Comparing hundreds of machine learning classifiers and discrete choice models in predicting travel behavior: an empirical benchmark

Preprint · February 2021

CITATIONS
0

READS
518

4 authors, including:

Shenhao Wang
University of Florida
40 PUBLICATIONS **408** CITATIONS
SEE PROFILE

Baichuan Mo
Lyft
40 PUBLICATIONS **434** CITATIONS
SEE PROFILE

Jinhua Zhao
Massachusetts Institute of Technology
181 PUBLICATIONS **4,100** CITATIONS
SEE PROFILE

Comparing hundreds of machine learning classifiers and discrete choice models in predicting travel behavior: an empirical benchmark

Shenhao Wang[1,4,*], Baichuan Mo[3], Stephane Hess[2], Jinhua Zhao[1]

1 - Department of Urban Studies and Planning, Massachusetts Institute of Technology
2 - Choice Modeling Centre & Institute for Transport Studies, University of Leeds
3 - Department of Civil and Environment Engineering, Massachusetts Institute of Technology
4 - Media Lab, Massachusetts Institute of Technology

January 2021

Abstract

A growing number of researchers have compared machine learning (ML) classifiers and discrete choice models (DCMs) in predicting travel behavior, but the generalizability of the findings is often limited by the specifics of data, contexts, and authors' expertise. This study seeks to provide a definitive and generalizable empirical benchmark by comparing hundreds of ML and DCM classifiers in a highly structured manner. The experiments evaluate both prediction accuracy and computational cost by spanning four hyper-dimensions, including 105 ML and DCM classifiers from 12 model families, 3 datasets, 3 sample sizes, and 3 outputs. This experimental design leads to an immense number of 6,970 experiments, which are further corroborated with a meta dataset of 136 experiment points from 35 previous studies. This study is hitherto the most comprehensive and an almost exhaustive comparison of the classifiers for travel behavioral prediction.

We found that the **ensemble methods**, including boosting, bagging, and random forests, and **deep neural networks**, achieve the highest predictive performance, but at a relatively high computational cost. **Random forests** are the most computationally efficient, thus balancing between prediction and computation. While **discrete choice models** can offer accuracy with only 3-4 percentage points lower than the the top ML classifiers, they have much longer computational time and become computationally impossible with large sample size, high input dimensions, or simulation-based estimation. The relative ranking of the ML and DCM classifiers is highly stable, while the absolute values of the prediction accuracy and computational time have large variations. Prediction accuracy varies significantly with datasets, and larger sample size leads to moderately higher prediction but significantly longer computational time. Overall, this paper suggests to use deep neural networks, model ensembles, and random forests as baseline models for future travel behavior prediction. It also suggests a synergetic perspective between the ML classifiers and DCMs, and further research into versatile neural network architectures. For choice modeling, the DCM community should switch more attention from fitting models to improving computational efficiency, so that the DCMs can be widely adopted in the big data context.

Keywords: Machine learning; choice modeling; travel behavior; prediction.

Corresponding author: Shenhao Wang; email: shenhao@mit.edu

1. Introduction

Travel behavior prediction is a fundamental topic in transportation research [3, 10, 4, 63, 15]. Although this topic has traditionally been addressed by discrete choice models (DCMs) [67], researchers have started to predict travel behavior by using a long list of machine learning (ML) classifiers and comparing them to the classical DCMs [28, 72, 12]. The comparative studies are important since they provide insights into model selection, reveal novel behavioral mechanisms different from random utility maximization, and improve the performance of demand forecasting with the new ML classifiers. However, the comparative studies can rarely provide a definitive and generalizable conclusion about the predictive power of the ML and DCM classifiers[1], since they are often limited by the specifics of data, contexts, or authors' expertise. For example, researchers might choose a specific ML classifier simply due to coding convenience or based on their own expertise. Or they may find that ML classifiers outperform DCMs only because of the particularity of the datasets or the research contexts. These two issues, i.e. model selection as well as application context, inevitably limit the generalizability of their conclusions. Therefore, a useful empirical benchmark is still lacking in the travel demand modeling community, since a critical question is unresolved: how to use a single study to provide a generalizable empirical benchmark for the comparative performance of the ML and DCM classifiers in predicting travel behavior?

To answer this question, we first define the concepts of **hyper-dimensions**, **experiment point**, and **experiment space**. Hyper-dimensions are defined as the aspects that significantly influence the predictive performance of a classifier, and these include the type of classifiers, datasets, sample sizes, outputs[2], among others. An experiment point is defined as a model trained with given fixed hyper-dimensions. For example, an experiment point can be a multinomial logit model trained with a specific dataset of 1,000 observations to predict travel mode choices. The experiment space is defined as the space including all the experiment points spanned by the hyper-dimensions. These three concepts can be intuitively understood from the perspective of a meta-analysis, which draws conclusions by integrating the results from many past studies that aim to solve the same problem [19]. A meta-analysis can be seen as a summary of an experiment space, in which each past study serves as an experiment point. Therefore, a single study can yield generalizable results when it resembles the nature of a meta-analysis by incorporating critical hyper-dimensions, a typically vast experiment space, and a comprehensive, representative, and relevant list of experiment points. Unfortunately, none of the past studies comparing the performance of ML and DCM classifiers has achieved this goal because they did not make these concepts explicit, let alone satisfying the criteria.

The present study seeks to provide a definitive and generalizable empirical benchmark about the performance of the ML and DCM classifiers in predicting travel behavior by investigating

[1] We will use the term *classifiers* throughout the rest of this paper. It is the common term used in ML, while in choice modelling, the term *model* is more common.

[2] *Inputs* and *outputs* are commonly used in the ML literature, and they are corresponding to *independent variables* and *dependent variables* in choice modeling.

a tremendous experiment space consisting of four hyper-dimensions and 1,394 experiment points. Empirically, we seek to (1) identify the globally best model family and classifiers for travel behavior prediction, (2) provide insights into how model performance varies across datasets, sample size, and outputs, and (3) discuss the pros and cons of ML classifiers and DCMs in terms of both prediction accuracy and computational time. We chose prediction accuracy as the metric to evaluate predictive performance, since it is nearly the only metric allowing the comparison across a large number of classifiers. We documented the computational time to evaluate the computational burden, which is important since a theoretically feasible model can be deemed as practically impossible when the computational time[3] is extremely long. We designed a large-scale experiment spanning four hyper-dimensions, including 105 classifiers from 12 model families as the primary hyper-dimension, and 3 datasets, 3 sample sizes, and 3 travel behaviors as three secondary hyper-dimensions. The hyper-dimensions are organized as follows.

- Classifiers. The primary hyper-dimension is the classifiers. The experiments examine 105 classifiers from the 12 most important model families, including (1) discrete choice models (DCM; 3 models), (2) deep neural networks (DNN; 15 models), (3) discriminant analysis (DA; 12 models), (4) Bayesian methods (BM; 7 models), (5) support vector machines (SVM; 9 models), (6) K nearest neighbors (KNN; 4 models), (7) decision trees (DT; 14 models), (8) generalized linear models (GLM; 10 models), (9) Gaussian process (GP; 3 models), (10) bagging (3 models), (11) random forests (RF; 2 models), and (12) boosting (23 models).
- Datasets. Three datasets are used, including the US national household travel survey 2017 (NHTS2017), the London travel demand survey 2015 (LTDS2015), and a stated preference survey collected in Singapore in 2017 (SGP2017). The three datasets cover two important data collection procedures - revealed and stated preference surveys - and three geographical locations in America, Europe, and Asia.
- Sample sizes. The sample size includes 1,000, 10,000, and 100,000 observations, which can represent the main scales of the sample sizes in the travel behavior research.
- Outputs. The outputs include three travel behaviors: travel mode choice, car ownership, and trip purposes. Owing to data limitation, car ownership and trip purposes are only available in the NHTS2017 dataset.

In this experiment, a single classifier trained on a dataset, a sample size, and a targeting output is an experiment point; for example, an experiment point can refer to a multinomial logit model trained on NHTS2017 with 1,000 observations to predict travel mode choice. All the experiment points together constitute our experiment space. Only with this highly structured experiment design and a massive number of experiments can it be possible to identify a generalizable result for the comparative performance of ML and DCM classifiers in predicting travel behavior.

We defined the concepts of hyper-dimensions, experiment points, and experiment space so that we can jointly analyze the large-scale experiments and a meta dataset, which is a literature review

[3]In choice modelling, the term *estimation time* is used more commonly.

summarizing 136 experiment points from 35 previous studies. The joint analysis is possible since the modeling results from these previous studies are also valid experiment points with insightful hyper-dimension information. The meta dataset is organized by the four hyper-dimensions and used to inform our experiment design and corroborate with our empirical results. The meta dataset and the empirical experiments complement each other: the empirical experiments prevail in the completeness of classifiers, while the meta dataset is particularly strong in the richness of contexts and data types. A joint analysis can further improve the generalizability and the validity of our findings.

The remainder of this paper is organised as follows. The next section introduces the meta dataset of previous studies, serving as our literature review. Section 3 discusses the experiment design for the four hyper-dimensions, with a specific focus on the list of classifiers. Section 4 summarizes other aspects of the experiment design, such as data processing and computational process. Section 5 presents the empirical results of the large-scale experiments. Section 6 compares the results from our empirical experiments to the meta dataset of the previous studies. Section 7 summarizes the findings about model performance, computational time, and the effects of the hyper-dimensions. Section 8 presents discussions and future research possibilities. Appendix I includes a table of acronyms and terminology to translate the terminology between ML classifiers and DCMs.

2. Literature review: a meta dataset

The authors collected a meta dataset of 35 previous studies and 136 experiment points, as shown in Table 1. This meta dataset is organized by the four hyper-dimensions, with the first column indexing the studies, the second column presenting the author-year information, the third column the three outputs, the fourth column the sample size, the fifth column the number of experiment points and compared models, and the last column the best model. Each row compiles several experiment points trained on the same dataset, so that the models on every row are comparable. Four studies, indexed as 12, 22, 34, and 35, used more than one dataset, so they are divided into more than one row [82, 60, 74, 73]. Although this meta dataset is not exhaustive, the structure in Table 1 can inform our experiment design. This meta dataset is also used to validate our empirical findings, which will be discussed in Section 6.

Study Index	**Author (Year)**	**Tasks**	**Sample Size**	**Number of Experiment Points - Compared Models**	**Best Model**
1	Nijkamp et al. (1996) [50]	MC	1,396	2 - DNN, MNL	DNN
2	Rao et al. (1998) [56]	MC	4,335	2 - DNN, MNL	DNN
3	Hensher and Ton (2000) [31]	MC	1,500	2 - DNN, NL	DNN/NL
4	Sayed and Razavi (2000) [58]	MC	7,500	2 - MNL, DNN	DNN
5	Cantarella et al. (2002) [9]	MC	2,808	3 - DNN, MNL, NL	DNN
6	Mohammadian et al. (2002) [46]	CO	597	2 - DNN, NL	DNN

7	Doherty and Mohammadian (2003) [16]	TP	5,583	2 - DNN, GLM	DNN
8	Xie et al. (2003) [80]	MC	4,747	3 - DT, DNN, MNL	DNN
9	Cantarella et al. (2005) [10]	MC	1,067	2 - DNN, MNL	DNN
10	Celikoglu (2006) [11]	MC	381	4 - DNN, MNL	DNN
11	Tortum et al. [66]	MC	441	3 - MNL, GLM, DNN	DNN
12	Zhang and Xie (2008) [82]	MC	1,000	3 - MNL, DNN, SVM	SVM
12	Zhang and Xie (2008) [82]	MC	2,000	3 - MNL, DNN, SVM	SVM
13	Biagioni et al. (2008) [5]	MC	19,118	4 - MNL, BM, BOOSTING, DT	BOOSTING
14	Xian and Jian (2011) [79]	MC	4,725	3 - SVM, NL, DNN	SVM
15	Allahviranloo and Recker (2013) [2]	TP	3,671	2 - SVM, MNL	SVM
16	Omrani et al. (2013) [52]	MC	3,673	6 - DNN, SVM, KNN, MNL, DT, BM	DNN
17	Ermagun et al. (2015) [18]	MC	4,700	2 - NL, RF	RF
18	Jahangiri et al. (2015) [37]	MC	N.A.	6 - KNN, SVM, DT, BAGGING, RF, MNL	RF
19	Tang et al. (2015) [65]	MC	72,536	2 - DT, MNL	DT
20	Omrani (2015) [51]	MC	9,500	4 - DNN, RBFNN, MNL, SVM	DNN
21	Shafique et al. (2015) [60]	MC	1,968	4 - SVM, BOOSTING, DT, RF	RF
21	Shafique et al. (2015) [60]	MC	1,488	4 - SVM, BOOSTING, DT, RF	RF
21	Shafique et al. (2015) [60]	MC	2,754	4 - SVM, BOOSTING, DT, RF	RF
22	Shukla et al. (2015) [62]	MC	100,000	2 - DNN, DT	DNN
23	Sekhar and Madhu (2016) [59]	MC	4,976	3 - RF, DT, MNL	RF
24	Hagenauer and Helbich (2017) [28]	MC	230,608	8 - MNL, DNN, NB, SVM, CTs, BOOSTING, BAGGING, RF	RF
25	Paredes et al. (2017) [53]	CO	15,211	5 - MNL, BOOSTING, DT, SVM, RF	RF
26	Hillel et al. (2018) [32]	MC	N.A.	8 - DNN, BAGGING, BOOSTING, KNN, GLM, BM, RF, SVM	BOOSTING
27	Golshani et al. (2018) [25]	MC	9,450	2 - MNL, DNN	DNN
28	Tang et al. (2018)	MC	14,000	2 -MNL, DT	DT
29	Wang and Ross (2018) [72]	MC	51,910	2 - BOOSTING, MNL	BOOSTING
30	Lee et al. (2018)	MC	4,764	2 - MNL, DNN	DNN
31	Cheng et al. (2019) [12]	MC	7,276	4 - RF, SVM, BOOSTING, MNL	RF
32	Zhou et al. (2019) [83]	MC	30,000	8 - MNL, KNN, DT, SVM, BM, BOOSTING, BAGGING, RF	BAGGING
33	Wang et al. (2020) [73]	MC	8,418	8 - MNL, NL, DNN, SVM, BM, KNN, BOOSTING, GLM	DNN
33	Wang et al. (2020) [73]	MC	2,929	8 - MNL, NL, DNN, SVM, BM, KNN, BOOSTING, GLM	DNN
34	Wang et al. (2020) [74]	MC	80,000	2 - MNL, DNN	DNN
34	Wang et al. (2020) [74]	MC	8,418	2 - MNL, DNN	DNN
35	Wang et al. (2020) [75]	MC	8,418	3 - NL, MNL, DNN	DNN

Notes: MC - mode choice; CO - car ownership; TP - trip purposes

Table 1: A meta dataset of 35 studies and 136 experiment points

Table 1 demonstrates that the most common classifiers are MNL, NL, DNN, GLM, BM, BOOSTING, BAGGING, DT, RF, and SVM, all of which will be incorporated into our empirical experiments. However, it also presents at least four weaknesses of the previous comparisons in choosing the list of classifiers. First, each study incorporated only a small number of experiment points, which refers to not only the small number of classifiers, but also the failure to illustrate the potential variations in datasets and sample sizes, among many other hyper-dimensions. Second, past studies often failed to recognize the richness within each classifier. Many ML classifier categories, such as DNN and BOOSTING, should not be seen as a single model. For example, DNNs can adopt a variety of architectures and hyperparameters, so a single DNN with an arbitrary architecture cannot represent the vast DNN model family. Third, many important classifiers are still missing. On the ML side, the missing classifiers are discriminant analysis (DA) models and generalized linear models (GLM). On the DCM side, the vast majority of the previous studies did not incorporate the mixed logit model (MXL), which is a workhorse in practical travel demand modeling. Lastly, these studies predominantly used predictive performance as the criterion, but failed to acknowledge the practical challenges, such as the computational cost and the variation in coding languages. Although predictive performance is important to reveal the underlying behavioral mechanism, computational time can have practical implications even more important than prediction, since practitioners can decide against using a theoretically ideal model when it requires hours or days in training. These four weaknesses will be addressed in our large-scale experiments.

Every past study contains information on the hyper-dimensions - datasets, sample sizes, and outputs; however, they failed to analyze the variation or acknowledge the impacts of the hyper-dimensions on model performance. On the one hand, the meta dataset in Table 1 informs our experiment design by illustrating the important values in the hyper-dimensions. In Table 1, the sample size ranges from 381 to 230,608 observations[4], suggesting that the scale of 1,000, 10,000, and 100,000 should be able to represent the most important sample sizes in travel behavioral studies. The previous studies analyzed travel mode choice, car ownership, and trip purposes, which can be deemed as the most common travel behaviors. On the other side, these studies ignored the potential variation in predictive performance caused by the hyper-dimensions: it is reasonable to postulate that the predictive performance can vary significantly with the type of dataset and sample size, among many other factors.

This meta dataset is **not** intended to be exhaustive. It excludes a vast number of studies that used only one ML or DCM classifier, and many others that focused on spatio-temporal travel behavior, such as traffic flow and accidents, with a data structure more complex than the classical cross-sectional data [48, 54, 78]. The studies germane to this paper but outside the transportation field are Fernandez-Delgado et al. (2014) and Kotsiantis et al. (2007) [20, 40], which similarly used a large number of ML classifiers for comparison. Our comparison is limited to prediction and computation, but other perspectives are also possible. For example, interpretability and robustness are both critical for successfully deploying the ML methods in practice [43, 17, 47, 49, 26, 74, 73].

[4]The observations are the trips, so the number of individuals and households is smaller.

However, these topics are beyond the scope of the present paper and are left for future work.

3. Experimental design: hyper-dimensions

The experiment design needs to balance the feasibility of conducting a large-scale experiment and the ideal of incorporating as many hyper-dimensions H and values of hyper-dimensions T_h as possible. This balance is a key challenge throughout the present benchmark work. Roughly speaking, the number of experiment points T equals to:

$$T = \prod_{h \in H} |T_h|;\ h \in H = \{f, s, n, y\}, \tag{1}$$

in which f represents classifiers; s datasets; n sample sizes; y outputs; T the total number of experiment points; and T_h the cardinality of each hyper-dimension. Hence our experiments can be treated as a grid search for the experiment space along the four hyper-dimensions. While larger H and T_h always render our results more compelling and generalizable, both values need to be chosen in a quite parsimonious manner to make our experiments feasible, because the complexity T is exponential in the number of hyper-dimensions $|H|$ and polynomial in the number of values $|T_h|$. This study chose four hyper-dimensions ($|H| = 4$), 105 classifiers ($|T_f| = 105$), 3 datasets ($|T_s| = 3$), 3 sample sizes ($|T_n| = 3$), and 3 outputs ($|T_y| = 3$). For each hyper-dimension, this section will demonstrate how and why certain values are incorporated while others are not, based on the meta data of the past studies (Table 1) and the principles of **completeness**, **relevance**, and **representativeness**.

3.1. Hyper-dimension 1: classifiers

The authors seek to provide a **complete** list of classifiers, which is summarized in Table 2 including 105 classifiers from 12 model families. Instead of treating the ML classifiers as single models, we recognize the richness by incorporating many models into each model family. For example, to recognise the significant impacts of DNN architectures on model performance, the DNN family includes 16 DNNs with different architectures and from different coding languages. However, even with our tremendous effort, it is literally impossible to exhaust all classifiers, simply because the true number of potential classifiers is infinite. For example, with a slight variation in model architecture, the DNN family can present a new DNN, which might yield a predictive performance drastically different from the 16 benchmark DNN models in our list. Therefore, to make the large-scale experiments practically feasible, the authors only choose the **representative** classifiers within each model family and the most **relevant** ones for travel behavioral analysis. Intuitively, DCMs are the most relevant given their long time use in the transportation field for behavioral analysis, and DNNs are the second most important due to the rising popularity in many subdomains in transportation [38, 73, 75]. The authors incorporate MNL, NL, and MXL into the model family of DCMs, but even so, the nest structures in NL and the coefficients' heterogeneity in MXL can only be prespecified

following the common standards without further detailed investigations. Nonetheless, Table 2 has presented a relatively complete, highly representative, and highly relevant list of ML and DCM classifiers for travel behavioral analysis. While not completely exhaustive, it is substantially more comprehensive than the past studies.

Classifiers	**Model Families**	**Description**	**Language & Function**
1. Discrete Choice Models (3 Models)			
mnl_B	DCM	Multinomial logit model	Python Biogeme
nl_B	DCM	Nested logit model (see Section 3.4 for how to nest)	Python Biogeme
mxl_B	DCM	Mixed logit model (ASC's as random variables)	Python Biogeme
2. Deep Neural Networks (15 Models)			
mlp_R	DNN	Multi-layer perceptrons (MLP)	R RSNNS mlp
avNNet_R	DNN	Neural network with random seeds with averaged scores; [57]	R Caret avNNet
nnet_R	DNN	Single layer neural network with BFGS algorithm	R Caret nnet
pcaNNet_R	DNN	PCA pretraining before applying neural networks	R Caret pcaNNet
monmlp_R	DNN	MLP with monotone constraints [81]	R Caret monmlp
mlp_W	DNN	MLP with sigmoid hidden neurons and unthresholded linear output neurons	Weka MultilayerPerceptron
DNN_1_30_P	DNN	MLP with one hidden layer and 30 neurons in each layer	Python Tensorflow
DNN_3_30_P	DNN	MLP with three hidden layers and 30 neurons in each layer	Python Tensorflow
DNN_5_30_P	DNN	MLP with five hidden layer and 30 neurons in each layer	Python Tensorflow
DNN_1_100_P	DNN	MLP with one hidden layer and 100 neurons in each layer	Python Tensorflow
DNN_3_100_P	DNN	MLP with three hidden layers and 100 neurons in each layer	Python Tensorflow
DNN_5_100_P	DNN	MLP with five hidden layers and 100 neurons in each layer	Python Tensorflow
DNN_1_200_P	DNN	MLP with one hidden layer and 200 neurons in each layer	Python Tensorflow
DNN_3_200_P	DNN	MLP with three hidden layers and 200 neurons in each layer	Python Tensorflow
DNN_5_200_P	DNN	MLP with five hidden layers and 200 neurons in each layer	Python Tensorflow
3. Discriminant Analysis (12 Models)			
lda_R	DA	Linear discriminant analysis (LDA) model	R Caret lda
lda2_R	DA	LDA tuning the number of components to retain up to #classes - 1	R Caret lda2
lda_P	DA	LDA solved by singular value decomposition without shrinkage	Python sklearn LinearDiscriminantAnalysis
sda_R	DA	LDA with Correlation-Adjusted T (CAT) scores for variable selection	R Caret sda

lda_shrink_P	DA	LDA solved by least squares with automatic shrinkage based on Ledoit-Wolf lemma used.	Python sklearn LinearDiscriminantAnalysis
slda_R	DA	LDA developed based on left-spherically distributed linear scores	R Caret ipred
stepLDA_R	DA	LDA model with forward/backward stepwise feature selection	R Caret stepLDA
pda_R	DA	Penalized discriminant analysis (PDA) with shrinkage penalty coefficients [29]	R mda gen.ridge
mda_R	DA	Mixture discriminant analysis (MDA) where the number subclass is tuned to 3 [30]	R mda
rda_R	DA	Regularized discriminant analysis (RDA) with regularized group covariance matrices [23]	R klaR
hdda_R	DA	High dimensional discriminant analysis (hdda) assuming each class in a Gaussian subspace [7]	R HD
qda_P	DA	Quadratic discriminant analysis (qda)	Python sklearn QuadraticDiscriminantAnalysis
4. Bayesian Models (7 Models)			
naive_bayes_R	BM	Naive Bayes (NB) classifier with the normal kernel density (Laplace correction factor = 2 and Bandwidth Adjustment = 1)	R naivebayes
nb_R	BM	NB classifier with the normal kernel density (Laplace correction factor = 2 and Bandwidth Adjustment = 1)	R Caret nb
BernoulliNB_P	BM	NB model with Bernoulli kernel density function	Python sklearn BermoulliNB
GaussianNB_P	BM	NB model with Gaussian kernel density function (smoothing = 5, according to the variance portions)	Python sklearn GaussianNB
MultinomialNB_P	BM	NB model with multinomially distributed data (smoothing = 1 and learn class prior probabilities)	Python sklearn MultinomialNB
BayesNet_W	BM	Bayes network models by hill climbing algorithm [13]	Weka BayesNet
NaiveBayes_W	BM	NB model with Gaussian kernel density function	Weka NaiveBayes
5. Support Vector Machines (9 Models)			
svmLinear_R	SVM	Support Vector Machine (SVM) model with linear kernel (inverse kernel width = 1)	R Caret kernlab
svmRadial_R	SVM	Support Vector Machine (SVM) model with Gaussian kernel (inverse kernel width = 1)	R Caret kernlab
svmPoly_R	SVM	SVM with polynomial kernel	R Caret kernlab
lssvmRadial_R	SVM	Least Squares SVM model with Gaussian kernel	R Caret kernlab
LinearSVC_P	SVM	SVM with linear kernel and l2 penalty	Python sklearn LinearSVC
SVC_linear_P	SVM	SVM with linear kernel (regularization parameter = 1)	Python sklearn SVC
SVC_poly_P	SVM	SVM with polynomial kernel (regularization parameter = 1)	Python sklearn SVC
SVC_rbf_P	SVM	SVM with radial basis function (rbf) kernel (regularization parameter = 1)	Python sklearn SVC
SVC_sig_P	SVM	SVM with sigmoid function kernel (regularization parameter = 1)	Python sklearn SVC
6. K Nearest Neighbors (4 Models)			

KNN_1_P	KNN	k-nearest neighbors (KNN) classifier with number of neighbors equal to 1	Python sklearn KNeighborsClassifier
KNN_5_P	KNN	KNN classifier with number of neighbors equal to 5	Python sklearn KNeighborsClassifier
lBk_1_W	KNN	KNN classifier with number of neighbors equal to 1 (brute force searching and Euclidean distance) [1]	Weka lBk
lBk_5_W	KNN	KNN classifier with number of neighbors equal to 5 (brute force searching and Euclidean distance) [1]	Weka lBk

7. Decision Tree (14 Models)

rpart_R	DT	Recursive partitioning and regression trees (RPART) model (max depth = 30)	R rpart
rpart2_R	DT	RPART (max depth = 10)	R Caret klaR
C5.0Tree_R	DT	C5.0 decision tree (confidence factor = 0.25)	R Caret C5.0Tree
C5.0Rules_R	DT	Rule-based models using Quinlan's C5.0 algorithm [55]	R Caret C5.0Rules
ctree_R	DT	Conditional inference trees [35]	R Caret ctree
ctree2_R	DT	Conditional inference trees (max depth = 10)	R Caret ctree2
DecisionTree_P	DT	Decision tree classification model with Gini impurity split measure	Python sklearn DecisionTreeClassifier
ExtraTree_P	DT	Tree classifier with best splits and features chosen from random splits and randomly selected features [24]	Python sklearn ExtraTreeClassifier
DecisionStump_W	DT	Tree model with decision stump	Weka DecisionStump
HoeffdingTree_W	DT	An incremental tree with inductive algorithm. [36]	Weka HoeffdingTree
REPTree_W	DT	Tree model using information gain/variance	Weka REPTree
J48_W	DT	Pruned C4.5 decision tree model	Weka J48
Attribute Selected_W	DT	Use J48 trees to classify patterns reduced by attribute selection (Hall, 1998)	Weka AttributeSelected
DecisionTable_W	DT	Simple decision table majority classier that uses BestFirst as search method [39]	Weka DecisionTable

8. Generalized Linear Models (10 Models)

Logistic Regression_l1_P	GLM	Logistic regression model with l1 penalty	Python sklearn LogisticRegression
Logistic Regression_l2_P	GLM	Logistic regression model with l2 penalty	Python sklearn LogisticRegression
Logistic_W	GLM	Logistic regression model with a ridge estimator [42]	Weka Logistic
SimpleLogistic_W	GLM	Linear logistic regression models fitted by using LogitBoost [41]	Weka SimpleLogistic
Ridge_P	GLM	Classifier using Ridge regression	Python sklearn RidgeClassifier
Passive Aggressive_P	GLM	Passive-aggressive algorithms for classification with hinge loss [14]	Python sklearn PassiveAggressiveClassifier
SGD_Hinge_P	GLM	Linear classifier with hinge loss and SGD training	Python sklearn SGDClassifier
SGD_Squared Hinge_P	GLM	Linear classifiers of SGD training with squared hinge loss function	Python sklearn SGDClassifier
SGD_Log_P	GLM	Linear classifiers of SGD training with log loss function	Python sklearn SGDClassifier

SGD_Modified Huber_P	GLM	Linear classifiers of SGD training with modified huber loss function	Python sklearn SGDClassifier

9. Gaussian Process (3 Models)

GP_Constant_P	GP	Gaussian Processes classification model with constant kernel	Python sklearn GaussianProcessClassifier
GP_DotProduct_P	GP	Gaussian Processes classification model with DotProduct kernel	Python sklearn GaussianProcessClassifier
GP_Matern_P	GP	Gaussian Processes classification model with Matern kernel	Python sklearn GaussianProcessClassifier

10. Bagging (3 Models)

Bagging_SVM_P	BAGGING	A bagging classifier that fits base classifiers based on random subsets of the original dataset; SVM is the base classifier	Python sklearn BaggingClassifier
Bagging_Tree_P	BAGGING	A bagging classifier with DecisionTree as the base classifier	Python sklearn BaggingClassifier
Voting_P	BAGGING	A classifier which combine machine learning classifiers and use a majority vote. We use lda_P, LinearSVM and Logistic classifiers here.	Python sklearn VotingClassifier

11. Random Forests (2 Models)

RandomForest_P	RF	A random forest model with 10 trees in the forest	Python sklearn RandomForestClassifier
ExtraTrees_P	RF	A meta estimator that fits 10 ExtraTree classifiers	Python sklearn ExtraTreeClassifier

12. Boosting (23 Models)

AdaBoost_P	BOOSTING	AdaBoost classifier. The DecisionTree with maximum depth =10 is set as the base estimator. [21]	Python sklearn AdaBoostClassifier
AdaBoostM1_W	BOOSTING	Boosting method with DecisionStump as the base classifier	Weka AdaboostM1
AdaBoostM1_R	BOOSTING	Boosting method with DecisionTree as the base classifier	R adabag Adaboost.M1
LogitBoost_R	BOOSTING	Logitboost classification algorithm using decision stumps (one node decision trees) as base learners.	R LogitBoost
Gradient Boosting_P	BOOSTING	An additive model trained in a forward stage-wise fashion [22]	Python sklearn GradientBoostingClassifier
DNN_1_30 AdaBoost_P	BOOSTING	AdaBoosting method with DNN_1_30_P as the base classifier	Python Tensorflow
DNN_3_30 AdaBoost_P	BOOSTING	AdaBoosting method with DNN_3_30_P as the base classifier	Python Tensorflow
DNN_5_30 AdaBoost_P	BOOSTING	AdaBoosting method with DNN_5_30_P as the base classifier	Python Tensorflow
DNN_1_100 AdaBoost_P	BOOSTING	AdaBoosting method with DNN_1_100_P as the base classifier	Python Tensorflow
DNN_3_100 AdaBoost_P	BOOSTING	AdaBoosting method with DNN_3_100_P as the base classifier	Python Tensorflow
DNN_5_100 AdaBoost_P	BOOSTING	AdaBoosting method with DNN_5_100_P as the base classifier	Python Tensorflow
DNN_1_200 AdaBoost_P	BOOSTING	AdaBoosting method with DNN_1_200_P as the base classifier	Python Tensorflow

DNN_3_200 AdaBoost_P	BOOSTING	AdaBoosting method with DNN_3_200_P as the base classifier	Python Tensorflow
DNN_5_200 AdaBoost_P	BOOSTING	AdaBoosting method with DNN_5_200_P as the base classifier	Python Tensorflow
DNN_1_30 GradientBoost_P	BOOSTING	Gradient boosting method with DNN_1_30_P as the base classifier	Python Tensorflow
DNN_3_30 GradientBoost_P	BOOSTING	Gradient boosting method with DNN_3_30_P as the base classifier	Python Tensorflow
DNN_5_30 GradientBoost_P	BOOSTING	Gradient boosting method with DNN_5_30_P as the base classifier	Python Tensorflow
DNN_1_100 GradientBoost_P	BOOSTING	Gradient boosting method with DNN_1_100_P as the base classifier	Python Tensorflow
DNN_3_100 GradientBoost_P	BOOSTING	Gradient boosting method with DNN_3_100_P as the base classifier	Python Tensorflow
DNN_5_100 GradientBoost_P	BOOSTING	Gradient boosting method with DNN_5_100_P as the base classifier	Python Tensorflow
DNN_1_200 GradientBoost_P	BOOSTING	Gradient boosting method with DNN_1_200_P as the base classifier	Python Tensorflow
DNN_3_200 GradientBoost_P	BOOSTING	Gradient boosting method with DNN_3_200_P as the base classifier	Python Tensorflow
DNN_5_200 GradientBoost_P	BOOSTING	Gradient boosting method with DNN_5_200_P as the base classifier	Python Tensorflow

Table 2: List of 105 ML classifiers from 12 model families

The list of classifiers is designed to address the weaknesses in past studies. For example, we acknowledge the diversity of the classifiers by introducing the two-level structure of model families and individual classifiers. We create this list of classifiers based on the meta dataset of the literature review, transcending our own knowledge limitations. The classifiers in Table 2 are implemented in four predominant coding languages: Python, R, Biogeme, and Weka, abbreviated as _P, _R, _B, and _W, attached after the name of each classifier in the first column of Table 2. We intentionally kept some redundancy by using multiple coding languages to train the same model, recognising that differences in coding languages might lead to differences in performance and computation. The redundancy is necessary to minimize unnecessary variations in coding modules.

Unfortunately only a relatively small number of DCMs - three in total - are incorporated. Theoretically, we can further enrich the DCMs. For example, the NL models can be expanded using different nest structures; the MXL models can be expanded by incorporating a flexible correlation matrix between alternatives, modeling the panel structure caused by the same individuals' repeated choices, and perhaps most importantly incorporating heterogeneity in the sensitivities to explanatory variables (as opposed to only the ASCs). However, the limited scope of DCMs in the present experimental design is caused by their much higher computational cost. Within a reasonable amount of time (< 10 hours), the MXL models can be trained when only 1,000 observations are used and only the ASCs are specified to be random. When alternatives are specified to be correlated

or sample sizes become larger, a single MXL model can take days to train.[5] Nonetheless, the three DCMs are repeatedly trained for different sample sizes, behavioral outputs, and datasets, leading to hundreds of experiments and thus still providing broad insights across a variety of contexts.

3.2. Hyper-dimension 2: datasets

The hyper-dimension of datasets includes NHTS2017, LTDS2015, and SGP2017. NHTS2017 refers to the national household travel survey 2017, collected by the Federal Highway Administration[6]. It provides information about daily travel, vehicles, and individuals' and households' characteristics for all the states and specifically the major metropolitan areas in the United State. LTDS2015 refers to the London travel demand survey 2005, a dataset collected in London with trip histories from April 2012 to March 2015 [33]. This dataset was based on the trip diary data from the initial London travel demand survey collected by the local transit agency (Transport for London), and further augmented by adding individual- and mode-specific level-of-service variables (e.g. in-vehicle travel time, public transport fares, fuel cost, etc.) to the historical trip records. As such, LTDS2015 includes the information about individuals' travel mode choices, socio-demographics, and alternative specific variables. SGP2017 was collected in 2017 in Singapore by a stated preference survey [76, 61, 45]. The choice experiment in the survey followed the standard orthogonal survey design [44]. Similar to LTDS2015, SGP2017 includes (stated) mode choices, socio-demographics, and alternative specific variables. Both NHTS2017 and LTDS2015 are publicly available, allowing future studies to work on the same datasets to improve our results.

We chose the three datasets to cover a variety of data collection procedures, geographical locations, and local contexts. The NHTS2017 was collected through revealed preference surveys conducted by the US government, LTDS2015 by combining a transit agency's survey with simulation-based travel information, and SGP2017 by a standard stated preference survey. The three datasets span the continents of America, Europe, and Asia, thus jointly creating a geographically diverse set for experiments. Both LTDS2015 and SGP2017 focus on metropolitan areas, while NHTS2017 covers both metropolitan and rural areas. The diversity of the three datasets can enhance the generalizability of our findings, improving upon the past studies that predominantly rely on only one convenient dataset.

3.3. Hyper-dimension 3: sample sizes

Informed by Table 1, we design the hyper-dimension of sample sizes as 1,000, 10,000, and 100,000. A dataset with large sample size (>100,000) is re-sampled once per sample size to create three different samples to test the impact of sample size on performance.

As the travel surveys have a hierarchical structure of households-individuals-trips, they are re-sampled at different levels depending on the target outputs. For example, after filtering out missing values, the NHTS2017 data contains 781,831 trips by 183,111 individuals from 110,565

[5] This computational difficulty of DCMs will be fully discussed in Section 5.5.

[6] Available at https://nhts.ornl.gov/

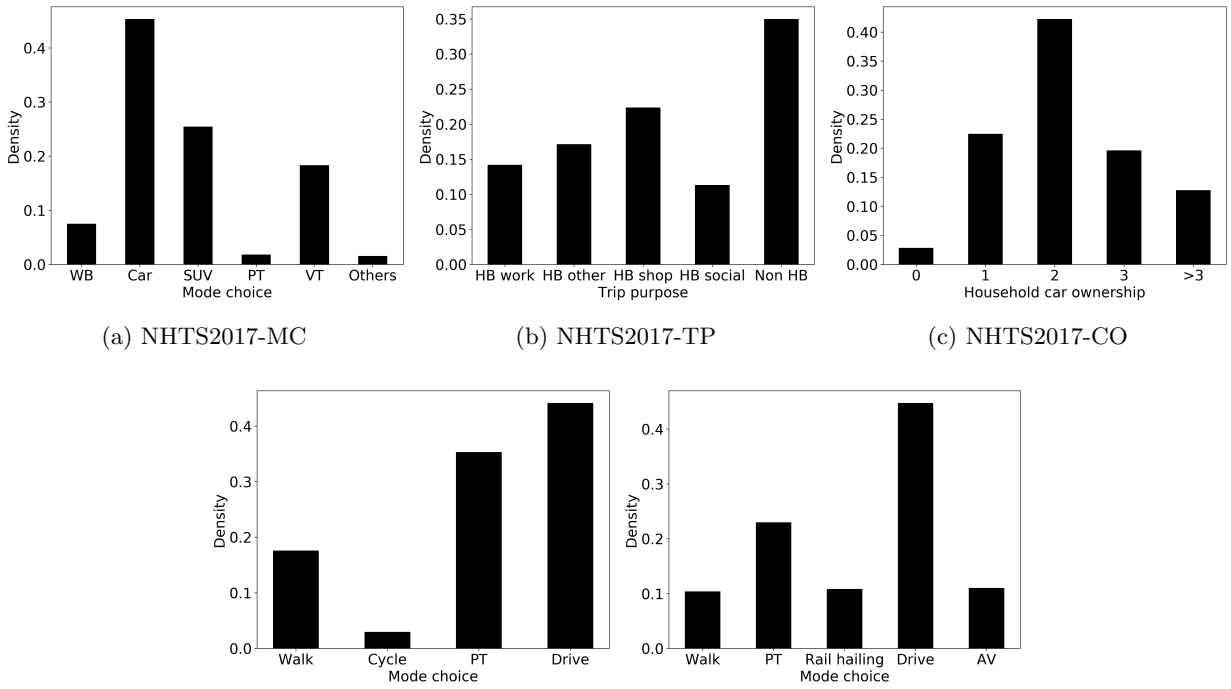
households. Since the choices of travel modes and trip purposes happen at the trip level, the 1,000, 10,000, and 100,000 observations are randomly drawn from 781,831 trips, while for the car ownership prediction, the samples are redrawn from the 110,565 households because car ownership is counted at the household level. Similarly the LTDS2015 dataset has 81,086 trips, so 1,000 and 10,000 samples are randomly drawn from the 81,086 trips for travel mode prediction, and the full dataset represents the 100,000 sample size. The SGP2017 dataset has 11,613 trip-level observations from 2,003 individuals. This dataset is used only for mode choice prediction, so 1,000 and 10,000 samples are randomly drawn from the 11,613 trips, while the 100,000 sample size is unavailable for the SGP2017 dataset.

3.4. Hyper-dimension 4: outputs

The experiments test three travel behavior variables, including travel modes, trip purposes, and car ownership, which are the most commonly used choice dimensions in classical travel demand modeling studies. Travel mode choice is available for all three datasets, while trip purposes and car ownership are only available for the NHTS2017 dataset.

These different dependent variables are summarised using histograms in Figure 1. In the NHTS dataset, travel modes were initially classified into 21 categories, including many special modes such as airplanes and boats. To facilitate the modeling, we aggregate the 21 travel modes into 6 modes by combining similar ones and those with only small proportions. The final six travel modes are (1) walk and bike, (2) car, (3) SUV, (4) van and track, (5) public transit, and (6) others. Cars and SUVs are separate to avoid a dominating alternative, since the two modes jointly account for more than 80% of choices. Some travel mode alternatives may not be available to everyone, but we simplified the experiments by assuming the availability of all the travel modes. The NHTS dataset has five trip purposes, including home-based work, home-based shopping, home-based social, home-based others, and none-home-based trips. The car ownership in the NHTS dataset has five categories, ranging from zero to more than three cars. The LTDS2015 and SGP2017 datasets have four and five travel mode alternatives, respectively.

As shown in Figure 1, in the NHTS2017 dataset, the most common travel mode is car, accounting for 45.3%. More than 80% of trips are taken by motor vehicles (car, SUV, and van and track), and public transit only accounts for 1.8%, which is expected in the US since most people rely on motor vehicles. Since SUV, cars, and van and trucks are similar modes and account for a large proportion, we may expect the accuracy of predicting mode choice in NHTS2017 to be only modestly higher than the largest proportion (i.e. 45.3%). In the LTDS2015 and SGP2017 datasets, the largest mode share is driving, which accounts for 44.1% and 44.7%, respectively. Public transit is the second largest mode share in LTDS2015 and SGP2017 datasets, accounting for 35.3% and 23.0% of the total number of the trips.



(a) NHTS2017-MC (b) NHTS2017-TP (c) NHTS2017-CO

(d) LTDS2015-MC (e) SGP2017-MC

Fig. 1. Output distributions of different datasets

3.5. Evaluation criteria

This study evaluates both predictive performance and computational time. Predictive performance is measured by prediction accuracy with the zero-one loss function[7], because it is the most common metric used in the ML community and potentially the only criterion widely comparable across datasets, outputs, and sample sizes. The prediction accuracy is based on the out-of-sample test, so it addresses the potential overfitting problem caused by overly complex models. Computational time is also a critical metric, since a theoretically feasible classifier with extremely long computational time is highly unlikely to be adopted in practice, particularly in the case of repeated experiments. Our computational time is the sum of training and testing time, measuring the burden on both model estimation and implementation.

Equally important are the metrics this study does **not** use. We decided not to use log-likelihood because this metric relies on the probabilistic nature of a classifier, and unfortunately many machine learning classifiers such as KNN and SVM do not model the discrete outputs in a probabilistic manner. Therefore, although the log-likelihood score is the most widely used metric in choice modeling and has satisfactory statistical properties, it is impossible to use in a large-scale empirical experiment covering hundreds of machine learning classifiers. Other alternatives, such as hinge and quadratic losses, are not commonly used in choice modeling yet. Our large scope inevitably limits the extent to which alternative metrics can be applied, and researchers should investigate them in

[7]The zero-one loss function is defined as $\mathbb{1}\{y_i \neq \hat{y}_i\}$, in which $\hat{y}_i$ is the predicted choice.

future.

3.6. Excluded hyper-dimensions

Many important hyper-dimensions are excluded from our experimental design, including the number of inputs, training algorithms, number of choice alternatives, feature transformation[8], among many others. These hyper-dimensions are not adopted for a variety of reasons. For example, while different numbers of inputs can significantly influence model performance, researchers typically use all the relevant inputs to improve model performance. The question about inputs is often about data availability in practice rather than modeling concerns. Training algorithms also have many possibilities for every classifier, but they are often not a variable to consider in practice, since nearly every classifier already uses some standard training algorithm - for example, it would be deemed unnecessary to attempt to implement Newton methods for DNNs. It is also possible to improve model performance by using complex feature transformations, such as quadratic or polynomial series. However, feature transformation is very difficult to implement since a simple quadratic feature transformation can expand 50 features to at least 100. The 100 inputs would impose too much computational burden on many classifiers, such as DCMs, to the extent that the training cannot be completed. In the end, our experiment is very large-scale, so one more hyper-dimension will lead to the explosion of the total number of experiments.

The alternative hyper-dimensions are excluded partially because of computational limitations, but more importantly, because they might be irrelevant to our core findings. We posit that many potential hyper-dimensions, such as the number of alternatives and feature transformation, might have an impact on the absolute value of predictive performance, but less so on the relative ranking of the classifiers. For example, when the number of alternatives in a choice set increases, the prediction accuracy tends to decrease and the computational time tends to increase for both DCMs and DNNs. However, it is likely that the relative performance of DCMs and DNNs in both prediction and computation varies less than the absolute values. As a result, the substantial computational costs of including more hyper-dimensions might not outweigh the additional benefits. In fact, our postulation that relative ranking is more stable is indeed true, which will be demonstrated in the next few sections.

As a summary, our experiment design has to strike a delicate balance between feasibility and ideal. Although an ideal experiment is to iterate over all possible datasets, inputs, outputs, and classifiers, it would be impossible to complete such an experiment in a single study. In the end, the experiment incorporates the hyper-dimensions as complete as possible, but they are also parsimoniously chosen based on their representativeness and relevance.

[8]Feature transformation refers to the process of transforming and enriching the input variables, e.g. turning the linear input variables to a quadratic series.

4. Experiment setup: data processing and computation

4.1. Data processing

The most relevant features were selected and normalized before the formal modeling process. In the NHTS2017 dataset, the most important 50 features were chosen from the initial 115 by using χ^2 feature selection, which examines the dependence between inputs and outputs by the condensed information. In the LTDS2015 dataset, the best 40 features were chosen from the initial 48 features using the same method. The SGP2017 dataset has only 22 independent variables, so it is not necessary to limit the number of input features. After feature selection, all the input variables were normalized to Gaussian distributions centered on zero with unit variance.

4.2. Computational process

The number of computed experiments is smaller than the theoretically maximum, because of the limitations on data availability, data storage, and computational difficulty. Although the theoretical maximum of experiment points should be $105 \times 3 \times 3 \times 3 = 2,835$, our experiments only examined 1,394 experiment points because of several computational difficulties. Specifically, trip purposes and car ownership are not available in both the LTDS2015 and SGP2017 datasets. For the three GP classifiers, the complexity for storage space is $\mathcal{O}(N^2)$, where N is the sample size, so the GP classifiers with larger than 1K sample sizes can lead to memory errors in a personal computer (PC) with 32G RAM. The MXL model can only be implemented for the 1K sample size, because its training time can exceed 10 hours when the sample size reaches 10K.

The five-fold cross-validation was used to compute the average prediction accuracy in the testing sets for each classifier, leading to a total number of $1,394 \times 5 = 6,970$ experiments. For each experiment, the data was split into the training and testing sets with a ratio of $4:1$. The classifiers were trained on a PC with a single core I9-9900K CPU and 32G RAM. By limiting to the single core and the same hardware, the computational time of the experiment points become comparable, since the hardware variations are removed from the computational process.

5. Results

This section first evaluates the predictive performance by ranking model families and classifiers, discusses how the relative ranking and the absolute prediction accuracy vary with the hyper-dimensions (datasets, sample size, and outputs), and finally evaluates the computational time of the models.

5.1. Hyper-dimension 1: classifiers

5.1.1. Prediction accuracy of 12 model families

Figure 2 visualizes the distribution of the 12 model families' prediction accuracy, sorted from the highest to the lowest according to their average prediction accuracy. The x-axis represents the

model families, and the y-axis represents the prediction accuracy. A white dot for each classifier represents the accuracy of an experiment point, which is a unique combination of a classifier, a dataset, a sample size, and an output, and the grey areas represent the accuracy distributions. The blue, red, and green dots represent the mean, median, and maximum values.

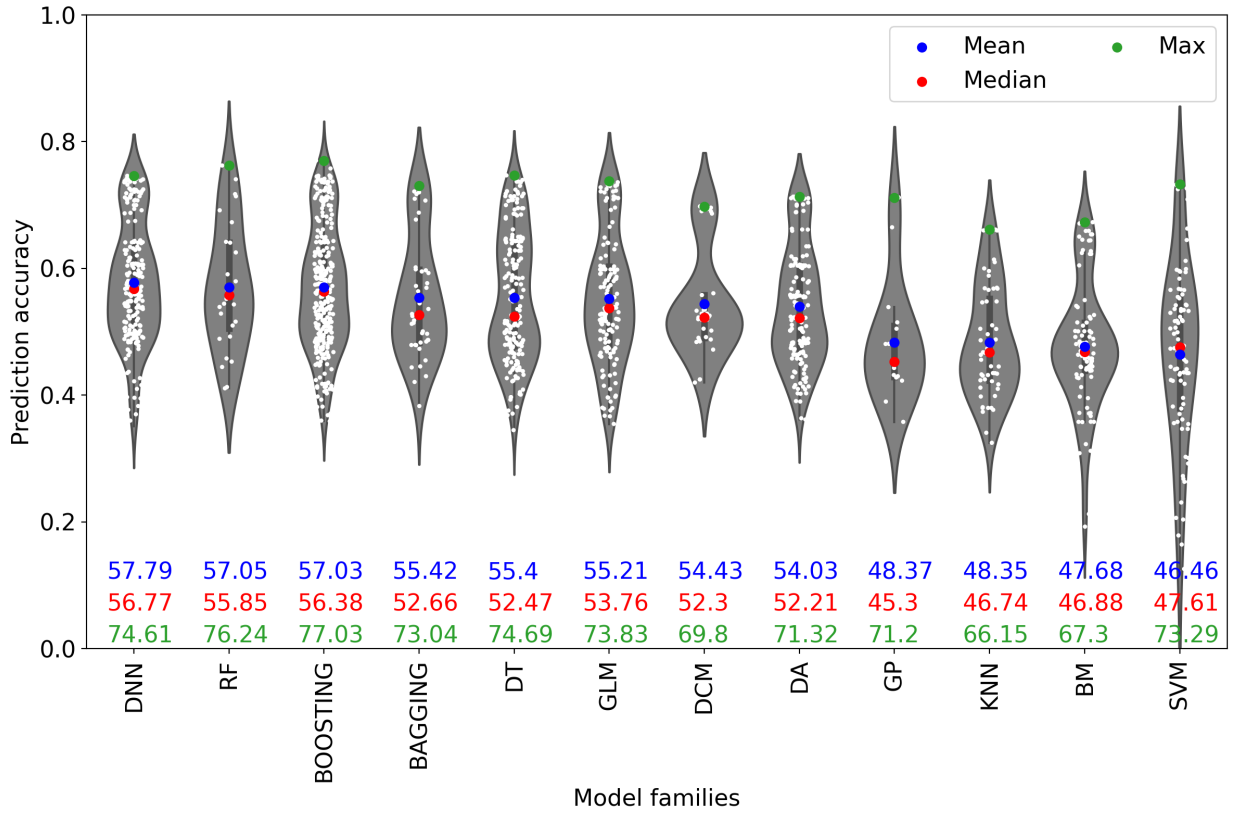


Fig. 2. Distributions of 12 model families' prediction accuracy

The DNN model family on average achieves the highest predictive performance, followed by three ensemble methods (RF, BOOSTING, and BAGGING). The DNN model family, consisting of 15 DNN classifiers, achieves the mean and median accuracy at 57.79% and 56.77%. The three ensemble methods (RF, BOOSTING, and BAGGING) also show high prediction accuracy with the mean (median) values equal to 57.05% (55.85%), 57.03% (56.38%), and 55.42% (52.66%), respectively. The DNNs and the ensemble methods differ modestly in terms of average prediction accuracy, by only around 1-2 percentage points. DNNs can achieve high prediction accuracy, because the DNN model family is known as a universal approximator which can capture any complex relationship between inputs and outputs [34]. The ensemble methods achieve high performance because they are effective for both model approximation and regularization. Ensembling multiple models together can be more effective than individual models in model approximation. Model ensemble also resembles the typical regularization methods, such as dropouts in DNNs [64] and Bayesian prior [8], in reducing the high estimation errors for the models with high model complexity

[6, 27].

The DCM family ranks only 7th out of the 12 model families, showing that DCMs are far from the best predictive model for the travel behavioral analysis. Whereas previous studies limited their scope of analysis to mainly the MNL models, our results can demonstrate the performance of NL and MXL models for a variety of outputs and data sets. The results suggest that DCMs cannot outperform the ensemble models and DNNs in prediction even after incorporating more complex structures of the randomness in the error terms. However, the authors postulate that the performance of DCMs can be improved with richer MXL and NL specifications. Our NL models prespecify the nest structures based on the authors' prior knowledge and the MXL models only naively impose randomness on the alternative-specific constants. Had these structural limitations been removed, the DCMs' performance could have been further improved. Unfortunately, these structural limitations are hard to be removed in this work, because the DCMs are computationally inefficient with large sample sizes, which hinders our intention of designing a richer set of DCMs for comparison and leads to a limited set of experiments on the DCM side.

Although our observations above hold on average, the large variation in the distributions of the prediction accuracy in Figure 2 might lead to different implications. First, the large variation suggests that the absolute values of prediction accuracy can be highly unstable, largely depending on the context. For example, the average accuracy of the DNN family is about 57.79%, while its maximum prediction accuracy is around 74.61%. Similarly, the average accuracy of the DCM family is 54.43%, which is about 15.37% lower than the maximum value (69.8%). In fact, every model family can achieve around 70-80% as the maximum values, about 20-30 percentage points higher than the average ones. Second, the small gaps between the average prediction values suggest that the ranking of models can vary with contexts. For example, it seems plausible that certain DCMs can outperform some DNNs, particularly when the architectures and hyper-parameters of DNNs are configured poorly. In fact, the small gap in the average prediction accuracy and the large variations suggest that, if a statistical test were used, it would be difficult to reject the null hypothesis that the DNNs and other model families (e.g. DCMs) have the same levels of predictive performance. Lastly, the distributions of the prediction accuracy for the model families often present a multi-modal pattern. This is caused by the different levels of predictability for each data set. This observation points out the importance of our framework that incorporates multiple hyper-dimensions, only through which can the large variations be revealed. It also suggests the importance of further decomposing the model families into individual classifiers for specific data sets, outputs, and sample sizes for comparison, which will be examined in the next few subsections.

5.1.2. Prediction accuracy of 105 classifiers

Figure 3 expands the 12 model families to 105 classifiers, sorted by the average prediction accuracy from the highest to the lowest, with the blue dots representing the mean values, red the median values, and the blue bars the variances. This expansion can reduce the large variations in model families and illustrate a subtler pattern for individual classifiers.

As shown in Figure 3, the top 2 classifiers are the LogitBoosting_R and GradientBoosting_P from the BOOSTING model family, followed by many DNNs. For example, the nnet_R, avNNet_R, and monmlp_R are ranked as 4th, 5th, and 10th out of 105 classifiers. The DNN architecture with medium width and depth (DNN_200_3; 200 neurons, 3 hidden layers) presents the best performance, and the DNN_100_3 (100 neurons, 3 hidden layers) ranks as the second. These findings are largely intuitive. The high performance of the boosting methods demonstrate the importance of combining classifiers to achieve high performance, rather than choosing one over the other. The performance of the individual DNNs illustrates a non-monotonic relationship between predictive performance and DNN complexity. An overly complex DNN model can underperform owing to the high estimation error, and a overly simplifed DNN model can also underperform owing to the high approximation error [77]. Hence a DNN architecture with medium complexity achieves the best performance. But interestingly, the combination of the boosting methods and DNNs (DNN_GradientBoost_P and DNN_AdaBoost_P) fail to outperform the single DNNs. This may be because boosting methods are effective in combining under-parameterized weak classifiers, but less so for the over-parameterized DNNs.

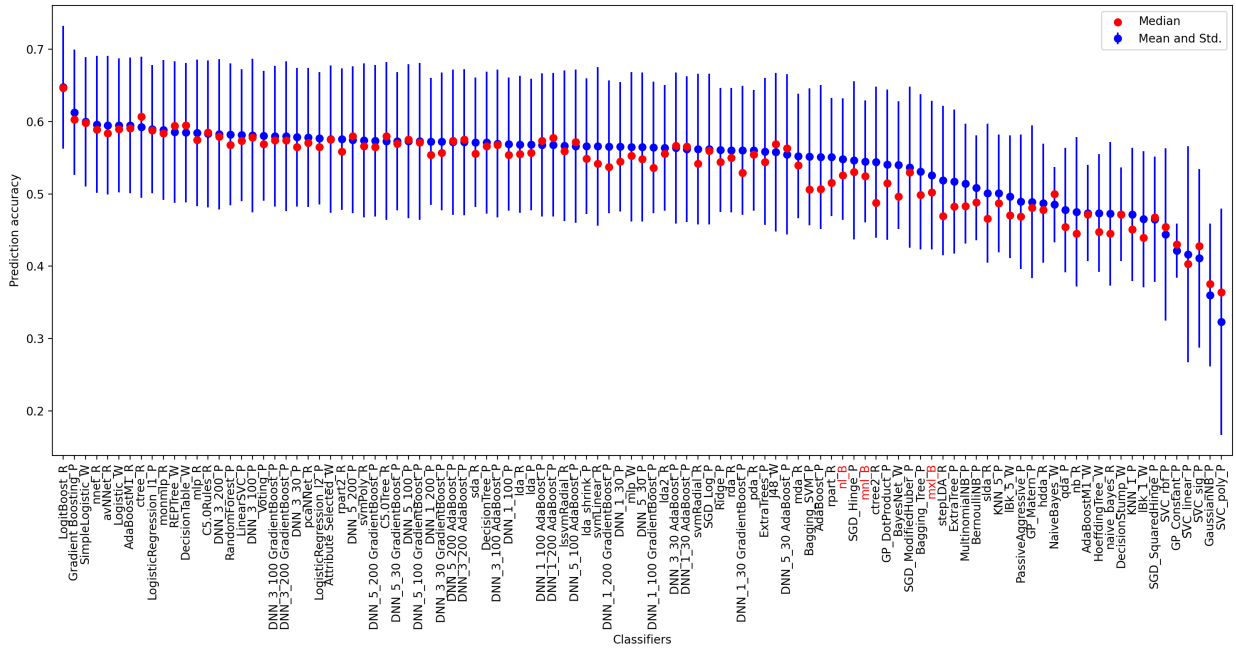


Fig. 3. Prediction accuracy of 105 ML classifiers

MNL, NL, and MXL in the DCM model family (highlighted in red) perform at the medium to lower end of the 105 classifiers, with the NL outperforming MNL and MXL. The average prediction accuracy of MNL, NL, and MXL are 54.50%, 55.01%, and 52.58%, which have a very small deviation from their average value, suggesting that individual DCMs cannot significantly improve their average performance. Within the three models, the NL model outperforms MNL since the NL model captures the correlation of alternatives within the same nest. The NL model outperforms

the MXL model because our MXL model does not incorporate the full covariance matrix for the random error terms. Therefore, the relatively poor performance of the MXL model is attributed more to its computational complexity rather than the model structure. The MXL model cannot converge within a reasonable amount of time ($<$ 10 hours) when the sample size reaches 10,000 or the full covariance matrix is used. Since larger sample size is typically associated with better performance (Section 3.3), the MXL models cannot show high predictive performance when they cannot be trained for the large sample sizes.

Using top-N classifier as another evaluation metric, Figure 4 also demonstrates that the DNNs and the ensemble methods achieve the highest predictive performance, similar to the findings above. The top-N classifier measures the chance of a classifier being among the N classifiers with the highest performance. Figure 4 ranks the classifiers from the highest to the lowest top-N value along the x-axis, based on the $N = 20$ scenario. As shown in Figure 4, LogitBoost_R and GradientBoosting_P perform as the best two classifiers, and the nnet and avNNet are ranked as the fourth and sixth. The three DCMs do not belong to any of the top-N models.

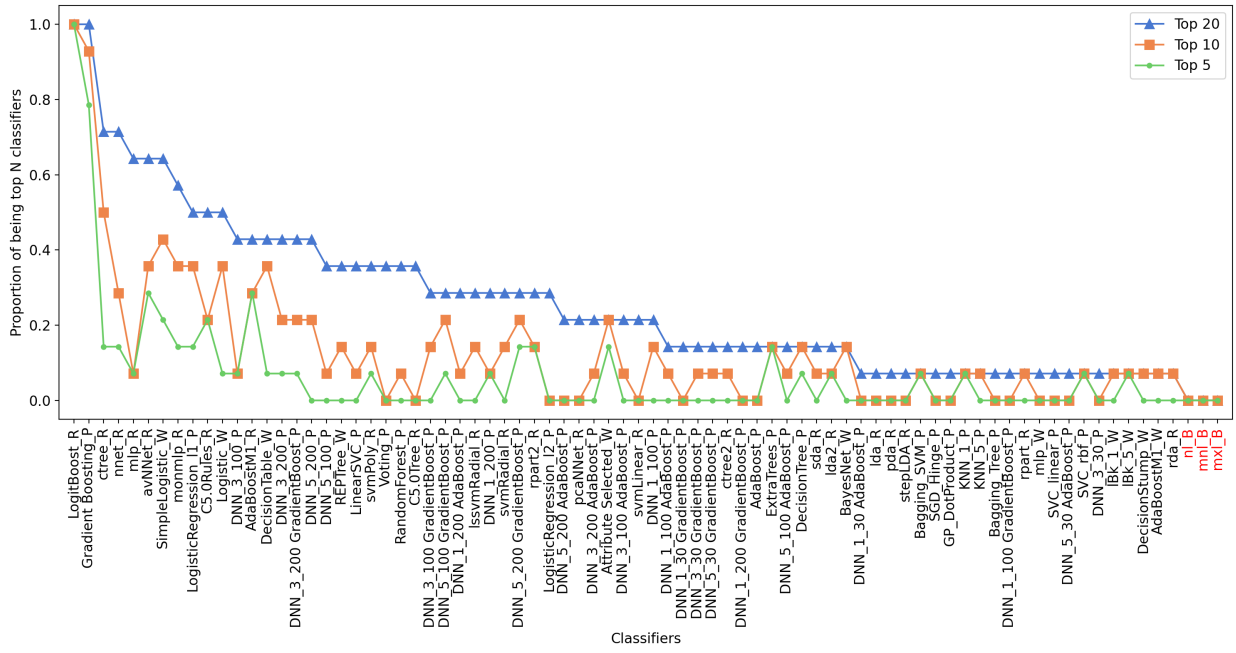


Fig. 4. Proportion of being top N classifiers

5.2. Hyper-dimension 2. datasets

The classifiers are further compared for each data set, given the large variations and the multi-modal patterns illustrated in Figure 2. The prediction accuracy and ranking of the top 10 classifiers and the three DCMs are presented in Table 3. The prediction accuracy in Table 3 represents the average of the five-fold cross-validation for three sample sizes. The accuracy distributions of the three datasets are visualized in Figure 5. The data-specific analysis yields the following four major

findings.

Table 3: Top 10 classifiers and DCMs for three datasets

NHTS2017-MC			LTDS2015-MC			SGP2017-MC		
Rank	Model	Accuracy	Rank	Model	Accuracy	Rank	Model	Accuracy
1	LogitBoost_R	0.5443	1	LogitBoost_R	0.7658	1	LogitBoost_R	0.5827
2	Gradient Boosting_P	0.5130	2	Gradient Boosting_P	0.7431	2	avNNet_R	0.5564
3	SimpleLogistic_W	0.5001	3	avNNet_R	0.7386	3	nnet_R	0.5552
4	LogisticRegression_l1_P	0.4919	4	nnet_R	0.7351	4	Gradient Boosting_P	0.5550
5	Logistic_W	0.4904	5	SimpleLogistic_W	0.7350	5	DNN_1_200_P	0.5487
6	ctree_R	0.4881	6	Logistic_W	0.7339	6	DNN_1_100 AdaBoost_P	0.5484
7	DecisionTable_W	0.4861	7	svmPoly_R	0.7329	7	DNN_3_100 AdaBoost_P	0.5481
8	monmlp_R	0.4834	8	lssvmRadial_R	0.7320	8	DNN_3_100_P	0.5480
9	LinearSVC_P	0.4822	9	monmlp_R	0.7317	9	DNN_1_100_P	0.5474
10	REPTree_W	0.4808	10	LogisticRegression_l1_P	0.7312	10	DNN_5_200 AdaBoost_P	0.5474
36	nl_B	0.4674	71	mxl_B	0.6980	34	nl_B	0.5270
41	mnl_B	0.4657	76	nl_B	0.6933	35	mnl_B	0.5262
94	mxl_B	0.4250	77	mnl_B	0.6917	44	mxl_B	0.5200

First, predictability varies dramatically across datasets. The highest accuracy is achieved in the LTDS2015 (76.58%), which is around 20 percentage points higher than the NHTS2017 and the SGP2017 datasets. This gap of around 20 percentage points is consistent across the top-10 classifiers. Second, although the specific prediction values vary with datasets, the ensemble methods and DNNs still deliver the highest performance. LogitBoost shows a dominant performance in all three datasets; GradientBoosting ranks as the second in two datasets; the DNNs, such as avNNet, nnet, and DNN_200_1_python, are among the top 5 classifiers in the LTDS2015 and SGP2017 datasets. Third, the DCMs still perform at the medium to low end of the performance distributions in this data-specific analysis. The MNL, MXL, and NL models rank around 75th in the LTDS2015 and around 40th in the SGP2017. In the NHTS2017 dataset, the MXL model performs significantly worse than the MNL and NL models, because of the limited scope of the MXL models' training. The DCMs have the highest ranking in the SGP2017 dataset, probably because this dataset is designed as the standard orthogonal stated preference survey with all the alternative-specific variables included. Lastly, decision trees and some logistic regressions can predict with relatively high accuracy in both NHTS2017 and LTDS2015 datasets. It may be because many input variables in the NHTS2017 are discrete variables measuring individuals' characteristics and the discrete inputs may fit the tree-based methods better than DNNs. As a short summary, the data-specific analysis demonstrates the robustness of our previous findings. Although the absolute values of the prediction accuracy vary with datasets, the relative ranking of the key models is highly consistent.

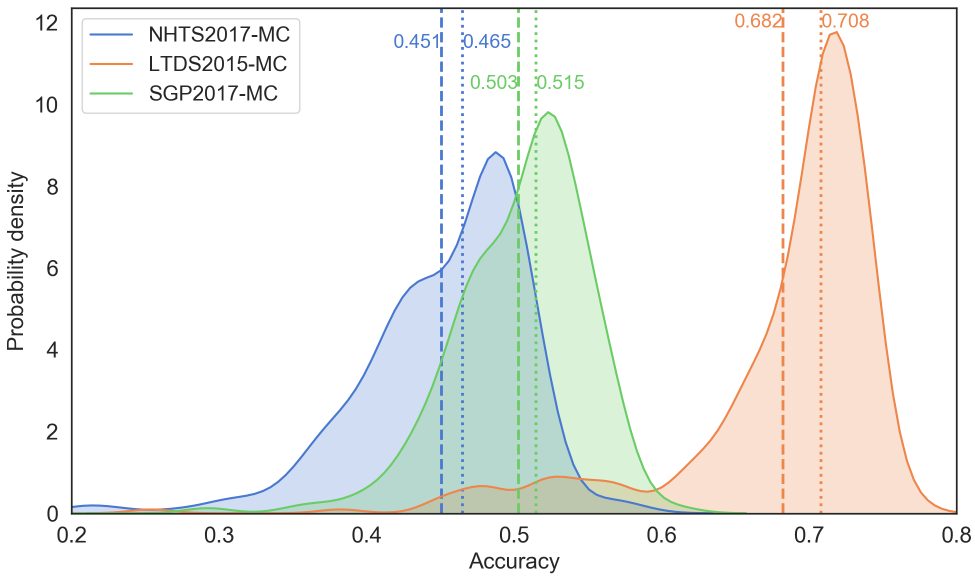


Fig. 5. Prediction accuracy distributions by datasets (dashed lines are mean values and dotted lines are median values)

5.3. Hyper-dimension 3. sample sizes

Table 4 summarizes the prediction accuracy and the relative ranking of the top 10 and three highlighted classifiers for the three sample sizes. Figure 6 and 7 visualize how the distribution of prediction accuracy varies with sample size, datasets, and outputs. As a high-level summary, Table 4 demonstrates several patterns similar to the discussions above. The ensemble methods (e.g. LogitBoost and GradientBoosting) perform the best among all the classifiers and the DNN models (e.g. nnet, avNNet) achieve relatively high performance. The MNL and NL models still fail to provide high-quality performance, as they rank at around 50-70th for each sample size. The absolute prediction values vary with sample size, while the relative ranking of the models is more stable. Decision tree methods (e.g., ctree and REPTree_W) perform well in large sample size, and SVM models (e.g., svmPoly, lssvmRadial) show decent performance in small sample size. Besides these patterns already discussed above, we would like to point out two new observations.

Table 4: Top 10 and three highlighted classifiers for three sample sizes

1k			10k			100k		
Rank	Model	Accuracy	Rank	Model	Accuracy	Rank	Model	Accuracy
1	LogitBoost_R	0.6244	1	LogitBoost_R	0.6476	1	LogitBoost_R	0.6760
2	Gradient Boosting_P	0.5874	2	Gradient Boosting_P	0.6205	2	Gradient Boosting_P	0.6353
3	AdaBoostM1_R	0.5772	3	avNNet_R	0.6082	3	C5.0Rules_R	0.6303
4	SimpleLogistic_W	0.5752	4	monmlp_R	0.6070	4	ctree_R	0.6295
5	svmPoly_R	0.5741	5	nnet_R	0.6068	5	DNN_5_200_P	0.6261
6	nnet_R	0.5724	6	Logistic_W	0.6054	6	monmlp_R	0.6257
7	avNNet_R	0.5708	7	SimpleLogistic_W	0.6049	7	DNN_3_200_P	0.6253
8	lssvmRadial_R	0.5666	8	DNN_3_200_P	0.6001	8	Logistic_W	0.6248
9	LogisticRegression_l1_P	0.5664	9	DNN_3_100_P	0.5974	9	REPTree_W	0.6244
10	svmLinear_R	0.5658	10	ctree_R	0.5972	10	DNN_5_100_P	0.6244
50	mnl_B	0.5270	14	DNN_5_200_P	0.5943	70	nl_B	0.5683
52	nl_B	0.5256	64	nl_B	0.5600	73	mnl_B	0.5584
57	DNN_5_200_P	0.5144	67	mnl_B	0.5523	-	-	-

The relative ranking of the DCMs and DNNs demonstrates an opposite trend, as the ranking of DNNs increases with sample size, while the DCMs' ranking decreases. This pattern is exemplified by the comparison between the three models: MNL_B, NL_B, and DNN_200_5_P. When the sample size increases from 1,000 to 10,000 and 100,000, the relative ranking of the MNL_B decreases from 50th, to 67th and 73rd, and the relative ranking of the NL_B also decreases from 51st to 67th and 73th. On the other hand, the prediction accuracy of the DNN_200_5_P model is about 51.4% when the sample size is 1,000, and its ranking increases significantly from 57th, to 14th and 5th as the sample size increases. This opposite trend between the DCMs and DNNs is caused by their different model complexity [69, 68, 70, 71]. The estimation error of simple models such as MNL converges quickly when sample size is relatively small, whereas that of complex models such as DNNs converges slowly with sample size [77]. In other words, the complex models such as DNNs tend to outperform simple models when sample size is large.

The prediction accuracy increases with sample size, although the increasing pattern seems to have a marginally decreasing shape. When the sample size is 1,000, the mean (median) prediction accuracy is 51.63% (50.20%), and it increases to about 55.29% (55.49%) and 57.42% (59.17%) as the sample size becomes 10,000 and 100,000, as shown in Figure 6. Therefore, the increasing trend seems to have a slightly concave shape, as the accuracy increment from 1,000 to 10,000 is larger than that from 10,000 to 100,000. This pattern of marginally diminishing improvement is consistent across the datasets and outputs, as shown in the five subfigures in Figure 7. For example in Figure 7a, the mean accuracy increment between 1K and 10K observations is around 5.38 percentage points (46.62-41.24), while that between 10K and 100K is only about 1.13 percentage points (47.75-46.62). This marginally decreasing accuracy suggests that a larger magnitude of the sample size contributes to the prediction accuracy less effectively as the sample size grows.

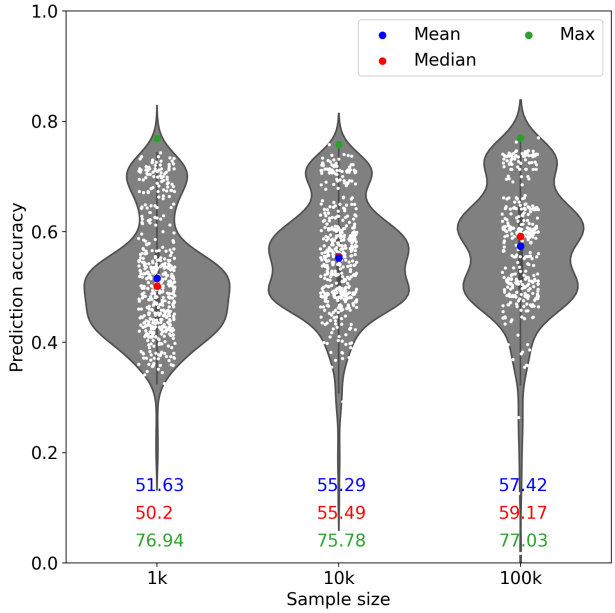
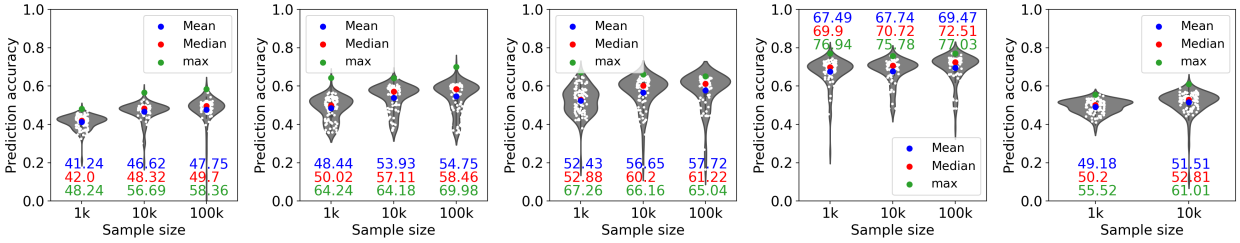


Fig. 6. Prediction accuracy by sample sizes



(a) NHTS2017-MC (b) NHTS2017-TP (c) NHTS2017-CO (d) LTDS2015-MC (e) SGP2017-MC

Fig. 7. Prediction accuracy by sample size, datasets, and outputs

5.4. Hyper-dimension 4. outputs

The predictive performance of the three outputs and the example classifiers is presented in Table 5. The first column represents the performance of predicting travel mode choices, and the second and third represent that of trip purposes and car ownership in the NHTS2017 dataset.

Table 5: Top 10 classifiers for predicting three outputs

MC (NHTS2017+LTDS2015+SGP2017)			NHTS2017-TP			NHTS2017-CO		
Rank	Model	Accuracy	Rank	Model	Accuracy	Rank	Model	Accuracy
1	LogitBoost_R	0.6369	1	LogitBoost_R	0.6613	1	LogitBoost_R	0.6615
2	Gradient Boosting_P	0.6098	2	Gradient Boosting_P	0.5901	2	rpart2_R	0.6464
3	SimpleLogistic_W	0.5953	3	LogisticRegression_l1_P	0.5859	3	AdaBoostM1_R	0.6464
4	avNNet_R	0.5926	4	ctree_R	0.5825	4	Attribute Selected_W	0.6459
5	nnet_R	0.5921	5	mlp_R	0.5809	5	DecisionTable_W	0.6450
6	Logistic_W	0.5920	6	SimpleLogistic_W	0.5802	6	Gradient Boosting_P	0.6440
7	LogisticRegression_l1_P	0.5894	7	avNNet_R	0.5774	7	ctree_R	0.6400
8	monmlp_R	0.5882	8	Logistic_W	0.5771	8	REPTree_W	0.6380
9	AdaBoostM1_R	0.5834	9	LogisticRegression_l2_P	0.5768	9	SimpleLogistic_W	0.6315
10	LinearSVC_P	0.5832	10	Voting_P	0.5762	10	nnet_R	0.6301
52	nl_B	0.5670	52	nl_B	0.5473	81	mnl_B	0.5124
57	mnl_B	0.5656	69	mnl_B	0.5228	82	nl_B	0.5078
73	mxl_B	0.5477	74	MXL_B	0.5020	86	mxl_B	0.4840

This decomposition of the model families into the three outputs again demonstrate the robustness of our previous findings. LogitBoost and GradientBoosting can provide the highest predictive performance, as they rank the top two for the travel mode choice and trip purpose predictions. DNN models, such as avNNet and nnet, enter the list of the top-10 classifiers for all three outputs. DTs and some GLM (e.g. logistic regressions) perform with high quality, which is probably caused by the discrete nature of the inputs facilitating the tree-based regressions. DCMs perform at the medium and lower end of the performance distributions, ranking at around 50th-80th for the three outputs. Within the three DCMs, MXL models perform with the lowest prediction accuracy, caused by the limited scope of our experiments on the MXL models. In addition to the specifics, the top classifiers have modest differences across the three outputs, similar to our initial postulation. The top classifiers can achieve 58-63% for mode choice prediction, 57-66% for trip purpose prediction, and 63-66% for car ownership prediction.

It is intriguing to observe the different performance between the GLMs and DCMs, since their model structures are similar and the DCMs can be treated as one type of generalized linear models. This difference can be attributed to both model specification and regularization. DCMs are often sparsely specified to guarantee parameter identification, which might somewhat sacrifice model prediction. DCMs are implemented in Biogeme, through which the authors cannot easily implement L1 or L2 regularizations to further improve the model performance, while the GLMs have a high ranking potentially owing to the regularizations (LogisticRegression_l1 and LogisticRegression_l2). Therefore, the relatively low predictive performance of DCMs may not be caused by the simple linear model structure, but the lack of regularizations in the current mainstream DCMs' implementation.

5.5. Evaluating computational cost

Models are not practical if they entail too much computational cost. Even moderately long computational time can hinder practical use, because models often need to be repeatedly adjusted and trained in practice. To facilitate this discussion, Figures 8, 9, and 10 visualize how computational

time varies across model families, individual classifiers, and sample size. The formats of the figures are similar to the Figures above.

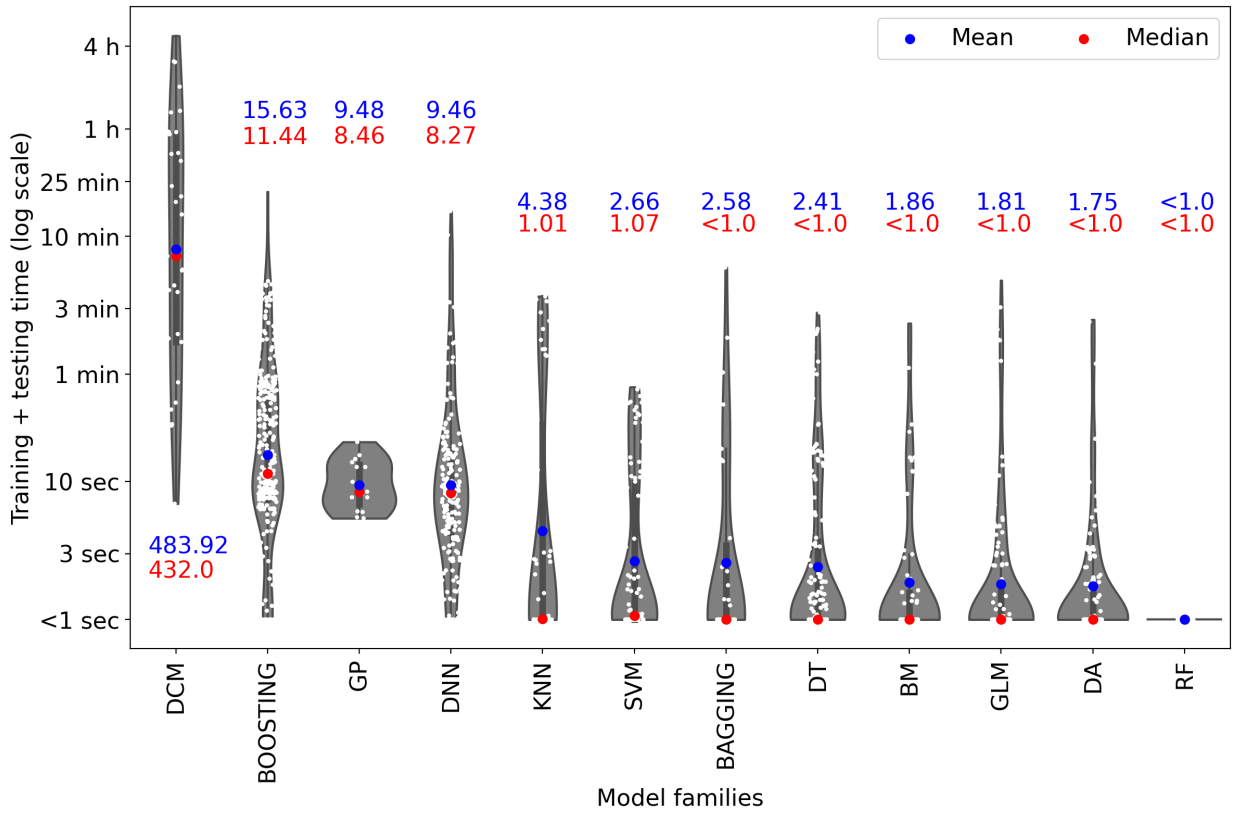


Fig. 8. Running time distribution of 12 model families (blue texts are means and red medians. The units are in seconds, same for the following graphs)

The most salient result is the longest computational time of the DCM family, as shown in Figure 8. On average, the DCM family takes 483 seconds (around 8 min) to train, which is more than 30 times longer than the second worst model family (Boosting). Many DCMs can take more than even 1 hour to train, and in fact, the full picture is even worse than Figure 8. Since many MXL models cannot be computed within 10 hours, the authors did not report the computational time of these models. Had all the MXL models been completed and incorporated into Figure 8, we would expect an even longer computational time for the DCM family. This exceedingly large computational cost of the DCMs can be attributed to at least four factors. First, the typical DCM training uses the Newton or quasi-Newton family methods, which involve the inversion of large matrices. Even the most efficient quasi-Newton method (e.g. BFGS) takes a computational time quadratic to the input dimensions ($\mathcal{O}(D^2)$), as opposed to the linear complexity ($\mathcal{O}(D)$) of the first-order methods in ML classifiers. Therefore, when the dimension of inputs is high, such as 250 parameters in the NHTS dataset, the computational burden is 250 times larger in the DCMs than the ML classifiers for each step in training. However, it is worth noting that this burden

of computing the Hessian matrix is a necessary step of computing the covariance matrix for the parameters in DCMs. Only after obtaining the covariance matrix, can the researchers conduct statistical tests to understand the statistical significance of the findings. Second, a typical DCM training uses all the observations to compute the Hessian or the gradients of the loss function regarding the parameters in every step, while many ML classifiers - such as DNNs - rely on only a sub-sample for each step using the algorithm of stochastic gradient descent (SGD). When the sample size reaches 100K, this difference in computational cost is large. Third, it is very slow to use simulation-based estimation in optimizing MXLs. The MXL model for just 1,000 samples can cost a lot of time ($>$ 4 hours) to train, owing to the slow convergence in the Monte Carlo drawing-based likelihood function evaluation. Lastly, this computational difficulty can be specific to the Python Biogeme module. Python Biogeme leverages the optimization algorithm in Python `scipy` package, which could be inefficient for large sample sizes with high input dimensions.

Among the ML classifiers, the ensemble methods (Boosting and bagging) and the DNNs take relatively long computational time, while the RF achieves a reasonable balance between prediction and computation. The boosting methods use sequential updating for multiple classifiers, thus costing long computational time. DNNs use the back propagation for gradient descent, which can cost a long time partially due to their complex model structures. The RFs rank as the most computationally efficient in Figure 8 and the second best in prediction in Figure 2. The RFs even have a small variation in their computational time, consistently taking less than 1 second to train for a very large sample size (100,000).

The findings above hold for not only model families but also individual classifiers, as shown in Figure 9, which decomposes the 12 model families into 105 classifiers. The three DCMs take the 1st (MXL), 2nd (NL), and 4th (MNL) longest computational time, and the MXL model is significantly more inefficient than the other two and the rest 104 classifiers. The MNL and NL are relatively more computational efficient than the MXL model, because the former two use the closed-form analytical form for the gradients of their log-likelihood functions, while the latter has to rely on numerical simulation. The ML classifiers with high predictive performance, including GradientBoosting, LogitBoost, and DNNs, take a relatively long time to train.

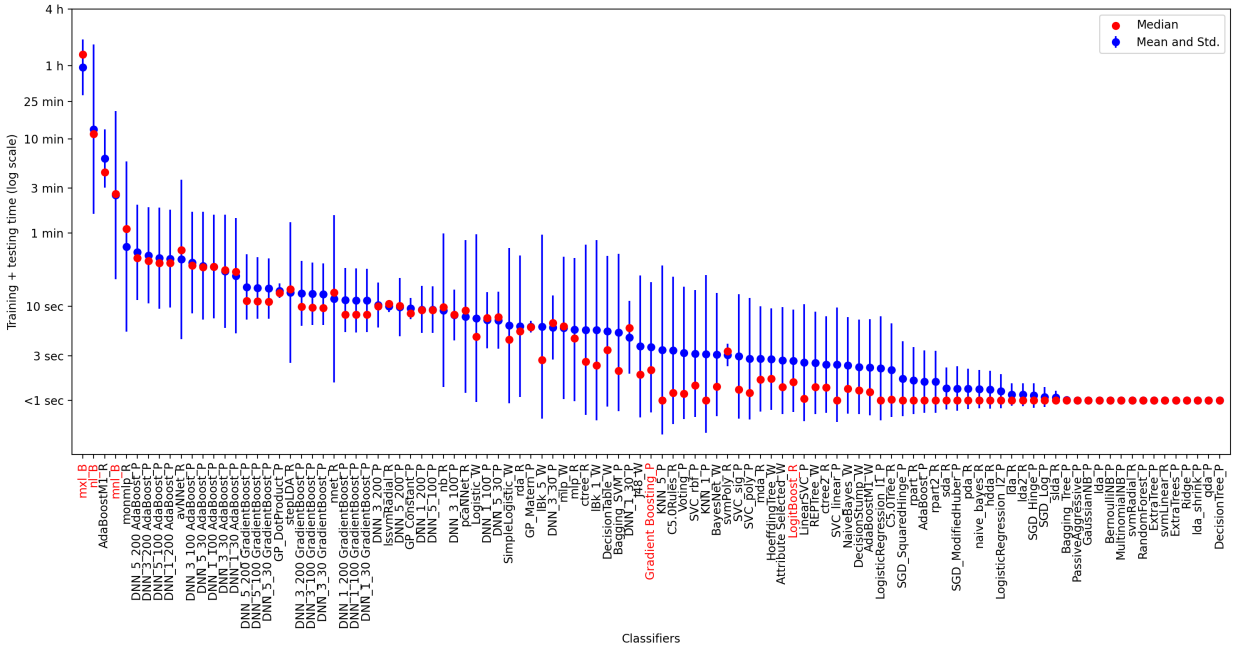


Fig. 9. Running time of 105 ML classifiers

Computational burden increases with sample size, and the marginal cost in computation seems to outweigh the marginal benefit in prediction. Larger sample size leads to longer computational time, as shown by the visualization in Figure 10. The median computational time for the three sample sizes are 2.39, 3.57, and 19.36 seconds, which approximately represents a linear relationship under a log scale. This marginally increasing computational time contrasts with the marginally decreasing predictive performance with respect to sample size. As discussed in Section 5.3, the increment of the predictive performance decreases as sample size becomes larger, while that of the computational burden increases. This contrast suggests that the ML classifiers, although often discussed in the big data context, may not be the most effective in an infinitely large sample size.

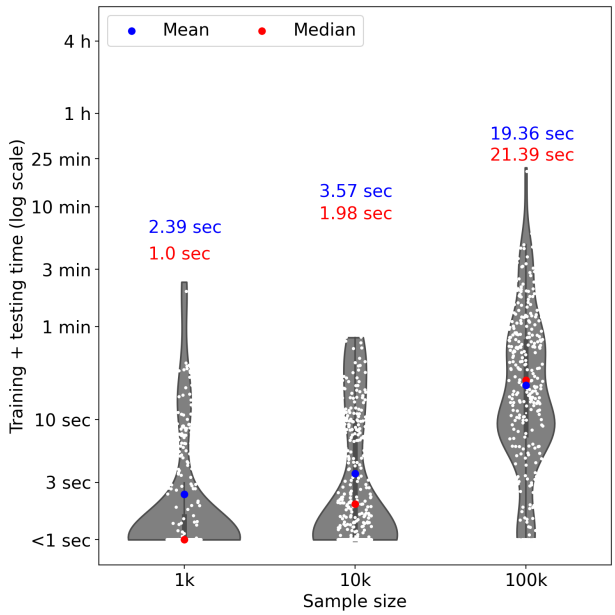


Fig. 10. Running time distribution for different sample sizes

Although we treat computational cost and predictive performance as two evaluation metrics, they are inextricably interwoven in practice, because a low computational cost enables researchers to conduct more experiments and thus present better prediction results. In fact, this problem is already revealed in this present work, as the computational inefficiency of DCMs limits the scope of the corresponding experiment design. Since the computational cost of DCMs is quadratic to the input dimensions, the authors cannot effectively enrich the input features of the DCMs to boost the performance. Since the computational cost of MXLs become tremendous when sample size is large, the authors cannot train the MXL models for larger sample sizes to reveal the potentially higher predictive performance. This problem is not specific to our work, but generally exists for all the practices. Currently, as the typical sample sizes in travel demand modeling are around 10^3-10^4 and the typical input dimensions are often smaller than 20 (Table 1), this computational issue is not salient in common practice yet. However, the computational inefficiency of DCMs could become a much more pronounced limitation in future under the big data context, when both the input dimension and the sample size become much larger.

6. Comparing to meta dataset

The results from our large-scale experiments are comparable to the meta dataset, since both are organized around the concepts of hyper-dimensions, experiment points, and experiment space. The experiments and meta dataset are complementary, because the large-scale experiments are richer in the number and structure of the classifiers, while the meta dataset prevails in terms of the diversity in contexts and datasets. Figures 11 and 12 visualize the histogram of the best models and the

prediction accuracy of the models in our meta dataset. Unfortunately the past studies have never reported their computational time, so we cannot discuss the computational aspect here.

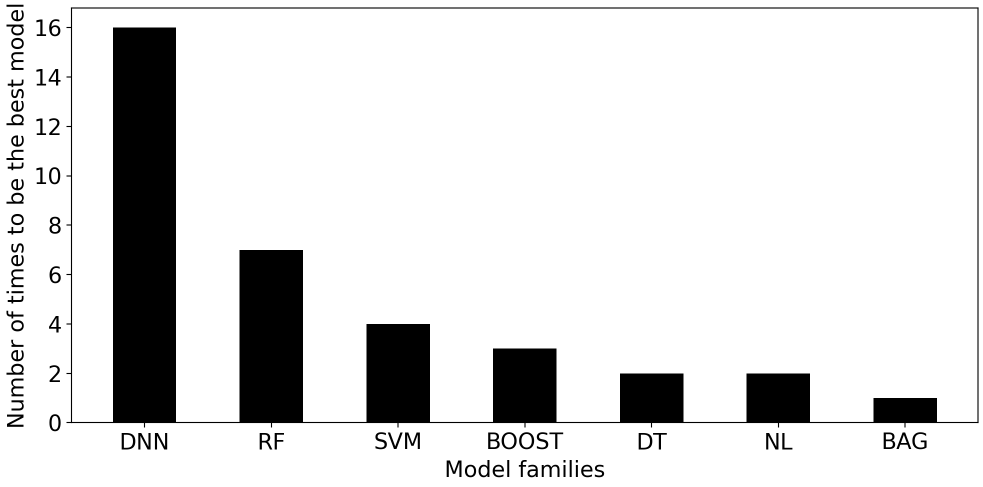


Fig. 11. Histogram of the best models from the meta dataset

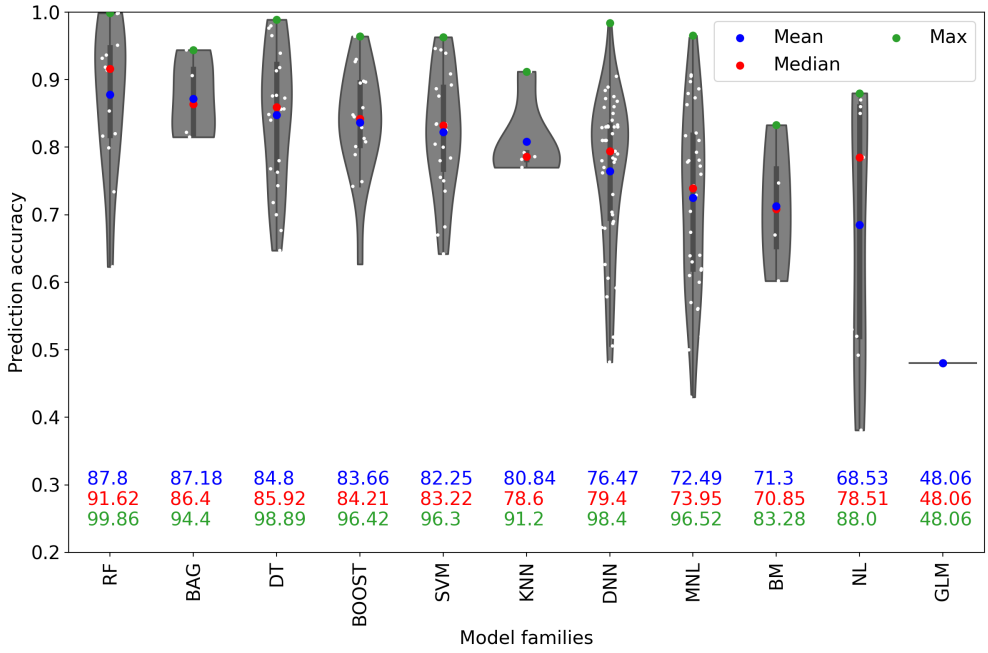


Fig. 12. Prediction accuracy of the examined models from the meta dataset

The meta dataset confirms our main findings that the ensemble methods and the DNNs have the highest predictive performance, while the DCMs rank at the medium or lower ends of the predictive distributions. As shown in Figure 11, 16 out of the 35 studies found that DNNs achieve the highest performance and 11 of them found that the ensemble methods (RF, BOOSTING, and BAGGING) outperform others. The ensemble methods achieve the highest prediction accuracy

at 87.8% by RFs, 87.18% by BAGGING, and 83.66% by BOOSTING, and the accuracy of the three ensemble families can be as high as nearly 99.86%, as shown in Figure 12. DNNs' average prediction accuracy is around 76.47%, which can be as high as 98.4%. The MNL and NL achieve about 72.49% and 68.53% on average, which are about 4-8 percentage points lower than the DNNs and about 15-20 percentage points lower than the ensemble methods.

The meta dataset differs from our large-scale experiments in terms of the absolute values of the prediction accuracy, although this difference does not conflict with our findings. The average prediction accuracy of the classifiers in the meta dataset ranges between 75 and 85 percentage points, which are higher than the range of 50-60 percentage points found in our experiments. This gap is reasonable since we did not seek to carefully adjust each classifier in our experiment to achieve the highest performance, which is computationally difficult given the large scope of our experiment. This gap is also partially explained in our experiments by the large variations in each model family. Since the classifiers have a large variation in performance, it is highly likely to achieve 75-85% accuracy with careful model tuning. In this meta dataset, the accuracy difference between the DNNs and DCMs is around 3-4 percentage points on average, which are similar to our findings, while the difference between the ensemble methods and DCMs is larger at about 15-20 percentage points, which are much larger than our experiments' results. Again, the absolute values of prediction accuracy are highly context-specific and have high variations, while the relative ranking is much more stable and consistent across our experiment and the meta-dataset.

7. Conclusions

Although many past studies have compared the predictive performance of DCMs and ML classifiers, they are limited by the particularities of data, contexts, and authors' expertise. To address this challenge, this study starts with the conceptualization of experiment space, experiment points, and hyper-dimensions, and clarifies how to provide a generalizable empirical benchmark for the performance of the ML and DCM classifiers in predicting travel behavior. This study formulates its large experiment space by spanning four hyper-dimensions and an immense number of 6,970 experiments, thus transcending the limitations of specific datasets, geographical locations, data collection procedures, outputs, and sample sizes. It includes 105 classifiers from 12 model families, which represents hitherto the most comprehensive and an almost exhaustive list for travel behavior prediction. The large-scale experiment is combined with a meta dataset of 35 past studies and 136 experiment points for a joint analysis, further enhancing the generalizability and validity of our findings. Our results provide broad insights into the prediction and the computational burden of hundreds of classifiers, and how they vary with the hyper-dimensions, as summarized below.

The ensemble methods (boosting, bagging, and random forests) and the DNNs achieve the highest predictive performance, while the DCM family performs at only the medium to lower range in the distribution of the prediction accuracy. On average, the DCM family modestly underperforms the top ML model families by around 3-4 percentage points in prediction accuracy. These

findings hold for the whole experiment space, and for different sample sizes, datasets, and outputs. They are also consistent with the results from the meta dataset of the past studies. The exact magnitude of DCMs' under-performance varies, depending on the specifics of datasets, sample size, and outputs, although the relative ranking of the DCM family is quite consistently positioned at the medium to lower range of all the classifiers.

The absolute values of the prediction accuracy have large variation. The prediction accuracy of the top ML classifiers can vary between 40 and 75 percentage points, centered around 55 percentage points. With further fine-tuning and model selection, the prediction accuracy can be boosted to around 70-90 percentage points, as shown in the meta dataset. This large variation corroborates with our initial postulation that model prediction highly depends on the hyper-dimensions. The predictive performance varies drastically with datasets, as shown by the 20 percentage points higher accuracy of the LTDS2015 than the other two. The prediction accuracy increases with larger sample sizes, and modestly varies with the prediction tasks.

RF is the most computationally efficient classifier, while other high-performance ML classifiers, such as DNNs and ensemble methods, take a relatively long time to train. The RF model family, which ranks as the second best in predictive performance and the best in computational time, successfully balances between prediction and computation. The DNNs are relatively expensive in computation owing to their complex model structures, and the ensemble methods are also relatively cumbersome owing to the complexity of ensemble over multiple models. Computational time also depends on the hyper-dimensions. Particularly, the computational cost marginally increases with sample size, in contrast to the marginally decreasing benefits in the predictive performance. In other words, larger sample size leads to both the benefit of higher prediction and the cost of more computational burden, but the marginal cost looms larger than the benefit. This finding questions the importance of collecting big data in the sense of sheer sample size. Nonetheless, the average computational time of the ML classifiers for our largest sample size (100,000 observations) takes about 84 seconds, which are still reasonable in practice.

The adoption of DCMs can be severely limited owing to their computational inefficiency, particularly when sample size is large, input dimensions are high, or simulation-based estimation is used. Many MXL models cannot be completed even within 10 hours. For the DCMs that can be completed within 10 hours, the MXL, NL, and MNL models rank the lowest, the second lowest, and the fourth lowest among the 105 classifiers in computational efficiency. The DCM family is significantly inefficient compared to all the ML classifiers, since it takes 30 times longer training time than the second worst model family (Boosting). The poor computational performance of the NL and MNL models can be attributed to the second-order Newton and Quasi-Newton methods, and that of the MXL models to the slow simulation-based estimation. These mainstream algorithms in DCMs are more computationally expensive than those in ML classifiers, such as the widely used first-order stochastic gradient descent (SGD). The computational challenge in DCMs is not salient in small sample size (e.g. $<$ 1,000 observations), but much more so in large same size (e.g. $>$ 100,000 observations).

8. Contributions, Discussions, and Future Research Directions

This benchmark paper makes important contributions by filling in a vast number of research gaps. It starts by asking how to provide a definitive and generalizable conclusion within a single study, leading to our innovative conceptualization of experiment space, experiment points, and hyper-dimensions. This conceptualization is meaningful in at least two senses. First, it allows us to examine how prediction performance and computational cost depend on datasets, sample size, and outputs in the highly structured experiment. Second, it enables a joint analysis of the large-scale experiments and the meta dataset, by taking advantage of their complementary nature in terms of the completeness of classifiers and the richness of contexts. This present work designs a complete, representative, and relevant list of classifiers informed by the meta dataset, constituting hitherto the most comprehensive comparison for the ML and DCM classifiers. It differentiates the two concepts of model families and individual classifiers to reflect the richness within each model family. It examines not only the predictive performance but also the computational burden to evaluate the practicality of the classifiers. As a result, this work can overcome the context-, data-, and author-specific limitations in the past studies. It also has enhanced validity in findings since the joint analysis demonstrates consistent results between our experiments and past studies. Therefore, we believe that this empirical benchmark work is innovative in terms of conceptualization, experiment design, and findings, successfully serving as an benchmark work for the performance of the ML and DCM classifiers in predicting travel behavior.

However, more important than filling in past research gaps, this empirical benchmark work presents many future-oriented values. It provides insights into modeling practices and future research directions, and will facilitate building a coherent knowledge ground for the community of travel behavior research.

First, our conclusions can be treated as a practical guidance of model selection for travel behavioral analysis in the future. When high prediction accuracy is the sole purpose, researchers should use DNNs and ensemble methods to achieve high performance. When both high prediction and efficient computation are of interest, researchers can consider using RFs as a starting point, owing to its balance between high performance and computational efficiency. Researchers and practitioners should be more aware of the computational problem in DCMs, particularly with large sample and high-dimensional inputs. Nonetheless, the model performance and computational time highly depend on specific contexts. The models suggested by this work are only high-quality baselines, but not guaranteed to provide the best performance in all the contexts.

Second, this empirical benchmark work yields insights for promising future research directions. Although this work motivates with a comparative perspective, the high performance of the ensemble methods implies that it is more likely to yield the best prediction by combining classifiers. This insight is quite reasonable, since every classifier is limited by its own underlying model assumptions, and a heterogeneous ensemble across a large number of classifiers can counteract their individual disadvantages. The high performance of the DNN family also demonstrates its potentials. In this study, the DNNs are the most standard feedforward DNNs with different width and depth, but

more refined DNN architectures and hyper-parameters are highly likely to improve the predictive performance and even interpretation, as demonstrated by recent studies [73]. This direction is also promising, since the DNN model family is arguably the most versatile among all the ML families, as shown by the tremendous progress in the ML and choice modeling communities.

The DCM community should switch more attention from predictive performance to computational efficiency. Computational efficiency is not only an important factor influencing the practical adoption, but also reversely influences the predictive performance. Although DCMs can predict at the medium and lower end of the accuracy distribution, their computational efficiency is ranked as the lowest and is significantly more inefficient than the second lowest model family. This computational problem of DCMs will become more pronounced as researchers increasingly adopt higher-dimensional inputs and larger sample sizes in the big data context. The DCMs are particularly strong in their underlying economic theory for policy and welfare analysis, so to further DCMs' impacts, researchers should improve the computational efficiency at least to the degree of the mainstream ML classifiers. In fact, the authors believe that some simple steps can be adopted to mitigate this problem. For example, the DCM community can consider simplifying the expensive Newton methods by SGD, and using deterministic function approximation instead of the slow simulation-based estimation in MXL. The DCM community can also consider implementing the ensemble methods or regularization tools for choice modeling. The authors postulate that these steps should not be too complicated, since they have been widely adopted in standard ML packages such as scikit-learn and tensorflow.

Lastly, this empirical benchmark work presents an agenda of streamlining future research about travel demand modeling to facilitate knowledge sharing by accumulating the findings of individual studies. Our framework allows the augmentation of future studies as experiment points into the meta-dataset, so the findings from the future studies can always be compared to our benchmark results. This work used two publicly available datasets (NHTS2017 and LTDS2015), which represent two important geographical areas and will be continuously updated by the governments. Therefore, the two datasets might suffice as the benchmark datasets for future research. It is important for the travel behavior community to work on shared public datasets, particularly for the methodological studies. It is because publicly shared datasets allow a wide and consistent comparison across individual researchers' work and facilitate persistent methodological development. This vision is already a fact in the ML community, where individual researchers use the same datasets, such as MNIST and ImageNet, to continuously improve their methodological understanding. Since this work uses only the average prediction and computational time to construct the argument, the authors have no doubt that future studies will beat our benchmark performance in prediction and computation by innovating in specific modeling structures and applications.

Acknowledgement

We thank Singapore-MIT Alliance for Research and Technology (SMART) for partially funding this research. We thank Alex Guo for his collection of the references in the literature review. Shenhao thanks Stefanie Jegelka to provide key guidance about this paper idea in a casual talk in 2017.

CRediT Author Statement

The authors confirm contribution to the paper as follows. **Shenhao Wang:** Conceptualization, Data curation, Formal analysis, Investigation, Methodology, Project administration, Validation, Writing (original draft), and Writing (review and editing). **Baichuan Mo:** Data curation, Formal analysis, Investigation, Methodology, Software, and Visualization. **Stephane Hess:** Writing (review and editing). **Jinhua Zhao:** Funding acquisition, Resources, and Supervision.

References

[1] David W Aha, Dennis Kibler, and Marc K Albert. "Instance-based learning algorithms". In: *Machine learning* 6.1 (1991), pp. 37–66.

[2] Mahdieh Allahviranloo and Will Recker. "Daily activity pattern recognition by using support vector machines with multiple classes". In: *Transportation Research Part B: Methodological* 58 (2013), pp. 16–43.

[3] Moshe Ben-Akiva, John L Bowman, and Dinesh Gopinath. "Travel demand model system for the information era". In: *Transportation* 23.3 (1996), pp. 241–266.

[4] Moshe Ben-Akiva et al. *Discrete Choice Analysis*. 2014.

[5] James P Biagioni et al. "Tour-based mode choice modeling: using an ensemble of (un-) conditional data-mining classifiers". In: *88th Annual Meeting of the Transportation Research Board, Washington, DC*. 2008.

[6] Christopher M Bishop. *Pattern recognition and machine learning*. springer, 2006.

[7] Charles Bouveyron, Stéphane Girard, and Cordelia Schmid. "High-dimensional discriminant analysis". In: *Communications in Statistics—Theory and Methods* 36.14 (2007), pp. 2607–2623.

[8] Stephen Boyd and Lieven Vandenberghe. *Convex optimization*. Cambridge university press, 2004.

[9] Giulio Erberto Cantarella and Stefano de Luca. "Modelling mode choice behaviour for extra-urban journeys". In: *Proceedings of European Transport Conference*. 2002.

[10] Giulio Erberto Cantarella and Stefano de Luca. "Multilayer feedforward networks for transportation mode choice analysis: An analysis and a comparison with random utility models". In: *Transportation Research Part C: Emerging Technologies* 13.2 (2005), pp. 121–155.

[11] Hilmi Berk Celikoglu. "Application of radial basis function and generalized regression neural networks in non-linear utility function specification for travel mode choice modelling". In: *Mathematical and Computer Modelling* 44.7 (2006), pp. 640–658.

[12] Long Cheng et al. "Applying a random forest method approach to model travel mode choice behavior". In: *Travel behaviour and society* 14 (2019), pp. 1–10.

[13] Gregory F Cooper and Edward Herskovits. "A Bayesian method for the induction of probabilistic networks from data". In: *Machine learning* 9.4 (1992), pp. 309–347.

[14] Koby Crammer et al. "Online passive-aggressive algorithms". In: *Journal of Machine Learning Research* 7.Mar (2006), pp. 551–585.

[15] Juan De Dios Ortuzar and Luis G Willumsen. *Modelling transport*. John Wiley and Sons, 2011.

[16] Sean T Doherty and Abolfazl Mohammadian. "Application of artificial neural network models to activity scheduling time horizon". In: *Transportation research record* 1854.1 (2003), pp. 43–49. ISSN: 0361-1981.

[17] Finale Doshi-Velez and Been Kim. "Towards a rigorous science of interpretable machine learning". In: (2017).

[18] Alireza Ermagun, Taha Hossein Rashidi, and Zahra Ansari Lari. "Mode choice for school trips: long-term planning and impact of modal specification on policy assessments". In: *Transportation research record* 2513.1 (2015), pp. 97–105. ISSN: 0361-1981.

[19] Reid Ewing and Robert Cervero. "Travel and the built environment: a meta-analysis". In: *Journal of the American planning association* 76.3 (2010), pp. 265–294.

[20] Manuel Fernández-Delgado et al. "Do we need hundreds of classifiers to solve real world classification problems". In: *Journal of Machine Learning Research* 15.1 (2014), pp. 3133–3181.

[21] Yoav Freund and Robert E Schapire. "A decision-theoretic generalization of on-line learning and an application to boosting". In: *Journal of computer and system sciences* 55.1 (1997), pp. 119–139.

[22] Jerome H Friedman. "Greedy function approximation: a gradient boosting machine". In: *Annals of statistics* (2001), pp. 1189–1232.

[23] Jerome H Friedman. "Regularized discriminant analysis". In: *Journal of the American statistical association* 84.405 (1989), pp. 165–175.

[24] Pierre Geurts, Damien Ernst, and Louis Wehenkel. "Extremely randomized trees". In: *Machine learning* 63.1 (2006), pp. 3–42.

[25] Nima Golshani et al. "Modeling travel mode and timing decisions: Comparison of artificial neural networks and copula-based joint model". In: *Travel Behaviour and Society* 10 (2018), pp. 21–32. ISSN: 2214-367X.

[26] Ian J Goodfellow, Jonathon Shlens, and Christian Szegedy. "Explaining and harnessing adversarial examples". In: *arXiv preprint arXiv:1412.6572* (2015).

[27] Ian Goodfellow et al. *Deep learning*. Vol. 1. MIT press Cambridge, 2016.

[28] Julian Hagenauer and Marco Helbich. "A comparative study of machine learning classifiers for modeling travel mode choice". In: *Expert Systems with Applications* 78 (2017), pp. 273–282.

[29] Trevor Hastie, Andreas Buja, and Robert Tibshirani. "Penalized discriminant analysis". In: *The Annals of Statistics* (1995), pp. 73–102.

[30] Trevor Hastie and Robert Tibshirani. "Discriminant analysis by Gaussian mixtures". In: *Journal of the Royal Statistical Society: Series B (Methodological)* 58.1 (1996), pp. 155–176.

[31] David A Hensher and Tu T Ton. "A comparison of the predictive potential of artificial neural networks and nested logit models for commuter mode choice". In: *Transportation Research Part E: Logistics and Transportation Review* 36.3 (2000), pp. 155–172.

[32] Tim Hillel, Mohammed ZEB Elshafie, and Ying Jin. "Recreating passenger mode choice-sets for transport simulation: A case study of London, UK". In: *Proceedings of the Institution of Civil Engineers-Smart Infrastructure and Construction* 171.1 (2018), pp. 29–42. ISSN: 2397-8759.

[33] Tim Hillel, Mohammed ZEB Elshafie, and Ying Jin. "Recreating passenger mode choice-sets for transport simulation: A case study of London, UK". In: *Proceedings of the Institution of Civil Engineers-Smart Infrastructure and Construction* 171.1 (2018), pp. 29–42.

[34] Kurt Hornik, Maxwell Stinchcombe, and Halbert White. "Multilayer feedforward networks are universal approximators". In: *Neural networks* 2.5 (1989), pp. 359–366.

[35] Torsten Hothorn, Kurt Hornik, and Achim Zeileis. "Unbiased recursive partitioning: A conditional inference framework". In: *Journal of Computational and Graphical statistics* 15.3 (2006), pp. 651–674.

[36] Geoff Hulten, Laurie Spencer, and Pedro Domingos. "Mining time-changing data streams". In: *Proceedings of the seventh ACM SIGKDD international conference on Knowledge discovery and data mining*. ACM, 2001, pp. 97–106.

[37] Arash Jahangiri and Hesham A Rakha. "Applying machine learning techniques to transportation mode recognition using mobile phone sensor data". In: *IEEE transactions on intelligent transportation systems* 16.5 (2015), pp. 2406–2417. ISSN: 1524-9050.

[38] Matthew G Karlaftis and Eleni I Vlahogianni. "Statistical methods versus neural networks in transportation research: Differences, similarities and some insights". In: *Transportation Research Part C: Emerging Technologies* 19.3 (2011), pp. 387–399.

[39] Ron Kohavi. "The power of decision tables". In: *European conference on machine learning*. Springer, 1995, pp. 174–189.

[40] Sotiris B Kotsiantis, I Zaharakis, and P Pintelas. "Supervised machine learning: A review of classification techniques". In: *Emerging artificial intelligence applications in computer engineering* 160 (2007), pp. 3–24.

[41] Niels Landwehr, Mark Hall, and Eibe Frank. "Logistic model trees". In: *Machine learning* 59.1-2 (2005), pp. 161–205.

[42] Saskia Le Cessie and Johannes C Van Houwelingen. "Ridge estimators in logistic regression". In: *Journal of the Royal Statistical Society: Series C (Applied Statistics)* 41.1 (1992), pp. 191–201.

[43] Zachary C Lipton. "The mythos of model interpretability". In: *arXiv preprint arXiv:1606.03490* (2016).

[44] Jordan J Louviere, David A Hensher, and Joffre D Swait. *Stated choice methods: analysis and applications*. Cambridge university press, 2000.

[45] Baichuan Mo et al. *Impacts of subjective evaluations and use of existing travel modes on adoption of autonomous mobility-on-demand*. Tech. rep. 2020.

[46] Abolfazl Mohammadian and Eric J Miller. "Nested logit models and artificial neural networks for predicting household automobile choices: comparison of performance". In: *Transportation Research Record* 1807.1 (2002), pp. 92–100. ISSN: 0361-1981.

[47] Gregoire Montavon, Wojciech Samek, and Klaus-Robert Muller. "Methods for interpreting and understanding deep neural networks". In: *Digital Signal Processing* 73 (2018), pp. 1–15.

[48] Mikhail Mozolin, J-C Thill, and E Lynn Usery. "Trip distribution forecasting with multilayer perceptron neural networks: A critical evaluation". In: *Transportation Research Part B: Methodological* 34.1 (2000), pp. 53–73.

[49] Anh Nguyen, Jason Yosinski, and Jeff Clune. "Deep neural networks are easily fooled: High confidence predictions for unrecognizable images". In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. 2015, pp. 427–436.

[50] Peter Nijkamp, Aura Reggiani, and Tommaso Tritapepe. "Modelling inter-urban transport flows in Italy: A comparison between neural network analysis and logit analysis". In: *Transportation Research Part C: Emerging Technologies* 4.6 (1996), pp. 323–338.

[51] Hichem Omrani. "Predicting travel mode of individuals by machine learning". In: *Transportation Research Procedia* 10 (2015), pp. 840–849.

[52] Hichem Omrani et al. "Prediction of individual travel mode with evidential neural network model". In: *Transportation research record* 2399.1 (2013), pp. 1–8. ISSN: 0361-1981.

[53] Miguel Paredes et al. "Machine learning or discrete choice models for car ownership demand estimation and prediction?" In: *Models and Technologies for Intelligent Transportation Systems (MT-ITS), 2017 5th IEEE International Conference on*. IEEE, 2017, pp. 780–785.

[54] Nicholas G Polson and Vadim O Sokolov. "Deep learning for short-term traffic flow prediction". In: *Transportation Research Part C: Emerging Technologies* 79 (2017), pp. 1–17.

[55] Ross Quinlan. "C4. 5". In: *Programs for machine learning* (1993).

[56] PV Subba Rao et al. "Another insight into artificial neural networks through behavioural analysis of access mode choice". In: *Computers, environment and urban systems* 22.5 (1998), pp. 485–496.

[57] Brian D Ripley and NL Hjort. *Pattern recognition and neural networks*. Cambridge university press, 1996.

[58] Tarek Sayed and Abdolmehdi Razavi. "Comparison of neural and conventional approaches to mode choice analysis". In: *Journal of Computing in Civil Engineering* 14.1 (2000), pp. 23–30. ISSN: 0887-3801.

[59] Ch Ravi Sekhar and E Madhu. "Mode Choice Analysis Using Random Forrest Decision Trees". In: *Transportation Research Procedia* 17 (2016), pp. 644–652.

[60] Muhammad Awais Shafique and Eiji Hato. "Use of acceleration data for transportation mode prediction". In: *Transportation* 42.1 (2015), pp. 163–188. ISSN: 0049-4488.

[61] Yu Shen et al. *Built environment and autonomous vehicle mode choice: A first-mile scenario in singapore*. Tech. rep. 2019.

[62] Nagesh Shukla et al. "Tour-based travel mode choice estimation based on data mining and fuzzy techniques". In: (2015).

[63] Kenneth A Small, Erik T Verhoef, and Robin Lindsey. "Travel Demand". In: *The economics of urban transportation*. Vol. 2. Routledge, 2007.

[64] Nitish Srivastava et al. "Dropout: a simple way to prevent neural networks from overfitting". In: *Journal of machine learning research* 15.1 (2014), pp. 1929–1958.

[65] Liang Tang, Chenfeng Xiong, and Lei Zhang. "Decision tree method for modeling travel mode switching in a dynamic behavioral process". In: *Transportation Planning and Technology* 38.8 (2015), pp. 833–850.

[66] Ahmet Tortum, Nadir Yayla, and Mahir Gkda. "The modeling of mode choices of intercity freight transportation with the artificial neural networks and adaptive neuro-fuzzy inference system". In: *Expert Systems with Applications* 36.3 (2009), pp. 6199–6217. ISSN: 0957-4174.

[67] Kenneth E Train. *Discrete choice methods with simulation*. Cambridge university press, 2009.

[68] Vladimir Vapnik. *The nature of statistical learning theory*. Springer science and business media, 2013.

[69] Vladimir Naumovich Vapnik. "An overview of statistical learning theory". In: *IEEE transactions on neural networks* 10.5 (1999), pp. 988–999.

[70] Ulrike Von Luxburg and Bernhard Schölkopf. "Statistical learning theory: Models, concepts, and results". In: *Handbook of the History of Logic*. Vol. 10. Elsevier, 2011, pp. 651–706.

[71] Martin J Wainwright. *High-dimensional statistics: A non-asymptotic viewpoint*. Vol. 48. Cambridge University Press, 2019.

[72] Fangru Wang and Catherine L Ross. "Machine learning travel mode choices: Comparing the performance of an extreme gradient boosting model with a multinomial logit model". In: *Transportation Research Record* 2672.47 (2018), pp. 35–45.

[73] Shenhao Wang, Baichuan Mo, and Jinhua Zhao. "Deep neural networks for choice analysis: Architecture design with alternative-specific utility functions". In: *Transportation Research Part C: Emerging Technologies* 112 (2020), pp. 234–251. ISSN: 0968-090X.

[74] Shenhao Wang, Qingyi Wang, and Jinhua Zhao. "Deep neural networks for choice analysis: Extracting complete economic information for interpretation". In: *Transportation Research Part C: Emerging Technologies* 118 (2020), p. 102701. ISSN: 0968-090X.

[75] Shenhao Wang, Qingyi Wang, and Jinhua Zhao. "Multitask learning deep neural networks to combine revealed and stated preference data". In: *Journal of Choice Modelling* (2020), p. 100236. ISSN: 1755-5345.

[76] Shenhao Wang and Jinhua Zhao. "Risk preference and adoption of autonomous vehicles". In: *Transportation Research Part A: Policy and Practice* 126 (2019), pp. 215–229. ISSN: 0965-8564.

[77] Shenhao Wang et al. "Deep Neural Networks for Choice Analysis: A Statistical Learning Theory Perspective". In: *arXiv preprint arXiv:1810.10465* (2018).

[78] Yuankai Wu et al. "A hybrid deep learning based traffic flow prediction method and its understanding". In: *Transportation Research Part C: Emerging Technologies* 90 (2018), pp. 166–180.

[79] Jian-Chuan Xian-Yu. "Travel mode choice analysis using support vector machines". In: *ICCTP 2011: Towards Sustainable Transportation Systems*. 2011, pp. 360–371.

[80] Chi Xie, Jinyang Lu, and Emily Parkany. "Work travel mode choice modeling with data mining: decision trees and neural networks". In: *Transportation Research Record: Journal of the Transportation Research Board* 1854 (2003), pp. 50–61.

[81] Hong Zhang and Zhen Zhang. "Feedforward networks with monotone constraints". In: *IJCNN'99. International Joint Conference on Neural Networks. Proceedings (Cat. No. 99CH36339)*. Vol. 3. IEEE, 1999, pp. 1820–1823.

[82] Yunlong Zhang and Yuanchang Xie. "Travel mode choice modeling with support vector machines". In: *Transportation Research Record: Journal of the Transportation Research Board* 2076 (2008), pp. 141–150.

[83] Xiaolu Zhou, Mingshu Wang, and Dongying Li. "Bike-sharing or taxi? Modeling the choices of travel mode in Chicago using machine learning". In: *Journal of transport geography* 79 (2019), p. 102479. ISSN: 0966-6923.

Appendix I: Summary of Acronyms and Terminology

Table 6: Acronyms and other terminology in this study;

Panel 1. Acronyms	
DCM	Discrete choice models
DNN	Deep neural networks
DA	Discriminant analysis
BM	Bayesian models
SVM	Support vector machines
KNN	K nearest neighbors
DT	Decision tree
GLM	Generalized linear models
GP	Gaussian process
BAGGING	Bagging methods
RF	Random forests
BOOSTING	Boosting methods
MNL	Multinomial logit
NL	Nested logit
MXL	Mixed logit
NHTS2017	National household travel survey 2017
LTDS2015	London travel demand survey 2015
SGP2017	Singapore stated preference survey 2017
MC	Mode choice
CO	Car ownership
TP	Trip purposes
Panel 2. Other terminology	
Classifiers	Models using discrete values as outputs
Hyperparameters	A parameter used to control the learning process
0-1 loss	An indicator function that returns 1 when the target and output are not equal and zero otherwise