Probabilistic Lane Change Prediction in Highway Scenarios

Master’s Thesis
Embedded and Intelligent Systems

Hector Montemayor Ayala
Probabilistic Lane Change Prediction in Highway Scenarios

School of Information Science, Computer and Electrical Engineering
Halmstad University
Box 823, S-301 18 Halmstad, Sweden

October 23, 2011
Acknowledgement

Give me the serenity to accept the things I cannot change,  
courage to change the things I can  
and the wisdom to tell the difference.  
Reinhold Niebuhr

I would like to thank:

My supervisor Stefan Byttner whose help was fundamental in the development and writing  
of this thesis.

My dad who made possible for me to study my master abroad and my mother for being there  
every minute when I needed her.

Very special thanks to Laura who has been there always encouraging me to continue and to  
ever give up.
Details

First Name, Surname: Hector Montemayor
University: Halmstad University, Sweden
Degree Program: Embedded and Intelligent Systems
Title of Thesis: Probabilistic Lane Change Prediction in Highway Scenarios
Academic Supervisor: Dr. Stefan Byttner

Contact       Details
Hector Montemayor  hecmon@gmail.com
Dr. Stefan Byttner  Stefan.Byttner@hh.se
Abstract

The concern for more energy efficient vehicles has grown in the last years; the increase of oil prices and the need to curb greenhouse gas emissions has called for the search of solutions and alternatives to the fossil-fuel usage.

On one hand there is the search for better fuel sources while on the other, the development for better driver assistance systems. These systems should be capable of maximizing the driver’s security and comfort while minimizing the fuel consumption. This can be achieved by the understanding of the environment surrounding the vehicle. Therefore the assistance should use this knowledge of its environment to alert the driver from any situation of risk so that they can be prevented. Moreover, reducing the number of accidents and unnecessary maneuvers will help reduce the fuel consumption by avoiding emergency breaking and other unnecessary situations.

This thesis explains the development of a maneuver prediction system for highway scenarios. The system should identify lane changing vehicles with enough time to allow the driver to take the necessary precautions and maintain an efficient and more secure driving. To achieve this goal a set of features that describe the environment around the Ego vehicle is obtained. Afterwards, these Features are analyzed using state of the art data mining techniques and their performance is evaluated using a set of classification algorithms (Linear Discriminant Analysis, Neural networks and Random Forests).

According to the present set of input data, it is possible to identify a left lane change from a right lane change with low misclassification error. But it is not possible to identify with the same efficiency a lane change from a no lane change. The best performance was obtained with a random forest where half of the lane changes are recognized while at the same time achieving a low number of false alarms; in another test, using a sub set of data by filtering noisy observations, it was possible to recognize left and right lane changes with more than 70% efficiency.
# Contents

<table>
<thead>
<tr>
<th>Section</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>Acknowledgement</td>
<td>iii</td>
</tr>
<tr>
<td>Details</td>
<td>v</td>
</tr>
<tr>
<td>Abstract</td>
<td>vi</td>
</tr>
<tr>
<td>List of Definitions</td>
<td>x</td>
</tr>
<tr>
<td>List of Figures</td>
<td>xii</td>
</tr>
<tr>
<td>List of Tables</td>
<td>xiii</td>
</tr>
<tr>
<td><strong>1 Introduction</strong></td>
<td>1</td>
</tr>
<tr>
<td>1.1 Problem Statement</td>
<td>1</td>
</tr>
<tr>
<td>1.2 Project goal</td>
<td>2</td>
</tr>
<tr>
<td>1.3 Earlier research</td>
<td>2</td>
</tr>
<tr>
<td>1.4 Project description</td>
<td>6</td>
</tr>
<tr>
<td>1.5 Thesis Structure</td>
<td>7</td>
</tr>
<tr>
<td><strong>2 Lane Change Fundamentals</strong></td>
<td>8</td>
</tr>
<tr>
<td>2.1 The lane change process</td>
<td>8</td>
</tr>
<tr>
<td>2.2 Definition of a lane change</td>
<td>9</td>
</tr>
<tr>
<td>2.3 The driver model</td>
<td>10</td>
</tr>
<tr>
<td><strong>3 Methodology</strong></td>
<td>12</td>
</tr>
<tr>
<td>3.1 Data Visualization</td>
<td>12</td>
</tr>
<tr>
<td>3.1.1 Histograms and Density Plots</td>
<td>13</td>
</tr>
<tr>
<td>3.1.2 Box and Whisker Plot</td>
<td>13</td>
</tr>
<tr>
<td>3.1.3 2D/3D Plots</td>
<td>13</td>
</tr>
<tr>
<td>3.2 Features and Sampling</td>
<td>14</td>
</tr>
<tr>
<td>3.2.1 Pearson’s Correlation Coefficient</td>
<td>14</td>
</tr>
<tr>
<td>3.2.2 Mutual Information</td>
<td>15</td>
</tr>
</tbody>
</table>
3.2.3 Fisher Index ................................................. 16
3.2.4 Resampling .................................................. 17
3.3 Algorithms ..................................................... 17
  3.3.1 Linear Discriminant Analysis ......................... 17
  3.3.2 Random Forests ........................................ 18
    Decision Tree .............................................. 18
    Split Criterion .............................................. 19
    Random Forests ............................................. 20
3.3.3 Neural Networks .......................................... 20
  Perceptron .................................................. 21
  Multilayer Perceptron ...................................... 21
  Levenberg-Marquardt Algorithm .......................... 21

4 Data Acquisition .................................................. 23
  4.1 Software setup ............................................. 23
  4.2 Transformation of the Data .............................. 24
  4.3 Description of the Data .................................. 25

5 Results and Discussion .......................................... 26
  5.1 Lane Correction ........................................... 26
    5.1.1 Discussion ........................................... 29
  5.2 Data Analysis ............................................. 29
    5.2.1 Outlier Detection .................................... 29
    5.2.2 Resampling ............................................ 30
      Random Sampling With Replacement .................. 30
      SMOTE .................................................. 33
    5.2.3 Features Distribution and Density Plots ........ 33
    5.2.4 Pearson’s Correlation Coefficient ................ 35
    5.2.5 Mutual Information .................................. 39
    5.2.6 Fisher Index .......................................... 41
    5.2.7 Discussion ............................................ 41
  5.3 Performance Evaluation .................................. 42
    5.3.1 Test 1: Random test and training set ............ 43
    5.3.2 Test 2: Continuous test and training set ....... 45
    5.3.3 Test 3: Using objects for test and training set 45
    5.3.4 Discussion ............................................ 48

6 Summary and Conclusion ....................................... 51
  6.1 Future Work .............................................. 52
List of Definitions

ACC
Adaptive Cruise Control

ADTF
Automotive Data and Time triggered Framework

Backpropagation
Backward propagation of errors

BN
Bayesian Network

CAN information
Controller Area Network, data gathered by the sensors within the vehicle and provide information like speed, acceleration, orientation, engine conditions, gear being used among others.

DBN
Dynamic Bayesian Network

Ego Vehicle
Vehicle Containing the Assistance System

ELA
Ego Lane Assignment

FI
Fisher Index

GNSS
Global Navigation Satellite System

GPS
Global Positioning System

IMM
Interactive Multiple Model

IQR
Inter Quartile Range

LC
Lane Change
LDA
    Linear Discriminant Analysis

LM
    Levenberg-Marquardt Algorithm

Map Information
    Digitalized map of the road which can be used as reference in order to find the position and
    information regarding road signs and exits

MLP
    Multi Layer Perceptron

RVM
    Relevance Vector Machines

SBL
    Sparse Bayesian Learning

SMOTE
    Synthetic Minority Over-sampling Technique
# List of Figures

<table>
<thead>
<tr>
<th>Figure</th>
<th>Description</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>1.1</td>
<td>Example of the highway scenario</td>
<td>3</td>
</tr>
<tr>
<td>1.2</td>
<td>Graphical view of the project process</td>
<td>6</td>
</tr>
<tr>
<td>2.1</td>
<td>Classes of lane change models</td>
<td>9</td>
</tr>
<tr>
<td>2.2</td>
<td>Driver model</td>
<td>11</td>
</tr>
<tr>
<td>3.1</td>
<td>Box and whisker plot example</td>
<td>14</td>
</tr>
<tr>
<td>3.2</td>
<td>3D plot example</td>
<td>14</td>
</tr>
<tr>
<td>3.3</td>
<td>Decision tree example</td>
<td>19</td>
</tr>
<tr>
<td>3.4</td>
<td>Example of a multilayer perceptron</td>
<td>21</td>
</tr>
<tr>
<td>4.1</td>
<td>Data generation process</td>
<td>24</td>
</tr>
<tr>
<td>5.1</td>
<td>Box and whisker plot of feature 1</td>
<td>30</td>
</tr>
<tr>
<td>5.2</td>
<td>Box and whisker plot of feature 4</td>
<td>31</td>
</tr>
<tr>
<td>5.3</td>
<td>Box and whisker plot of feature 5</td>
<td>31</td>
</tr>
<tr>
<td>5.4</td>
<td>Imbalance of the classes</td>
<td>32</td>
</tr>
<tr>
<td>5.5</td>
<td>Comparison of resampling algorithms</td>
<td>34</td>
</tr>
<tr>
<td>5.6</td>
<td>Distribution of feature 4</td>
<td>35</td>
</tr>
<tr>
<td>5.7</td>
<td>Distribution of feature 9</td>
<td>36</td>
</tr>
<tr>
<td>5.8</td>
<td>Distribution of feature 19</td>
<td>36</td>
</tr>
<tr>
<td>5.9</td>
<td>Correlation coefficient results</td>
<td>37</td>
</tr>
<tr>
<td>5.10</td>
<td>3D plot features 15, 17 and 18</td>
<td>38</td>
</tr>
<tr>
<td>5.11</td>
<td>3D plot features 11, 12 and 13</td>
<td>39</td>
</tr>
<tr>
<td>5.12</td>
<td>Mutual information results</td>
<td>40</td>
</tr>
<tr>
<td>5.13</td>
<td>Example of trajectories for keeping lane and changing lane</td>
<td>50</td>
</tr>
</tbody>
</table>
## List of Tables

<table>
<thead>
<tr>
<th>Table</th>
<th>Title</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>2.1</td>
<td>Boundaries of the lane change</td>
<td>10</td>
</tr>
<tr>
<td>2.2</td>
<td>Parameters for Ego vehicle</td>
<td>11</td>
</tr>
<tr>
<td>2.3</td>
<td>Parameters for surrounding vehicles</td>
<td>11</td>
</tr>
<tr>
<td>5.1</td>
<td>Format of the output file of lane correction</td>
<td>27</td>
</tr>
<tr>
<td>5.2</td>
<td>Stability counter example</td>
<td>28</td>
</tr>
<tr>
<td>5.3</td>
<td>Outlier detection</td>
<td>32</td>
</tr>
<tr>
<td>5.4</td>
<td>Statistic information for the resampling algorithms</td>
<td>33</td>
</tr>
<tr>
<td>5.5</td>
<td>Fisher index results</td>
<td>41</td>
</tr>
<tr>
<td>5.6</td>
<td>Results of classification by randomly selecting test and training sets</td>
<td>44</td>
</tr>
<tr>
<td>5.7</td>
<td>Results from training with a continuous block of data</td>
<td>46</td>
</tr>
<tr>
<td>5.8</td>
<td>Results from training with a selection of objects</td>
<td>48</td>
</tr>
</tbody>
</table>
1

Introduction

During the 20th century, the automobile was transformed from being an expensive toy into the standard way of transportation. The growth of the car industry brought several advantages and disadvantages to society. Advantages include personal mobility, the creation of jobs, possibility of traveling long distances in short periods of time and making life more comfortable. On the other hand the disadvantages consist of the use of non-renewable fuels, accidental death and the generation of air and noise pollution amongst others. These disadvantages need to be minimized as much as possible since they present an increasing concern for both the car industry and society.

1.1 Problem Statement

Energy efficient vehicles are a concern for society all around the world; the rapid increase of oil prices due to the limited supply [1] and the need to curb the greenhouse gas emissions which are increasing the planet’s temperature [2] have called for the search of solutions and alternatives to the fossil-fuel usage. In the last decade, vehicles that use alternative fuel sources have been developed, the Tesla Roadster created in 2006 or the Nissan Leaf developed in 2009 are just two examples of vehicles that no longer use oil-based fuels [3,4]. It can be expected that in the upcoming years the demand for vehicles with improved fuel-economy will increase.

Aside to the exploration for better fuel sources, there has been an extensive research for better driver assistance systems [5,7]. These systems should be capable of maximizing the driver’s security and comfort while minimizing the fuel consumption. To achieve these goals, it is of major interest to understand what is happening around the vehicle. Therefore the assistance system should use this knowledge of its environment to alert the driver from any
situation of risk so that they can be prevented. Moreover, reducing the number of accidents and unnecessary maneuvers will help reduce the fuel consumption by avoiding emergency breaking and leading to a more efficient fuel-economy.

1.2 Project goal

The objective of this thesis is to generate data in the shape of meaningful features for the task of maneuver prediction, analyze these data and evaluate its quality using data mining techniques; then, depending on the results of the analysis, reach a decision whether the selected features and the data set can be used for the prediction of maneuvers. The prediction should happen with enough time to allow the driver to take the necessary precautions and maintain an efficient and more secure driving. To achieve this goal, it is necessary to recognize the vehicles around the Ego vehicle (vehicle that possess the prediction system) and extract information such as position, speed and acceleration to predict their possible trajectories. The use of sensors installed around the Ego vehicle allows the gathering of the necessary information to get familiar with the surroundings and identify objects in it.

An example of a scenario is shown on Figure 1.1. The Ego vehicle (red) using sensors and map information creates a digital environment of its surroundings. The environment contains the data, such as speed, position and acceleration of all the surrounding vehicles. The most relevant data from the model is used for the prediction of maneuvers of the vehicles of interest, in this case, the upcoming car from the right (yellow). Then the system uses the prediction algorithm to assess the likelihood that the upcoming car has to merge in front of the Ego vehicle and depending on the result, send out a warning to the driver so that the proper precautions can be taken.

1.3 Earlier research

To solve the maneuver prediction problem, three main characteristics have to be considered. First, the method needs to be able to model complex scenarios; this is because the number of features that can be considered in the maneuver is very large. This characteristic makes deterministic methods like propositional or first order logic not suitable for the task since they are difficult to use for such a complex scenario. Second, handling uncertain knowledge since is not possible to have all the knowledge from all the possible features all the time, the method needs to handle incomplete data. The third characteristic is related to the nature
of the application, because the traceability of the conclusion is important, this means that it is important to know all the process step by step to see why the result was obtained. This characteristic makes black box methods like Neural Networks not the best option for the system final algorithm since the data that were not contained in the training set can lead to unexpected results.

Earlier research done in maneuver prediction is broad; several authors have built different approaches with diverse algorithms. In the next part I include a summary of the earlier research mentioning their most important characteristics and their results. These researches were used to generate the driver model which is the key element for determining the features that describe the environment surrounding the Ego vehicle.

Liu and Salvucci, [9] used a Markov Dynamic Model to predict lane changes by using eye movements and CAN (Controller Area Network) information, their system predicted lane changes made the Ego vehicle with 95% accuracy.

Althoff, et al. [10] implemented Markov Chains for lane change prediction in an autonomous vehicle. They considered several lane change restrictions and motivations, for instance, considering lane changes only around the Ego vehicle, lane changes can happen only once and with independent probabilities, they also defined the motivations for driving on the current lane and for driving in the neighbor lane. At the end of their paper they present a mathematical example.

Toledo, et al. [11] applied method called Interactive Multiple Model (IMM) which uses two or more Kalman Filters running in parallel. Each of the filters was running with a different model to identify target motion and errors. Then IMM forms an ideal weighted sum.
of the output of all the filters. Their results showed predictions within 0.2 - 0.3 seconds. It would be useful to know information about efficiency (percentage of correct classifications and misclassifications).

Freyer, et al. [13] did research on the types of drivers dividing them in fluid, moderate and comfortable. Furthermore they use a cascade method where the criterion for lane change is analyzed, motivations are collected, possible trajectories are drawn and possible lane changes are recognized using cluster analysis.

Dagli, et al. [14] suggested a Bayesian Network model that uses time gaps between the Ego vehicle and the surrounding vehicles as well as trajectory and orientation for predicting the lane change. This was done using the CAN bus and a front radar; the results reported 77% recognition of the lane changes with one second prediction time.

Dagli and Reichardt [15] proposed in 2002 an analysis that uses the possible motivations of the driver to change lane as key factor for the identification. From all the possible motivations, they define goals which are used afterwards to generate plans. From the possible plans, possible world structures are created; one world contains the possible development of the trajectories of all the involved vehicles. A heuristic search algorithm is used to find the world that has the best probability of success. In the results is mentioned that they could predict lane changes before they happen, although there is no specific information about miss classifications, time of prediction, or number of data tested which would be useful to know.

Another work from Dagli, et al. [16] built a Dynamic Bayesian Network (DBN) for the prediction of maneuvers by means of two features, time to collision and the net time gap between the Ego and the surrounding vehicles. The simulation results showed up to 80% of probability of lane change with 1.5 seconds of prediction. Is important to note that simulation does not always work the same way as the real situation.

Berndt, et al. [17, 18] predicted lane changes of the Ego vehicle by building a Hidden Markov Model which uses the CAN bus data and Map information as input, their results showed 76% efficiency in the detection of lane changes from the Ego Vehicle. The Ego’s lane change can be seen as a linear model in contrast to the maneuver prediction of the surrounding vehicles, which is a more complex system.

Another work with Hidden Markov Models is the one done by Polling, et al. [19] where using different types of models, (driver model, acceleration model, tactical model and steer-
Doshi, et al. [20] did research in the use of eye gaze and head motion to detect lane changes employing Relevance Vector Machines (RVM). RVM are vector machines with a probabilistic output which gives the advantage to tune the decision boundary to achieve the desired performance. Their results showed that using eye gaze was less efficient than head motion, this is because the driver’s eyes are always distracted with the surroundings and not only when a lane change occurs.

A Bayesian Network approach for Ego vehicle lane change prediction is compared with Hidden Markov Models by Tezuka, et al. [21] where defining three types of driver behavior (normal lane keeping, normal right lane keeping and emergency right lane change). The results showed that a Bayesian Network perform better than previous research.

McCall, et al. [23] used a camera for lane tracking, CAN bus data and a camera inside the vehicle to track head motion. They carried out Sparse Bayesian Learning (SBL) to prune features even when the number of candidates is very large. SBL simplifies the assimilation of multiple modalities of sensor information (information coming from different types of sensor sources) making it good for computer vision applications. Their results showed 95% accuracy with two seconds of prediction time for lane change maneuvers of the Ego vehicle.

A very interesting comparison was made by Dogan, et al. [22] where they tried to find the best model for the lane change prediction of the Ego vehicle using Feed Forward Neural Network, Recurrent Neural Network and Support Vector Machines. They used CAN bus data and the distance to the front vehicle as features. Their results showed that Support Vector Machines had a slightly better performance over the Neural Network methods.

The earlier research covers lane change prediction of both the Ego vehicle and the surrounding vehicles. These researches suggest, on one hand, that to solve the first task several methods can be carried out. This is because the type of lane change has a linear behavior and having all the information of the CAN bus makes it easier for the classifier to identify when the maneuver will happen. Note that the information that can be obtained from within the Ego vehicle contains less noise and is more stable than the one observed from the surroundings (with camera, radar and ultrasonic sensors).

On the other hand, the lane change of the vehicles around the Ego vehicle is a more complex problem, where data is often incomplete and the number of features that analyze it is
very broad. This makes the choice of features difficult since the more of them we use, the more complex the system becomes. This has to be considered seriously in the data analysis process.

1.4 Project description

To predict maneuvers in the system, a driver model is defined and used to select the features that will work as input for the probabilistic algorithm; the driver model is explained in Chapter 2. Afterwards, state of the art data mining techniques are necessary to evaluate the quality of the data and see if the maneuver prediction can be achieved. The analyzed data will be tested in different learning algorithms to see its performance and conclude if a real implementation can be achieved.

Figure 1.2 has a graphical view of the process that this thesis follows. The first step is to obtain the data in the shape of features with significant information regarding the surroundings of the Ego vehicle. Afterwards, perform data analysis to improve its quality and obtain as much information as possible about the dependencies and independencies of the features. When the data analysis is exhausted, the performance of the feature set is measured using a set of learning algorithms and then judge if an implementation is possible.

This thesis considers the general case of lane changes in highways. The driver model used is also fixed as well as the set of features created with it. The objective of this thesis is to decide whether or not the current configuration of features and the data set available can be used for the prediction of maneuvers.
1.5 Thesis Structure

The structure of this thesis goes as follows; the fundamental knowledge regarding lane change is defined in Chapter 2, where the basic terms that are involved in a common lane change are explained, as well as the driver model used. The third Chapter shows the techniques and algorithms that will analyze the data and evaluate the performance. Chapter 4 explains the process for gathering and preparing the data for the maneuver prediction task. The fifth Chapter presents the results and discussion from the data analysis and the performance evaluation. The sixth and last Chapter contains the summary and conclusion of the thesis.
Lane Change Fundamentals

This Chapter explains the basic concepts which are related to a lane change as well as the driver model that is used as basis for this work. First the lane change process will be described, including its phases and types which are of relevance to this work. Second the characterization of the lane change process, summarizing some attributes and some limitations that will work as boundaries for the identification of the maneuver. This Chapter ends with the introduction of the driver model which serves as a guideline for the creation of useful features that provide relevant information for the prediction task.

2.1 The lane change process

From the literature search regarding lane change, the prediction task is divided in two types. The first type predicts lane changes of the Ego vehicle (vehicle that possess the prediction system), while the second type focuses on the lane change for the surrounding vehicles. Literature search performed with respect to lane change of the surrounding vehicles present many approaches that are generalized in two main classes. The first class of models describes the lane change psychologically from the perspective of the driver, to weigh the advantages and disadvantages of the lane change and make a decision. The second class is describing the models from a technical point of view, consisting of mathematical equations which describe the lane change process. These lane change models have parameters that allow certain driving styles and situations to be modeled and compared. A chart, including the two classes, is shown in Figure 2.1.

With the research performed in “accident cause” by Fastenmeier, et al. 24, the lane change can be divided into four phases:
1. Decision: Motivators and external stimuli that are favorable for a lane change such as gaining acceleration or having more space between vehicles are weighted against the inhibitors like loss of acceleration or the reduction of space between vehicles. Based on these weights a decision to undertake in a lane change maneuver can be taken.

2. Preparation: If the decision from the first step is to undertake into a lane change maneuver; in this step the proper adjustments are made, in example, adjusting the speed to the one of the target lane.

3. Introduction: The lane change is performed, the maneuver is considered started when the front tire touches the lane marking.

4. Culmination: The lane change is completed and final adjustments to speed and position within the target lane are done.

Figure 2.1: Detection of a lane change can be divided in two types: detection for the Ego vehicle and detection of the surrounding objects. From the latter the research suggests two main approaches, psychological point of view of the driver (motivators and inhibitors to change lane) and mathematical approach (equations that can describe the maneuver).

2.2 Definition of a lane change

Lane changes vary from one to the other therefore is important to delimit the lane change within certain boundaries to know the distance, duration and speed that contain it. For identifying a lane change it is important to define the lane change within certain boundaries; these boundaries allow for a better definition of the data that are analyzed to do the prediction.

For purpose of this work a lane change shall be considered as started when the front tire of the vehicle touches the lane separation marking. For defining the other boundaries the work of Sporrer, et al. is used to define the duration, speed limits and prediction time of a lane change. A summary of Sporrer results is shown in Table 2.1. These boundaries are used to define the data; for example, only the data allocated 2-3 seconds before the lane
change happens will be marked as lane change.

<table>
<thead>
<tr>
<th>Term</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Duration of a lane change</td>
<td>2-6 Seconds</td>
</tr>
<tr>
<td>Speed boundaries</td>
<td>69-177km/h</td>
</tr>
<tr>
<td>Time allowed before detecting a</td>
<td>2-3 seconds</td>
</tr>
<tr>
<td>lane change</td>
<td></td>
</tr>
</tbody>
</table>

Table 2.1: Boundaries of the lane change as defined by Sporrer [25].

2.3 The driver model

The driver model used in this project is based on the one suggested by Dagli, et al. [14] this model includes the following actions:

- Longitudinal regulation
- Lateral control
- Lane change to the left
- Lane change to the right
- Deceleration
- Acceleration

These actions are divided into two groups, longitudinal and lateral as shown in Figure 2.2. The actions within a group are disjointed from each other, meaning that two actions from the same group cannot occur simultaneously. These actions allow for the creation of many features related to speed, acceleration, position in the lane, distance and time gaps which could be useful for identifying a lane change and therefore to predict it.

Using the defined action model, a set of parameters that describe the position and the trajectory of the surrounding vehicles can be defined. The data preparation step of the project should extract these parameters from the raw data so that significant features can be created and the lane change prediction can be studied. The parameters are divided into two sets, one for the Ego vehicle, see Table 2.2 and one for all the surrounding objects, Table 2.3.

The features created from the action model parameters relate the time and distance gaps among the vehicles, the changes in relative speed and acceleration and the position within the lanes. Knowing all these features ease the modeling of the environment surrounding the
Figure 2.2: Driver model from Dagli, et al. [14] Actions are divided into longitudinal (LO) and lateral (LA). Actions within a group are disjointed from each other.

<table>
<thead>
<tr>
<th>Parameters for Ego vehicle</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>$x$</td>
<td>Global x-Position from point of origin [m]</td>
</tr>
<tr>
<td>$y$</td>
<td>Global y-Position from point of origin [m]</td>
</tr>
<tr>
<td>$s$</td>
<td>Driven distance from point of origin [m]</td>
</tr>
<tr>
<td>$v_c$</td>
<td>Velocity [m/s]</td>
</tr>
<tr>
<td>$a_c$</td>
<td>Acceleration [m/s$^2$]</td>
</tr>
<tr>
<td>$\psi$</td>
<td>Angle with respect to the point of origin [$^\circ$]</td>
</tr>
<tr>
<td>$\dot{\psi}_x$</td>
<td>Yaw rate [$^\circ$/s]</td>
</tr>
</tbody>
</table>

Table 2.2: Parameters for Ego vehicle.

Ego vehicle and therefore the implementation of a driver assistance system that can support the driver with the decisions that have to be taken to prevent unwanted scenarios.

This Chapter presented the fundamental ideas that are necessary for the understanding of the lane change and the development of the maneuver prediction. All this knowledge has been used for the identification of relevant features for the lane change prediction. The next Chapter describes the process for extracting data and convert it into meaningful features.

<table>
<thead>
<tr>
<th>Parameters for surrounding vehicles</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>$x$</td>
<td>Gap in x direction from Ego vehicle [m]</td>
</tr>
<tr>
<td>$y$</td>
<td>Gap in y direction from Ego vehicle [m]</td>
</tr>
<tr>
<td>$v_x$</td>
<td>Relative velocity in x direction with respect to the Ego vehicle [m/s]</td>
</tr>
<tr>
<td>$v_y$</td>
<td>Relative velocity in y direction with respect to the Ego vehicle [m/s]</td>
</tr>
<tr>
<td>$a_x$</td>
<td>Relative acceleration in x with respect to the Ego vehicle [m/s$^2$]</td>
</tr>
<tr>
<td>$a_y$</td>
<td>Relative acceleration in y with respect to the Ego vehicle [m/s$^2$]</td>
</tr>
<tr>
<td>$w$</td>
<td>Width [m]</td>
</tr>
<tr>
<td>$l$</td>
<td>Length [m]</td>
</tr>
</tbody>
</table>

Table 2.3: Parameters for surrounding vehicles.
3

Methodology

3.1 Data Visualization

Data mining (or data analysis) consists in the extraction of useful information from large data sets [27]. Data mining comes from the necessity of finding significant relationships among different variables, when there is no sufficient information about their nature.

There exist several techniques that can be used for data mining; these techniques can be put together in two categories: computational methods and data visualization methods. The next table shows some of the methods that belong to each category [27, 28]. Some of these methods are explained in more detail since they were carried out in the data analysis of this research.

• Computational methods:
  – Descriptive statistics (distribution, mean, median, variance)
  – Multivariate exploratory techniques (cluster analysis, principal components, discriminant analysis, amongst others)
  – Correlation
  – Mutual information

• Data visualization methods (represent information in a visual form and are widely used for data exploration):
  – Histograms
3.1.1 Histograms and Density Plots

Histograms and density plots help visualize if the data have a specific type of distribution. The distribution may give information about the modality of the data, this means that if the data are formed from more than one distribution the problem can be separated into multiple simpler problems [27,28].

3.1.2 Box and Whisker Plot

The box and whisker plot, also known as boxplot is a helpful tool for graphically analyzing data. The plot uses five number summaries to represent the data and how it is distributed using the median, the largest and the smallest observation [27].

The five numbers summary consists of:

- Smallest observation
- Lower quartile (25% of the data between the median and the lowest observation)
- Median
- Upper quartile (25% of the data between the median and the largest observation)
- Largest observation

For this research, the box and whisker plot is used to aid the identification of values in the data that are outside the interest of the prediction task (outliers). Figure 3.1 shows an example of the plot, an outlier is defined as any value outside the whiskers. These whiskers represent the lowest datum still within 1.5 Inter Quartile Range (IQR) of the first quartile and 1.5 IQR of the third quartile.

3.1.3 2D/3D Plots

2D and 3D plots help visualize dependencies and independencies amongst groups of features. An ideal example can be seen in Figure 3.2 where three different classes can be recognized.
from the combination of 3 features. It is important to mention that if there is a visible relation in the plot, there exist a relation among the features but this does not work both ways, the fact that it is not possible to visualize a relation does not mean there is not one [27].

### 3.2 Features and Sampling

#### 3.2.1 Pearson’s Correlation Coefficient

To understand the correlation coefficient it is necessary to describe the covariance. Covariance is the measure of how much does two variables change together and it is calculated with the equation

\[
C(X,Y) = E[(X - E[X])(Y - E[Y])'] = E[XY'] - E[X]E[Y]'
\]  

(3.1)

where \(X, Y\) correspond to two random variables, \(E\) is the expected value and \(Y'\) is the transpose of \(Y\).
Pearson’s correlation coefficient \[31\] indicates if there exist a linear relationship between two random series.

\[
\frac{C(x, y)}{\sqrt{\sigma_{xx} \sigma_{yy}}} = R_{xy} \tag{3.2}
\]

where the variance \((\sigma_{xx}, \sigma_{yy})\) and \((C(x, y))\) is the covariance between \(x\) and \(y\). Therefore we have that

\[
y = a + bx \rightarrow R_{xy} = 1, \tag{3.3}
\]

\[
y = a - bx \rightarrow R_{xy} = -1, \tag{3.4}
\]

which demonstrate that when the absolute value of \(|R_{xy}|\) is close to one, then there exist a linear relationship; while a value close to zero indicates linear independence.

The standard deviation for the correlation coefficient between two random series of length \(N\) is \(1/\sqrt{N}\). This denotes that if

\[
|R_{xy}| > \frac{1.96}{\sqrt{N}} \tag{3.6}
\]

then the correlation between \(x\) and \(y\) is significant at the 95% level.

The correlation coefficient is presented as a matrix where all the features \((F_1, F_2...F_n)\) are analyzed against one another this matrix is related to the covariance matrix by the following equation

\[
\frac{C(F_1, F_2)}{\sqrt{C(F_1, F_1)C(F_2, F_2)}} = R(F_1, F_2) \tag{3.7}
\]

\(C\) and \(R\) correspond to the covariance and correlation coefficient respectively for the pair of features \(F_1\) and \(F_2\).

3.2.2 Mutual Information

The mutual information measures the mutual dependence of two variables. When the mutual information is high then there is a big dependence among the corresponding variables and vice versa.
The mutual information of two discrete random variables $X$ and $Y$ is calculated by

$$I(X, Y) = \sum_{y \in Y} \sum_{x \in X} p(x, y) \log \left( \frac{p(x, y)}{p(x)p(y)} \right)$$  \hspace{1cm} (3.8)$$

where $p(x, y)$ is the joint probability distribution of the pair of variables $X$, $Y$ and $p(x), p(y)$ are the marginal probability distributions of $X$ and $Y$ respectively.

Mutual information has values bigger than zero but for better understanding whether or not there is significance between the features it is preferred to normalize the mutual information making its limits between zero and one. The mutual information can be normalized by

$$NI(X, Y) = \frac{2I(X, Y)}{\max(H(A)) + \max(H(B))}$$  \hspace{1cm} (3.9)$$

using the maximum entropy $H(X), H(Y)$ and the mutual information $I(X, Y)$ for the two variables $X$ and $Y$.

### 3.2.3 Fisher Index

Fisher index (FI) allows to identify features that are useful for classification, it does this by analyzing the between class spread of two classes and the variance that exists in each class respectively. A high fisher index will indicate that the classes are well separated by the variable in question; note that a low fisher index does not necessarily mean that the feature is bad for classification [30].

Fisher index is defined as:

$$FI(K) = \frac{(\mu_{k,1} - \mu_{k,2})^2}{(N_1 - 1)\sigma_{k,1}^2 + (N_2 - 1)\sigma_{k,2}^2}$$  \hspace{1cm} (3.10)$$

where the indices 1 and 2 refer to two categories and $\mu_k$ and $\sigma_k^2$ correspond to the mean and variance respectively.
3.2.4 Resampling

In order to perform a successful data analysis and testing of the algorithms, it is necessary to compensate the imbalances that exist within the classes of the data; for doing this, two over sampling algorithms are used.

- Random sampling with replacement: This method consists of the resampling of the less populated classes, by randomly selecting and duplicating samples until the imbalance disappears. The data generated will not contain any extra information but allows for a better training of the learning algorithms.

- Synthetic Minority Over-sampling TEnchnique (SMOTE): This method creates new observations by interpolating between several minority class examples. The idea is to find the nearest neighbors from an observation and to calculate the distance between them; then this distance is multiplied by a random number between zero and one and added to the vector under consideration [32].

3.3 Algorithms

In order to corroborate the information gathered in the data analysis stage, it is necessary to train a classification algorithm and evaluate its performance. The results from the training and testing of the classification algorithm will aid in the decision of following a final implementation or go back to either the feature selection process to modify the features or measure more data to double check that the bad performance is not coming from poor data quality.

For the purpose of this work, three classification algorithms are implemented, linear discriminant analysis (LDA), neural networks (NN) and random forests: LDA will allow to see the performance of a linear model so that it can be compared to the non linear models (NN and random forests); the performance of all three methods, will tell what the maximum performance of the feature set can be. [27, 33].

3.3.1 Linear Discriminant Analysis

The idea of LDA is to transform a set of multivariate observations to univariate observations such that the univariate observation classes will be separated as much as possible [34].
Suppose that there is a ser of \( m \) p-dimensional samples \( x_1, x_2, ..., x_m \) that belong to one of \( n \) classes \( C_1, C_2, ..., C_n \).

The scatter matrix for one class is given by:

\[
S_i = \sum_{X \in C_i} (X - \mu_i)(x - \mu_i)',
\]

where \( \mu_i \) is the mean of the class. Hence the total intra-class scatter matrix is given by:

\[
\varepsilon_w = S_1 + S_2 + ... + S_n
\]

The inter-class scatter is given by:

\[
\varepsilon_b = \sum_{i=1}^{n} m_i(x - \mu_i)(x - \mu_i)'
\]

were \( m_i \) is the number of training samples for each class and \( \mu_i \) is the mean of each class.

Then a linear transformation \( \Phi \) is used to maximize the ratio of the determinant of the inter-class scatter matrix of the projected samples to the intra-class scatter matrix of the projected samples:

\[
\gamma(\Phi) = \frac{|\Phi^T \varepsilon_b \Phi|}{|\Phi^T \varepsilon_w \Phi|}
\]

were the transformation \( \Phi \) can be obtained by solving the generalized eigenvalue problem:

\[
\varepsilon_b \Phi = \lambda \varepsilon_w \Phi
\]

then the classification is performed using a distance metric like Euclidean or Mahalanobis distance.

### 3.3.2 Random Forests

**Decision Tree**

A decision tree is a classification tool that is used to predict the membership of objects to different classes taking into account the values that correspond to their attributes. A simple example, see Figure 3.3, shows a decision tree for a data set consisting of two features (age and car type) and two class outputs (high accident risk and low accident risk) [27,33].

The learning performed by the decision tree is characterized by the following:
Figure 3.3: Example of a decision tree created with two features (age, car type) and two classes (high, low) [27].

- Each tree’s internal node (non-terminal node) expresses the testing based on a certain attribute.
- Each branch of the tree expresses the test’s result.
- The leaf nodes (terminal nodes) represent the decision classes.

**Split Criterion**

For the split criterion, the Gini index is used, this index is a measure of how often will a value taken randomly from the set will be classified incorrectly. The Gini index is given by:

$$G = 1 - \sum_i p_i^2$$  \hspace{3cm} (3.16)

where $p$ is the probability of each class. The optimal split is chosen by means of the information gain:

$$I(i) = G_{parent} - \sum_k \frac{k}{n} G_{subset}$$  \hspace{3cm} (3.17)

where $G_{parent}$ is the Gini index of the parent node, $G_{subset}$ is the Gini index of each value $k$ of subset table $S_i$. 

<table>
<thead>
<tr>
<th>Age</th>
<th>Car Type</th>
<th>Accident Risk</th>
</tr>
</thead>
<tbody>
<tr>
<td>20</td>
<td>Sport</td>
<td>High</td>
</tr>
<tr>
<td>18</td>
<td>Convertible</td>
<td>High</td>
</tr>
<tr>
<td>40</td>
<td>Minivan</td>
<td>Low</td>
</tr>
<tr>
<td>50</td>
<td>Premium</td>
<td>Low</td>
</tr>
<tr>
<td>35</td>
<td>Compact</td>
<td>Low</td>
</tr>
<tr>
<td>30</td>
<td>Sport</td>
<td>High</td>
</tr>
<tr>
<td>32</td>
<td>Sport</td>
<td>High</td>
</tr>
<tr>
<td>40</td>
<td>Full size van</td>
<td>Low</td>
</tr>
<tr>
<td>33</td>
<td>Mini</td>
<td>High</td>
</tr>
<tr>
<td>39</td>
<td>Convertible</td>
<td>Low</td>
</tr>
</tbody>
</table>
Some advantages of decision trees are:

- Easy to understand and interpret.
- They require a small amount of training data in order to be built.
- Allow the use of both numerical and categorical data without any restriction.
- Represent models of the gray box type, meaning that the logic underlying the decision process can be followed if necessary.
- They can process large data sets.
- They use classical statistical techniques to make the model validation possible.

**Random Forests**

A random forest consists of an ensemble of many decision trees to classify a new object from an input vector; the vector is passed to all the trees in the forest. Each tree gives a classification which is taken as a “vote” for the class; then, the forest chooses the class with the largest amount of votes.

Each of the trees is grown in the following manner:

1. If the number of cases in the training set is \( N \), sample \( N \) cases randomly with replacement from the original data. This sample will be the training set for growing the tree.
2. If there are \( M \) input observations, a number \( m << M \) is specified such that at each node, \( m \) observations are selected at random out of the \( M \) and the best split on these \( m \) is used to split the node. The value of \( m \) is held constant during the growth of the forest.
3. Each tree is grown to the largest extent possible (there is no pruning).

**3.3.3 Neural Networks**

Neural networks (NN) represent non-programmed adaptive information processing systems. These networks learn from examples and behave like “black boxes’ since the way they process information remains inexplicit. Neural networks can be seen as a massive parallel distributed computing structure. It was inspired in the way that the human brain works and processes information. The NN acquires knowledge through a learning process. The connections
Methodology

Figure 3.4: Example of a multilayer perceptron (MLP). An MLP is formed of at least three layers, a layer for input nodes, a layer for output nodes and one or more layers of hidden nodes.

among neurons, known as weights, are used to store the knowledge acquired in the learning process \[27,36\]. The neural network built in this work is a Multi Layer Perceptron (MLP).

**Perceptron**

A perceptron, seen as a single element, contains a defined number of inputs which are weighted by respective values, then summed up and passed to an activation function to produce a certain output depending on a predetermined threshold \[27\].

**Multilayer Perceptron**

A multilayer perceptron (MLP), see Figure 3.4, is a feedforward neural network model that maps input data onto a desired output. Feedforward means that the connections between the units do not form a directed cycle. A MLP is formed by three or more layers consisting of an input layer, an output layer and one or more hidden layers. The learning of a MLP is done by means of backpropagation \[37\].

**Levenberg-Marquardt Algorithm**

The Levenberg-Marquardt (LM) optimization function is used as training function for the learning of the multi-layer perceptron in this work; this algorithm builds on the assumption that the error has a quadratic form \[37\]. LM optimization performs very fast with the cost of memory; but thanks to the use of the Matlab server it is possible to use it for the large amount of observations.
The algorithm consists in solving the equation:

\[ J^t E = (J^t J + \lambda I) \delta \]  \hspace{1cm} (3.18)

where \( J \) is the Jacobian matrix of the system, \( \lambda \) is the Levenberg’s damping factor, \( \delta \) is the weight update vector that needs to be found and \( E \) is the error vector containing the output errors for each input vector used on training the network. \( \delta \) will tell how much does the network weights need to change to possibly achieve a better solution. The \( \lambda \) damping factor is adjusted on each iteration and works as guide for the optimization process.

When performing tests in the MLP, it is important to define two variables: the number of hidden layers and the number of hidden nodes. Increasing these values will reduce the error but it is important to avoid overfitting. Overfitting happens when the model describes the errors instead of the underlying relationship that exists in the data. Therefore it is important to perform several tests with different values of hidden nodes and hidden layers and use an error measure to identify the best performance without overfitting. To avoid overfitting, cross validation is used for the training and testing of the different networks.

The methods and algorithms for data analysis have been presented in this Chapter, giving an overview of the tools that will be used to get a better understanding about the features. The next Chapter will present the result of the data analysis as well as the algorithms built for performance evaluation.
Data Aquisition

The process of gathering data for the task of maneuver prediction is very complex. It involves a combination of hardware and software working together to convert raw sensor information into significant and understandable features. Then, these features are analyzed for the prediction of lane change maneuvers. This chapter gives an overview of the software and hardware setup used for acquiring the data, the process followed to convert these data into features and the detailed description of these features.

4.1 Software setup

The implementation on the Ego vehicle is done with the Assist Automotive Data and Time triggered Framework (ADTF) Software which plays a major role in today’s vehicle development. ADTF supports the vehicle application designer in easily creating new functionalities with a multitude of ready-to-use modules which in the ADTF environment are known as filters. These filters are programmed in C++ using Microsoft visual studio 2003. To each module a make-file is built allowing the creation of the filters. For the monitoring and analysis of the CAN bus data, CANalyzer is used.

The analysis of the data is performed using Matlab R2010a, using the statistics toolbox, parallel computing toolbox and the neural network toolbox. Due to the amount of memory and computing power that is required to analyze and evaluate the large number of data, (a vehicle could generate 36000 observations per hour, each observation consisting of 21 or more features) a simulation server is used, consisting of 24 cores and 15 GB of RAM.
4.2 Transformation of the Data

The process of data generation and analysis can be observed in Figure 4.1. This process is done in six steps:

1. **Raw Data**: In this step the raw sensor information is put together by means of a Kalman filter. This step is already built and functional.

2. **Ground Truth**: This step takes the data from step 1 filter and identifies objects and lanes in it. Then the objects are saved in a database with a corresponding id number and their parameters created with the driving model (gaps, relative velocities, etcetera), see Table 2.3. Aside from the identification of the objects, the ground truth lanes are discovered, this is done with the use of the mono camera in the front of the vehicle and because the camera cannot have a complete overview of the highway, the algorithm will add a new lane every time the camera discovers one. This means that if the vehicle traveled from a three lane highway into a ramp and then another ramp, by the time it enters the highway again the ground truth lanes will have five lanes identified. This step is already built and functional.

3. **Lane Correction**: This step has the task of analyzing the ground truth lanes and omit the extra lanes by means of another source of information called Ego Lane Assignment (ELA). The ELA contains a better description of the actual position of the vehicle and the correct number of lanes. Extra lanes are omitted with the use of a mask, the mask...
contains the information of how many lanes have to be deleted from the left and right side of the Ego and the time stamp when the validity of the mask stops; these masks are saved into an output file that is used by the next step in the process. This step is performed as part of this thesis.

4. **Continuous Learn Data:** Once the lane correction is finished, the objects and lanes are used to create the features observations with their corresponding classification. To obtain the classification, the program goes through the database of objects generated in the second step; then, when a new object is detected, the program looks in the database whether or not that object will change lane. Then if the object changes lane, the program assign to the observations that correspond to two seconds prior to the lane change, the lane change output.

5. **Data Analysis:** With the use of the learning data, the data analysis can be performed in this step. Data mining techniques are used to visualize the distributions of the features and the relationship that the features have with each other as well as with the output.

6. **Discretized Learn Data:** The results of the data analysis will be used to conclude if the final implementation should be carried out and obtain discretized learn data to train Bayesian Network algorithm proposed for this project.

The implementation of all the steps with the exception of the data analysis (step five) were carried out on the ADTF Environment while the data analysis was performed in Matlab.

### 4.3 Description of the Data

The data generated by the ADTF Configuration consist of a set of observations with 21 features and three possible outputs. Each observation corresponds with an object being tracked and the features give an overview of the object surroundings.

This chapter went through the data generation process, starting by the hardware and software setup needed to gather raw data and the process to convert these data into meaningful features that can be used later for the prediction of maneuvers. Note that the complete data generation process made in ADTF, represents a project all by itself and was performed before this work. In the next chapter, the algorithms and techniques used for data mining and performance evaluation are introduced.
5

Results and Discussion

This Chapter is divided into three sections, first the ADTF configuration that had to be added for the lane correction step is presented. This section will explain the logic followed and the built algorithm used for solving the false lane problem. Afterwards the results of the data analysis will be illustrated and explained. The last section includes the performance evaluation where the results of the classification algorithms (LDA, NN and Random Forest) are shown.

5.1 Lane Correction

As it was previously mentioned on Section 3.2, during the ground truth step of the data transformation. The mono camera on the vehicle which has the task of identifying lanes in the road is not able to see all the lanes at once. Therefore every time a new lane is observed, it is added to the data as a new lane. This behavior leads to the incorporation of false lanes in the data.

In the third step of the data transformation process it is necessary to solve this problem by comparing two sources of information; the lane information generated in the ground truth and the Ego Lane Assignment (ELA) which containing the accurate position of the Ego vehicle. It is necessary to check if these two sources are different and if they are then create an entry to the output file with a new mask value to omit the extra lanes. The format of the output file is shown in Table 5.1.

The following aspects should be taken into account for the elaboration of the program:
Results and Discussion

<table>
<thead>
<tr>
<th>Parameter of output file</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Bit Mask Section</td>
<td>Counter of the number of masks added, every new entry will increment the mask value by 1</td>
</tr>
<tr>
<td>End Time</td>
<td>The ending time of the assigned mask</td>
</tr>
<tr>
<td>Remove Left Lane Markings</td>
<td>Number of lanes to remove on the left side of the Ego</td>
</tr>
<tr>
<td>Remove Right Lane Markings</td>
<td>Number of lanes to delete on the right side of the Ego</td>
</tr>
</tbody>
</table>

Table 5.1: Format of the output file of the lane correction.

- Both the lane information and the ELA need to be available in order to correct the lane markings.
- The ELA has an accuracy value which indicates its integrity.
- There exist noise in both the lane markings and the ELA, this noise has to be taken into account to avoid creating a large number of entries in the output file. To correct this noise problem a stability counter is created, this counter will make sure that only the changes that occur continuously are taken into account for a new mask entry.

There are two main methods working together for the creation of the output file. The first method verifies that the lane information and ELA are available and also that there is more than 90% integrity of the ELA information. When the conditions are met, a new offset is calculated from the difference between the two sources. If the conditions are not met, the offset receives a value that corresponds to the deletion of all possible lanes (offset = 20). The reason for deleting all the lanes is that is not possible to know for sure where the car is located. The pseudo code for the first method can be seen next:

\[
\text{IF } (\text{there is ELA-Information AND Lane-Information AND ELA-Accuracy} > .9) \\
\quad \text{New-Offset} = \text{Offset (Lane-Information and ELA-Information)} \\
\text{ELSE} \\
\quad \text{New-Offset} = \text{Delete All}
\]

The second method receives the new offset value from the first method and compares it with the offset from the current working mask. If they are different then it checks the value of the stability counter and if the counter has reached its limit (defined as 12 occurrences) then a new entry is created. If the counter has not reached the limit then it is increased. If the previous offset is the same as the new offset then the stability counter will be decreased reaching its minimum value of zero. The pseudo code for the second method is shown below:

\[
\text{IF } (\text{Previous-Offset DIFFERENT New-Offset}) \\
\quad \text{IF } (\text{Stability-Counter} > \text{Stability-Limit})
\]


insert End-Time for Previous Mask
create New-Mask entry for output file
assign the new value to Previous-Offset
reset Stability-Counter
ELSE
increase the Stability-Counter
ELSE
IF (Stability-Counter > 0 )
decrese Stability-Counter
ELSE
do nothing

<table>
<thead>
<tr>
<th>Time stamp</th>
<th>ELA Accuracy</th>
<th>New Offset</th>
<th>Previous Offset</th>
<th>Stability Counter</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>.9</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>2</td>
<td>.9</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>3</td>
<td>.9</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>4</td>
<td>.7</td>
<td>20</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>5</td>
<td>.6</td>
<td>20</td>
<td>0</td>
<td>2</td>
</tr>
<tr>
<td>6</td>
<td>.9</td>
<td>0</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>7</td>
<td>.9</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>8</td>
<td>.9</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>9</td>
<td>.7</td>
<td>20</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>11</td>
<td>.6</td>
<td>20</td>
<td>0</td>
<td>2</td>
</tr>
<tr>
<td>12</td>
<td>.6</td>
<td>20</td>
<td>0</td>
<td>3</td>
</tr>
<tr>
<td>13</td>
<td>.6</td>
<td>20</td>
<td>0</td>
<td>4</td>
</tr>
<tr>
<td>14</td>
<td>.6</td>
<td>20</td>
<td>0</td>
<td>5</td>
</tr>
<tr>
<td>15</td>
<td>.6</td>
<td>20</td>
<td>20</td>
<td>0</td>
</tr>
</tbody>
</table>

Table 5.2: Example of the stability counter, only after five consecutive changes in new offset and previous offset, will the change be taken into account and the previous offset will receive the value of the new offset.

It is easier to understand the performance of the stability counter with an example; presume that the Ego vehicle is running on a straight line of a three lane highway; both the ELA and the lane information have same value (offset = 0). During the run the accuracy of the ELA will decrease due to the noise and increase back to the stable value these small changes should not be taken into account, only when they happen more frequently than the previous condition.

Table 5.2 shows the example of 15 consecutive samples of the data from ELA and the lane markings. As it can be seen from the table, the ELA accuracy may change to an unstable value (< 0.9) or the offset from the lane information and ELA be different. But the output will not be modified unless the change in stability occurs for more than the stability limit set
in the parameters (in this example after five consecutive unstable values it is considered a new entry for the mask, in the real application this value is 15, equivalent to one and a half seconds).

5.1.1 Discussion

The disadvantage of the current implementation is that the new entries of the output file will take one and a half seconds longer to be placed; this time will not be included in the mask and therefore the data are lost. This loss of data is necessary since is the time that it takes to verify the stability of the change and it is preferred to have shorter stable masks than many unstable masks. This is an acceptable behavior since the duration of the stable runs is very large (hundreds of seconds each).

5.2 Data Analysis

This section will show the results for the data analysis, most of the analysis presents a graphical result and since there exist 21 features, the amount of images is too large to include it here. Therefore, only a selection of images is presented in this Chapter.

5.2.1 Outlier Detection

The whisker and box plot was used in cooperation to the expert knowledge for the recognition of values that are outside the interest of the prediction task. All outliers identified by the box and whisker algorithm are shown as red marks in the plots. Outlier limits were defined for all the features except the binary features. The whisker values along with the theoretical concepts from Chapter 2, will help define the limits of the data, every value observed in the data outside the limits will not be taken into account for the prediction of maneuvers.

Figure 5.1 shows the initial distribution (upper plot) and the box and whisker plot (lower plot) for feature 1. The values of the whiskers for this feature are located in -16 and +34; but because the feature corresponds to a time unit, the minimum limit is set to zero since there are is no negative time in this context.

In the case of feature 4, see Figure 5.2, the box and whisker plot helps detect possible outliers and define the upper and lower limit. These limits are set to -0.6 and 0.6. Figure 5.3 shows the outlier detection for feature 5; is possible to see that there exist several distributions on the upper plot (around 2, 12 and 24). The limits found with the whiskers (-14, 25)
5.2.2 Resampling

The data classes are highly imbalanced, see Figure 5.4, in order to do a better training of the classification algorithms, resampling techniques have to be used in the data in order to have the same number of observations for each output.

Table 5.4 shows the number of samples that correspond to each class and the corresponding number of samples that should be replaced in order to balance the classes.

Random Sampling With Replacement

For this algorithm, the method has to take 600716 samples from the 10446 that correspond to left lane change and copy them together. With a resample rate of 5816% this will be
Figure 5.2: Outlier detection for feature 4, top plot shows the initial distribution while bottom plot shows the box and whisker plot.

Figure 5.3: Outlier detection for feature 5, top plot shows the initial distribution while bottom plot shows the box and whisker plot.
Results and Discussion

<table>
<thead>
<tr>
<th>Feature</th>
<th>Minimum</th>
<th>Maximum</th>
<th>Lower Limit</th>
<th>Upper Limit</th>
</tr>
</thead>
<tbody>
<tr>
<td>1 Feature 1</td>
<td>0</td>
<td>300</td>
<td>0</td>
<td>5</td>
</tr>
<tr>
<td>2 Feature 2</td>
<td>0</td>
<td>300</td>
<td>0</td>
<td>5</td>
</tr>
<tr>
<td>3 Feature 3</td>
<td>-2.5</td>
<td>2.5</td>
<td>-1.5</td>
<td>1.5</td>
</tr>
<tr>
<td>4 Feature 4</td>
<td>-6</td>
<td>6</td>
<td>-6</td>
<td>.6</td>
</tr>
<tr>
<td>5 Feature 5</td>
<td>-30</td>
<td>40</td>
<td>-14</td>
<td>25</td>
</tr>
<tr>
<td>6 Feature 6</td>
<td>0</td>
<td>300</td>
<td>0</td>
<td>3.6</td>
</tr>
<tr>
<td>7 Feature 7</td>
<td>-40</td>
<td>40</td>
<td>-9.5</td>
<td>17</td>
</tr>
<tr>
<td>8 Feature 8</td>
<td>-30</td>
<td>40</td>
<td>-9.5</td>
<td>25</td>
</tr>
<tr>
<td>9 Feature 9</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>10 Feature 10</td>
<td>-40</td>
<td>30</td>
<td>-9</td>
<td>18</td>
</tr>
<tr>
<td>11 Feature 11</td>
<td>0</td>
<td>250</td>
<td>0</td>
<td>4.5</td>
</tr>
<tr>
<td>12 Feature 12</td>
<td>0</td>
<td>180</td>
<td>0</td>
<td>125</td>
</tr>
<tr>
<td>13 Feature 12</td>
<td>-30</td>
<td>40</td>
<td>-20</td>
<td>11</td>
</tr>
<tr>
<td>14 Feature 14</td>
<td>0</td>
<td>90</td>
<td>0</td>
<td>3.5</td>
</tr>
<tr>
<td>15 Feature 15</td>
<td>-30</td>
<td>40</td>
<td>-20</td>
<td>11</td>
</tr>
<tr>
<td>16 Feature 16</td>
<td>-30</td>
<td>40</td>
<td>-12</td>
<td>26</td>
</tr>
<tr>
<td>17 Feature 17</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>18 Feature 18</td>
<td>-30</td>
<td>40</td>
<td>-18.5</td>
<td>9.5</td>
</tr>
<tr>
<td>19 Feature 19</td>
<td>0</td>
<td>250</td>
<td>0</td>
<td>5</td>
</tr>
<tr>
<td>20 Feature 20</td>
<td>0</td>
<td>180</td>
<td>0</td>
<td>144</td>
</tr>
<tr>
<td>21 Feature 21</td>
<td>-40</td>
<td>40</td>
<td>-9.5</td>
<td>18</td>
</tr>
</tbody>
</table>

Table 5.3: Outlier detection results, column 3 and 4 contain the minimum and maximum values of each feature and column 5 and 6 present the lower limit and upper limit chosen from the outlier detection analysis.

Figure 5.4: Output distribution of the original data, more than 97 % of the data corresponds to a no lane change class.


<table>
<thead>
<tr>
<th>No Lane Change</th>
<th>Samples</th>
<th>Resamples Needed</th>
<th>% of Resampling</th>
</tr>
</thead>
<tbody>
<tr>
<td>Left Lane Change</td>
<td>10446</td>
<td>600716</td>
<td>5816%</td>
</tr>
<tr>
<td>Right Lane Change</td>
<td>3576</td>
<td>607586</td>
<td>16900%</td>
</tr>
</tbody>
</table>

Table 5.4: Second column contains the amount of observations corresponding to each class, third the amount of observations that should be resampled, the last column has the percentage of resample that has to be done.

similar to copy the data from left lane change 58 times and put it together. Similar for the right lane change the result will be equivalent to copying the data set 169 times.

**SMOTE**

This algorithm creates new observations, for each observation of the left lane change, SMOTE will have to find 58 nearest neighbors and then multiply the distances by a random number to create the new samples. For the right lane change the algorithm will need to find 169 nearest neighbors and then create the new observations.

The result of both algorithms affects the density of the classes, this is because random sampling will copy the same values and therefore increase those values densities. SMOTE will create new values that are very close together this can be seen on Figure 5.5 It is preferred to have a plot with more frequency so that it can have a heavier weight against the no lane change plot since the frequency of this one is very high. The rest of the results will be presented using random sampling with replacement.

### 5.2.3 Features Distribution and Density Plots

The features distribution will help identify the behavior that the features follow and by means of the density plots, identify if there is some significance between the values of the features and the three classes (no lane change, left lane change and right lane change). Figure 5.6 shows the distribution for the fourth feature, the Figure is divided into four plots where, from top to bottom, the first plot displays the initial distribution of the data, the next image shows the distribution of the data classified as lane change to the left, the third image the distribution for lane change to the right and the last image for no lane change.

It would be ideal to identify clusters in different positions for the distributions of left, right and no lane change, indicating that is possible to identify the three classes. In some cases like feature 9 (Figure 5.7) or feature 19 (Figure 5.8), it is possible to identify left lane
Figure 5.5: Distributions for feature 12 using both resampling algorithms, the most evident change is the frequency of the plots for the left and right lane change.
Results and Discussion

Figure 5.6: Distributions for feature 4, from top to bottom, first plot shows the initial distribution, the other three plots present the distributions with respect to the three classes (left lane change, right lane change and no lane change). The lateral velocity shows a normal distribution with its mean in zero. The frequency limit for no lane change overlaps with that of left and right lane change; which indicates that a vehicle not changing lane can have the same value for this feature as a vehicle changing lane.

change from right lane change. But the distribution for no lane change in all cases overlaps completely with the initial distribution making the differentiation of lane change (to either way) and no lane change not possible with single features.

The results of the distributions indicate that a single feature is not able to identify the lane change from a no lane change. Still, there exist the possibility that combinations of features can make the recognition of the lane change possible and for this it is necessary to do correlation analysis.

5.2.4 Pearson’s Correlation Coefficient

The correlation coefficient matrix is shown as an image in Figure 5.9 to have a better view of the correlation level in the data. The 21 features and the three different outputs are compared against each other. Correlated features will have a color closer to the limits (red and dark blue). Uncorrelated features can be seen for correlation values close to zero (teal).

From this matrix it is possible to assess two concepts:
Results and Discussion

Figure 5.7: Distributions for feature 4, from top to bottom, first plot shows the initial distribution, the other three plots present the distributions with respect to the three classes (left lane change, right lane change and no lane change). Most of the lane changes to the left happen when the Feature has a value of 0.

Figure 5.8: Distributions for feature 19, from top to bottom, first plot shows the initial distribution, the other three plots present the distributions with respect to the three classes (left lane change, right lane change and no lane change). It is possible to recognize a lane change to the left from a lane change to the right since this feature will most of the time have zero when the lane change to the left occurs.
Figure 5.9: Pearson’s correlation coefficient; correlated features will have a color closer to the limits (red and dark blue). Uncorrelated features can be seen for correlation values close to zero (teal).
Results and Discussion

There exist a high correlation among certain features which represent the same area in the highway scenario (front left, front right, back left, back right).

The features are not separating the classes.

From the matrix it can be observed that features 15 and 17 have a negative dependence while feature 18 has a positive dependence with feature 15. These correlation can be observed in Figure 5.10 the blue, green and red marks in the plot correspond to no, left and right lane change respectively. It is possible to observe that many right lane changes (red) occur when there is not a continuous line on the right side. Still it is not possible to recognize clearly the blue marks from either the red or the green marks which complicate the classification.

Features 11, 12 and 13 are displayed in Figure 5.11. Like in the previous image, the three classes seem to overlap one on top of the other one, complicating the classification task.

This information should lead to the conclusion that some of the features can indicate when a lane change to the left or to the right happen which can be helpful for classification. Still there do not seem to be a clear separation between no lane changes and lane changes which follows the same situation seen in the distribution analysis; this means that, feature values
Results and Discussion

Figure 5.11: 3D Plot of features 11, 12 and 13 There does not seem to be a separable cluster between the three classes no lane change (blue), left lane change (green) and right lane change (red).

from vehicles that are not changing lane are spread over all possible values of the lane changes.

5.2.5 Mutual Information

Mutual information should help confirm the results obtained by Pearson’s correlation coefficient, Figure 5.12 displays the mutual information for the features and the output. Note that mutual information contains values from [0 to 1] while Pearson’s correlation coefficient contains values from [−1 to 1].

Both algorithms clearly indicate that the features possess certain correlation with each other, this is to be expected because some groups of the features describe the same characteristic with different values. For example, features 11, 12 and 13 from Figure 5.11 describe the time, distance and change in speed between the Ego vehicle and the vehicle on the front right lane. These features are all related to each other, since the change in the distance gap will also affect a change in the time gap. On the contrary to features that explain different regions of the highway scenario where there is nearly no correlation between them.
Figure 5.12: Mutual information; correlated features will have values close to one (red) while uncorrelated features will have values around zero (dark blue).
The data analysis shows that there exist some correlation among related features, it also

### 5.2.6 Fisher Index

A Fisher index with high value indicates that the feature under consideration is useful for separating the two classes being analyzed. A low value of FI does not indicate that the feature is not useful. Since there are three output classes and because the FI compares two classes, several tests were performed to analyze the three combinations of the classes, left lane change against right lane change and no lane change against left and right lane change respectively.

For this data, the fisher index did not provide any information regarding good classification, since all of the indices were near zero. Table 5.5 shows the values obtained for the different comparisons of the classes for each feature. Note just as was mentioned before in Chapter 3, the fact that the FI does not indicate that the feature is useful for classification, it does not mean it is without use.

### 5.2.7 Discussion

The data analysis shows that there exist some correlation among related features, it also indicates that it is not possible to visualize a separation among the different classes; this

---

Table 5.5: Fisher index results, a high value of fisher index shows that the feature is useful for classification. A low value of fisher index does not suggest that the feature is not useful. In this analysis all the values for the FI for the different types of classes are very low.

<table>
<thead>
<tr>
<th>Feature</th>
<th>No vs. Left LC</th>
<th>No vs. Right LC</th>
<th>Right vs. Left LC</th>
</tr>
</thead>
<tbody>
<tr>
<td>1 Feature 1</td>
<td>6.01e-7</td>
<td>1.32e-8</td>
<td>2.14e-5</td>
</tr>
<tr>
<td>2 Feature 2</td>
<td>9.42e-8</td>
<td>2.04e-7</td>
<td>6.08e-7</td>
</tr>
<tr>
<td>3 Feature 3</td>
<td>9.92e-10</td>
<td>2.82e-8</td>
<td>1.86e-6</td>
</tr>
<tr>
<td>4 Feature 4</td>
<td>5.61e-8</td>
<td>1.62e-7</td>
<td>9.63e-6</td>
</tr>
<tr>
<td>5 Feature 5</td>
<td>1.78e-7</td>
<td>1.84e-11</td>
<td>8.80e-6</td>
</tr>
<tr>
<td>6 Feature 6</td>
<td>1.96e-7</td>
<td>3.89e-7</td>
<td>7.98e-5</td>
</tr>
<tr>
<td>7 Feature 7</td>
<td>1.09e-7</td>
<td>2.19e-8</td>
<td>1.68e-5</td>
</tr>
<tr>
<td>8 Feature 8</td>
<td>4.18e-7</td>
<td>7.11e-8</td>
<td>7.10e-5</td>
</tr>
<tr>
<td>9 Feature 9</td>
<td>6.56e-7</td>
<td>1.55e-8</td>
<td>5.108e-5</td>
</tr>
<tr>
<td>10 Feature 10</td>
<td>3.80e-8</td>
<td>2.99e-7</td>
<td>1.68e-5</td>
</tr>
<tr>
<td>11 Feature 11</td>
<td>5.36e-7</td>
<td>1.57e-8</td>
<td>5.97e-5</td>
</tr>
<tr>
<td>12 Feature 12</td>
<td>3.22e-7</td>
<td>2.30e-8</td>
<td>4.29e-5</td>
</tr>
<tr>
<td>13 Feature 13</td>
<td>2.69e-7</td>
<td>7.58e-8</td>
<td>6.20e-6</td>
</tr>
<tr>
<td>14 Feature 14</td>
<td>3.73e-7</td>
<td>2.24e-7</td>
<td>4.99e-5</td>
</tr>
<tr>
<td>15 Feature 15</td>
<td>1.90e-8</td>
<td>1.51e-7</td>
<td>9.08e-6</td>
</tr>
<tr>
<td>16 Feature 16</td>
<td>4.89e-7</td>
<td>1.34e-8</td>
<td>2.18e-5</td>
</tr>
<tr>
<td>17 Feature 17</td>
<td>8.19e-7</td>
<td>7.71e-8</td>
<td>7.06e-5</td>
</tr>
<tr>
<td>18 Feature 18</td>
<td>9.88e-9</td>
<td>1.73e-7</td>
<td>9.32e-6</td>
</tr>
<tr>
<td>19 Feature 19</td>
<td>6.12e-8</td>
<td>1.33e-7</td>
<td>1.74e-5</td>
</tr>
<tr>
<td>20 Feature 20</td>
<td>1.33e-8</td>
<td>9.79e-8</td>
<td>9.14e-6</td>
</tr>
<tr>
<td>21 Feature 21</td>
<td>1.27e-9</td>
<td>4.84e-8</td>
<td>2.38e-6</td>
</tr>
</tbody>
</table>
would be ideal but not expected, since the problem involves a complex environment with highly imbalanced classes. Also the values of the fisher index do not help identify a sub set of features that might be more useful for classification. Still it can be possible to classify the data using a larger combination of features which is the objective of the performance valuation.

One step to follow would be to make new measurements using a newer version of the sensors since the data set comes from two years ago. The sensors installed in the test vehicles are upgraded continuously with more robust versions, therefore a new test could generate data with less noise and false objects. Also a data set should be generated without the ultrasonic sensor since from experience on other projects, this sensor generates more noise than the radars and therefore many false objects which affect the result of the analysis. Having a new and more up to date data set should help identify if the problem relies in the previous data set (used in this work) or in the driver model which is used to extract the set of features that describe the environment around the Ego vehicle.

5.3 Performance Evaluation

In behalf of the results of the data analysis where the features alone and in small groups, do not seem to separate the three classes; an evaluation of the performance from algorithms that use all the features together to perform the classification is suggested. The evaluation will assess the likelihood that these features have to perform the prediction of the lane change maneuvers and depending on the result decide whether it is necessary to pursue more tests or to change the features and try a different approach.

The performance evaluation is divided into three sections, each section corresponds to a different selection of test and training sets. For all the tests, a linear classifier is implemented to compare its results with the non linear models. For the non linear models, multilayer perceptron’s (MLP) with two, five, seven and ten hidden nodes where built along with a random forest with up to 15 trees. Since the results using both resampling algorithms gave similar performances, only the results using random sampling with replacement are shown in this section.
5.3.1 Test 1: Random test and training set

The first test was performed using a random selection of test and training set, note that after dividing both sets, the training set was resampled to have a balance in the classes so that the algorithm consider them equally important.

The test set remains unbalanced and is selected to shape the real conditions that are met in the highway where most of the observations should belong to no lane change and very few to left and right lane change respectively.

This test contains the following structure:

- Train Size = 620267
  - Number of no lane change = 206756
  - Number of left change = 206756
  - Number of right change = 206755

- Test size = 62523
  - Number of no lane change = 61117
  - Number of left change = 1039
  - Number of right change = 367

From the results, see Table 5.6, two behaviors can be observed, on one hand, the MLP and the linear classifier tried to maximize the classification of all the lane changes and upon this get the highest value possible for the classification of no lane change (54%). The behavior from the linear classifier and the MLP reached similar results.

On the other hand the random forest, which reached is smallest error in 10 trees, maximizes the no lane change classification (99.65%) and then reaches a lower performance in the other two classes (in comparison the other two algorithms). Also the misclassification that exist between lane changes to the right and lane changes to the left is more accurate in the random forest.

It should be considered not to use a random selection of observations but instead, using a full section of the data set for training and another section for testing since with this we can ensure that the observations belonging to one object are either taken for training or for testing. The next test will use this idea in the selection of the test and training sets.
<table>
<thead>
<tr>
<th>LDA Classifier</th>
<th>Classified No LC</th>
<th>Classified Left LC</th>
<th>Classified Right LC</th>
</tr>
</thead>
<tbody>
<tr>
<td>No LC</td>
<td>52.69%</td>
<td>27.16%</td>
<td>20.16%</td>
</tr>
<tr>
<td>Left LC</td>
<td>17.15%</td>
<td>71.07%</td>
<td>11.78%</td>
</tr>
<tr>
<td>Right LC</td>
<td>20.17%</td>
<td>10.36%</td>
<td>69.47%</td>
</tr>
<tr>
<td><strong>MLP 2 Hidden Nodes</strong></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>No LC</td>
<td><strong>42.88%</strong></td>
<td>36.42%</td>
<td>20.70%</td>
</tr>
<tr>
<td>Left LC</td>
<td>7.12%</td>
<td><strong>81.23%</strong></td>
<td>11.65%</td>
</tr>
<tr>
<td>Right LC</td>
<td>13.62%</td>
<td>14.17%</td>
<td><strong>72.21%</strong></td>
</tr>
<tr>
<td><strong>MLP 5 Hidden Nodes</strong></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>No LC</td>
<td>51.12%</td>
<td>32.72%</td>
<td>16.16%</td>
</tr>
<tr>
<td>Left LC</td>
<td>11.65%</td>
<td><strong>76.13%</strong></td>
<td>12.12%</td>
</tr>
<tr>
<td>Right LC</td>
<td>18.80%</td>
<td>10.90%</td>
<td><strong>70.30%</strong></td>
</tr>
<tr>
<td><strong>MLP 7 Hidden Nodes</strong></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>No LC</td>
<td>54.72%</td>
<td>29.82%</td>
<td>15.45%</td>
</tr>
<tr>
<td>Left LC</td>
<td>12.13%</td>
<td><strong>76.23%</strong></td>
<td>11.65%</td>
</tr>
<tr>
<td>Right LC</td>
<td>13.08%</td>
<td>13.90%</td>
<td><strong>73.02%</strong></td>
</tr>
<tr>
<td><strong>MLP 10 Hidden Nodes</strong></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>No LC</td>
<td>54.22%</td>
<td>31.38%</td>
<td>14.40%</td>
</tr>
<tr>
<td>Left LC</td>
<td>10.30%</td>
<td><strong>75.94%</strong></td>
<td>13.76%</td>
</tr>
<tr>
<td>Right LC</td>
<td>13.08%</td>
<td>8.99%</td>
<td><strong>77.93%</strong></td>
</tr>
<tr>
<td><strong>Random Forest</strong></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>No LC</td>
<td><strong>99.65%</strong></td>
<td>0.35%</td>
<td>0.0%</td>
</tr>
<tr>
<td>Left LC</td>
<td>46.87%</td>
<td><strong>52.36%</strong></td>
<td>0.77%</td>
</tr>
<tr>
<td>Right LC</td>
<td>45.78%</td>
<td>3.00%</td>
<td><strong>51.23%</strong></td>
</tr>
</tbody>
</table>

Table 5.6: Results of classification by randomly selecting test and training sets, bold numbers correspond to correct percentage of classifications.
5.3.2 Test 2: Continuous test and training set

Ten-fold cross validation was used for selecting ten percent of the data set for testing and the rest for training. This test should help identify if the method for selecting the test and training sets is meaningful in comparison to randomly selecting them.

This test contains the following structure:

- Train Size = 1937802
  - Number of no lane change = 645934
  - Number of left change = 645934
  - Number of right change = 645934

- Test size = 175053
  - Number of no lane change = 172139
  - Number of left change = 2063
  - Number of right change = 851

The average misclassification error from the cross validation is shown in Table 5.7. There is a slightly better performance in comparison with the first test for the MLP and the random forest; while the linear classifier improved the no lane change classification with the trade-off of left and right classification. Although, the difference in performance is not significant enough to say that choosing or not continuous data for training and testing can be relevant for the performance.

The first two tests were performed with very large data sets, these data sets contain much noise and false objects which can affect the learning of the algorithms. Hence, a selection of a smaller data set where there is more understanding of the observations is pursued in the third test.

5.3.3 Test 3: Using objects for test and training set

The objective of this test is to extract a small set of objects and then use these objects for the training and testing. The motivation for this test is that, an object that changes lane should follow a path, this path should be visible in the data by small positive or negative changes in the values of the features. Therefore if it is possible to extract a small set of defined objects with their path. Then, it would be interesting to evaluate the performance of the algorithms.
<table>
<thead>
<tr>
<th>LDA Classifier</th>
<th>Classified No LC</th>
<th>Classified Left LC</th>
<th>Classified Right LC</th>
</tr>
</thead>
<tbody>
<tr>
<td>No LC</td>
<td>62.81%</td>
<td>20.93%</td>
<td>16.26%</td>
</tr>
<tr>
<td>Left LC</td>
<td>24.40%</td>
<td>60.57%</td>
<td>15.02%</td>
</tr>
<tr>
<td>Right LC</td>
<td>41.62%</td>
<td>15.64%</td>
<td>42.74%</td>
</tr>
<tr>
<td>MLP 2 Hidden Nodes</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>No LC</td>
<td>43.01%</td>
<td>30.73%</td>
<td>26.26%</td>
</tr>
<tr>
<td>Left LC</td>
<td>9.21%</td>
<td>78.43%</td>
<td>12.36%</td>
</tr>
<tr>
<td>Right LC</td>
<td>20.45%</td>
<td>9.28%</td>
<td>70.27%</td>
</tr>
<tr>
<td>MLP 5 Hidden Nodes</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>No LC</td>
<td>61.56%</td>
<td>21.57%</td>
<td>16.87%</td>
</tr>
<tr>
<td>Left LC</td>
<td>17.64%</td>
<td>78.43%</td>
<td>8.77%</td>
</tr>
<tr>
<td>Right LC</td>
<td>32.08%</td>
<td>8.34%</td>
<td>59.58%</td>
</tr>
<tr>
<td>MLP 7 Hidden Nodes</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>No LC</td>
<td>53.77%</td>
<td>23.42%</td>
<td>22.81%</td>
</tr>
<tr>
<td>Left LC</td>
<td>11.59%</td>
<td>73.53%</td>
<td>14.88%</td>
</tr>
<tr>
<td>Right LC</td>
<td>23.97%</td>
<td>11.40%</td>
<td>64.63%</td>
</tr>
<tr>
<td>MLP 10 Hidden Nodes</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>No LC</td>
<td>62.08%</td>
<td>21.61%</td>
<td>16.31%</td>
</tr>
<tr>
<td>Left LC</td>
<td>15.46%</td>
<td>76.88%</td>
<td>7.66%</td>
</tr>
<tr>
<td>Right LC</td>
<td>31.73%</td>
<td>11.13%</td>
<td>57.11%</td>
</tr>
<tr>
<td>Random Forest</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>No LC</td>
<td>96.45%</td>
<td>1.45%</td>
<td>2.10%</td>
</tr>
<tr>
<td>Left LC</td>
<td>49.54%</td>
<td>48.18%</td>
<td>2.18%</td>
</tr>
<tr>
<td>Right LC</td>
<td>45.24%</td>
<td>3.17%</td>
<td>51.59%</td>
</tr>
</tbody>
</table>

Table 5.7: Results from the tests using a continuous block of data for training the classifiers, bold numbers correspond to correct percentage of classifications.
Because the sampling time of the observations is 100 milliseconds, the values of one observation should not change drastically to the next observation unless the next observation corresponds to a different object. Following this idea, it is possible to extract a small set of observations that belong to the same object and mark them with an identification number. The objects represent vehicles that are changing lane and therefore an object is formed by a set of observations containing both no lane changes and lane changes. This small set of objects is used for testing and training the algorithms.

This test was performed without resampling since the objective is to see how good the algorithms can learn by the use of objects. This test contains the following structure:

- **Train Size = 3358**
  - Number of no lane change = 2060
  - Number of left change = 1151
  - Number of right change = 147

- **Test size = 169**
  - Number of no lane change = 65
  - Number of left change = 43
  - Number of right change = 61

The idea behind this test is to work with a more controlled set of data; but still, the number of objects that could be extracted successfully was not very large, especially for right lane changes. The results show an overall better performance than with the previous tests, the linear model increased classification of left and right lane changes with the trade-off of no lane change increased misclassification. It could be that the extraction of this small set of objects discarded a large amount of noise that exist within the full data set. Note that the training set was not resampled for this tests to keep the objects intact.

The random forest gave the best performance, using all 15 trees, where more than half of the lane changes were recognized and there was no misclassification between left and right changes. This could indicate that the noise in the data and the false objects could be the responsible for the bad performance in the previous two tests.
## Results and Discussion

<table>
<thead>
<tr>
<th>LDA Classifier</th>
<th>Classified No LC</th>
<th>Classified Left LC</th>
<th>Classified Right LC</th>
</tr>
</thead>
<tbody>
<tr>
<td>No LC</td>
<td>27.69%</td>
<td>26.15%</td>
<td>46.15%</td>
</tr>
<tr>
<td>Left LC</td>
<td>6.98%</td>
<td>93.02%</td>
<td>0.00%</td>
</tr>
<tr>
<td>Right LC</td>
<td>16.39%</td>
<td>0.00%</td>
<td>83.61%</td>
</tr>
<tr>
<td><strong>MLP 2 Hidden Nodes</strong></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>No LC</td>
<td>58.46%</td>
<td>23.08%</td>
<td>18.46%</td>
</tr>
<tr>
<td>Left LC</td>
<td>25.58%</td>
<td>74.42%</td>
<td>0.00%</td>
</tr>
<tr>
<td>Right LC</td>
<td>29.51%</td>
<td>0.00%</td>
<td>70.49%</td>
</tr>
<tr>
<td><strong>MLP 5 Hidden Nodes</strong></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>No LC</td>
<td>49.23%</td>
<td>25.76%</td>
<td>25.01%</td>
</tr>
<tr>
<td>Left LC</td>
<td>24.12%</td>
<td>75.88%</td>
<td>0.00%</td>
</tr>
<tr>
<td>Right LC</td>
<td>27.59%</td>
<td>0.00%</td>
<td>72.41%</td>
</tr>
<tr>
<td><strong>MLP 7 Hidden Nodes</strong></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>No LC</td>
<td>27.69%</td>
<td>26.15%</td>
<td>46.15%</td>
</tr>
<tr>
<td>Left LC</td>
<td>6.98%</td>
<td>93.02%</td>
<td>0.00%</td>
</tr>
<tr>
<td>Right LC</td>
<td>47.54%</td>
<td>0.00%</td>
<td>52.46%</td>
</tr>
<tr>
<td><strong>MLP 10 Hidden Nodes</strong></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>No LC</td>
<td>67.69%</td>
<td>1.54%</td>
<td>30.77%</td>
</tr>
<tr>
<td>Left LC</td>
<td>41.86%</td>
<td>58.14%</td>
<td>0.00%</td>
</tr>
<tr>
<td>Right LC</td>
<td>34.43%</td>
<td>0.00%</td>
<td>65.57%</td>
</tr>
<tr>
<td><strong>Random Forest</strong></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>No LC</td>
<td>92.31%</td>
<td>0.00%</td>
<td>7.69%</td>
</tr>
<tr>
<td>Left LC</td>
<td>18.60%</td>
<td>81.40%</td>
<td>0.00%</td>
</tr>
<tr>
<td>Right LC</td>
<td>29.51%</td>
<td>0.00%</td>
<td>70.49%</td>
</tr>
</tbody>
</table>

Table 5.8: Results from the tests using a selection of objects (each object corresponds to a vehicle).

### 5.3.4 Discussion

In the first two tests, the algorithms could identify lane changes to the left from lane changes to the right even when the data analysis did not make a clear division of these classes. Still there was a higher misclassification level between the lane changes and no lane changes for all three algorithms. Lowering the classification error on the left and right lane changes (MLP behavior), leads to a higher misclassification of no lane changes. In comparison to the performance of the random forest where the classification error for no lane changes was minimized with the trade-off of higher misclassification of the other two classes. The linear classifier could not outperform the other two methods, although it had a similar behavior to the MLP this is to be expected since the environment where the lane change occurs is very complex.

All three algorithms performed better on the third test, the random forest reached a misclassification error of around 25% for lane changes vs. no lane changes which is an improvement over the 50% from the first tests. This suggests that the original data set contains many false objects which affect the training phase. Still the amount of data used in the third test is very small, the number of objects that performed left lane changes was relatively
small. Further study in a selection of objects and creation of a better training set should be pursued by incorporating object ids in the data generation process making the selection of objects for training and testing easier.

In a real implementation, where a driving assistant system wants to be integrated to alert the driver of possible lane changes, the random forest performance would be preferred over the MLP, since in a normal drive most of the observations will correspond to no lane changes and very few will correspond to lane changes; making the assistant to give very few false positives.

Putting the performance of the MLP algorithm in a concrete example; assuming 1000 observations from a test drive which correspond to 100 seconds. These 1000 observations will be formed of 970 no lane changes, 19 left lane changes and 11 right lane changes (according to the distribution seen in the previous test drives). The MLP will send out 417 false warnings of lane changes when no lane change happen; it will warn wrongly 3 lane changes and it will not warn three lane changes when it should have. The results of the LDA classifier, will suggest a similar behavior to the multi layer perceptron having a large number of false warnings.

For the same example the random forest will send 70 false warnings of lane changes when no lane change happen, while not warning about 15 lane changes that occur. The performance from the random forest will be preferred since it produces less false warnings which are more disturbing to the driver than warnings that should have occur but they did not.

Currently each observation is analyzed independently, but the observations that belong to the same object are not independent; the values of the features that belong to one object change gradually as this object prepares the lane change maneuver. If it could be possible to trace not only the current state of the features, but also how those features have been changing since the object was identified; identifying when this object changes its trajectory from keeping a lane into changing a lane could improve the result.

Figure 5.13 shows an example of this scenario, the blue vehicle has a trajectory that will not change lane while the yellow vehicle that had a similar trajectory to the blue at the beginning changes this behavior and then performs the lane change maneuver. If it could be possible to trace how the vehicles features were changing from the moment it was first seen then it might help visualize when something out of the ordinary, a lane change in this case, is going to happen.
Figure 5.13: Example of the difference in trajectory from a vehicle that stays in the same lane and a vehicle that changes lane.

This Chapter presented the results of the lane correction algorithm, the data analysis and the performance evaluation done in this thesis, also the discussions regarding the results and what should be pursued next. The next Chapter includes the summary and conclusion for this thesis.
Summary and Conclusion

This thesis presented an approach for predicting lane change maneuvers on highway scenarios. Using a driver model, a set of features that describe the environment around the vehicle is defined. These features allow for the full description of the position of the Ego and the surrounding vehicles.

To gather data, a test vehicle equipped with a set of sensors is used in different highway drives. The set of sensors allows the vehicle to keep track of all possible objects around it; this object information is then passed through a data preparation process that will convert the raw data information into lanes and objects with aspects such as velocity, acceleration, position. Then, these objects and lanes are used to generate the features that were defined by the driver model and store them in a database so that they can be used to create the output classes.

For the classification, the program goes through the database of objects and in the moment a new object is detected, the program looks in the database whether or not that object will change lane; if the object changes lane, the program assigns to the observations that correspond to two seconds before the lane change, the corresponding class.

The classified data is then analyzed using state of the art data mining techniques, these techniques allow to identify the level of correlation that is present on the features and also help identify which features are more helpful for the classification task. Afterwards, two classification algorithms are built to evaluate the performance that can be achieved with the current data set and the current set of features.

According to the present set of input data, it is possible to identify a left lane change...
Summary and Conclusion

from a right lane change with low misclassification error. But it is not possible to identify, with the same efficiency, a lane change from a no lane change. The best performance using all the data was obtained with a random forest where no lane changes are recognized almost completely but only half of the lane changes are recognized. Using a sub set of data the performance of the random forest increased to have around 70% correct classification between lane changes and no lane changes. This would suggest that the amount of noise in the data could be the responsible for the performance of the first tests where all the data was used.

6.1 Future Work

The problem of noise and false objects could be tested by making new measurement drives using the new versions of the sensors which are more robust and produce less noise than the implementation of two years ago (when the data for this thesis was measured). If the results from a new data analysis and performance evaluation show the same results then the problem lies in the features.

The dependency that exist between continuous observations of the same object could be used by incorporating time dynamics as additional information. Time dynamics means to have information not only of the current state of the vehicle in question, but also of the conditions that the vehicle had in previous observations and how these conditions have changed. Using time dynamics as additional features will increase the information that the Ego has about the surrounding objects. This extra knowledge could help identify the behavior that the vehicle follows and therefore making it easier to recognize when something out of the ordinary, like a lane change, will occur.

A different solution to time dynamics could be the incorporation of extra technology that allows the exchange of information with the surrounding vehicles (inter-vehicle communication). Inter vehicle communication is another field of research for driving assistance systems where the communication among vehicles is used to improve traffic conditions and make driving more secure, an example of the research done in this field is the Grand Cooperative Driving Challenge, which had its first occurrence on May of 2011 [39].

There exists the possibility that communicating with the surrounding vehicles will improve the prediction task by knowing extra information about the other vehicles; some examples of this information are: the final destination, average speed of the vehicle in its whole trip, style of driver, trajectory that their GPS could have planned. This new information added
to the previous measured features could improve the prediction of maneuvers.

It is the conclusion of this thesis that further analysis of the system should be pursued, making a new test with new data and determining whether or not it is necessary to incorporate more features or more technology; so the performance of the prediction system can achieve an implementation robust enough to be brought to the public.
Bibliography


[25] A. Sporrer, G. Prell, J. Buck, S. Schaible, **Realsimulation von Spurwechselvorgän-
gen im Straenverkehr**. Technical report, Ingenieurburo Nickl, Karl-Mangold Str. 17


[27] F. Gorunescu, **Data Mining - Concepts, Models and Techniques**. Springer-Verlag
Berlin, Heidelberg 2011.

[28] K.R.Venugopal, K.G. Srinivasa, L.M. Patnaik, **Soft Computing for Data Mining

[29] D. J. Ketchen, Jr and C. L. Shook, **The application of cluster analysis in Strategic
Management Research: An analysis and critique**. Strategic Management Journal
17, 1996.

**Support Vector Machine Classification of Microarray Gene Expression Data**.

[31] P. R. Cohen, **Empirical Methods for Artificial Intelligence**. MIT Press, Cambridge,
Massachusetts, 1995.

[32] Q. Gu, Z. Cai, L. Zhu, B. Huang, **Data Mining on Imbalanced Sets**. IEEE Interna-

[33] J. Fulcher, C. Lakhmi, **Computational Intelligence: A Compendium**. Springer-

[34] T. Li, S. Zhu, M. Ogihara **Using Discriminant Analysis for Multi-Class Classifi-
cation: An Experimental Investigation**. Knowledge and Information Systems,
Volume 10, Number 4, 2006.

breiman/RandomForests/](http://www.stat.berkeley.edu/~breiman/RandomForests/)
Visited, 07.2011.

[36] A. A. Aboul-Ella Hassanien, A. Ponce de Leon, F. de Carvalho, **Foundations of Com-
putational, IntelligenceVolume 6 - Data Mining**. Springer-Verlag Berlin, Heidelberg
2009.

[37] S. Haykin, **Neural Networks, A Comprehensive Foundation**. Prentice Hall New

[38] R. Rojas, **Neural Networks, A Systematic Introduction**. Springer-Verlag New
York, 1996.