Modeling Mandatory Lane Changing Using Bayes Classifier and Decision Trees

Yi Hou, Praveen Edara, and Carlos Sun

Abstract—A lane changing assistance system that advises drivers of safe gaps for making mandatory lane changes at lane drops is developed. Bayes Classifier and Decision Tree methods were applied to model lane changes. Detailed vehicle trajectory data from the Next Generation Simulation (NGSIM) dataset were used for model development (US Highway 101) and testing (Interstate 80). The model predicts driver decisions on whether to merge or not as a function of certain input variables. The best results were obtained when both Bayes and Decision Tree classifiers were combined into a single classifier using a majority voting principle. The prediction accuracy was 94.3% for non-merge events and 79.3% for merge events. In a lane change assistance system, the accuracy of non-merge events is more critical than merge events. Misclassifying a non-merge as a merge event could result in a traffic crash while misclassifying a merge event as a non-merge event would only result in a lost opportunity to merge. Sensitivity analysis performed by assigning higher misclassification cost for non-merge events resulted in even higher accuracy for non-merge events but lower accuracy for merge events.

Index Terms—Intelligent Transportation System, Bayesian Methods, Decision Trees, Driver Behavior, Lane Changing Assistance

I. INTRODUCTION

With the increase in deployment of sensor technology in automobiles, driver assistance systems such as adaptive cruise control, collision avoidance system, and lane departure warning system have become a reality in the recent years. In terms of lane changing assistance, the current technology focuses primarily on blind spot identification and warning. Limited research exists on other forms of lane changing assistance systems. In this paper, a lane changing assistance system that advises drivers of safe and unsafe gaps for making mandatory lane changes is developed.

Lane changing models describe drivers’ lane changing behaviors under various traffic conditions. These models are an essential component of microscopic traffic simulation and have been extensively studied in the literature. Much of the literature on lane change models is based on gap acceptance. A driver makes a lane change when both the lead and the lag gaps in the target lane are acceptable. Given the assumption on the distribution of critical lead and lag gap lengths, various gap acceptance models were built in 1960s and 1970s. Herman and Weiss [1] assumed an exponential distribution for critical gap, Drew et al. [2] assumed lognormal distribution, and Miller [3] assumed a normal distribution. Daganzo [4] modeled driver’s merging from the minor leg of a stop controlled T-intersection to the major leg using a Probit model. Gipps [5] designed a lane changing model that was implemented in a microscopic traffic simulator. Kita [6] modeled driver’s merging behavior from freeway on-ramp using a Logit model for gap acceptance. Yang and Koutsopoulos [7] established a rule-based lane changing model that was incorporated into the microscopic simulator MITSIM. Ahmed et al. [8] developed a generic lane changing model that captures lane changing behavior under both mandatory and discretionary lane changes. Kita [9] also developed a game theoretic lane changing model. A two-person non-zero non-cooperative game was developed to model the interaction of drivers in the target lane and the merging lane. Hidas [10] used intelligent agent based techniques to model driver’s lane changing behavior and implemented the model in the ARTEMiS traffic simulator. Toledo et al. [11] proposed an integrated driving behavior model that captures both lane changing and acceleration behaviors. Recently, Meng and Weng [12] used statistical methods such as the classification and regression tree (CART) to predict merging behavior near work zone tapers. In a recent study [13] the authors developed a genetic fuzzy model to predict merging behavior of drivers at lane drops.

In summary, several types of lane changing models have been proposed in the literature with the main goal of developing accurate traffic simulation models. However, none of these models were intended for use in a real-time lane changing assistance system that advises drivers of when it is safe or unsafe to merge. One main difference between the simulation and the lane change assistance system applications is the relative importance of misclassification between a merge and a non-merge decision. In a simulation model, the effect of a non-merge event misclassified as a merge event only affects the mobility measures. The same misclassification, however, in a lane change assistance system could impact traffic safety significantly. Thus, any model of...
lane change targeted for use in vehicles as part of an assistance system must give more importance to not misclassifying non-merge events as merge events. On the other hand, misclassifying a merge event as a non-merge event would result in a lost opportunity to merge but would not have a negative safety effect. Many of the models proposed in the literature are not appropriate for this new application.

In this paper, Bayes classifier and Decision Tree methods are applied to develop models for mandatory lane changes at lane drops. Both methods have been applied extensively in machine learning systems built for decision making in many disciplines. They have several advantages for modeling lane changing. Both of them relax the assumptions of traditional lane changing model’s mathematical forms and variable distributions. Therefore, they can mimic the complex nonlinear nature of driver’s lane changing behavior more realistically. One additional advantage of Bayes classifier is its ability to take into account the cost of misclassification. In a Bayes classifier, it is possible to assign a higher cost of misclassification to non-merge events than merge events.

Bayes classifier and Decision Tree models were developed using the same training and validation data. Then, both classifiers were combined into a single hybrid classifier. The combined classifier when tested on a new dataset from a different highway segment outperformed the individual classifiers in terms of the accuracy of non-merge events. In this paper, mandatory lane changes at lane drops refer only to those executed by traffic entering from a ramp. The lane changes made by vehicles exiting the mainline, although also mandatory, were not included in this study. Discretionary lane changes performed when drivers perceive the driving conditions in the target lanes to be better are also beyond the scope of this study.

II. DATA

A. Data Reduction

In this study, traffic data provided by the Federal Highway Administration’s (FHWA) Next Generation Simulation (NGSIM) project [14] was used to build the lane changing models. NGSIM dataset is an open source dataset that has been used in previous research for simulation model development and testing ([15, 16]). NGSIM data include vehicle trajectories on a segment of southbound US Highway 101 (Hollywood Freeway) in Los Angeles, California and a segment of Interstate 80 in San Francisco, California. US Highway 101 data was collected for 45 minutes from 7:50 a.m. to 8:35 a.m., on June 15, 2005. Interstate 80 data was also collected for 45 minutes from 4:00 p.m. to 4:15 p.m. and from 5:00 p.m. to 5:30 p.m., on April 13, 2005. Both datasets represents two traffic states – conditions when congestion was building up (period of the first 15 minutes) denoted as the transition period and congested conditions (period of the remaining 30 minutes). Table I shows the aggregate speed and volume statistics of NGSIM dataset for every 15 minutes. For the congested period, the flows and speeds both decreased. As depicted in Figure 1, the study segment of US Highway 101
was located between an on-ramp and off-ramp and was 2100 feet long with five freeway lanes and an auxiliary lane. The study segment of Interstate 80 was 1650 feet in length, and also had five freeway lanes and an auxiliary lane, and one on-ramp.

Past research studies [18, 19, 20, 21] have shown that NGSIM speed measurements exhibit noises (random errors). Data smoothing techniques such as moving average [21], Kalman filtering [22] and Kalman smoothing [23] have been used to improve speed data quality. In this paper, the moving average method was adopted to smooth the speed measurements.

The longitudinal and lateral coordinates, speed, acceleration, and headway for each vehicle were obtained from trajectory data at a resolution of 10 frames per second. Given the focus of this study on mandatory lane changes, only trajectory data of vehicles in the auxiliary lane and the adjacent lane were used for model development. Hereafter, the auxiliary lane is referred to as the merge lane and the adjacent lane as the target lane. The speed and position of each vehicle were identified in 1-second intervals. The 1-second interval produced data with comparable sample sizes for both lane changing and non-lane changing events. Other researchers [12] have also used a 1-second interval for analyzing lane changing behavior of drivers. Since it is impossible to determine the intent of the driver using trajectory data alone, the observed behavior of drivers is modeled. During every 1-second interval, a driver’s behavior is identified as either merge or no-merge. Merge events occurred when a vehicle’s lateral coordinate began to shift toward the adjacent target lane direction without oscillations. Otherwise it was deemed as a non-merge event. A single driver could participate in several non-merge events but only one merge event.

A total of 686 observations were obtained from US Highway 101, 373 of them being non-merge and 313 of them being merge events. As discussed in Hastie et al. [24], there is no general rule on how many observations should be assigned to training and validation. In order to obtain high accuracy, a large training data size is required. Other studies have used 80% of the dataset for training and 20% for validating the model [25, 26]. Base on these studies, the dataset was divided into two groups – 80% of observations were used for training and 20% were used for validation. The model was tested using the Interstate 80 dataset consisting of 667 observations, 459 of them being non-merge and 208 of them being merge events.

B. Input Variables

At any given instant, a driver traveling in the merge lane assesses traffic conditions in both the target lane and the merge lane in order to decide whether to merge or not. Several factors may affect a driver’s lane changing decision. In this study, five factors or dimensions that were found to affect a driver’s merging decision in previous studies [8, 10] are considered as input variables for the models. These factors are shown in Figure 2 and defined below.

- \( \Delta V_{\text{lead}} (\text{m/s}) \): The speed difference between the lead vehicle in the target lane and the merging vehicle, in feet per second. \( \Delta V_{\text{lead}} \) can be expressed as
  \( \Delta V_{\text{lead}} = V_{\text{lead}} - V_{\text{merge}} \),
  where \( V_{\text{lead}} \) is the speed of the lead vehicle and \( V_{\text{merge}} \) is the speed of merge vehicle.

- \( \Delta V_{\text{lag}} (\text{m/s}) \): The speed difference between the lag vehicle in the target lane and the merging vehicle, in feet per second. \( \Delta V_{\text{lag}} \) can be expressed as
  \( \Delta V_{\text{lag}} = V_{\text{lag}} - V_{\text{merge}} \),
  where \( V_{\text{lag}} \) is the speed of the lag vehicle.

- \( D_{\text{lead}} (\text{m}) \): The gap distance between the lead vehicle in the target lane and the merging vehicle, in feet.

- \( D_{\text{lag}} (\text{m}) \): The gap distance between the lag vehicle in the target lane and the merging vehicle, in feet.

- \( S (\text{m}) \): The distance from the merging vehicle to the beginning of the merge lane.

\[ P(y_1|x) = \frac{P(x|y_1)P(y_1)}{P(x)}, \quad i = 1,2 \] (1)

III. METHODOLOGY

A. Bayes Classifier

1) Bayes Decision Theory

Let \( y_1, y_2 \) denote the merge and non-merge classes. According to the Bayesian classification rule [27],
where \( \mathbf{x} \) is the input vector, \( P(\cdot) \) is the probability, and \( p(\cdot) \) is the probability density function. The Bayes classification rule [24] is stated as follows:

- If \( P(Y_1|\mathbf{x}) > P(Y_2|\mathbf{x}) \), \( \mathbf{x} \) is classified to \( Y_1 \).
- If \( P(Y_1|\mathbf{x}) < P(Y_2|\mathbf{x}) \), \( \mathbf{x} \) is classified to \( Y_2 \).
- If \( P(Y_1|\mathbf{x}) = P(Y_2|\mathbf{x}) \), \( \mathbf{x} \) can be assigned to either \( Y_1 \) or \( Y_2 \).

Using (1), the classification decision is equivalently based on the inequalities

\[
p(\mathbf{x}|Y_1)P(Y_1) > (\leq)p(\mathbf{x}|Y_2)P(Y_2)
\]

2) **Risk of Misclassification**

Risk considers both the likelihood of misclassification and the cost of the misclassification. A penalty term \( \lambda_X \) denotes the cost of misclassifying \( \mathbf{x} \) to a wrong class \( Y_i \) while belonging to class \( Y_k \) [27]. In order to minimize the average risk, the classification decision inequalities (2) become

\[
(\lambda_{12} - \lambda_{11}) p(\mathbf{x}|Y_1)P(Y_1) > (\leq)(\lambda_{21} - \lambda_{22}) p(\mathbf{x}|Y_2)P(Y_2)
\]

Adopting the assumption that \( \lambda_{ij} > \lambda_{il} \) and \( \lambda_{il} = 0 \), the Bayes classification rule becomes

\[
\mathbf{x} \text{ belongs to } Y_1(Y_2) \text{ if } l_{12} = \frac{p(\mathbf{x}|Y_1)}{p(\mathbf{x}|Y_2)} > (\leq) \frac{p(Y_2)\lambda_{21}}{p(Y_1)\lambda_{12}}
\]

where \( l_{12} \) is likelihood ratio.

3) **k Nearest Neighbor Density Estimation**

A driver’s merging behavior can be predicted using the class-conditional probability density function, \( p(\mathbf{x}|Y_1) \). In this study, the k Nearest Neighbor (kNN) density estimation method [28] is used to estimate the class-conditional probability density functions. The kNN estimation method was chosen because similar to kernel estimation, it is a non-parametric method; thus, there is no need to assume a distributional form unlike maximum likelihood. By using this method, the class-conditional probability density functions is estimated as

\[
p(\mathbf{x}|Y_1) = \frac{k}{N|V_i|}, \quad i = 1,2
\]

where \( N_i \) is the total number of training samples in class \( Y_i \), and \( V_i \) is the volume of the five-dimensional hypersphere (i.e. input data space) centered at \( \mathbf{x} \) that contains \( k \) points from class \( Y_i \).

\[
P(Y_i) \text{ is easily estimated from observations as follows:}
\]

\[
P(Y_i) = \frac{N_i}{N}, \quad i = 1,2
\]

where \( N_i \) is the total number of training samples in class \( Y_i \), and \( N \) is the total number of training samples.

By substituting equation (5) and (6) into equation (4), Bayes classification rule is equivalent to

\[
\mathbf{x} \text{ belongs to } Y_1(Y_2) \text{ if } l_{12} = \frac{k}{N|V_i|} > (\leq) \frac{\lambda_{21}}{\lambda_{12}}
\]

Let \( r_i \) denote the radius of the hypersphere centered at \( \mathbf{x} \) that contains \( k \) points from class \( Y_i \). Since hypersphere dimension in this study is five (the total number of input variables), the likelihood ratio can be computed as

\[
l_{12} = \frac{V_2}{V_1} = \left(\frac{r_2}{r_1}\right)^5
\]

4) **Distance Measurement**

The hypersphere radius \( r_i \) can be easily obtained by searching for the \( k \)th nearest distance from all the training vectors of class \( Y_i \). The weighted distance measure was used to calculate hypersphere radius. Let \( \mathbf{x}_j \) and \( \mathbf{x}_k \) denote two vectors of \( l \) features. The weighted distance is:

\[
D(\mathbf{x}_j, \mathbf{x}_k) = \sqrt{\sum_{i=1}^{l}w_i(\mathbf{x}_j - \mathbf{x}_k)^2}
\]

where \( w_i \) is the weights associated with features. The maximum margin decision boundary established by support vector machines (SVMs) is used to determine the weights [30]. Let \( \mathbf{q} \) be the query point whose class label is to be predicted. SVMs classifier gives decision hyperplane \( g(\mathbf{x}) \). Let \( \mathbf{p} \) be the point with the closest Euclidean distance to \( \mathbf{q} \) on decision hyperplane \( g(\mathbf{x}) \). \( R(\mathbf{q})_j \) is defined as

\[
R(\mathbf{q})_j = [\mathbf{e}_j^T \nabla g(\mathbf{p})]
\]

where \( \mathbf{e}_j \) denote the canonical unit vector along input feature \( j \). The weights are given by

\[
w(\mathbf{q})_j = \frac{(R(\mathbf{q})_j)^t}{\sum_{t=1}^{t}(R(\mathbf{q})_j)^t}
\]

where \( t \) is a positive integer. In this study, \( t \) values ranging from 1 to 4 were applied and \( t = 2 \) produced the best model performance. In this case, the SVMs decision hyperplane is in linear form \( g(\mathbf{x}) = \mathbf{b}^T \mathbf{x} + b_0 = 0 \), Thus, \( R(\mathbf{q})_j \equiv b_j \).

**B. Decision Tree Model**

A decision tree achieves a classification decision by performing a sequence of tests on feature vectors along a path of nodes [31]. Each internal node in the tree provides a question, “Is feature \( X_i \geq a \)?”, where \( a \) is a threshold value. The binary answer to the question corresponds to a descendant node. At the end, each terminal node returns a class label. The size of a decision tree is the key factor in developing the decision tree model. If the size of a tree is too small, the tree results in high misclassification rates. On the other hand, if a tree grows too large, it could overfit the training data and perform poorly on testing data. Therefore, the suggested approach is to grow a tree with a large enough size and then prune branches according to a set of pruning rules.

1) **Node Splitting**

In order to construct a decision tree, the set of questions at tree nodes are to be determined. Each node \( t \) is associated with a subset of training set \( \mathbf{X}_t \). The root node is assigned with the
The goal of the binary split at each node is to produce subsets that are more homogeneous or purer than the parent subset. In this model, Shannon’s information theory [32] is adopted to measure the impurity of subset \( X_t \), also known as node impurity.

Let \( y_1, y_2 \) denote the two classes: merge and non-merge. Let \( P(y_i | t) \) denote the probability that a sample in subset \( X_t \) belongs to class \( y_i, i = 1,2 \). Node impurity is then defined as

\[
I(t) = -\sum_{i=1}^2 P(y_i | t) \log_2 P(y_i | t)
\] (12)

\( P(y_i | t) \) can be easily estimated by \( N_{iy} / N_t \), where \( N_{iy} \) is the number of vectors in subset \( X_t \) that belongs to class \( y_i \), and \( N_t \) is the total number of vectors in subset \( X_t \). After performing a binary split at node \( t \), a subset \( X_{ty} \) with an answer “Yes” is assigned to node \( t_y \), and a subset \( X_{tn} \) with answer “No” is assigned to node \( t_N \). The decrease in node impurity \( \Delta I(t) \) is given by

\[
\Delta I(t) = I(t) - \frac{N_{iy}}{N_t} I(t_y) - \frac{N_{tn}}{N_t} I(t_N)
\] (13)

where \( N_{iy} \) and \( N_{tn} \) are the numbers of vectors in subsets \( X_{ty} \) and \( X_{tn} \). By exhaustively searching for all candidate questions, the one that leads to the maximum impurity decrease is chosen.

2) Stop-Splitting Criteria and Class Assignment

A threshold probability value \( P_0 \) is necessary to stop the node splitting process at any node. Splitting stops when more than \( P_0 \times 100\% \) of vectors in the subset belong to any one single class, i.e., \( \max_i P(y_i | t) > P_0 \). In this model, 0.9 is selected to be the threshold value as this value will also ensure the tree grows large enough for pruning. Once a terminal node is determined, the class label is given by \( y_j \) where

\[
j = \arg \max_i P(y_i | t)
\] (14)

3) Tree Pruning

Minimal cost-complexity pruning [33] is employed as the pruning rule in this paper. Due to its computational efficiency the minimal cost-complexity pruning is one of the most common methods to prune a decision tree. The sequence of subtrees generated by this pruning process is nested meaning that the nodes that were previously cut off will not reappear in subsequent subtrees. The cost-complexity measure \( R_\alpha (T) \) of decision tree \( T \) is defined as

\[
R_\alpha (T) = R(T) + \alpha |T|
\] (15)

where \( R(T) \) is the substitution estimate for the overall misclassification rate of tree \( T \), \( \alpha \geq 0 \) is the complexity parameter, and \( |T| \) is the total number of terminal nodes in tree \( T \). Each value of \( \alpha \) is associated with a subtree \( T(\alpha) \) that minimizes \( R_\alpha (T) \). As \( \alpha \) increase from 0 to a sufficiently large number, the size of \( T(\alpha) \) decrease from its largest size to the smallest size (only for the root node). If a subtree that minimizes \( R_\alpha (T) \) for a given value of \( \alpha \), it will remain minimizing \( R_\alpha (T) \) until \( \alpha \) increases to a jump point. Let \( \{\alpha_k\} \) be the increasing sequence of the jump points. For any \( \alpha_k \leq \alpha \leq \alpha_{k+1} \), \( T(\alpha) = T(\alpha_k) = T_k \). Finally, a sequence of minimal cost-complexity trees \( \{T_k\} \) are generated. The right sized tree \( T^* \) can be selected by test sample estimates

\[
R^{ts}(T^*) = \min_k R^{ts}(T_k)
\] (16)

where \( R^{ts}(\cdot) \) denotes misclassification rate for test sample.

C. Combining Classifiers

Majority-voting rule [27] is used as the combination rule to combine both Bayes classifier and Decision Tree methods. The majority-voting rule was chosen due to its robust performance in previous research. The majority-voting rule is simple to apply and belongs to the class of hard type combination rules. Let \( L \) denote the number of classifiers, the majority-voting rule is stated as follows:

- If \( L \) is odd, the unknown pattern is classified to a class when at least \( \frac{L+1}{2} \) of classifiers agree on the class label.
- If \( L \) is even, the unknown pattern is classified to a class when at least \( \frac{L}{2} + 1 \) of classifiers agree on the class label.

In this paper, a vehicle will merge (class) only if both Bayes classifier and Decision Tree agree on the decision to merge (same class label). Thus, the combined classifier is more conservative than either of the individual classifiers it’s constructed from. This is a valuable attribute for safety applications such as the lane change assistance system where non-merge decisions are more critical and erring on conservativeness is safer.

IV. RESULTS

A. Bayes Classifier

For the Bayes classifier the weights were first estimated using SVMs. The estimated weights shown in Table II reveals that \( \Delta V_{lead} \) has the largest weight, which indicates \( \Delta V_{lead} \) is the most relevant feature in classifying the merge and non-merge events, indicating that a slight change in \( \Delta V_{lead} \) may greatly change the distance. Speed differences \( \Delta V_{lag} \), and lag gap \( D_{lag} \) are more relevant than lead gap \( D_{lead} \) and lag gap \( D_{lag} \). The distance from the beginning of the merge (auxiliary) lane, \( S \), turned out to be the least relevant feature.

A Bayes classifier was developed from \( k = 3 \) and \( \frac{4\alpha}{\alpha} = 1 \). The model’s prediction accuracy on validation and test data.
for merge and non-merge events is shown in Table III. For the test data the accuracy of merge events was high (92.3%) but it was only 79.3% for the critical non-merge events.

### B. Decision Tree Model

A decision tree with 62 terminal nodes was constructed using training data before pruning. After applying the pruning rules, a sequence of 16 minimal cost-complexity trees were generated. The total numbers of terminal nodes $|T_k|$ are shown in Table IV.

The relationship between total number of terminal nodes $|T_k|$ and estimated misclassification rate for both training and testing data is presented in Figure 3. In Figure 3 the estimated misclassification rate for training data $R(T_k)$ decreases sharply as the tree initially increases in size and then decreases slowly. The estimated misclassification rate for testing data $R^*$ decreases sharply initially, but after reaching its minimum value at 18 terminal nodes, the rate begins to climb as tree size grows. Thus, the tree $T_k$ with 18 terminal nodes was selected as the right size decision tree model for predicting merge and non-merge events.

![Fig. 3. Relationship between total number of terminal nodes and misclassification rate.](image)

The tree structure is presented in Figure 4, where terminal nodes are represented by shaded squares and decision nodes are represented by circles. Number of observations, class labels, and prediction accuracies for terminal nodes are displayed beneath them. Node 1 was first split by using the relative speed between the lead and merging vehicles, $\Delta V_{\text{lead}}$. This result further supports the finding from the Bayes classification model that $\Delta V_{\text{lead}}$ is the most relevant driver feature in making merging decisions. The decision making process of the decision tree model is intuitive. For example, as shown by terminal node $t_8$, a driver merges if the merging vehicle is slower ($\Delta V_{\text{lead}} \geq 0 \text{ m/s}$) or slightly faster ($0 > \Delta V_{\text{lead}} \geq -2.7 \text{ m/s}$) than the lead vehicle and both the lead and lag gap is large ($D_{\text{lag}} = 2.4 \text{ m}, D_{\text{lead}} = 7.6 \text{ m}$). In contrast, terminal node $t_7$ is interpreted in natural language as: if the merging vehicle is much faster ($\Delta V_{\text{lead}} < -2.7 \text{ m/s}$) than the lead vehicle and the lead gap is small ($D_{\text{lead}} < 8.9 \text{ m}$), then driver does not merge. For terminal node $t_{14}$, if the merging vehicle speed is much greater ($\Delta V_{\text{lead}} \geq -2.7 \text{ m/s}$) than the lead vehicle; lead gap is large ($D_{\text{lead}} \geq 8.9 \text{ m}$); distance from the beginning of the merge lane is far ($S \geq 138.7 \text{ m}$); and even lag gap is not too large ($D_{\text{lag}} \geq 0.76 \text{ m}$); driver decides to merge, because as driver approaches the end of merge lane his or her merge behavior become more aggressive. These rules generated by decision tree are representative of everyday driving experiences. The prediction results of decision tree are presented in Table III. The accuracy of both merge and non-merge events for test data was above 80%. However, a higher accuracy for non-merge events is desirable for a lane changing assistance system.

### C. Combining Classifiers

The Bayes classifier and Decision Tree models were combined using the majority voting rule. The resulting model was tested using the test data and the results are shown in

<table>
<thead>
<tr>
<th>Decision</th>
<th>Validation data</th>
<th>Test data</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Bayes Classifier</td>
<td>Decision Tree</td>
</tr>
<tr>
<td></td>
<td>Observations</td>
<td>Accuracy</td>
</tr>
<tr>
<td>Non-merge</td>
<td>73</td>
<td>82.2%</td>
</tr>
<tr>
<td>Merge</td>
<td>56</td>
<td>91.1%</td>
</tr>
</tbody>
</table>

### Table III: Accuracies of Different Models

| Tree | $|T_k|$ | Validation data | Test data |
|------|------|-----------------|-----------|
| $T_1$ | 62 | | |
| $T_2$ | 58 | | |
| $T_3$ | 53 | | |
| $T_4$ | 46 | | |
| $T_5$ | 29 | | |
| $T_6$ | 27 | | |
| $T_7$ | 22 | | |
| $T_8$ | 18 | | |
| $T_9$ | 12 | | |
| $T_{10}$ | 10 | | |
| $T_{11}$ | 9 | | |
| $T_{12}$ | 7 | | |
| $T_{13}$ | 5 | | |
| $T_{14}$ | 3 | | |
| $T_{15}$ | 2 | | |
| $T_{16}$ | 1 | | |
Table III. The accuracy for non-merge improved to 94.3% while the accuracy for merge events dropped slightly to 79.3%. As previously discussed, the accuracy of non-merge events is more critical than the merge events for a safety application like the lane change assistance system. Misclassifying a merge event as a non-merge event would result in a lost opportunity to merge but would not have a negative safety effect. The misclassification parameter $\lambda_{12}$ can be adjusted to give greater weight to the prediction accuracy of non-merge events. For illustration, the model performance for $\lambda_{12} = 2$ and 5 is shown in Table V. The prediction accuracy of non-merge events increased while the accuracy of merge events decreased since the model becomes more conservative.

D. Performance of Other Models

Two models from the literature, Genetic Fuzzy model and a Binary Logit model, were also evaluated. They were estimated using the same dataset and the same set of variables. The coefficients of the Binary Logit model are presented in Table VI. For Genetic Fuzzy System, a total of 120 rules were generated from the training data. The performance of the two models for the test data is shown in Table VII. Both models performed poorly compared to the classifier models developed in the study. The low accuracy of non-merge events is a concern for their use in real-time lane changing assistance systems. However, it is noted that the estimated Logit model is a Binary Logit model and is based on existing research. In the future, advanced discrete choice models could be developed to increase the prediction accuracies.

<table>
<thead>
<tr>
<th>Variable</th>
<th>Coefficient</th>
<th>$p$-value</th>
</tr>
</thead>
<tbody>
<tr>
<td>$\Delta V_{lead}(\text{m/s})$</td>
<td>0.163</td>
<td>&lt;.0001</td>
</tr>
<tr>
<td>$\Delta V_{lag}(\text{m/s})$</td>
<td>0.070</td>
<td>0.0043</td>
</tr>
<tr>
<td>$D_{lead}$ (m)</td>
<td>0.061</td>
<td>&lt;.0001</td>
</tr>
<tr>
<td>$D_{lag}$ (m)</td>
<td>0.003</td>
<td>0.4721</td>
</tr>
<tr>
<td>$S$ (m)</td>
<td>-0.004</td>
<td>&lt;.0001</td>
</tr>
<tr>
<td>Intercept</td>
<td>1.967</td>
<td>&lt;.0001</td>
</tr>
</tbody>
</table>

* Not significant at 0.05 significance level

TABLE VII

<table>
<thead>
<tr>
<th>Decision</th>
<th>Observed</th>
<th>Accuracy</th>
<th>Accuracy</th>
</tr>
</thead>
<tbody>
<tr>
<td>Genetic Fuzzy</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Non-merge</td>
<td>459</td>
<td>73.6%</td>
<td>20.9%</td>
</tr>
<tr>
<td>Merge</td>
<td>208</td>
<td>71.6%</td>
<td>95.7%</td>
</tr>
</tbody>
</table>

V. CONCLUSIONS AND CONTRIBUTIONS

In this paper, models for mandatory lane changes at lane drops were developed using Bayes Classifier and Decision Tree methods. The publicly available NGSIM vehicle trajectory dataset that consists of traffic conditions approaching congestion and congested conditions was used for model development and testing. The model employed factors such as vehicle speeds relative to lead and lag vehicles in the target lane, lead and lag gap distances, and the distance from the beginning of merge lane. Previous research focused on developing models for use in microscopic simulation whereas the current study focused on the design of a lane changing assistance system. One main difference between the simulation and the lane change assistance system applications is the relative importance of misclassification between a merge and a non-merge decision. In a simulation model, the effect of a non-merge event misclassified as a merge event only affects the mobility measures. The same misclassification, however, in a lane change assistance system would likely result in a traffic crash.

The combined classifier that merges both Bayes classifier and Decision Tree models generated high prediction accuracy for the critical non-merge events. By assigning values of 1, 2,
and 5 to the cost of misclassification, the classifier produced accuracies of 94.3%, 95.4% and 96.7% for non-merge events, and 79.3%, 73.6% and 49.5% for merge events. Cost of misclassification can be treated as a surrogate to driver conservativeness. The greater the cost the more conservative or less aggressive a driver is in pursuing the gap to change lane. As the cost of misclassification increases, the accuracy for non-merge events also increases but the accuracy for merge events decreases. Although the paper illustrated the performance of two other models from literature, the Genetic Fuzzy system and binary Logistic, these models as proposed in the literature were targeted at microscopic simulation. In future research, other modeling techniques can be applied for the design of lane changing assistance system and the accuracies compared with the combined classifier accuracies obtained in this paper.

REFERENCES

Yi Hou received his B.S degree in civil engineering in Southwest Jiaotong University, Chengdu, China, in 2004, and received his M.S degree in civil engineering in University of Missouri-Columbia, MO, in 2011.

He is currently pursuing his Ph.D. degree in civil engineering in University of Missouri-Columbia. His research interests include intelligent transportation system, traffic modeling and simulation, traffic safety, machine learning, and data mining.

Praveen Edara received his Ph.D. degree in transportation systems from Virginia Tech, Blacksburg, VA. He was a Research Scientist with the Virginia Transportation Research Council, and a Research Contractor with the Federal Highway Administration’s Turner Fairbank Highway Research Center.

Since 2007, he has been a faculty at the University of Missouri. His current research interests include traffic modeling, simulation, and ITS.

Carlos Sun received his B.S degree in electrical engineering, M.S. and Ph.D. degrees in civil engineering from the University of California-Irvine and J.D. from the University of Missouri. He is a licensed professional engineer and attorney. His research interests include ITS, safety, work zones and legal issues.