A Machine Learning Decision Support Tool for Travel Demand Modeling

CScott Brown
School of Computing
University of South Alabama

Yi Hou
NREL

Venu M. Garikapati
NREL

***Abstract*—In this paper we present a tool for applying an array of models, including logit, nested logit, neural network, naive bayes and decision tree classifiers, applicable to arbitrary classification problems and specifically tailored to aid in the construction of transportation models. Our processing pipeline is designed to be easily extensible, accounts for common pitfalls in the application of various models, automatically selects optimum hyperparameters and reports a variety of metrics commonly employed to evaluate model performance. The tool can be used to choose an appropriate model family or to construct a model ensemble to improve upon current modeling standards in transportation engineering. We test our proposed system on Fisher's classic Iris dataset, and household vehicle count and work schedule targets from the 2017 National Household Travel Survey (NHTS), which are common variables of interest in transportation simulations. Results demonstrate that for some variables, logit are not the most effective models, and the proposed system can aid in selecting a better model.**

I. Introduction

Logistic regression has long been the golden standard for classification modeling in transportation problems. State-of-the-art transportation simulation software [1], [2] in use by transportation departments around the globe have at their core this tried-and-true family of functions. However, as available data grows in size and scope and desired models move from agglomerate trip-based to more intricate activity-based designs, models must capture increasingly complex relationships between variables. It is not clear that the restrictive assumptions accompanying logistic regression remain valid in this increasingly complex domain.

Despite a wealth of work demonstrating situations in which alternative model families can produce superior results, adoption of alternative model families amongst transportation practitioners remains low. Partly this is accounted for by the simplicity of interpretation of logit models and the fact that they are so deeply ingrained in the current infrastructure of transportation modeling software. Further, most transportation simulation problems involve a great number of models for a variety of parameters. For example, the Comprehensive Econometric Micro-simulator for Daily Activity-travel Patterns (CEMDAP) [3] involves at least 57 separate models, 26 of which are logit models, most of the remainder being regression or ordinal probit. For any problem, there are a great many possible model families that can be applied, and it is a priori unclear whether investing in the development of more complex models will be valuable for a given set of predictors and targets. Finally, applying different families of models can involve a range of necessary preprocesing steps and implementation pitfalls that can make casting a final judgement between one model family or another difficult.

Given existing barriers to replacing logit models with more robust alternatives, we propose a black-box pipeline to apply an array of machine learning algorithms alongside logit models to provide practitioners with a simple means of guaging the cost vs benefit of investigating alternative model families. We believe that the ability to choose an appropriate model family for each of the great number of individual problems in a transportation model will improve the predictive power of overall models and the resulting simulations. Furthermore, in situations where interpretability is not a concern, a battery of models can be ensembled to create a model that accounts for the inevitable shortcomings of each family resulting in a single more robust model.

In the proceeding sections, we investigate the abilities of the proposed system on the Iris dataset and also variables from the 2017 NHTS. The goals of our experiments are the following: First, to examine the hypothesis that fire-and-forget modeling pipelines can be applied to a variety of problems, and provide reasonable models to facilitate model family performance comparison. Such a pipeline is important, since different model families may be subject to different data preprocessing considerations such as hyperparameter selection, feature selection, class balance and variable scaling. Second, to provide a tool, TEAM-TDM (A Tool for Evaluating an Array of Machine Learning Travel Demand Models) that automates the entire process, allowing practitioners to effortlessly evaluate a range of model families on a variety of problems and datasets. Although we will use TEAM-TDM to compare the efficiency and efficacy of various model families, including logit models on the examined variables of interest, we consider this to be of secondary importance to the research, but some readers may find this aspect informative nonetheless.

Section II reviews related work and works demonstrating the value of various machine learning algorithms in transportation modeling. Section III describes the proposed battery of models, and lays out the order, means and reasonings behind the experiments performed. Section IV summarizes the exact proceedings and results of our experiments. Finally, section V discusses important takeaways, limitations and future work.

II. Related Work

In this section we review works evaluating the relative performance of logit regression (MNL) and machine learning algorithms for classification in transportation problems. The body of works comparing the abilities of logit regression to other model families on classification and discrete choice problems in transportation is extensive, and dates back decades. As such, we attempt to provide diverse and popular examplars of the subject, covering the most common model families, and especially focusing on works that evaluate multiple models. Commonly applied model families (besides logit) include support vector machines (SVM), multi-layer perceptrons (MLP) (a subset of neural networks (NN)), naive bayes (NB), decision trees (DT), and various derivatives, such as bagged and boosted varieties and other model ensembles. NN models are the most commonly evaluated model family in the literature, and an entire review [4] is dedicated to NNs vs classical statistical models (including MNL) for transportation problems.

Mode choice is by far the most popular target variable for evaluating model families, despite the fact that alternative model families repeatedly demonstrate mediocre gains in performance vs MNL models for this problem. [5] perform a comparison of SVM, MLP and MNL models for mode choice modeling. Notably, they compare against a known and thoroughly vetted MNL model, painstakingly conceived and refined in [6] to even include non-linear variable combinations. The compared SVM and MLP models, in contrast, have a simple specification, and their hyperparameters are automatically chosen using cross-validation, thus being amenable to a fire-and-forget style application. All three models perform comparably, with the SVM model having a slight edge, even without having included the same non-linear variable combinations fed to the MNL model. [7] compare DT, MLP and MNL models also for mode choice. All three modes again perform comparably, with the DT and MLP models performing slightly better. The authors point out the interesting observation that an MNL model is actually a special case of MLP, and also point out the problem of class balance in training certain types of classifier. [8] also compare MNLs with a NN architecture, but go to great lengths to develop a NN architecture that captures various decision structures hypothesized to be analagous to natural human reasoning. Even so, the NN and logistic regression models perform comparably. [9] compare a number of model families, including MNL, MLP, DT, several ensemble varieties of DT, NB, and SVM using the Dutch National Travel Survey. Interestingly, they observe a performance difference between MNL and other models that is significantly greater than observed in previous studies, with random forests achieving upward of 90% accuracy. The reason for this distinction is unclear. Perhaps the Dutch are simply more predictable with respect to mode choice. Unfortunately, no accounting is given in the paper. However, this suggests hope that in some circumstances a black box battery of models may distinguish situations where alternatives to MNL models shine.

Since MNL is so entrenched in transportation modeling culture, other target variables for evaluating the performance of MNL vs machine learning methods span the gamut of topics related to transportation. [10] compare a nested MNL with NNs for modeling vehicle ownership. The nesting structure for the MNL is specified a priori, and the NN architecture is optimized to give the best accuracy for individual predictions. They find that the NN gives a better accuracy for individual predictions, but note that there are a variety of metrics by which models might be compared, including aggregate market share and log loss. [11] compare MNL and DT models for route choice modeling. The models perform comparably on their dataset, with the DT model notably performing quite well even when utility maximizing (MNL-generated) routes were taken to be ground truth. [12] compare MNL and SVM models for predicting daily activity sequences. Although the MNL performs slightly better for predicting the initial activity of the day, the SVM model is significantly better at predicting subsequent activities. It is possible that the MNL model could be improved by incorporating a nesting structure, since activity choices are not a priori independent. However, the SVM model does not take the nesting structure into account, and is trained without such a posteriori contrivances. [13] compare NB and DT models for predicting vechicle ownership rates, and provide some discussion comparing their results with those of other studies employing MNL models. They also find little difference in the predictive capabilities of machine learning models and more classical approaches. However, they do note that the NB model is particularly good at classifying certain classes, which suggests that an ensemble of models may be viable.

Many other works focus entirely on MNL models solely, or on some other specific model family. Despite the large corpus of work applying machine learning models to transportation problems, the studies are difficult to compare, owing to models having been built on diverse datasets and with different designs concerning model hyperparameters, data preprocessing, etc. Furthermore, all of the above studies were conducted with the intention of constructing the best possible model for a given problem, rather than creating a model battery that can be applied to many problems. In the proceeding sections, we address these concerns by using TEAM-TDM to compare a wider range of model families than most previous studies, that can be applied with little or no alteration to different datasets to provide maximum consistency between studies. We also show that TEAM-TDM can be quickly and easily adapted to new problems, allowing practitioners and researchers to easily and fairly compare the performance of different model families on individual tasks.

III. Methodology

In this section we describe the proposed battery of models and related data processing pipeline comprising TEAM-TDM and the reasonings behind individual choices. First, we provide a brief overview of the applied models, benefits and shortcomings of each, as well as references for a more detailed understanding of the related mathematics. Also, we examine the training procedures, data (pre)processing and hyperparameter selection. Finally, we outline the experiments that we will perform and the datasets upon which we will apply our models.

A. Models

The models we evaluate in our expereiments with TEAM-TDM include logit (MNL), random forests (RF), multi-layer perceptrons (MLP) and naive-bayes (NB). We also include other models when appropriate, such as ordered probit (OP) and nested logit models (NMNL) to demonstrate the ease with which additional models can be added to TEAM-TDM when desired.

Logit models [14] are statistical models that are the traditional standard for classification problems, being some of the earliest model families for classification. MNL are linear models, in the sense that MNL models can only be 100% effective on data that are perfectly linearly separable. A related but not identical problem to linear separability is that MNL models assume that, for multi-class problems, the choice between any pair of alternatives is independent of the distribution of other alternatives. This is referred to as the independence of irrelevant alternatives assumption (IIA), and, when problematic, is sometimes but not always addressable by 'nesting' logit models [15]. Despite these quite restrictive assumptions, MNL models enjoy broad success in a wide range of applications, due to the fact that many datasets satisfy the assumptions at least approximately, because the learned parameters of an MNL model can often be interpreted easily in a direct cause-and-effect manner, and because training and predictions are relatively speedy. In TEAM-TDM, we include a simple MNL model over all available variables, without regularization.

Decision trees [16] are a popular class of models due to their speed in both training and prediction, ready interpretability and general ease-of-use. A DT model performs a classification by recursively choosing a feature and then 'deciding' a branch of the tree based on the value of that feature, on an on until a prediction can be made on the target variable. The training procedure for a DT model family consists of finding a tree so that the series of 'decision' branches results in accurate predictions. Choosing the optimal decision tree from the set of all possible trees is known to be an NP-complete problem. As such, various methods exist for choosing an optimal tree from some subset of all possible trees, of which we employ CART [17], which creates branches based on the criterion of Gini Importance. DT models have a number of known issues, including most notably a tendancy to overfit without considerable supervision. One popular method of dealing with the overfitting issue, known as a random forest (RF) [18], is to construct and ensemble of weak decision trees each trained on a random subset of the data. This trades in speed and interpretability for a more robust classifier. In TEAM-TDM, we include an RF classifier where the depth of the individual trees is an optimized hyperparamter.

Multi-layer perceptrons [19] can be seen as a simple extension of MNL. MNL applies a linear transformation to the features from n feature dimensions into m target class dimensions, followed by a softmax function to obtain probability-esque output. It is entirely possible to chain these functions to obtain a more complex decision surface. For example, we could apply a linear transformation to the features from n feature dimensions into h 'hidden' dimensions, followed by a softmax function (or some other 'activation' function), followed by yet another linear transformation from h dimensions to m target class dimensions, followed by another softmax function. This chaining process is the basic thought behind MLP models, and can be done to arbitrary depths, and with arbitrary activation functions. Specifically, the MLP architecture we apply consists of one hidden layer, treating h as a hyperparameter, and uses a rectified linear unit as the hidden layer activation function. The benefit of MLPs over MNLs is their ability to model non-linear decision boundaries, at the cost of less interpretability, slower training/prediction speeds, and a danger of overfitting, especially for very deep MLPs. In TEAM-TDM, we include an MLP with one hidden layer, rectified linear unit activations and L_2 regularized weights, where the number of 'hidden' dimensions h is an optimized hyperparameter, always trained for a fixed 1000 epochs.

Naive Bayes is a simple family of models that makes an assumption that the feature values are all independent conditioned on the target value. Thus, we can predict the probability of a target class given each feature value independently based on the distribution of the training data. In TEAM-TDM, for continuous valued features, we make a further assumption that the feature values are Gaussian distributed. For discrete valued features, we simply use the empirical distribution.

Ordinal regression problems, which occupy an intermediate ground between classification and regression can be handled in a number of ways. One common method is to simply cast the problem as a classification problem. Other methods include inclusion of latent variables to measure the varying degrees of separation characterized by ordinal problems, or by nesting classifiers in a one vs rest tree to cast the ordinal regression as a series of binary classification problems [20]. Since these type problems appear frequently in transportation modeling, we provide an option in the Pipeline for including them when appropriate. In TEAM-TDM, we include an OP and also an NMNL where nests are constructed into a tree of binary problems $y = i$ vs $y > i$.

Finally, for reference, we include a stratified dummy classifier which randomly predicts a class according to the empirical distribution of classes in the training data without regard to any features. This model is essentially a best guess when no features are known. Also, It was intended to include an RBF kernel SVM in the experiments, but the NHTS dataset was too large to train this model family on the available hardware in a reasonable amount of time, and it was abandoned at that stage and later removed from the Iris dataset analysis for continuity.

B. Stacking

In addition to applying the battery of models described in section III-A, TEAM-TDM constructs a stacked model by adding the model battery predictions to the features space and constructing a model on the extended features. The exact procedure for extending the feature space with model predictions is as follows: Following a procedure similar to cross-validation, split the training data into 5 approximately equally-sized sets. For each combination of 4 of those sets, train the entire battery of models on those 4 and make predictions on the probability of each class on the remaining set. Thus, each training point receives an additional $n_{\text{target classes}} \times n_{\text{models}}$ features consisting of out-of-training predictions by each model. For model families which do not naturally predict a vector of probabilities, TEAM-TDM substitutes a one-hot vector of the predicted class for uniformity. In our experiments, TEAM-TDM stacks the model battery using an MNL model, but this can be changed with a single line of code. However, since non-linearities in the decision surface are already captured by the constituent models, we find that stacking with a more complex model is generally unnecessary.

C. Feature Transformation

For a given dataset, categorical variables are identified a priori and a list of which variables are categorical is given as a parameter to TEAM-TDM. TEAM-TDM transforms these variables via a one-hot encoding, which is equivalent to the introduction of a dummy variable for each possible class of the variable. Where data is missing or has an otherwise non-standard value, TEAM-TDM does not make a distinction, and a class is created for that possibility, and it is included as a feature. For example, in the NHTS 2017 data, the variable corresponding to the race of the respondent includes categories for 'Don't know', and 'Refused'. These classes are treated exactly the same as all other classes, and are assigned a dummy variable.

In fact, TEAM-TDM does not include a means of dealing with missing or problematic data, partly because the data which we are using does not have such problems to a significant degree, and partly because it is impossible to construct a one-size-fits-all solution to the issue of messy data. As such, dropping, imputing or otherwise accounting for ugly data problems must be done by the practitioner prior to application of the model battery. However, dealing with messy data is an existing problem for application of classical statistical models, and since our model battery is designed as a comparison tool, accounting for messy data is beyond the scope of the current research.

Also of note is that TEAM-TDM has no general means of dealing with ordinal data. It is up to the user to determine if ordinal data would be best treated categorically (and thus losing the ordinal information), or numerically (and thus assuming a fixed scale between ordinals). This is owing to the fact that most machine learning model families have no intrinsic method of dealing with ordinal data, and methods for including ordinal features are not always available. Ordinal target variables are, however, fair game, and we include an ordinal-capable nested logit model in one of our experiments, to illustrate the ease with which TEAM-TDM can be extended. TEAM-TDM makes the decision to transform the target variable on a per-model level and, with the exception of the NMNL and OP models which explicitly deal in ordinal targets, all models treat target variables as categorical.

D. Feature Scaling

Having introduced dummies for the categorical variable classes, each numerical feature in the training data are scaled to lie on $[0, 1]$. The scaling formula, sometimes referred to as min-max scaling is given by:

$$x \leftarrow \frac{x - \min(x)}{\max(x) - \min(x)}$$

Where $\min(x)$ and $\max(x)$ are the minimum and maximum values for feature x in the training data. The purpose of scaling is that some model families (notably SVM models, but also NNs and others to some extent) are sensitive to the Euclidean distance between pairs of points. The Euclidean distance between points is, in turn, sensitive to the units in which we represent a feature. For example, representing income in dollars vs thousands of dollars can have a great impact on the outcome of the model. As such, we scale features to have units that place them on the same scale as the categorical dummy variables, to give all of the features equal a priori importance in the model. Importantly, logit models and other models that are not sensitive to the relative scale of features, such as decision trees, are, in fact, invariant to the scaling, and an equivalent logit model will be obtained up to scaling of the model parameters. Thus, TEAM-TDM simply scales the data for all models, whether they need it or not. Note that this method of feature scaling may be defeated in the presence of extreme outliers, in the sense that the scale of the majority of data might end up on a much smaller interval than $[0, 1]$. We make no attempt to confront this problem, and removal or mitigation of outliers is left as a consideration for the user.

E. Feature Selection

The next step in TEAM-TDM is feature selection. In order to streamline the selection of features, an RF classifier is fit to the data, and the mean decrease impurity (MDI) [21] is computed for each feature. MDI for a feature roughly corresponds to the degree to which we are able to divide the target classes cleanly by splitting the dataset on a value of that feature, aggregated over each node in a tree corresponding to that feature, over the entire forest. We include a feature selection step for a number of reasons. Selection of only the most relevant features is a standard step in building an MNL model for transportation problems. Also, the time complexity of some model classes is highly dependent on the number of features, and thus reducing the feature space improves training speed. Furthermore, superfluous features can exacerbate overfitting problems, and thus model families already prone to overfitting can benefit from a reduced feature set. In practice, the important features for one model tend to be similar to those that are important for another. However, this does not necessarily hold in all conceivable scenarios. Although TEAM-TDM reports per-model feature importances to enable investigation of this phenomenon, we do not delve into the implications of this fact in the present research.

F. Hyperparameter Selection

The next step in TEAM-TDM is hyperparameter selection. For many model families, there exist values that must be decided for a model that are not learned during the ordinary training procedure. Often, these values are parameters of the training procedure itself, or may simply not be amenable to the optimization method used to learn other model parameters. For example, the maximum depth of a DT family can be specified to help avoid overfitting problems. However, we cannot learn maximum depth during the ordinary training procedure, since increasing the depth *always* gives a better fit to the training data. Although hyperparameters such as maximum depth for a DT cannot be learned along with the other model parameters, they can still be learned. The typical procedure involves repeatedly splitting the training data, training the model with a particular setting of hyperparameters on one part of the split data, and then evaluating the model on the other part (the test split). The success of the model on the test split is taken to be a measure of the success of the hyperparameter selection, and after a number of iterations we can choose the best set of hyperparameters. Specifically, the splitting procedure that TEAM-TDM utilizes is 5-fold cross-validation [22], the metric of success is accuracy, and new candidate hyperparameters are iteratively selected using Bayesian optimization with an assumption that the hyperparameter vs test-accuracy surface is a Gaussian process with a squared-exponential covariance function [23].

Accuracy can pose problems as a metric of success, specifically whenever target classes are unbalanced and it is important that minority classes are also classified well. There exist various ways of dealing with this problem, however, the decision to implement these depends on various meta-considerations of the data, such as relative importance of target classes. Furthermore, other metrics such as precision and recall can be difficult to define for multi-class problems, and yet other metrics such as log loss are themselves sensitive to this issue in varying degrees, and may not be applicable to any arbitrary classification model. As such, we do not deal with this potential problem, and leave it as a consideration for the user.

The Gaussian process assumption can also potentially cause problems. For one, the hyperparameters are assumed to be continuous values whether they are or not. We deal with this problem by simply rounding candidate values to the nearest integer when appropriate (such as for maximum depth of a DT). The resulting hyperparameter vs test-accuracy surface is continuous, but almost certainly not differentiable, which violates the assumption that the surface is a Gaussian process with a squared-exponential covariance function. However, in practice, the optimization procedure still performs well in spite of this potential pitfall.

G. Model Training

Once features have been selected, scaled and transformed and model hyperparameters have been selected, the models can be trained on available data, using whatever procedure is standard for a given family of models. TEAM-TDM is implemented in python using the scikit-learn library, version 0.19.1 [24] with the exception of the MLP and OP models which are implemented in tensorflow. However, care was taken that these models implement the scikit API for seamless inclusion into TEAM-TDM.

Training is performed on a machine with dual Xeon E5-2640 10-core processors and a K80 GPU. Although care was taken to parallelize TEAM-TDM where possible, the API for some models, such as the SVM models, only supported single thread training. The MLP and OP models are trained on the GPU, and the remaining models are trained on the CPUs. Note that, since the models are trained simultaneously (and thus compete for resources), care should be taken in interpreting the reported training time.

H. Model Evaluation

The objective of the current research is to provide a means of evaluating various machine learning models on a given problem. It is therefore important to provide a robust set of metrics by which models can be evaluated. This is because, when attempting to distill the predictive performance of a model down to a single value, certain information is accentuated, and some information can be altogether lost. However, as humans, we are unable to take into account the exact details of a model's classification of each individual data point, and therefore some manner of distillation is necessary. As such, we take a moment at this point to describe each metric in detail, as well as the takeaways and shortcomings of each metric in comparing model performance, and discuss possible considerations when using the provided metrics in ultimately choosing a model.

By default, TEAM-TDM provides overall accuracy, two versions of precision, recall, and f1-score, the cross-entropy loss (or log-loss), mean absolute market share error (MAMSE) and training time. TEAM-TDM also provides the confusion matrix and feature importances for each model, but these are not examined here.

Cross-entropy loss provides a metric similar to accuracy that can be somewhat more appropriate for models that give probability-esque outputs. Specifically, if a model predicts a high probability for the correct class, but not the highest probability, then this tallies as a miss in the overall accuracy, but still 'counts toward' the calculation in the cross-entropy metric. Notably, cross-entropy loss is inverted from accuracy, so high values indicate an inaccurate classifier and low values indicate a more accurate classifier.

Accuracy is often the de facto standard in comparing predictive performance, and is often the ultimate metric of interest for many problems. As mentioned above (§III-F), it is problematic when accurate prediction of a minority class is of importance. In order to determine if there are problems with prediction of minority classes, we provide two versions of precision, recall and f1-score. The f1-score is simply the harmonic mean of precision and recall. Precision and recall are only naturally defined for binary classification problems. As such, it is quite easy to compute the precision and recall for each target class. The precision with respect to a particular target class can inform us about an abnormally high false-positive rate for that class. Recall with respect to a particular target class can inform us about an abnormally high false-negative rate for that class.

However, since transportation modeling problems are frequently multi-class, the way in which we combine the scores for individual classes can have different effects on the overall score. We summarize by averaging across classes according to two different schemes. The first scheme, which we refer to as 'macro' averaging, simply sums the relevant metric for each class and divides by the number of classes. This scheme puts an equal emphasis on each class, regardless of the relative representation in the data, so that if a minority class has poor precision or recall, this fact will reflect in the 'macro' averaged metric. The second scheme, which we refer to as 'weighted' averaging, weights the relevant metric for each class by its representation in the data. Thus, minority classes will have little impact on the overall metric. To sum up, a wide difference in the 'macro' and 'weighted' version of precision or recall can indicate that a model is having trouble classifying minority classes.

In many transportation applications it is less important that individual predictions are correct, and more important that predictions are correct in aggregate. For example, depending on the details of the simulation, it may be less of concern that the prediction of number of drivers in an individual household is accurate than it is that the aggregate number of drivers across an entire census block is accurate, even though our model is designed to predict for an individual household. As such, in parity with the considerations of precision and recall above, we provide two metrics 'macro' and 'weighted' mean absolute market share error (MAMSE), which is computed according to:

$$\text{MAMSE}_{\text{macro}} := \left|\left|\frac{\sum_i C_{i,j} - (\sum_j C_{i,j})^T)}{\sum_i C_{i,j}}\right|\right|_{L_1}$$

$$\text{MAMSE}_{\text{weighted}} := \frac{\left|\left|\sum_i C_{i,j} - (\sum_j C_{i,j})^T)\right|\right|_{L_1}}{\sum_{i,j} C_{i,j}}$$

Where vector division is performed pointwise, and C is the confusion matrix for the model. This metric computes the error in total market share for each class, and averages this value over all classes. Note that if accuracy is 100%, then both macro and weighted MAMSE will be 100%, but it is possible for accuracy to be 0% and both macro and weighted MAMSE to still be 100%. However, for certain applications, an analyst may wish to have a model with high MAMSE, even at the expense of lower accuracy, and we thus include it as a relevant metric.

I. Experiments

The experiments proceed according to a simple formula. Given a dataset, we randomly split the dataset $80/20$ by stratified sampling into a training set and a testing set. We identify independent variables that should be considered categorical or ordinal (which we treat as categorical), and feed this information along with the dataset into TEAM-TDM. We then give the resulting output metrics as a table and discuss the outcome. The datasets and targets upon which we evaluate TEAM-TDM are:

1) Fisher's famous Iris dataset and a modified version with some variables converted to ordinal data.
2) NHTS 2017 household level data to predict number of vehicles per household.
3) NHTS 2017 household, person and trip level data to jointly predict the time of leaving to go to work and the time of leaving work.

These particular datasets are chosen in part to evaluate TEAM-TDM on a known set of problems, and also to show that it can be used to effectively evaluate model families on real problems in transportation simulation. Although work schedule is perhaps less well-studied in the literature than some other transportation modeling problems, it is nonetheless an important part of many transportation simulations. Notably CEMDAP [3] includes a work-schedule MNL model, the general process of which we recreate for the work schedule experiment. This model was chosen because we believed it to be, a priori, a problem in which MNL might not satisfy the relevant assumptions, and where Note that, with the exception of adding OP and NMNL models for the vehicle count prediction problem, we do not alter TEAM-TDM for the different datasets. Thus, we attempt to show that TEAM-TDM, as is, can be applicable to a wide range of problems, and can be employed as a fire-and-forget evaluation method.

IV. RESULTS

In this section we detail the results of running TEAM-TDM on the proposed datasets and targets and discuss important takeaways, positive results and shortcomings.

A. Iris Dataset

The Iris dataset is a simple dataset comprising 4 numeric features and a categorical target. It is relatively straightforward to achieve near 100% accuracy on this task with even simple classifiers, and we include it primarily to illustrate the fact that TEAM-TDM can be applied as-is to simple and complex tasks alike. Table I shows the summary of performance metrics on the test data split as output by TEAM-TDM. TEAM-TDM chooses only 2 features as meaningful, ‘petal length’ and ‘petal width’, which is convenient for visualizing the output.

Notable takeaways are the relative speed and accuracy of the naive bayes model, and the relatively lengthy training time for the MLP and RF models. To some degree, the fact that we are optimizing hyperparameters for these models, which requires repeatedly training a full model often more than a hundred times is to blame. However, even accounting for this the difference in training time is orders of magnitude.

It is interesting that the stacked model actually performs *worse* than some of the other models. Recall, however, that the stacked model is built from individual models trained on subsets of the entire training data. Thus, there is no guarantee that the stacked model will improve upon the individual models. We see then, that the stacked model should really be viewed as just another model for evaluation by the user, and not an automated model-picking mechanism.

B. *NHTS vehicle prediction*

The NHTS 2017 dataset contains a much larger number of observations, and also a much larger number of features than the Iris dataset. Many features are categorical in nature, with a large number of classes, e.g. ”State”. Some features are meaningless, such as unique row identifiers. One feature is actually a ”weights” column, which gives the NHTS-provided weights for each row, accounting for the unstratified nature of the sample. The target feature is the number of vehicles owned by the household. All of the models in our battery are capable of accounting for weighted data, and we include this in our training pipeline. Besides the row identifiers, the weights, and the target, we initially include all features in our training pipeline, and let the model decide which are important.

For this experiment, only the features and data at the ‘household’ level is used. This consists of 129696 observations of 54 variables (not including the id, weights and target). TEAM-TDM transforms this into 503 variables by inclusion of dummy variables, and reduces this to 179 variables after the features selection stage. Results are summarized in the second section of Table I.

Since vehicle count admits an ordinal interpretation, we include an OP and an NMNL mode for comparison, at the cost of two additional lines of code. This allows for an indirect comparison with prior results along the same vein. For example, in [25], MNL and ordinal logit (devised using a different nesting structure than ours) give best case accuracies of around 59%. This is comparable to the results of our MNL and OP models, and suggests that TEAM-TDM produces models that are at least in the right ballpark.

The NMNL and OP models seem to perform best, although the MLP model is close behind. However, since the MLP does not benefit from our a pirori knowledge of ordinality in the target variable this might reasonably be expected. Also of note is that the NMNL model trains significantly faster than the MNL model. This fact, along with the good performance of the MLP model, suggest that a nested MLP model might be worth exploring.

Interestingly, the stacked model performs even better than the NMNL model across the board. Specifically, the macro precision score is significantly higher, suggesting that the stacked model performs much better on classes with less representation in the training data. Indeed, the disparity between weighted/macro precision/recall for all of the models suggests that model predictions for under-represented classes might be taken with a grain of salt, and that some manner of data balancing might be in order as a preprocessing step.

Lastly, comparing the training times of the models with the Iris dataset, it is clear that some model families scale better than others. Notably, RF does not seem to take significantly more time on the much larger NHTS dataset. This is most likely an artifact of a large quantity of time taken up by the Gaussian Process optimizer for picking hyperparameters.

C. *NHTS work times prediction*

For this experiment, the features and data at the ‘household’, ‘person’ and ‘trip’ level are used. As preprocessing for input into TEAM-TDM, we limit trips to those for which the reason given for either the origin or destination was ‘Work’. We then join these, and limit our observations to those for which there was only a single trip to work, and a single trip from work that day. We bin the times by hour on the hour, and combine the time to work and time from work into a single variable. In theory, since the time leaving work is allowed to be the next day, this could result in up to 828 classes. However, we only observe 419 of these in the data.

In order to prevent problems with our train/test splitting procedures, we further limit the data to classes whose observations number at least 10. This reduces the dataset to 53937 observations of 250 independent variables, with 222 target classes. This data also includes a weights column per trip, which is a function of the person weights, and which we include as weights during training. Results are summarized in the third section of Table I

V. Conclusions

In this paper we examine the ability of applying a cookie cutter battery of machine learning models to classification problems in transportation for the purpose of first-pass evaluation of the performance of various model families. The tool is hardly a panacea for modeling, and is not designed to be. Notably, there is no consideration for messy data, unbalanced data, the effects of outliers or all possible model tuning configurations. Even so, we show that, without modification, the same modeling pipeline can be applied to a variety of classification problems, and that this pipeline can aid the choice of an appropriate model family for a particular problem. Furthermore, we show that these models can be stacked to produce a model with superior predictive power, at the expense of interpretability. We provide this modeling pipeline as a tool for transportation engineers and researchers to further investigate the ability of various machine learning algorithms to successfully model transportation problems.

TABLE I
MODEL SCORES

	RF	MNL	MLP	NB	Dummy	OP	NMNL	Stacked	
accuracy	1.000	0.967	1.000	1.000	0.467			0.933	NHTS vehicle count
weighted f1	1.000	0.966	1.000	1.000	0.469			0.929	
weighted precision	1.000	0.969	1.000	1.000	0.510			0.942	
weighted recall	1.000	0.967	1.000	1.000	0.467			0.933	
macro f1	1.000	0.957	1.000	1.000	0.462			0.910	
macro precision	1.000	0.976	1.000	1.000	0.478			0.956	
macro recall	1.000	0.944	1.000	1.000	0.497			0.889	
mean log loss	0.048	0.166	0.163	0.023	21.875			0.089	
macro MAMSE	0.000	0.244	0.000	0.000	1.503			0.487	
weighted MAMSE	0.000	0.067	0.000	0.000	0.400			0.133	
training time	146.917	0.002	549.334	0.002	0.001			709.604	
accuracy	0.637	0.606	0.637	0.614	0.244	0.644	0.651	0.658	NHTS vehicle count
weighted f1	0.600	0.567	0.602	0.602	0.244	0.618	0.629	0.637	
weighted precision	0.627	0.568	0.576	0.602	0.243	0.625	0.623	0.640	
weighted recall	0.637	0.606	0.637	0.614	0.244	0.644	0.651	0.658	
macro f1	0.206	0.194	0.202	0.229	0.072	0.230	0.227	0.241	
macro precision	0.285	0.216	0.198	0.247	0.072	0.266	0.244	0.304	
macro recall	0.204	0.197	0.209	0.222	0.073	0.221	0.223	0.234	
mean log loss	1.051	1.066	1.108	1.996	25.352	1.049	1.045	1.039	
macro MAMSE	10.015	32.054	9.458	25.644	4.667	15.207	8.693	14.374	
weighted MAMSE	0.349	0.330	0.228	0.203	0.048	0.282	0.217	0.220	
training time	268.247	289.164	6701.169	0.620	0.066	147.305	18.799	12238.811	
accuracy	0.212	0.572	0.626	0.260	0.032			0.593	NHTS work schedule
weighted f1	0.149	0.560	0.599	0.300	0.030			0.577	
weighted precision	0.214	0.571	0.585	0.438	0.029			0.587	
weighted recall	0.212	0.572	0.626	0.260	0.032			0.593	
macro f1	0.024	0.290	0.252	0.115	0.004			0.294	
macro precision	0.064	0.334	0.242	0.179	0.004			0.339	
macro recall	0.025	0.292	0.286	0.142	0.004			0.292	
mean log loss	3.656	3.942	1.968	20.363	33.417			12.839	
macro MAMSE	211.669	152.784	279.702	1116.607	250.511			154.545	
weighted MAMSE	1.146	0.235	0.348	0.886	0.304			0.249	
training time (s)	1383.772	2214.177	181095.857	10.359	1.171			194215.423	

We also demonstrate that, amongst the vast array of classification problems tackled by typical transportation simulations, there exist situations where standard model families such as logit models are inferior to other model families. Since these classification problems typically number in the dozens for a typical transportation simulation, we conclude that a point-and-click method for evaluation of different model families will be of great benefit as future simulations become more complex and include even more variables.

In the future, extention of the proposed pipeline to tackle regression targets, clustering targets and mixed discrete continuous targets would aid in evaluating model families in most scenarios arising out of transportation simulations. Further additions might include more model families, and extension to applications beyond travel demand modeling. Additionally, although the models in the proposed pipeline have a degree of tuning, it is unclear to what degree their performance differs from models chosen via fine manual tuning. For example feature selection for MNL problems is often an involved manual process, and further study is necessary to determine how such considerations might improve the performance of the individual models in TEAM-TDM. Lastly, if the objective is to find the best possible model, it is not clear that we need to train all available model families on all available data. Research into algorithms that decide which model families warrant the dedication of hardware time would help choose where to focus limited resources.

REFERENCES

[1] A. Horni, K. Nagel, and K. W. Axhausen, *The multi-agent transport simulation MATSim*. Ubiquity Press London, 2016.

[2] C. Sheppard, R. Waraich, A. Campbell, A. Pozdnukhov, and A. Gopal, "Modeling plug-in electric vehicle charging demand with beam," 2017.

[3] A. Pinjari, N. Eluru, S. Srinivasan, J. Y. Guo, R. B. Copperman, I. N. Sener, and C. R. Bhat, "Cemdap: Modeling and microsimulation frameworks, software development, and verification," in *Proceedings of the Transportation Research Board 87th Annual Meeting, Washington DC*, 2008.

[4] M. G. Karlaftis and E. I. Vlahogianni, "Statistical methods versus neural networks in transportation research: Differences, similarities and some insights," *Transportation Research Part C: Emerging Technologies*, vol. 19, no. 3, pp. 387–399, 2011.

[5] Y. Zhang and Y. Xie, "Travel mode choice modeling with support vector machines," *Transportation Research Record: Journal of the Transportation Research Board*, no. 2076, pp. 141–150, 2008.

[6] F. S. Koppelman and C. Bhat, "A self instructing course in mode choice modeling: multinomial and nested logit models," 2006.

[7] C. Xie, J. Lu, and E. Parkany, "Work travel mode choice modeling with data mining: decision trees and neural networks," *Transportation Research Record: Journal of the Transportation Research Board*, no. 1854, pp. 50–61, 2003.

[8] P. C. Vythoulkas and H. N. Koutsopoulos, "Modeling discrete choice behavior using concepts from fuzzy set theory, approximate reasoning and neural networks," *Transportation Research Part C: Emerging Technologies*, vol. 11, no. 1, pp. 51–73, 2003.

[9] J. Hagenauer and M. Helbich, "A comparative study of machine learning classifiers for modeling travel mode choice," *Expert Systems with Applications*, vol. 78, pp. 273–282, 2017.

[10] A. Mohammadian and E. Miller, "Nested logit models and artificial neural networks for predicting household automobile choices: comparison of performance," *Transportation Research Record: Journal of the Transportation Research Board*, no. 1807, pp. 92–100, 2002.

[11] K. Park, M. Bell, I. Kaparias, and K. Bogenberger, "Learning user preferences of route choice behaviour for adaptive route guidance," *IET Intelligent Transport Systems*, vol. 1, no. 2, pp. 159–166, 2007.

[12] M. Allahviranloo and W. Recker, "Daily activity pattern recognition by using support vector machines with multiple classes," *Transportation Research Part B: Methodological*, vol. 58, pp. 16–43, 2013.

[13] S. D. Clark, "The determinants of car ownership in england and wales from anonymous 2001 census data," *Transportation research part C: emerging technologies*, vol. 17, no. 5, pp. 526–540, 2009.

[14] J. Berkson, "Application of the logistic function to bio-assay," *Journal of the American Statistical Association*, vol. 39, no. 227, pp. 357–365, 1944.

[15] D. McFadden, "Modeling the choice of residential location," *Transportation Research Record*, no. 673, 1978.

[16] J. R. Quinlan, "Induction of decision trees," *Machine learning*, vol. 1, no. 1, pp. 81–106, 1986.

[17] L. Breiman, J. H. Friedman, R. A. Olshen, and C. J. Stone, *Classification and Regression Trees*. Monterey, CA: Wadsworth and Brooks, 1984.

[18] T. K. Ho, "Random decision forests," in *Document analysis and recognition, 1995., proceedings of the third international conference on*, vol. 1, pp. 278–282, IEEE, 1995.

[19] D. E. Rumelhart, G. E. Hinton, and R. J. Williams, "Learning internal representations by error propagation," tech. rep., California Univ San Diego La Jolla Inst for Cognitive Science, 1985.

[20] E. Frank and M. Hall, "A simple approach to ordinal classification," in *European Conference on Machine Learning*, pp. 145–156, Springer, 2001.

[21] G. Louppe, L. Wehenkel, A. Sutera, and P. Geurts, "Understanding variable importances in forests of randomized trees," in *Advances in neural information processing systems*, pp. 431–439, 2013.

[22] M. Stone, "Cross-validatory choice and assessment of statistical predictions," *Journal of the royal statistical society. Series B (Methodological)*, pp. 111–147, 1974.

[23] J. Snoek, H. Larochelle, and R. P. Adams, "Practical bayesian optimization of machine learning algorithms," in *Advances in neural information processing systems*, pp. 2951–2959, 2012.

[24] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay, "Scikit-learn: Machine learning in Python," *Journal of Machine Learning Research*, vol. 12, pp. 2825–2830, 2011.

[25] C. R. Bhat and V. Pulugurta, "A comparison of two alternative behavioral choice mechanisms for household auto ownership decisions," *Transportation Research Part B: Methodological*, vol. 32, no. 1, pp. 61–75, 1998.