comparable across different model types because each study's resulting importances are normalized to that model's own variance in objective values. Moreover, some hyperparameters sampled on a log scale occupy a different marginal distribution than those sampled linearly, so the log sampling concentrates trials unevenly across the domain and thus alters the estimated variance contributions compared to linear-scale parameters. Despite these caveats, the normalized importance scores still offer meaningful insight into each hyperparameter's relative influence on optimizing the objective function.

Figure 13 combines the variable importance ranking with a parallel coordinate plot of the random forest trials. In the parallel coordinate plot, each vertical axis represents a hyperparameter, and each line traces a single trial's setting across those axes. The top ten performing trials are highlighted in red, making it easy to identify which hyperparameter combinations yield the best results.

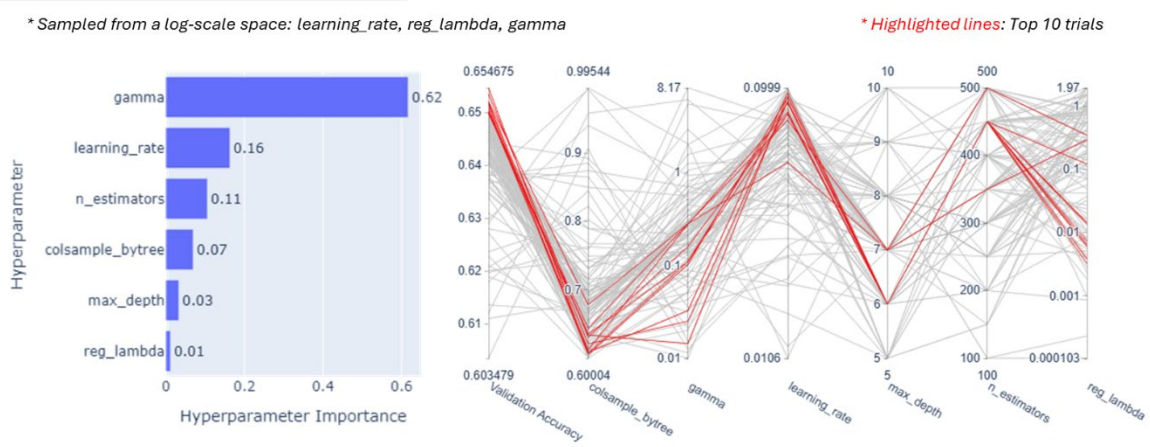


(a) Normalized relative importance (b) Parallel coordinate plot

Figure 13 Random forest's importance and parallel coordinate plots for hyperparameter analysis

The figure indicates that max_depth registers the highest importance score among the hyperparameters. However, because the accuracy of RF with its best feature engineering pipeline shows very little variance (see **Figure 10**), this dominance likely reflects the overall low variability rather than a truly decisive effect. That is, max_depth simply happens to capture the major source of the variance from the occasional bad, outlier trials (around accuracy 0.55 in the figure), and the normalization in **Equation (18)** likely expanded it to the near-one importance. The parallel coordinate plot reveals that all of the top trials use the binary option $max\_features$ set to "log2", that $max\_samples$ cluster around 0.7–0.8, that $min\_samples\_leaf$ favors lower values, and that $min\_samples\_split$ shows no consistent trend. Finally, the number of voting trees ($n\_estimators$) also correlates positively with accuracy.

Figure 14 shows the same pair of plots generated from XGBoost trials. The split-penalty hyperparameter $gamma$ emerges as the most important, while the L2 regularization parameter reg_lambda shows the least impact. In contrast to the RF results, max_depth plays a smaller role here, and the top trials cluster at depths of 6 and 7. The tree configuration hyperparameter $colsample\_bytree$—an equivalent version of RF's $max\_features$ that restricts the number of features used per tree—exhibits a clear "sweet spot" around 0.6–0.7.

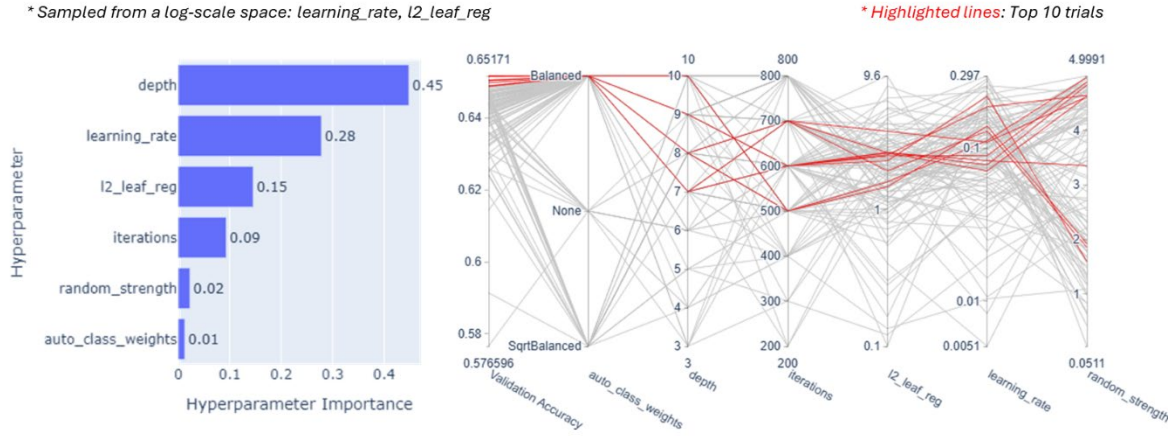


(a) Normalized relative importance (b) Parallel coordinate plot

Figure 14 XGBoost's importance and parallel coordinate plots for hyperparameter analysis

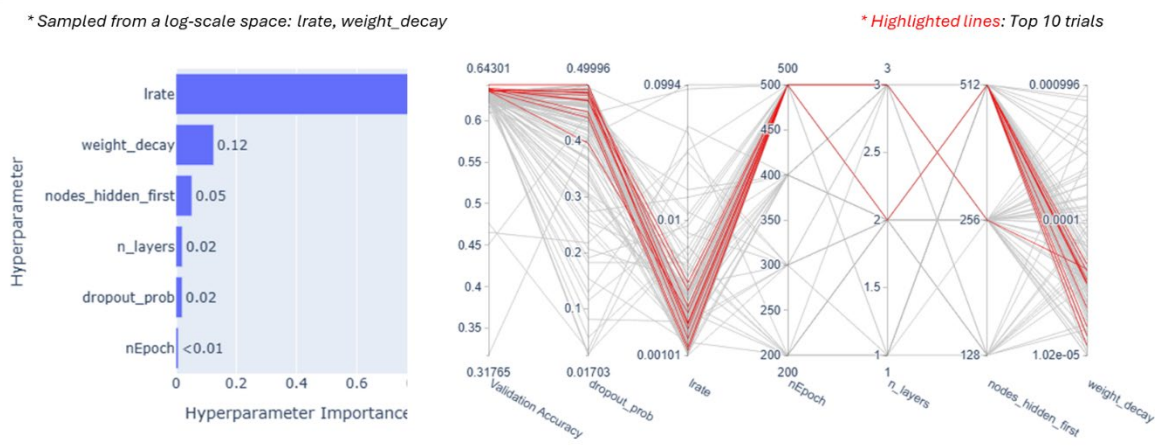
Likewise, **Figure 15** shows CatBoost's hyperparameter importance and coordinate plot. Here, $depth$ emerges as the single most critical parameter—much as in RF but unlike in XG. The tree-growing hyperparameter $random\_strength$, in contrast, shows minimal impact compared to XG's $colsample\_bytree$, even though they serve similar roles in randomly reshaping trees for regularization. Conversely, L2 regularization ($l2\_leaf\_reg$) ranks far higher in CB than in XG. This likely reflects CB's use of fully symmetric trees, which themselves impose a regularizing structure. As a result, there is less benefit from added split-randomness, making L2 penalties the primary driver of variation in accuracy.

Finally, **Figure 16** presents the same results for neural networks. Because the number of layers (n_layers) varies by trial, only the size of the first hidden layer or $nodes\_hidden\_first$, the shared hyperparameter for all trials, is plotted. Among the two regularization techniques—$weight\_decay$ and $dropout\_prob$ —the former was more influential hyperparameter. The top ten trials all use more than one hidden layers, indicating that a "deep" architecture is needed, while a larger number of nodes at the first hidden layer correlates with higher accuracy. The other hyperparameters, at least for this study context and feature engineering pipeline, had a well-established optimal area.



(a) Normalized relative importance (b) Parallel coordinate plot

Figure 15 CatBoost's importance and parallel coordinate plots for hyperparameter analysis



(a) Normalized relative importance (b) Parallel coordinate plot

Figure 16 Neural networks' importance and parallel coordinate plots for hyperparameter analysis

Because SVMs use different hyperparameters depending on the chosen kernel, we omit the importance and the coordinate plots here. Instead, **Figure 17** shows accuracy as a function of the shared regularization parameter $C = 1/\lambda$ (sampled from a log scale), with separate panels for runs with and without class weights to counteract minor-label underrepresentation. Even though our label distribution is fairly balanced (**Table 2**), Optuna consistently favored the weighted setting, producing more high-accuracy trials in that panel. Among kernels, the radial-basis function achieved the highest peak accuracies aligning with an activity pattern recognition with SVM from Allahviranloo and Recker (2013), while the linear kernel delivered the tightest, most stable performance across C and weights.

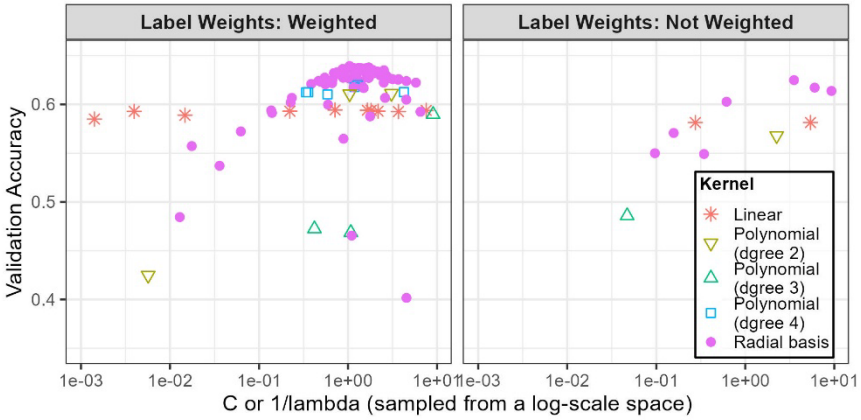


Figure 17 Support vector machine's regularization impact on accuracy by kernel and label weight

Up to this point, **Figures 13–17** have examined the marginal effects of individual hyperparameters. However, many hyperparameters interact—especially those serving similar functions such as regularization. In some cases, pairs of parameters create "sweet-spot" regions of high accuracy; in others, one parameter dominates and diminishes the influence of its counterpart. To explore these dynamics, we examined all possible pairs of two continuous, regularization-related hyperparameters for each model and plot joint effects in scatterplots. The selected hyperparameter pairs visualized in **Figure 18** are $gamma$ and $colsample\_bytree$ for XG illustrated in subplot (a); $l2\_leaf\_reg$ and $random\_strength$ for CB in (b); and $dropout\_prob$ and $weight\_decay$ for NN in (c).

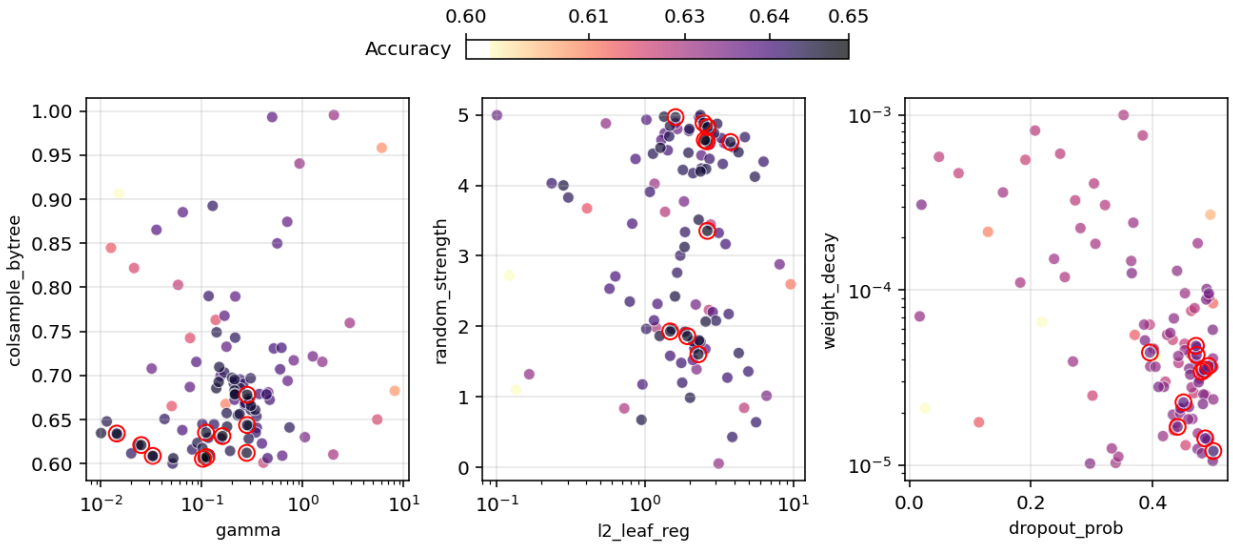


Figure 18 Regularization hyperparameter pairs plots for select models

For XG, both regularization hyperparameters settle in the lower half of their search ranges for the top ten trials, suggesting that stronger feature subsampling (fewer columns per tree) and a moderate split penalty yield the best regularization. In CB, the top trials cluster around a narrow band of L2 regularization strengths, while the randomness added for the tree-growing shows a "U-shaped" effect—small or large values outperform intermediate settings. Finally, in neural networks, the highest-performing trials all employ relatively high values of dropout but low values of weight decay. The results collectively indicate that diligent efforts are needed to identify optimal regularization hyperparameter combinations.

7. Case Study with AFC Data Fusion: Transit Commute Behavior During and After the COVID-19 Pandemic

The best-performing ML model from **Section 6**, an XGBoost classifier configured with its optimal feature engineering pipeline and finely tuned hyperparameters, is employed for our case study. Recall that this case study focuses on trip purpose inference in the AFC domain, utilizing insights gained from OBS surveys to demonstrate the practical application of our data fusion framework. The configuration is detailed below:

- **Dataset**: Raw, unconsolidated *season* and *day of week*
- **Temporal feature engineering**: Cyclical encoding
- **Spatial feature engineering**: Positional encoding (16,10K)
- **Categorical feature engineering**: One-hot encoding
- **Hyperparameter 1**—max_depth: 6
- **Hyperparameter 2**—$learning\_rate$: 0.0881
- **Hyperparameter 3**—$n\_estimators$: 450
- **Hyperparameter 4**—reg_lambda: 0.0039
- **Hyperparameter 5**—$gamma$: 0.1110
- **Hyperparameter 6**—$colsample\_bytree$: 0.6074

To capture moderate shifts in travel behavior while remaining close to the OBS survey period of 2022, we analyze two one-week windows in late March. The first phase, from March 21 to 27, 2021 (labeled "Pandemic"), occurred after Minnesota implemented major transit restrictions under the state's Emergency Executive Order 20-01 on March 13, 2020, and before their lifting under Order 21-23 on May 14, 2021. The second phase, from March 19 to 25, 2023 (labeled "Post-pandemic"), occurred well after the U.S. federal mask mandate for transit vehicles had been lifted (on April 18, 2022). By comparing these two comparable calendar weeks, we can assess how inferred transit travel behavior evolved.

For the trip-chained AFC dataset, 83,494 linked trips in the pandemic week and 148,777 in the post-pandemic week, our "Best" XGBoost model inferred destination place types as trip purposes. **Figure 19** compares the resulting label distributions with those collected from the OBS.

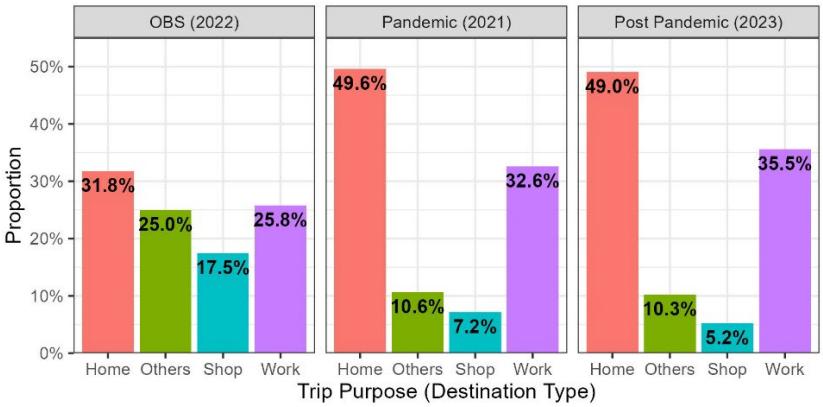


Figure 19 Compositions of trip purposes of on-board survey (recorded) and AFC (machine-learning-predicted) by COVID-19 phases

Unlike OBS, the AFC predictions show a larger share of "Home" trips in both periods. This discrepancy may reflect the model's limitation in non-perfect inference (the F1 Score for "Home" is 0.708) or differences in the sampling method between OBS and AFC. For example, our OBS dataset ensured that underrepresented routes and times were disproportionately represented to facilitate a broader behavioral understanding, as indicated by the "weight factor" recorded in each entry, although it was not used for our inference. In reality, since most transit travelers return home at day's end, a "Home" share near 50% may be more accurate in an exhaustive dataset like AFC.

Comparing the two AFC-inferred study weeks, we observe a slight decline in "Shop" trips and a corresponding rise in "Work" trips post-pandemic, echoing Balbontin et al. (2024), who reported increased shopping and decreased commuting during COVID-19 due to work-from-home policies. As those policies became more restrictive again by 2023 (Tahlyan et al., 2024), the rebound in "Work" trips seen here aligns with the gradual return to in-office routines.

Since the pandemic-era work-from-home policies permanently depressed peak-time conventional commutes, transit agencies cut back express bus services that once ran only during AM and PM peaks. This reduction in supply—especially for express routes—has, in turn, driven further declines in ridership, creating a feedback loop of shrinking service and falling demand, a phenomenon that happened extensively in our study area in the Twin Cities (Sok et al., 2025). Yet, off-peak transit commutes, which essential workers and transit-dependent populations disproportionately take, remain critical to system equity and resilience. The pandemic underscored the importance of understanding these non-traditional commutes (Parker et al., 2021). Additionally, many former peak-period riders have shifted their travel patterns during the pandemic (Wang et al., 2021). For these reasons, our next analysis focuses specifically on off-peak transit commute behaviors—how they have evolved post-pandemic and what they imply for equitable service planning.

Figure 20 illustrates the relative counts of inferred work trips by COVID phase, categorized by destination (the two core downtowns of Minneapolis/St. Paul and elsewhere) and time (AM Peak vs. Off-Peak). We define AM Peak as weekday work-inferred departures (first boarding) that happened between 6:00 AM and 9:00 AM and label all other work-inferred trips—weekday departures outside 6:00–9:00 AM and weekend trips—as Off-Peak. As the chart and its embedded statistics reveal, the share of conventional commutes (AM Peak or to-downtown work-inferred trips) was higher during the pandemic, while the post-pandemic proportions have rebounded toward more typical travel patterns, especially in terms of increased concentration on AM Peak commutes.

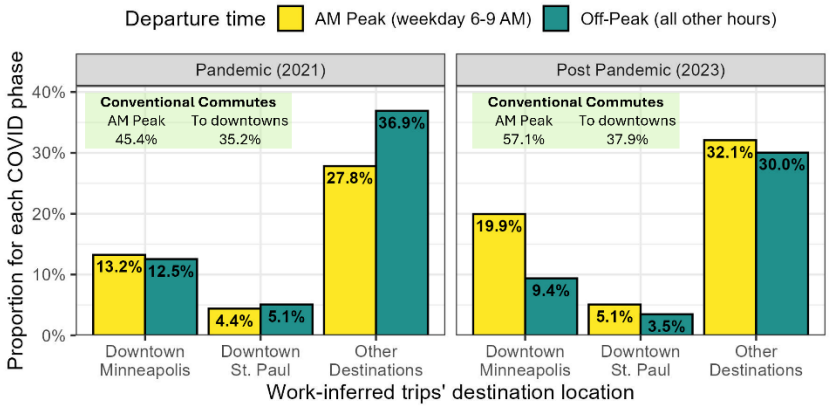


Figure 20 Inferred to-work commute trips' spatiotemporal distributions by the pandemic phase

Figure 21 maps the daily average counts of AFC-recorded destinations inferred as work-inferred trips across the Twin Cities. We created 3 km² hexagonal grids on all neighborhoods served by at least one transit stop and colored each cell by its mean work-trip count (log scale). Similar to the previous figure, the four panels split the data by COVID phase (Pandemic vs. Post-pandemic) and by time period (AM Peak vs. Off Peak).

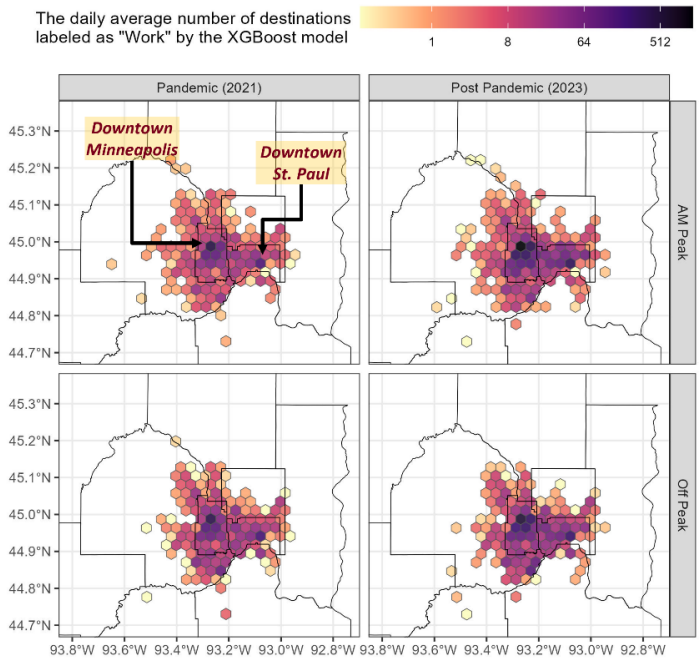


Figure 21 A demonstration of continuous transit commute behavior monitoring enabled by the OBS-AFC data fusion

While spatial patterns appear to remain broadly indistinguishable across panels with too much information, the map demonstrates how planners can leverage AFC-based inference to continuously monitor trip–purpose–specific travel behavior at fine geographic resolution—a primary benefit of the OBS-AFC data fusion via trip purpose inference we propose.

8. Conclusion

This study systematically examined the use of machine learning models to infer trip purposes from transit data, specifically intended for a fusion of actively collected survey and passively collected transit card transaction or AFC datasets. Following the introduction of the adopted models and various feature engineering techniques, we designed an extensive experiment involving five popular machine learning models and traditional/state-of-the-art feature engineering techniques to explore critical dimensions often overlooked by transportation practitioners.

Our two-stage experimental design revealed not only the significant influence but also the joint necessity of rigorous feature engineering and model-specific hyperparameter tuning for achieving accurate predictions. Specifically, the choice of feature engineering pipeline profoundly impacts a model's performance ceiling; the gap in accuracy between the "Best" and "Worst" pipelines spanned up to 6 percentage points for SVM, demonstrating that effective choice of feature engineering is far from trivial. In particular, positional encoding emerged as a particularly effective technique for representing spatial features common in travel behavior analysis, by outperforming other methods, such as zone-based or elliptical transformations, in nearly all tested scenarios. Additionally, advanced categorical encoding techniques, although not universally superior in our moderate-cardinality datasets, demonstrated potential advantages in some models that might bring more benefits with data of higher cardinality.

Furthermore, our results quantify how hyperparameter tuning matters: diligent tuning alone accounted for accuracy variations spanning 5 to 10 percentage points even within the same feature engineering pipeline. This finding establishes that hyperparameter tuning is crucial to achieving optimal predictive accuracy. The fact that a well-tuned model under a sub-optimal feature engineering pipeline can outperform a poorly-tuned model under the best pipeline underscores that neglecting either feature engineering or hyperparameter tuning risks leading to simplistic or inaccurate conclusions regarding model superiority and practitioners' decision making to select the final model to deploy.

Although caution is needed to generalize or transfer the following numeric results from hyperparameter tuning, under our study context, XGBoost achieved the highest test accuracy with optimal configurations, closely followed by CatBoost. These tree-based models consistently demonstrated strong performance and efficient convergence, underscoring their suitability for travel behavior analysis tasks. Conversely, SVM models, despite occasionally exhibiting peaks in label-specific performance, often display undesirable overfitting, as observed in a previous study in the travel mode choice context (Sun et al., 2022). Neural networks, while promising, have highlighted their sensitivity to hyperparameter settings and architectural choices, particularly emphasizing the critical role of regularization in avoiding overfitting.

Applying the optimal XGBoost configuration to AFC datasets from the Twin Cities region provided insights into behavioral shifts across pandemic phases, demonstrating practical implications of the proposed OBS-AFC data fusion framework. Specifically, the analysis revealed changes in commuting patterns during and after the COVID-19 pandemic, indicating a rebound in momentum towards more conventional AM-Peak and downtown-destined commute travel behavior, aligning with existing literature and reinforcing the methodology's relevance.

Despite rigorous experimental design and analyses, our study acknowledges several limitations. First, the initial hyperparameter search spaces, while comprehensive, could not cover all hyperparameters available within each ML model. Expanding the search spaces, particularly through broader ranges and additional parameters, may yield even better performance. Additionally, although Optuna-driven hyperparameter tuning involved extensive trials, its reliance on sampling inherently implies that higher accuracies or different optimal configurations may be achievable. To facilitate reproducibility and encourage further exploration, we provide the complete experiment code (feature engineering and hyperparameter tuning) in an accessible GitHub repository (https://github.com/KBaek0316/BaekKhani_TripPurposeInferenceML).

Second, the OBS and AFC datasets used in this study lacked potentially influential features highlighted in prior research—such as card type and fare type—which have demonstrated utility in similar analyses (Devillaine et al., 2012; Lee and Hickman, 2014). The absence of these variables limited the accuracy by constraining the number of informative features available for inference. Nonetheless, the proposed framework remains valuable for conducting aggregate behavioral analyses. Additionally, the effectiveness of specific feature engineering techniques,

particularly categorical encoding methods, can vary substantially depending on the data characteristics. While moderate cardinality constrained their efficacy in this context, contexts involving higher cardinalities could significantly benefit from these methods.

Third, model applicability may diminish if the AFC data significantly diverges temporally from the survey periods, as ML models "learn" the behavior captured in the survey. Future studies could employ multiple OBS datasets (e.g., 2016 and 2022 surveys for the Twin Cities) to independently train models and subsequently blend their predictions, which would be validated against a curated validation dataset. Related to this point, combining the results from multiple ML models could further enhance prediction accuracy. However, integrating outputs from distinct models (especially SVM with its non-probabilistic outputs) requires careful normalization and calibration, posing methodological challenges yet presenting valuable opportunities for future research.

Finally, investigating the relative importance of spatial versus temporal features and other predictors represents another crucial dimension of ML research (Duan et al., 2022), as well as other diverse external input data-side and/or contextual factors (Wang et al., 2024b). Although omitted from our analysis due to our specific focus on feature engineering and hyperparameter optimization, such studies could yield valuable insights for transit planners and modelers alike.

Building upon our findings and acknowledging these limitations, we propose directions for future research as follows:

- Expand the hyperparameter space and systematically explore additional model-specific parameters to refine performance further.
- Integrate richer datasets that contain critical missing variables, such as fare type and card type, to enhance predictive accuracy at both local and individual scales.
- Apply and validate advanced categorical encoding techniques in contexts with higher data cardinalities, to generalize our insights.
- Investigate ensemble techniques for combining multiple ML models, particularly addressing the normalization and integration challenges posed by heterogeneous model outputs.
- Conduct comprehensive variable importance analyses to identify the critical determinants of trip purposes, thereby enriching model interpretability and practical utility.

Ultimately, our study provides a robust framework and practical guidelines for transportation researchers and practitioners seeking to leverage machine learning effectively. By explicitly addressing often-overlooked methodological considerations—feature engineering and hyperparameter tuning—this research makes a meaningful contribution to advancing data-driven insights into transit travel behaviors.

Acknowledgment

This research was conducted at the University of Minnesota Transit Lab (http://umntransit.weebly.com/), supported by the following, but not limited to, projects at the time of conducting the research:

- National Science Foundation, awards CMMI-1831140 and TI- 2428029
- Minnesota Department of Transportation Contract No. 1036342 Work Order No. 08, 69, 73, 95, 119, and 121
- Metropolitan Council SG-2021-021
- Hennepin County Work Order No. UM0921
- Empowering Small Minnesota Communities (ESMC) program, Center for Transportation Studies, University of Minnesota

The authors thank Metro Transit for sharing AFC data. All the findings reflect the results obtained from the analysis conducted by the authors and do not represent the sponsors' opinions. Any limitation of the study remains the sole responsibility of the authors.

Declaration of Generative AI and AI-assisted Technologies in the Writing Process

During the preparation of this work the authors used OpenAI GPT-4o in order to improve the readability and language of the manuscript. After using this tool, the authors reviewed and edited the content as needed and take full responsibility for the content of the published article.

Appendix A. Zones Used as an Option for the Spatial Data Feature Engineering for Encoding Origin and Destination Locations

The zones used in **Section 4.2.4** for representing origin and destination locations were constructed from two existing geographical datasets, with an additional adjustment step:

- The Transit Market Areas (see **Figure 22**) defined by the regional government body (Metropolitan Council, 2015).
- The boundaries of the cities of Minneapolis, St. Paul, and the seven suburban counties that belongs to the Twin Cities Metropolitan Area: Hennepin (excluding Minneapolis), Ramsey (excluding St. Paul), Anoka, Carver, Dakota, Scott, and Washington Counties.
- Four specialized, independent zones added: Downtown Minneapolis, Downtown St. Paul, the University of Minnesota campuses, and the combined areas of Minneapolis–St. Paul (MSP) International Airport and the Mall of America shopping district.

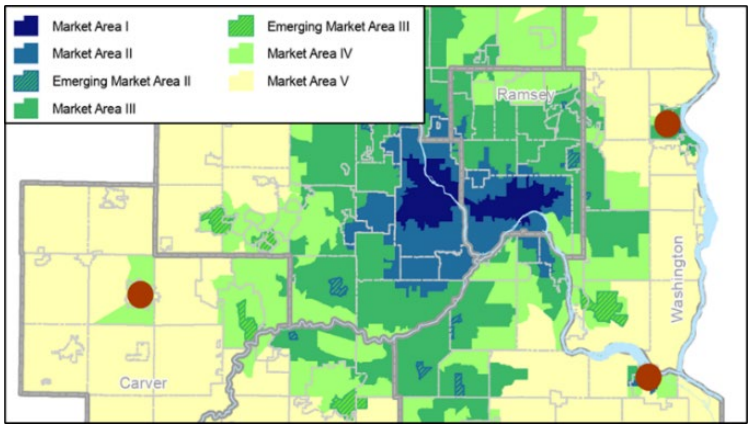


Figure 22 Transit Market Areas (Metropolitan Council, 2015)

While a straightforward combination of the 7 Transit Market Area categories over the 9 geographic areas (two cities plus seven suburban counties), plus the 4 specialized zones, would theoretically yield 67 distinct zones, we applied the following refinements to consolidate redundant categories. First, we did not intersect Market Areas IV and V with city or county boundaries and further consolidated those two categories, as these areas, with scarce boarding/alighting counts, are not proximate to Minneapolis or St. Paul and do not represent standalone suburban zones. This consolidated zone also encompasses locations outside the seven-county Twin Cities area, primarily representing long-distance commutes with limited commuter transit services. Second, for example, Market Area I exclusively overlaps with the core city boundaries of Minneapolis and St. Paul; thus, no additional intersections with suburban counties with Market Area I exist, resulting in no corresponding zones being created. This absence applies to other theoretical instances that do not have geographical intersections.

The borders of the 31 resulting zones (each of them is a multi-part polygon) are shown in **Figure 23**. **Table 8** summarizes the number of origin and destination locations aggregated by zone based on our OBS (after applying the SBSP transit assignment) observations. Notably, no trips originated from zones 26 and 30, or destined to zones 15 and 25. Therefore, while 31 zones are defined, only 29 one-hot encoded features are created for both origins and destinations in the discussion of **Figure 7 (Section 6.1)**, reflecting the absence of observations for specific zones.

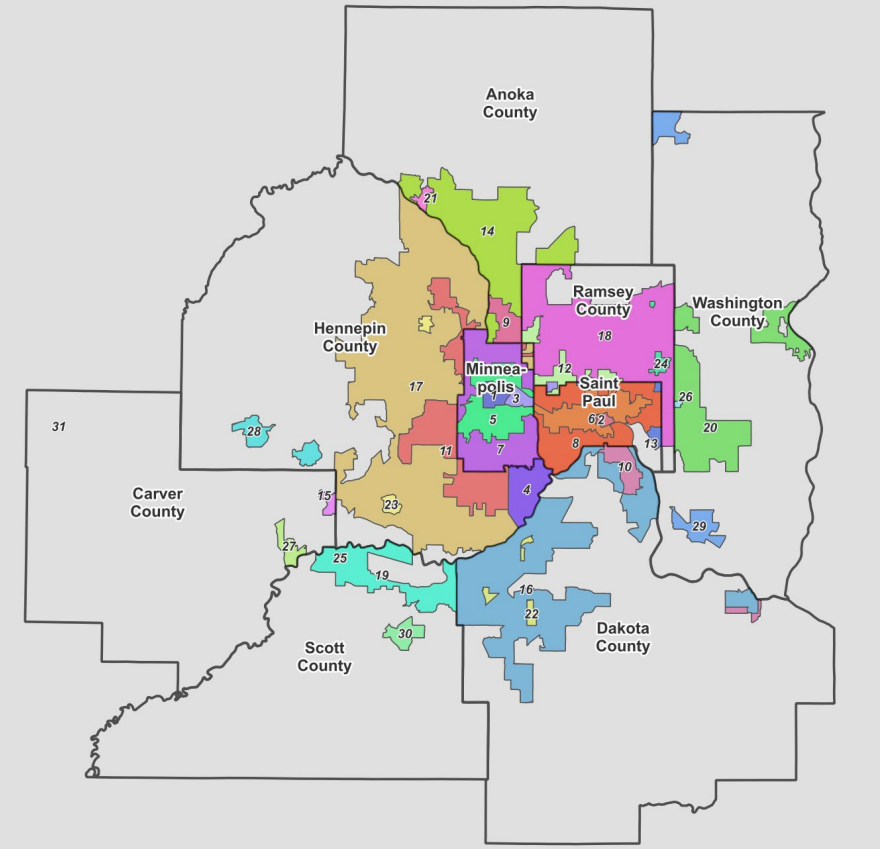


Figure 23 Borders of 31 zones; colors differentiate each zone, and labels display the assigned integer zone numbers (1-31)

Table 8 Zone description and OBS statistics in terms of trips originated from and destined to each zone

ID	Zone Name	# of trips originated	# of trips destined	ID	Zone Name	# of trips originated	# of trips destined
1	Downtown Minneapolis	3,055	3,436	17	Hennepin County × Market Area III	594	553
2	Downtown St. Paul	1,147	1,215	18	Ramsey County × Market Area III	525	504
3	University of Minnesota Campuses	2,241	2,318	19	Scott County × Market Area III	14	18
4	MSP Airport and Mall of America	1,240	1,636	20	Washington County × Market Area III	50	44
5	Minneapolis × Market Area I	3,465	2,966	21	Anoka County × Emerging Area II	20	22
6	Saint Paul × Market Area I	2,094	2,255	22	Dakota County × Emerging Area II	68	27
7	Minneapolis × Market Area II	1,681	1,473	23	Hennepin County × Emerging Area II	25	35
8	Saint Paul × Market Area II	1,375	1,270	24	Ramsey County × Emerging Area II	18	17
9	Anoka County × Market Area II	214	186	25	Scott County × Emerging Area II	2	0
10	Dakota County × Market Area II	99	100	26	Washington County × Emerging Area II	0	1
11	Hennepin County × Market Area II	1,434	1,345	27	Carver County × Emerging Area III	3	2
12	Ramsey County × Market Area II	196	189	28	Hennepin County × Emerging Area III	6	2
13	Saint Paul × Market Area III	15	20	29	Washington County × Emerging Area III	9	4
14	Anoka County × Market Area III	208	193	30	Scott County × Emerging Area III	0	1
15	Carver County × Market Area III	5	0	31	Undefined Area	86	85
16	Dakota County × Market Area III	344	316	**Total linked trips identified from OBS-SBSP:**			**20,233**

Appendix B. Hyperparameters Range and Drawing Methods Used in the Optuna-based Experiments

Table 9 Hyperparameters Range and Drawing Methods Used in the Optuna-based Experiments

	Hyperparameter	Primary role (simplified)	Range of sampling	Space drawn from
Random Forest (RF)				
	$n\_estimator$	Number of trees that vote	[200, 1000] with step size=50	Integer
	max_depth	Regularization with added restriction	[5, 50] with step size=5	Integer
	$min\_samples\_split$	Regularization with added restriction	[2, 10]	Integer
	$min\_samples\_leaf$	Regularization with added restriction	[1, 10]	Integer
	$max\_features$	Controls tree dimension	[“sqrt”, “log2”]	Categorical
	$max\_samples$	Controls tree dimension	[0.6, 1.0]	Integer
Extreme Gradient Boost or “XGBoost” (XG)				
	$n\_estimators$	Number of iterations	[100, 500] with step size=50	Integer
	max_depth	Regularization with added restriction	[5, 10]	Integer
	$colsample\_bytree$	Controls tree dimension	[0.6, 1.0]	Decimal
	$learning\_rate$	Controls contribution of each iteration	[0.01, 0.1]	Decimal (log)
	reg_lambda	Regularization with added restriction	[0.0001, 2]	Decimal (log)
	$gamma$	Regularization with added restriction	[0.01, 10]	Decimal (log)
CatBoost (CB)				
	$iterations$	Number of iterations	[200, 800] with step size=100	Integer
	$learning\_rate$	Controls contribution of each iteration	[0.005, 0.3]	Decimal (log)
	$depth$	Regularization with added restriction	[3, 10]	Integer
	$l2\_leaf\_reg$	Regularization with added restriction	[0.1, 10]	Decimal (log)
	$random\_strength$	Regularization with added randomness	[0, 5]	Decimal
	$auto\_class\_weights$	Addresses class-membership bias	[None, “Balanced”, “SqrtBalanced”]	Categorical
Neural Networks (NN): feed-forward, fully connected variation				
	n_layers	Model architecture	[1, 3]	Integer
	$n\_units\_layer\_\{s\}$[a]	Model architecture	[“128”, “256”, “512”]	Categorical
	$nEpoch$	Number of iterations	[200, 500] with step size=100	Integer
	$lrate$	Controls contribution of each iteration	[0.001, 0.1]	Decimal (log)
	$dropout\_prob$	Regularization with added randomness	[0, 0.5]	Decimal
	$weight\_decay$	Regularization with added restriction	[0.00001, 0.001]	Decimal (log)
Support Vector Machine (SVM)				
	$kernel$	Model architecture	[“linear”, “poly”, “rbf”]	Categorical
	C	Regularization with added restriction	[0.001, 10]	Decimal (log)
	$gamma$[b]	Kernel shape	[0.001, 1]	Decimal (log)
	$coef0$[c]	Kernel shape	[-0.5, 0.5]	Decimal
	$degree$[c]	Kernel shape	[2, 4]	Integer
	$class\_weight$	Addresses class-membership bias	[None, “Balanced”]	Categorical

[a] One to three $n\_units\_layer\_\{s\}$ (i.e., $s = \{1,2,3\}$) are defined per each model based on the number of hidden layers sampled via n_layers

[b] Not applies to “linear” kernel

[c] Only applies to “poly” kernel

All the other hyperparameters not listed in the table were not explicitly fed upon the model building (default constant/variable values from each package used)

References

Aghaabbasi, M., Shekari, Z.A., Shah, M.Z., Olakunle, O., Armaghani, D.J., Moeinaddini, M., 2020. Predicting the use frequency of ride-sourcing by off-campus university students through random forest and Bayesian network techniques. Transp Res Part A Policy Pract 136, 262–281. https://doi.org/10.1016/J.TRA.2020.04.013

Akiba, T., Sano, S., Yanase, T., Ohta, T., Koyama, M., 2019. Optuna: A Next-generation Hyperparameter Optimization Framework. ACM, New York, NY, USA, pp. 2623–2631. https://doi.org/10.1145/3292500.3330701

Allahviranloo, M., Recker, W., 2013. Daily activity pattern recognition by using support vector machines with multiple classes. Transportation Research Part B: Methodological 58, 16–43. https://doi.org/10.1016/j.trb.2013.09.008

Alsger, A., Tavassoli, A., Mesbah, M., Ferreira, L., Hickman, M., 2018. Public transport trip purpose inference using smart card fare data. Transp Res Part C Emerg Technol 87, 123–137. https://doi.org/10.1016/j.trc.2017.12.016

Aminpour, N., Saidi, S., 2025. Unveiling mobility patterns beyond home/work activities: A topic modeling approach using transit smart card and land-use data. Travel Behav Soc 38. https://doi.org/10.1016/j.tbs.2024.100905

Arnold, C., Biedebach, L., Küpfer, A., Neunhoeffer, M., 2024. The role of hyperparameters in machine learning models and how to tune them. Political Sci Res Methods 1–8. https://doi.org/10.1017/psrm.2023.61

Baek, K., Khani, A., 2024. Transfer Behavior and Off-Peak Commutes. Center for Transportation Studies, University of Minnesota, Minneapolis. https://doi.org/https://hdl.handle.net/11299/267839

Baek, K., Lee, H., Chung, J.-H., Kim, J., 2021. Electric scooter sharing: How do people value it as a last-mile transportation mode? Transp Res D Transp Environ 90, 102642. https://doi.org/10.1016/j.trd.2020.102642

Balbontin, C., Hensher, D.A., Beck, M.J., 2024. Relationship between commuting and non-commuting travel activity under the growing incidence of working from home and people’s attitudes towards COVID-19. Transportation (Amst) 2225–2251. https://doi.org/10.1007/s11116-023-10403-2

Barrero, J.M., Bloom, N., Davis, S.J., 2021. Why Working from Home Will Stick. NBER Working Paper Series. https://doi.org/10.3386/w28731

Baumgarte, F., Keller, R., Röhrich, F., Valett, L., Zinsbacher, D., 2022. Revealing influences on carsharing users’ trip distance in small urban areas. Transp Res D Transp Environ 105, 103252. https://doi.org/10.1016/J.TRD.2022.103252

Bergstra, J., Komer, B., Eliasmith, C., Yamins, D., Cox, D.D., 2015. Hyperopt: a Python library for model selection and hyperparameter optimization. Comput Sci Discov 8, 014008. https://doi.org/10.1088/1749-4699/8/1/014008
Berrebi, S.J., Lind, E., Brakewood, C., Erhardt, G., Watkins, K., 2022. Investigating the Ridership Impact of New Light-Rail Transit and Arterial Bus Rapid Transit Lines in the Twin Cities. Transp Res Rec 2676. https://doi.org/10.1177/03611981221078283
Bishop, C.M., 2006. Pattern recognition and machine learning, Information science and statistics. Springer, New York.
Boeing, G., 2017. OSMnx: New methods for acquiring, constructing, analyzing, and visualizing complex street networks. Comput Environ Urban Syst 65, 126–139. https://doi.org/10.1016/j.compenvurbsys.2017.05.004
Bohte, W., Maat, K., 2009. Deriving and validating trip purposes and travel modes for multi-day GPS-based travel surveys: A large-scale application in the Netherlands. Transp Res Part C Emerg Technol 17, 285–297. https://doi.org/10.1016/J.TRC.2008.11.004
Breiman, L., 2001. Random forests. Mach Learn 45, 5–32. https://doi.org/10.1023/A:1010933404324
Breiman, L., 1984. Classification and regression trees, First. ed. Routledge, London.
Chen, T., Guestrin, C., 2016. XGBoost: A Scalable Tree Boosting System, in: International Conference on Knowledge Discovery and Data Mining. ACM, pp. 785–794. https://doi.org/10.1145/2939672.2939785
Cui, Y., Meng, C., He, Q., Gao, J., 2018. Forecasting current and next trip purpose with social media data and Google Places. Transp Res Part C Emerg Technol 97, 159–174. https://doi.org/10.1016/J.TRC.2018.10.017
Devillaine, F., Munizaga, M., Trépanier, M., 2012. Detection of activities of public transport users by analyzing smart card data. Transp Res Rec 48–55. https://doi.org/10.3141/2276-06
Dorogush, A.V., Ershov, V., Gulin, A., 2018. CatBoost: gradient boosting with categorical features support. https://doi.org/https://doi.org/10.48550/arXiv.1810.11363
Duan, S.X., Tay, R., Molla, A., Deng, H., 2022. Predicting Mobility as a Service (MaaS) use for different trip categories: An artificial neural network analysis. Transp Res Part A Policy Pract 166, 135–149. https://doi.org/10.1016/J.TRA.2022.10.014
El Naqa, I., Murphy, M.J., 2015. What Is Machine Learning?, in: Machine Learning in Radiation Oncology: Theory and Applications. Springer International Publishing, Cham, pp. 3–11. https://doi.org/10.1007/978-3-319-18305-3_1
Ermagun, A., Fan, Y., Wolfson, J., Adomavicius, G., Das, K., 2017. Real-time trip purpose prediction using online location-based search and discovery services. Transp Res Part C Emerg Technol 77, 96–112. https://doi.org/10.1016/J.TRC.2017.01.020
Faroqi, H., Mesbah, M., 2021. Inferring trip purpose by clustering sequences of smart card records. Transp Res Part C Emerg Technol 127, 103131. https://doi.org/10.1016/J.TRC.2021.103131
Foumani, N.M., Tan, C.W., Webb, G.I., Salehi, M., 2024. Improving position encoding of transformers for multivariate time series classification. Data Min Knowl Discov 38, 22–48. https://doi.org/10.1007/s10618-023-00948-2
Gao, L., Huang, H., Ye, J., Wang, D., 2024. Activity type detection of mobile phone data based on self-training: Application of the teacher–student cycling model. Transp Res Part C Emerg Technol 161, 104550. https://doi.org/10.1016/J.TRC.2024.104550
Grądzki, P., Wójcik, P., 2024. Is attention all you need for intraday Forex trading? Expert Syst 41. https://doi.org/10.1111/exsy.13317
Hastie, T., Tibshirani, R., Friedman, J., 2009. The elements of statistical learning : data mining, inference, and prediction, 2nd ed, Springer series in statistics. Springer, New York.
Hossain, S., Habib, K.N., 2021. Inferring the Purposes of using Ride-Hailing Services through Data Fusion of Trip Trajectories, Secondary Travel Surveys, and Land Use Data. Transp Res Rec 2675, 558–573. https://doi.org/10.1177/03611981211003593
Hutter, F., Hoos, H., Leyton-Brown, K., 2014. An Efficient Approach for Assessing Hyperparameter Importance, in: Proceedings of the 31st International Conference on Machine Learning. Beijing, China, pp. 754–762.
International Business Machines, 1961. Reference manual, 709/7090 FORTRAN programming system. International Business Machines Corporation, White Plains, NY.
Khani, A., Hickman, M., Noh, H., 2015. Trip-Based Path Algorithms Using the Transit Network Hierarchy. Netw Spat Econ 15, 635–653. https://doi.org/10.1007/s11067-014-9249-3
Kim, E.J., Kim, D.K., Sohn, K., 2022. Imputing qualitative attributes for trip chains extracted from smart card data using a conditional generative adversarial network. Transp Res Part C Emerg Technol 137, 103616. https://doi.org/10.1016/J.TRC.2022.103616
Kim, E.J., Kim, Y., Kim, D.K., 2021. Interpretable machine-learning models for estimating trip purpose in smart card data. Proceedings of the Institution of Civil Engineers: Municipal Engineer 174, 108–117. https://doi.org/10.1680/jmuen.20.00003
Kingma, D.P., Ba, J., 2015. Adam: A Method for Stochastic Optimization, in: 3rd International Conference on Learning Representations (ICLR 2015). San Diego, CA.
Klemmer, K., Safir, N., Neill, D.B., 2023. Positional Encoder Graph Neural Networks for Geographic Data. pp. 1379–1389.
Kong, H., Jin, S.T., Sui, D.Z., 2020. Deciphering the relationship between bikesharing and public transit: Modal substitution, integration, and complementation. Transp Res D Transp Environ 85, 102392. https://doi.org/https://doi.org/10.1016/j.trd.2020.102392
Koushik, A.N., Manoj, M., Nezamuddin, N., 2020. Machine learning applications in activity-travel behaviour research: a review. Transp Rev 40, 288–311. https://doi.org/10.1080/01441647.2019.1704307
Kumar, P., Khani, A., He, Q., 2018. A robust method for estimating transit passenger trajectories using automated data. Transp Res Part C Emerg Technol 95, 731–747. https://doi.org/10.1016/J.TRC.2018.08.006
Lee, S.G., Hickman, M., 2014. Trip purpose inference using automated fare collection data. Public Transport 6, 1–20. https://doi.org/10.1007/S12469-013-0077-5/FIGURES/6
Li, Y., Si, S., Li, G., Hsieh, C.J., Bengio, S., 2021. Learnable Fourier Features for Multi-Dimensional Spatial Positional Encoding. pp. 15816–15829.
Loshchilov, I., Hutter, F., 2019. Decoupled Weight Decay Regularization, in: 7th International Conference on Learning Representations (ICLR 2019). OpenReview.net, New Orleans, LA.
Lu, Y., Zhang, L., 2015. Imputing trip purposes for long-distance travel. Transportation (Amst) 42, 581–595. https://doi.org/10.1007/s11116-015-9595-0
Lundberg, S., Lee, S.-I., 2017. A Unified Approach to Interpreting Model Predictions, in: 31st Conference on Neural Information Processing Systems (NIPS). Long Beach. https://doi.org/10.48550/arxiv.1705.07874
Luo, D., Bonnetain, L., Cats, O., van Lint, H., 2018. Constructing spatiotemporal load profiles of transit vehicles with multiple data sources. Transportation Research Record: Journal of the Transportation Research Board In press. https://doi.org/10.1177/0361198118781166
Luo, X., Cottam, A., Wu, Y.-J., Jiang, Y., 2021. Hybrid-Data Approach for Estimating Trip Purposes. Transp Res Rec 2675, 545–553. https://doi.org/10.1177/03611981211018474
Mai, G., Janowicz, K., Yan, B., Zhu, R., Cai, L., Lao, N., 2020. Multi-scale representation learning for spatial feature distributions using grid cells.
Memarian, B., Jeong, "David" Hyung Seok, Uhm, D., 2012. Effects of survey techniques on on-board survey performance. Transp Policy (Oxf) 21, 52–62. https://doi.org/10.1016/j.tranpol.2012.01.006
Metropolitan Council, 2023. Transit On-Board Survey [WWW Document]. URL https://metrocouncil.org/Transportation/Performance/Travel-Behavior-Inventory/Transit-On-Board-Survey.aspx (accessed 2.29.24).
Metropolitan Council, 2015. 2040 Transportation Policy Plan. St. Paul, MN.
Miah, M.M., Hyun, K.K., Mattingly, S.P., Khan, H., 2022. Estimation of daily bicycle traffic using machine and deep learning techniques. Transportation (Amst). https://doi.org/10.1007/s11116-022-10290-z
Montini, L., Rieser-Schüssler, N., Horni, A., Axhausen, K.W., 2014. Trip Purpose Identification from GPS Tracks. Transp Res Rec 2405, 16–23. https://doi.org/10.3141/2405-03
Nassir, N., Hickman, M., Ma, Z.L., 2015. Activity detection and transfer identification for public transit fare card data. Transportation (Amst) 42, 683–705. https://doi.org/10.1007/s11116-015-9601-6

Nguyen, S., Pallottino, S., 1988. Equilibrium traffic assignment for large scale transit networks. Eur J Oper Res 37, 176–186. https://doi.org/10.1016/0377-2217(88)90327-X
Ortúzar, J. de D., Willumsen, L.G., 2011. Modelling transport, 4th ed. John Wiley & Sons.
O'Sullivan, A., 2019. Urban economics, 9th ed. McGraw-Hill, Boston.
Paredes, M., Hemberg, E., O'Reilly, U.-M., Zegras, C., 2017. Machine learning or discrete choice models for car ownership demand estimation and prediction?, in: MTITS. IEEE, pp. 780–785. https://doi.org/10.1109/MTITS.2017.8005618
Pargent, F., Pfisterer, F., Thomas, J., Bischl, B., 2022. Regularized target encoding outperforms traditional methods in supervised machine learning with high cardinality features. Comput Stat 37, 2671–2692. https://doi.org/10.1007/s00180-022-01207-6
Parker, M.E.G., Li, M., Bouzaghrane, M.A., Obeid, H., Hayes, D., Frick, K.T., Rodríguez, D.A., Sengupta, R., Walker, J., Chatman, D.G., 2021. Public transit use in the United States in the era of COVID-19: Transit riders' travel behavior in the COVID-19 impact and recovery period. Transp Policy (Oxf) 111, 53–62. https://doi.org/10.1016/j.tranpol.2021.07.005
Paszke, A., Gross, S., Massa, F., Lerer, A., Bradbury, J., Chanan, G., Killeen, T., Lin, Z., Gimelshein, N., Antiga, L., Desmaison, A., Köpf, A., Yang, E., DeVito, Z., Raison, M., Tejani, A., Chilamkurthy, S., Steiner, B., Fang, L., Bai, J., Chintala, S., 2019. PyTorch: An Imperative Style, High-Performance Deep Learning Library.
Pedregosa, F., Varoquaux, G., Gramfort, A., Michel, V., Thirion, B., Grisel, O., Blondel, M., Prettenhofer, P., Weiss, R., Dubourg, V., Vanderplas, J., Passos, A., Cournapeau, D., Brucher, M., Perrot, M., Duchesnay, É., 2011. Scikit-learn: Machine Learning in Python. Journal of Machine Learning Research 12, 2825–2830.
Pérez, F.A.A., Ortiz, G.E.R., Muñiz, E.R., Sacarello, F.J.O., Kang, J.E., Rodriguez-Roman, D., 2020. Predicting Trip Cancellations and No-Shows in Paratransit Operations. Transp Res Rec 2674, 774–784. https://doi.org/10.1177/0361198120924661
Prokhorenkova, L., Gusev, G., Vorobev, A., Dorogush, A.V., Gulin, A., 2018. CatBoost: unbiased boosting with categorical features, in: Advances in Neural Information Processing Systems. Curran Associates, Inc.
Rahate, A., Walambe, R., Ramanna, S., Kotecha, K., 2022. Multimodal Co-learning: Challenges, applications with datasets, recent advances and future directions. Information Fusion 81, 203–239. https://doi.org/10.1016/J.INFFUS.2021.12.003
RAPIDS Development Team, 2023. RAPIDS: Libraries for End to End GPU Data Science.
Richardson, A.J., Ampt, E.S., Meyburg, A.H., 1995. Survey methods for transport planning. Eucalyptus Press, Parkville, Vic.
Robinson, S., Narayanan, B., Toh, N., Pereira, F., 2014. Methods for pre-processing smartcard data to improve data quality. Transp Res Part C Emerg Technol 49, 43–58. https://doi.org/10.1016/j.trc.2014.10.006
Sana, B., Castiglione, J., Cooper, D., Tischler, D., 2018. Using Google's Passive Data and Machine Learning for Origin-Destination Demand Estimation. Transp Res Rec 2672, 73–82. https://doi.org/10.1177/0361198118798298
Sari Aslam, N., Ibrahim, M.R., Cheng, T., Chen, H., Zhang, Y., 2021. ActivityNET: Neural networks to predict public transport trip purposes from individual smart card data and POIs. Geo-spatial Information Science 24, 711–721. https://doi.org/10.1080/10095020.2021.1985943
Schaller, B., 2005. TCRP Synthesis 63: On-Board and Intercept Transit Survey Techniques. Washington, D.C.
Schölkopf, B., Smola, A.J., 2002. Learning with kernels : support vector machines, regularization, optimization, and beyond, Adaptive computation and machine learning. MIT Press, Cambridge, Mass.
Shalit, N., Fire, M., Ben-Elia, E., 2022. A supervised machine learning model for imputing missing boarding stops in smart card data. Public Transport. https://doi.org/10.1007/s12469-022-00309-0
Shwartz-Ziv, R., Armon, A., 2022. Tabular data: Deep learning is not all you need. Information Fusion 81, 84–90. https://doi.org/10.1016/J.INFFUS.2021.11.011
Sok, S., Baek, K., Khani, A., 2025. The Changes in Transit Route Choice Behavior Between Pre- and Post-Pandemic, in: Transportation Research Board 104th Annual Meeting. Washington, DC.
Stopher, P., Meyburg, A., 1975. Urban transportation modelling and planning.
Sun, Y., Dong, Y., D. Waygood, E.O., Naseri, H., Jiang, Y., Chen, Y., 2022. Machine-Learning Approaches to Identify Travel Modes Using Smartphone-Assisted Survey and Map Application Programming Interface. Transp Res Rec 2677, 385–400. https://doi.org/10.1177/03611981221106483
Tahlyan, D., Mahmassani, H., Stathopoulos, A., Said, M., Shaheen, S., Walker, J., Johnson, B., 2024. In-person, hybrid or remote? Employers' perspectives on the future of work post-pandemic. Transp Res Part A Policy Pract 190, 104273. https://doi.org/10.1016/j.tra.2024.104273
Tomhave, B.J., Khani, A., 2022. Refined choice set generation and the investigation of multi-criteria transit route choice behavior. Transp Res Part A Policy Pract 155, 484–500. https://doi.org/10.1016/j.tra.2021.11.005
Tong, C.O., Richardson, A.J., 1984. A computer model for finding the time-dependent minimum path in a transit system with fixed schedules. J Adv Transp 18, 145–161. https://doi.org/https://doi.org/10.1002/atr.5670180205
Train, Kenneth., 2009. Discrete Choice Methods with Simulation. Cambridge University Press.
Trépanier, M., Tranchant, N., Chapleau, R., 2007. Individual Trip Destination Estimation in a Transit Smart Card Automated Fare Collection System. J Intell Transp Syst 11, 1–14. https://doi.org/10.1080/15472450601122256
Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A.N., Kaiser, Ł., Polosukhin, I., 2017. Attention is all you need, in: 31st Conference on Neural Information Processing Systems (NIPS 2017). Long Beach, CA, pp. 5999–6009. https://doi.org/https://doi.org/10.48550/arXiv.1706.03762
Wang, D., Tayarani, M., Yueshuai He, B., Gao, J., Chow, J.Y.J., Oliver Gao, H., Ozbay, K., 2021. Mobility in post-pandemic economic reopening under social distancing guidelines: Congestion, emissions, and contact exposure in public transit. Transp Res Part A Policy Pract 153, 151–170. https://doi.org/10.1016/J.TRA.2021.09.005
Wang, F., Ross, C.L., 2018. Machine Learning Travel Mode Choices: Comparing the Performance of an Extreme Gradient Boosting Model with a Multinomial Logit Model. Transp Res Rec 2672, 35–45. https://doi.org/10.1177/0361198118773556
Wang, J., Cui, M., Wang, H., Yang, H., Guo, X., Liu, X., Fu, X., 2024a. Trip Purpose Inference and Spatio-Temporal Characterization Based on Anonymized Trip Data: Empirical Study from Dockless Shared Bicycle Dataset in Xi'an, China. Transp Res Rec 2678, 251–266. https://doi.org/https://doi-org.ezp1.lib.umn.edu/10.1177/03611981231225654
Wang, L., Ma, W., Fan, Y., Zuo, Z., 2019. Trip chain extraction using smartphone-collected trajectory data. Transportmetrica B: Transport Dynamics 7, 255–274. https://doi.org/10.1080/21680566.2017.1386599
Wang, S., Mo, B., Zheng, Y., Hess, S., Zhao, J., 2024b. Comparing hundreds of machine learning and discrete choice models for travel demand modeling: An empirical benchmark. Transportation Research Part B: Methodological 190, 103061. https://doi.org/10.1016/J.TRB.2024.103061
Xiao, G., Juan, Z., Zhang, C., 2016. Detecting trip purposes from smartphone-based travel surveys with artificial neural networks and particle swarm optimization. Transp Res Part C Emerg Technol 71, 447–463. https://doi.org/10.1016/J.TRC.2016.08.008
Xu, R., Wang, X., Chen, K., Zhou, B., Loy, C.C., 2021. Positional Encoding as Spatial Inductive Bias in GANs, in: CVPR. IEEE, pp. 13564–13573. https://doi.org/10.1109/CVPR46437.2021.01336
Yin, G., Huang, Z., Fu, C., Ren, S., Bao, Y., Ma, X., 2024. Examining active travel behavior through explainable machine learning: Insights from Beijing, China. Transp Res D Transp Environ 127, 104038. https://doi.org/10.1016/J.TRD.2023.104038
Zailani, S., Iranmanesh, M., Masron, T.A., Chan, T.H., 2016. Is the intention to use public transport for different travel purposes determined by different factors? Transp Res D Transp Environ 49, 18–24. https://doi.org/10.1016/J.TRD.2016.08.038
Zhu, Y., 2020. Estimating the activity types of transit travelers using smart card transaction data: a case study of Singapore. Transportation (Amst) 47, 2703–2730. https://doi.org/10.1007/s11116-018-9881-8
Zou, Q., Yao, X., Zhao, P., Wei, H., Ren, H., 2018. Detecting home location and trip purposes for cardholders by mining smart card transaction data in Beijing subway. Transportation (Amst) 45, 919–944. https://doi.org/10.1007/s11116-016-9756-9