Exploring Ensemble Model for Delivery Predictions

Advit Sharma
Department of Computer Science
Golisano College of Computing and Information Sciences
Rochester Institute of Technology
Rochester, NY 14586
as3272@cs.rit.edu

***Abstract*—In the dynamic landscape of delivery services, predicting delivery duration stands as a pivotal factor. Leveraging historical data to forecast delivery time not only streamlines logistical operations but also enhances customer satisfaction. We show in this paper the applicability and usefulness of travel time i.e. delivery time prediction with a type of ensemble model. We investigate several methods such as tree models and tree based ensembles such as XGBoost and LightGBM. Results reveal that travel time prediction can help mitigate high delays. We show that the ensemlble model of a boosting tree model and Nearest Neighbor classifier have a higher performance in terms of accuracy.**

***Index Terms*—Last-mile Delivery Prediction; Origin Destination Approach; Ensemble model**

I. Introduction

In light of the escalating demands in volume and activities within the realm of e-commerce, postal services are confronted with escalating challenges in sustaining both efficiency and customer loyalty. At the same time, we need to find ways to reduce the environmental impact of all the extra transportation. Studying traffic patterns can give us important information about the problem. For example, we can look at how long it takes to travel and where things are going, to figure out the best routes and schedules.

Guessing when a delivery will arrive is really important for making things work smoothly and letting customers know when to expect their packages. If we're good at predicting delivery times, we can plan the routes and assignments better. Plus, telling customers when to expect their deliveries can help avoid wasted trips, like when someone isn't home. This saves both the customer and the delivery company time and effort.

This way of doing things can also help reduce traffic jams in cities just before deliveries, which in turn cuts down on long-term costs and helps fight climate change. And, it leads to happier customers who trust the postal service more, which is good for business. Plus, with big competitors offering super-fast next-day deliveries, stores are working hard to improve their delivery systems and keep their customers. It's especially important for the last part of the journey, from when the package is sent out for delivery to when it's handed over. This part is called the "last mile" in the delivery process. Getting this part right is crucial because it can make up a big chunk of the total cost of delivering the package. And for customers, it's the part they're most excited about and want to know about. Giving them accurate delivery times at this stage helps set the right expectations.

A lot of methods for guessing travel times need details about the exact route taken. While this works in theory, in real life, it's not always possible to plan out the whole route in advance. This creates uncertainty. That's why there's a growing need for solutions that only depend on knowing when the journey starts and when it ends. We call this "origin-destination travel time estimation" or OD-TTE for short. An example of a OD-TTE approach incorporating Internet-Of-Things(IOT) Technology is shown in Figure 1.

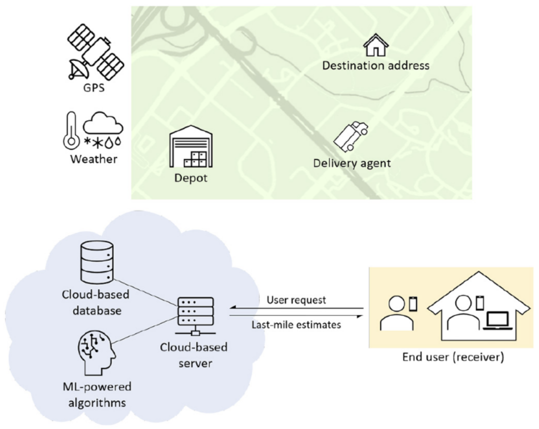


Fig. 1. In Related work, Architecture of a last-mile OD-TTE approach utilized with Internet-Of-Things(IOT) Technology

To tackle this challenge, we turn our attention towards leveraging the potential of Machine Learning techniques for predicting Delivery Durations. The objective is to estimate the time required to cover the crucial "last mile" for an order using a dataset comprising numerous order records within a specific geographic area, generously provided by DoorDash. Within the realm of prediction modeling, a variety of Machine Learning techniques have surfaced. These encompass models employing boosting methodologies, ensemble learning, and the application of Multi-Layer Perceptron approaches. In this paper, we set forth a comprehensive agenda that encompasses the following key tasks:

1) Establishing a Comparative Analysis: This involves cre-

ating a foundational and straightforward baseline approach, akin to a linear regression model. Subsequently, we will apply multiple other Machine Learning techniques to address the problem. This comparative analysis aims to elucidate which approach excels in terms of both accuracy in prediction and computational efficiency.
2) Evaluating Training Times: It is imperative to gauge the duration required for training each of the employed techniques. This step provides critical insights into the computational efficiency and resource requirements of each approach.
3) Determining the Optimal Approach: By considering factors such as training time, predictive accuracy, and runtime performance, we endeavor to offer a well-informed conclusion regarding which approach emerges as the most suitable solution for addressing the specific challenges posed by the given problem statement.

By undertaking these tasks, seek to not only shed light on the most effective approach for Delivery Duration Prediction but also provide valuable insights for optimizing operational processes in the domain of last-mile deliveries.

We explore a range of approaches to address the predictive modeling task. First, We employ Linear Regression as a Baseline model, a widely adopted and straightforward architecture for regression tasks. This model encapsulates a linear function within the data space, making it a popular choice for initial analyses. Next, We delve into Decision Trees, a versatile non-parametric supervised learning method extensively used for both classification and regression tasks. Decision Trees partition the data into subsets based on features, creating a hierarchical structure that aids in making accurate predictions. Following this, We delve into the power of Random Forests, a meta estimator technique. This method involves fitting numerous decision tree classifiers on various sub-samples of the dataset and employs averaging to enhance predictive accuracy and mitigate overfitting. Random Forests have proven to be exceptionally effective in capturing complex relationships within the data. Another pivotal technique in my exploration is XGBoost, or eXtreme Gradient Boosting. Widely acclaimed, this algorithm is renowned for its efficiency and effectiveness in handling an extensive array of predictive modeling tasks. Employing a gradient boosting framework, XGBoost iteratively builds a series of decision trees, each aimed at correcting the errors of its predecessor, resulting in a robust ensemble model. Finally, We investigate a Multi-Model Artificial Neural Network (ANN), an advanced deep learning approach that excels at learning intricate representations within the data space. This architecture encompasses multiple interconnected neural nodes and activation layers, working in tandem to generate highly effective predictions. The network's ability to capture complex patterns and relationships in the data makes it a compelling choice for sophisticated modeling tasks. By exploring these diverse approaches, I aim to gain comprehensive insights into their respective strengths and capabilities in the context of predictive modeling My key achievement lies in doing thorough tests to answer these questions and show how using machine learning on time-related information can be valuable in real-world industries. Next, We'll talk about other studies related to this issue. Then, We'll explain how I conducted my tests and set everything up. After that, We'll share and talk about the results. Finally, We'll suggest some possible future steps and wrap up my paper.

II. Related Work

Numerous studies have been conducted to develop and propose solutions for travel time prediction in the context of intelligent transportation systems. These solutions encompass various modes of transportation, including goods, public, and private transport. While many studies focus on short-term predictions within limited geographical areas, long-term predictions involving trips spanning several cities present significant challenges due to the complexity of the setting and the scarcity of real-time sensor data. In our work, we concentrate on employing machine learning techniques to address long-term travel time prediction for logistics. In this section, we present relevant research conducted in a similar context.

In a comprehensive review [1], the authors discuss primary approaches used to gather data and estimate travel times, which include statistical methods, machine learning techniques, and a historical data estimation method. Another study [2] introduces kriging, a statistical approach relying on spatial predictions for car travel time. Support vector regression is utilized in [3] to predict travel time for highway traffic data, while [4] focuses on urban networks and proposes a solution based on meta-rules using data from location-based services.

Furthermore, several studies, such as [5] and [6], emphasize the last mile delivery of goods and services. For instance, [5] employs a long short-term memory network to predict travel times associated with home delivery of large appliances. Meanwhile, [6] presents a food delivery use case and combines optimization heuristics with machine learning approaches, including linear models like the least absolute shrinkage and selection operator (LASSO) and ridge regression, as well as nonlinear models like support vector regression and random forest.

While much research focuses on proposing new formulations or improving specific aspects of last-mile delivery, there is limited literature specifically addressing the application of travel time estimation. One recent work, DeepETA [7], introduces a spatial-temporal sequential neural network model for estimating travel (arrival) time for parcel delivery with multiple destinations, considering the order of parcel deliveries. Although it achieves state-of-the-art performance, it relies on the availability of delivery routes, making it incompatible with our OD-based formulation.

Another approach of Travel time estimation involves the task of predicting travel durations from available data. While it encompasses last-mile delivery, the majority of works in this area have focused on other applications such as taxi trip durations. The literature on travel time estimation is typically categorized into two groups: route-based methods, which rely on knowledge of the traveled routes, and OD-TTE (Origin Destination - Travel Time Estimation) methods, which only require the start and end points of the trip. Recent efforts have leaned towards OD research, motivated by factors such as privacy concerns, tracking costs, or the inability to fully preplan the route.

A. Route-Based Travel Time Estimation

In [8], a neural network is utilized to predict bus travel times based on factors such as hour-of-day, day-of-week, and weather conditions. The network estimates travel time for each route segment and refines predictions using Kalman filtering and real-time bus locations. It estimates the duration of a bus ride based on a scheduled route and a source-destination pair. [9] predicts travel time based on floating-car data, employing a combination of linear models, deep, and recurrent neural networks to analyze road segment sequences. [10] introduces DeepTTE, an end-to-end deep learning framework that predicts travel time for an entire path, incorporating convolutional and recurrent stages. [11] presents DEEPTRAVEL, which estimates the travel time of a whole trip directly, using a feature extraction structure to capture spatial, temporal, and traffic dynamics.

B. OD Travel Time Estimation

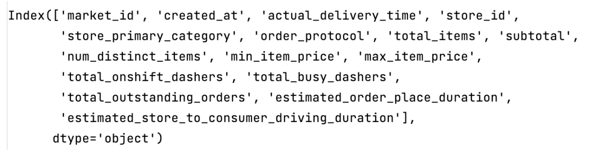
[12] introduces an approach to discover spatiotemporal patterns in OD displacements, utilizing spatial clustering of coordinates to identify meaningful places and extracting flow measures. In [13], taxi OD data is employed to estimate the hourly average travel times for urban links. [14] estimates daily OD trips from triangulated mobile phone records, converting the data into clustered locations where users engage in activities. Trips are then constructed for each user between consecutive observations in a day. It proposes a simple baseline to travel time estimation (SB-TTE) using large-scale trip data, estimating travel time between two points based on origin, destination, and actual distance traveled. However, it should be noted that the actual traveled distance may not be available before travel. In [15], insights from network optimization are applied to extract accurate travel time estimations from a large OD dataset, reconstructing traffic patterns in a city. It introduces the spatiotemporal neural network (ST-NN) model for OD travel time estimation, which jointly predicts trip durations and traveled distances. [16] presents MURAT, a multitask representation learning model for arrival time estimation leveraging road networks and spatiotemporal priors. Although an OD-based approach, MURAT requires route information for training. Finally, [17] employs deep neural networks for estimating travel times in an NYC taxi trip dataset within the context of OD-TTE.

In summary, our investigation into boosting approaches for travel time prediction in the logistics sector fills some gap in the existing literature. The diverse array of methodologies and techniques discussed in this related work section lays the foundation for our research.

III. Methodology

In this study, our focal point revolves around the comprehensive dataset procured from DoorDash, a prominent delivery service platform. This dataset serves as the bedrock of our exploration into the intricacies of delivery operations. By delving into the multifaceted information encompassed within DoorDash's data, our goal is to develop and evaluate predictive models tailored specifically to the challenges inherent in delivery logistics. This dataset encapsulates a diverse array of parameters, including order specifics, temporal nuances, geographical elements, store interactions, and delivery durations, providing a rich landscape for analysis and modeling. Leveraging machine learning and predictive analytics, our aim is to unearth underlying patterns and dependencies, contributing not only to a deeper understanding of delivery dynamics but also to the refinement of accurate delivery time estimations. Ultimately, our study seeks to bridge the gap between data-driven insights and operational improvements within the realm of logistics and e-commerce.

The data-set consists of 197,428 rows and 16 columns as raw data. The features include as shown in Figure 2.



```
Index(['market_id', 'created_at', 'actual_delivery_time', 'store_id',
       'store_primary_category', 'order_protocol', 'total_items', 'subtotal',
       'num_distinct_items', 'min_item_price', 'max_item_price',
       'total_onshift_dashers', 'total_busy_dashers',
       'total_outstanding_orders', 'estimated_order_place_duration',
       'estimated_store_to_consumer_driving_duration'],
      dtype='object')
```

Fig. 2. Raw Features

A. Data Pre-Prcoessing

The data-set underwent a meticulous process of data cleaning and formatting to ensure its quality and consistency. Cleaning involved removing duplicates and handling missing values, followed by standardizing formats across various data entries. To enhance our model's efficiency and computational tractability, we employed Principal Component Analysis (PCA) for dimensionality reduction, enabling us to distill essential information while minimizing redundancy. At the later stages we implemented our models to showcase their predicitive capabilites on the data-set. The testing phase involved computing the evaluating metrics such as Root Mean Squared Error (RMSE), Mean Absolute Error (MAE), and Fit Time. For our work we followed the phases listed ahead for Data Pre-Processing.

1) Data Cleaning and Formatting: The dataset underwent rigorous cleaning procedures to ensure data quality and integrity. Steps included removal of duplicates, handling missing values, and standardizing data formats for consistency across all entries.

2) Dimensionality Reduction using PCA: Principal Component Analysis (PCA) was applied to reduce the dataset's dimensionality, allowing for the extraction of essential information while minimizing redundancy and noise. The graph to explain the 90% of dataset with the give number of features is shown in the Figure 3.

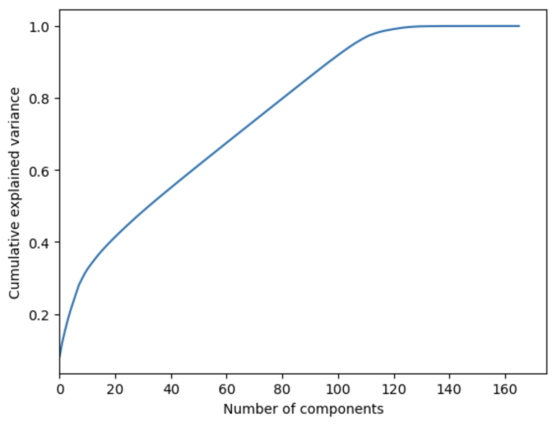


Fig. 3. PCA Analysis

3) Feature Engineering: Several additional features were engineered to enhance the dataset's predictive power, such as "Estimated_time_per_item," "total_duration_in_store," and "total_duration_out_of_store." These features were strategically created to capture nuanced aspects related to delivery durations and store interactions.

4) Date-Time Separation: The date-time feature was dissected to create distinct features representing temporal aspects such as "delivery_month," "delivery_year," "delivery_day," and "delivery_time." This segmentation enabled the model to capture time-related patterns effectively.

5) Feature Selection: Certain features were removed based on redundancy or insignificance for the predictive model. Retained features were carefully selected to maintain relevance and predictive power in estimating delivery times.

6) Data Summary: Key statistical summaries were generated to provide an overview of the dataset, including metrics such as "Number_of_orders," "Number_of_trips_per_day," and "Total_trips." These summaries offer insights into the volume and frequency of delivery-related activities within the dataset.

B. Baselines

To study the performance of predictive models, we tested on following baseline approaches.

1) Decision Trees: Decision trees are intuitive, hierarchical models used extensively in machine learning and data analysis. These models operate by recursively partitioning the dataset into smaller subsets based on the most influential features, creating a tree-like structure. Each node in the tree represents a decision based on a feature, and the branches represent the potential outcomes or decisions that stem from that feature. Decision trees excel in handling both categorical and numerical data while being relatively easy to interpret and visualize. They facilitate the understanding of complex decision-making processes and are highly versatile, capable of handling classification and regression tasks. However, they can be prone to overfitting, especially with deep or complex trees, and may require pruning or other techniques to optimize performance and generalization.

2) XGBoost: XGBoost stands as a potent algorithm in the realm of machine learning, particularly in predictive modeling and regression tasks. Short for eXtreme Gradient Boosting, XGBoost operates as a scalable and efficient gradient boosting framework. It excels in handling large datasets and is renowned for its superior performance due to its ability to harness the power of ensemble learning. By sequentially building multiple weak models and iteratively improving upon their weaknesses, XGBoost achieves remarkable accuracy and predictive power. Its strength lies in optimizing computational efficiency and model performance through parallel and distributed computing, feature importance analysis, and regularization techniques. XGBoost has emerged as a go-to choice in various machine learning competitions and real-world applications for its exceptional speed, accuracy, and ability to handle diverse data types.

3) LightGBM: LightGBM stands tall as a high-performance gradient boosting framework, acclaimed for its speed, efficiency, and scalability in handling large datasets. Developed by Microsoft, LightGBM leverages a novel decision tree algorithm that focuses on leaf-wise growth rather than level-wise, prioritizing nodes with higher loss for faster learning. This innovative approach significantly reduces the computational resources required while maintaining exceptional accuracy and model quality. LightGBM shines in scenarios where speed and scalability are paramount, making it a top choice for handling big data and real-time applications. Its ability to handle categorical features efficiently, along with robust regularization techniques, contributes to its effectiveness in various machine learning tasks, particularly in classification and regression problems.

4) Ensemble of XGBoost with K-NN: Combining the strengths of XGBoost, a high-performance gradient boosting framework, with the K-Nearest Neighbors (K-NN) algorithm creates a powerful synergy in machine learning. XGBoost's efficiency in handling large datasets and its ability to capture complex patterns through boosted decision trees complement the simplicity and intuitiveness of the K-NN algorithm in dealing with local patterns and instance-based learning. By integrating XGBoost with K-NN, the model benefits from XGBoost's speed and scalability while leveraging K-NN's proficiency in identifying neighboring data points. This amalgamation allows for a comprehensive approach that can tackle diverse data structures and patterns, enhancing predictive accuracy and model robustness across a wide range of machine learning tasks. The architecture of the Ensemble is explained in the Figure 4. The data-set is first aggregated to create clusters of samples. Theses sample clusters are assigned a weighted average of target variable as ”total_duration”. In the contrary, a predictive model like XGBoost or LightGBM is trained on the dataset with hyper-parameter optimizations and mentioned evalutaion metrics. For testing, the sample input vector is fed to a decision node, where decision is based on whether the input vector has a Euclidean distance to a cluster within a distance margin of *d*. If the decision is *Yes*, The centroid of the cluster representing the ”total_duration” is returned with K-NN model. K-nearest neighbors (K-NN) is a simple yet effective supervised learning algorithm used for regression tasks. It predicts the class or value of a new data point by identifying the majority class or averaging the values of its nearest neighbors in the feature space, determined by a predefined number *k* of neighboring data points. Here *k* is a hyperparameter, which is changed and tuned to get the best results. If the decision is *No*, the sample vector is fed the trained predictive model like XGBoost.

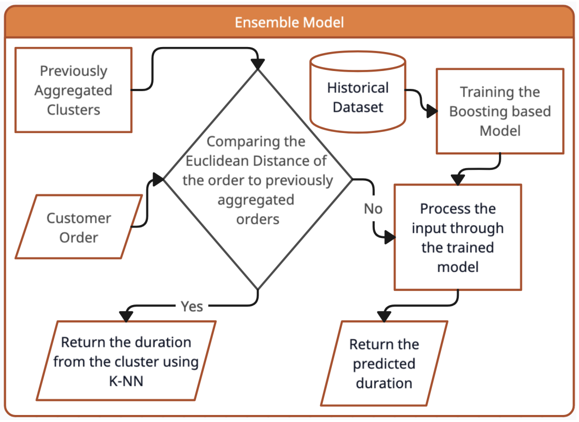

Ensemble Model
Previously Aggregated Clusters
Historical Dataset
Training the Boosting based Model
Comparing the Euclidean Distance of the order to previously aggregated orders
Customer Order
No
Process the input through the trained model
Yes
Return the duration from the cluster using K-NN
Return the predicted duration

Fig. 4. Ensemble Model

5) Ensemble of LightGBM with K-NN: The Implementation of this baseline is similar to previous one. The trained predictive model is changed to a LightGBM model when sample vector is fed from the decision node. This model is tested on the sample vector when there is no cluster in the dataset within a margin of *d* distance. The experimental setup, evaluation metrics and results are shown in the sections ahead.

IV. Evaluation

The evaluation of the aforementioned baselines is based on three different metrics. We detail about the experimental setup in this section.

A. Metrics

Following are the metrics used for testing.

1) Root Mean Squared Error (RMSE): The root mean squared error is defined as

$$\text{RMSE} = \sqrt{\frac{1}{n}\sum_{i=1}^{n}(y_i - \hat{y}_i)^2}$$

Here the Error is calculated by Summing the Squared error and taking a square root of it. Since the errors are squared before they are averaged, RMSE penalizes large errors more thus a good measure for larger deviations in the error.

2) Mean Absolute Error (MAE): The root mean squared error is defined as

$$\text{MAE} = \frac{1}{n}\sum_{i=1}^{n}|y_i - \hat{y}_i|$$

MAE is less sensitive to outliers compared to MSE because it takes the absolute differences between predicted and actual values, making it more robust in the presence of extreme values.

B. Training and Testing Procedures

The dataset is split into training and testing sets with a random sampling of 70-30 split. In adherence to standard machine learning practices, the dataset underwent a systematic partitioning into distinct training and testing sets. Employing a random sampling strategy, the dataset was segregated into a training subset, comprising 70% of the data, and a complementary testing subset accounting for the remaining 30%. This delineation ensured a balanced representation of instances across both partitions, preserving the integrity and diversity of the dataset. The training set was exclusively utilized for model training, allowing algorithms to learn patterns and relationships inherent in the data. Meanwhile, the testing set, independent of the training process, served as an unseen benchmark, enabling rigorous evaluation of the trained models’ performance and generalization capabilities. This meticulous split facilitated robust model development and validation, ensuring the derived algorithms were proficient in making accurate predictions on new, unseen data instances.

C. Parameters and Hyper parameters

To improve our models’ performance, we took a careful look at their settings, known as hyperparameters. We used a smart method called 5-fold cross-validation, splitting our training data into five parts. Then, we tested various hyperparameter combinations, like changing the settings of the models, to see which ones worked best. Each time, we held out 20% of the data to check how well the model did with new, unseen information. This process helped us find the best settings that made our models work really well, making sure they could handle new situations accurately.

V. Results and Discussion

After training individual baselines with the aforementioned pipeline, we show results in Table I.

In order to analyze the results, we consider in the following paragraphs different aspects separately.

TABLE I
EVALUATION RESULTS AFTER TRAINING (ALL VALUES IN SECONDS)

Model	RMSE	MAE	Fit Time
Decision Tree	19249.897	1783.544	**2**
XGBoost	973.499	605.925	356
LightGBM	686.784	557.112	427
XGBoost w/ K-NN	879.421	587.262	572
LightGBM w/ K-NN	**621.154**	**510.236**	623

A. MAE Comparison

We note from the Results Table I, the simple Decision Tree Predictive model performs works on the MAE, as compared to the boosting models. Our implemented ensemble model of LightGBM w/ K-NN performed the best. Other boosting models had a comparative results.

B. RMSE Comparison

We note that RMSE values show in Table I, have a more broader difference as compared to MAE values in the boosting models. The ensemble model again performs best for RMSE values.

C. Fit-Time Comparison

When training these models the ensemble had the worst fit-time but better accuracy than other models. This is useful when considering a trade-off between accuracy and fit-time for pre-training a predicitive model. The ensemble model has higher fit-time as it constitutes of time taken to train the model by LightGBM and K-NN.

The above comparisons show that our ensemble model performed the best when compared with other baselines.

VI. FUTURE WORK

The study faces limitations rooted in the scope and quality of available data, potentially constraining the predictive capacity of the models. While the chosen model exhibited promising performance, its adaptability to diverse scenarios or unforeseen data patterns remains a pertinent consideration. Future research endeavors might delve into incorporating additional data facets such as weather and GPS information, potentially enhancing the estimation of delivery times by capturing nuanced relationships within the dataset.

VII. CONCLUSION

In summary, A two stage regression model was developed for predicting delivery times. The initial classifier effectively manages typical delivery order scenarios, providing a foundational framework. The second model handles more complicated outliers. Model performance evaluation revealed that a meticulously trained ensemble of LightGBM and K-NN emerged as the most accurate predictor for delivery times within this dataset.

VIII. ACKNOWLEDGMENT

This work would not have been possible without the help of Professor Thomas B Kinsman, Ph.D. I am grateful of his support as a faculty advisor for devising ideas related to this research. I would also like to thank my parents for their continuous help.

IX. REFERENCES

1. H.-E. Lin, R. Zito, M. Taylor et al., "A review of travel-time prediction in transport and logistics," in Proceedings of the Eastern Asia Society for transportation studies, vol. 5. Bangkok, Thailand, 2005, pp. 1433–1448.
2. H. Miura, "A study of travel time prediction using universal kriging," Top, vol. 18, no. 1, pp. 257–270, 2010.
3. C.-H. Wu, J.-M. Ho, and D.-T. Lee, "Travel-time prediction with support vector regression," IEEE transactions on intelligent transportation systems, vol. 5, no. 4, pp. 276–281, 2004.
4. W.-H. Lee, S.-S. Tseng, and S.-H. Tsai, "A knowledge based real-time travel time prediction system for urban network," Expert systems with Applications, vol. 36, no. 3, pp. 4239–4247, 2009.
5. M. Gmira, M. Gendreau, A. Lodi, and J.-Y. Potvin, "Travel speedprediction based on learning methods for home delivery," EURO Journal on Transportation and Logistics, p. 100006, 2020.
6. S. Liu, L. He, and Z.-J. M. Shen, "On-time last mile delivery: Orderassignment with travel time predictors," Forthcoming in Management Science, 2018.
7. F. Wu and L. Wu, "DeepETA: A spatial–temporal sequential neural network model for estimating time of arrival in package delivery system," in Proc. AAAI Conf. Artif. Intell., vol. 33, Jul. 2019, pp. 774–781.
8. M. Chen, X. Liu, J. Xia, and S. I. Chien, "A dynamic bus-arrival time prediction model based on APC data," Comput.-Aided Civil Infrastruct. Eng., vol. 19, no. 5, pp. 364–376, 2004.
9. Z. Wang, K. Fu, and J. Ye, "Learning to estimate the travel time," in Proc. 24th ACM SIGKDD Int. Conf. Knowl. Disc. Data Min. (KDD), 2018, pp. 858–866.
10. D. Wang, J. Zhang, W. Cao, J. Li, and Y. Zheng, "When will you arrive? Estimating travel time based on deep neural networks," in Proc. AAAI, Jan. 2018, pp. 2500–2507.
11. H. Zhang, H. Wu, W. Sun, and B. Zheng, "DeepTravel: A neural network based travel time estimation model with auxiliary supervision," in Proc. 27th Int. Joint Conf. Artif. Intell. (IJCAI), 2018, pp. 3655–3661.
12. D. Guo, X. Zhu, H. Jin, P. Gao, and C. Andris, "Discovering spatial patterns in origin-destination mobility data," Trans. GIS, vol. 16, no. 3, pp. 411–429, 2012.
13. X. Zhan, S. Hasan, S. V. Ukkusuri, and C. Kamga, "Urban link travel time estimation using large-scale taxi data with partial information," Transp. Res. C Emerg. Technol., vol. 33, pp. 37–49, Aug. 2013.
14. L. Alexander, S. Jiang, M. Murga, and M. C. González, "Origin-destination trips by purpose and time of day inferred

from mobile phone data," Transp. Res. C Emerg. Technol., vol. 58, pp. 240–250, Sep. 2015.

15. I. Jindal, T. Qin, X. Chen, M. Nokleby, and J. Ye, "A unified neural network approach for estimating travel time and distance for a taxi trip," Oct. 2017. [Online]. Available: arXiv:1710.04350.
16. Y. Li, K. Fu, Z. Wang, C. Shahabi, J. Ye, and Y. Liu, "Multi-task representation learning for travel time estimation," in Proc. 24th ACM SIGKDD Int. Conf. Knowl. Disc. Data Min. (KDD), 2018, pp. 1695–1704.
17. A. C. de Araujo and A. Etemad, "Deep neural networks for predicting vehicle travel times," in Proc. IEEE SENSORS, 2019, pp. 1–4.