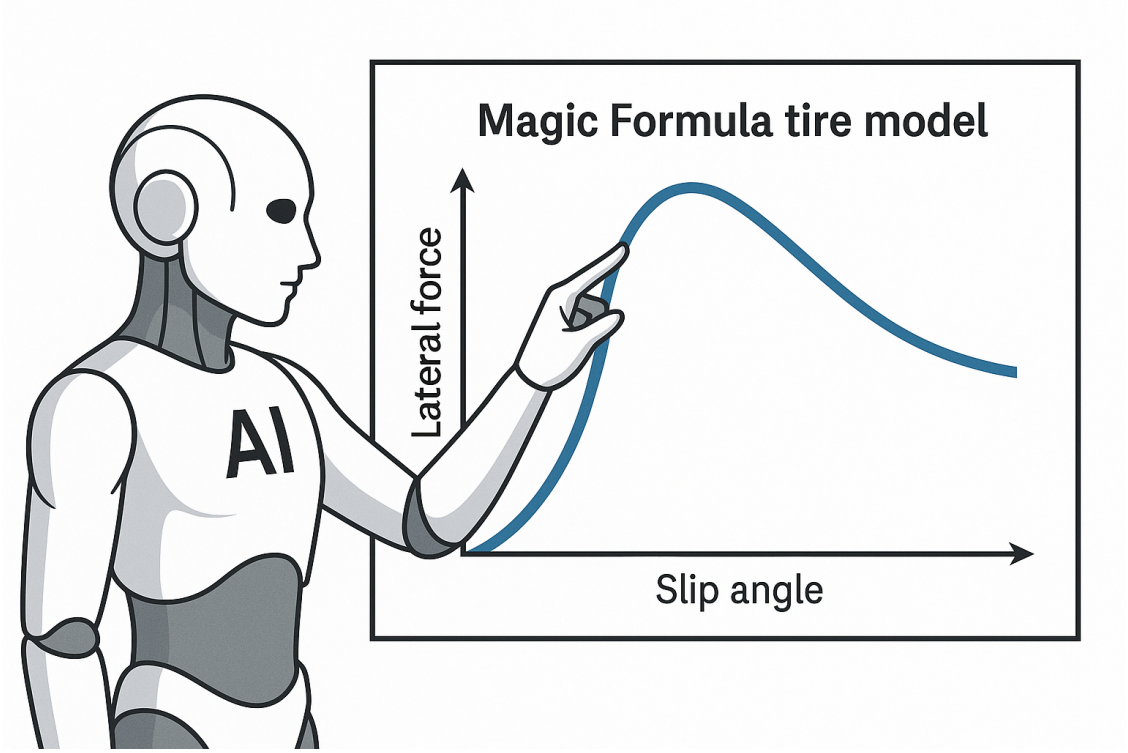


Machine Learning-Based Tire Model Optimization

Master's Thesis in Computer Science and Engineering

JOEL GRÅSJÖ & KEVIN TO

Department of Computer Science and Engineering
CHALMERS UNIVERSITY OF TECHNOLOGY
UNIVERSITY OF GOTHENBURG
Gothenburg, Sweden 2025

Master's Thesis 2025

Machine Learning-Based Tire Model Optimization

JOEL GRÅSJÖ & KEVIN TO

UNIVERSITY OF
GOTHENBURG

CHALMERS
UNIVERSITY OF TECHNOLOGY

Department of Computer Science and Engineering
Division of Data Science and AI
Chalmers University of Technology
University of Gothenburg
Gothenburg, Sweden 2025

Machine Learning-Based Tire Model Optimization
JOEL GRÅSJÖ & KEVIN TO

Supervisor: Hampus Gummesson Svensson, Department of Computer Science and Engineering
Advisor: Per Blomberg, Volvo Cars; Max Boerboom, Volvo Cars; Xin Li, Volvo Cars; Axel Villandseie, Volvo Cars
Examiner: Morteza Haghir Chehreghani, Department of Computer Science and Engineering

Master's Thesis 2025
Department of Computer Science and Engineering
Division of Data Science and AI
Chalmers University of Technology and University of Gothenburg
SE-412 96 Gothenburg
Telephone +46 31 772 1000

Cover: AI-generated image of a robot manipulating a Magic Formula force curve. (Created with OpenAI's ChatGPT-4-turbo model using the prompt "Create an image (4x3) where an AI is manipulating a Magic Formula tire model curve").

Typeset in LaTeX
Gothenburg, Sweden 2025

Machine Learning-Based Tire Model Optimization
JOEL GRÅSJÖ & KEVIN TO
Department of Computer Science and Engineering
Chalmers University of Technology and University of Gothenburg

Abstract

When developing new vehicles, accurate tire models are crucial for simulating real-world driving behavior. However, inconsistencies in measurement and fitting methods between tire manufacturers introduce bias, leading to different models for the same physical tire.

This thesis proposes a method to correct such bias by translating biased tire models into unbiased ones using supervised learning and reinforcement learning (RL). Initially, three supervised classifiers, Artificial Neural Network, Gaussian Naive Bayes, and Random Forest, were trained to identify the origin (manufacturer) of a tire model, serving as a proxy for the unbiased data. RL models were then formulated to optimize towards this classifier proxy, where the agents aimed to reach states the classifier predicted as "unbiased". To find a good policy, two different RL algorithms were investigated.

The Random Forest classifier demonstrated the highest accuracy, predicting the origin of a tire model with 97.0% confidence. However, the translation results, evaluated by Mean Squared Error, were mixed. While the RL agents learned to improve the classifier's assessed probability, only some translated curves showed improvement.

Future work may improve the results through systematic hyperparameter tuning of the RL environment and agent. Additionally, access to more labeled tire model data would enable a more precise assessment of variance and bias, supporting the identification and analysis of underlying measurement errors. Despite the limitations, this thesis demonstrates the potential of combining supervised learning and RL for bias correction in tire modeling.

Keywords: Bias Detection, Classification, Machine Learning, Magic Formula, Reinforcement Learning, Tire Model, Tire Property File, Vehicle Dynamics.

Acknowledgments

This thesis was conducted in collaboration with Volvo Cars, and we would like to send our deepest gratitude to the people that have made this thesis possible and have helped and encouraged us along the way.

Firstly, we would like to send our gratitude to Max Boerboom, Axel Villandseie, Hossein Abadikhah, Xin Li and Per Blomberg at Volvo Cars for their continuous feedback and their technical guidance.

We would also like to thank Hampus Gummesson Svensson and Morteza Haghir Chehreghani at Chalmers for their help and feedback during the thesis.

Joel Gråsjö & Kevin To, Gothenburg, 2025-06-10

Contents

List of Figures

List of Tables

1

Introduction

The handling and steering performance of a vehicle are critical to its driving characteristics, encompassing agility, predictability, controllability, steerability, and stability. Tires, as the sole interface between the vehicle and the road, play a decisive role in influencing these dynamics and directly impact ride quality and handling behavior.

In the concept phase of vehicle development, Computer-Aided Engineering simulations rely on tire models that accurately capture their real-life behavior to get reliable results. These models are often created using the Magic Formula (MF), a semi-empirical tire model that calculates the force and moment characteristics of a tire [1].

The process of going from a physical tire to a tire model involves several steps:

1. A tire is manufactured in a production plant or concept workshop.
2. The tire undergoes conditioning to "run-in" the tire.
3. The tire's forces and moments are measured using proprietary measuring methods under different loading conditions. These measurements are performed on either Flat-Trac machines[1] or using a physical vehicle.
4. The raw data is collected.
5. The data is prepared and turned into a TYDEX file, a standardized format for tire measurement data [3].
6. The generic MF tire model (see Section 2.1.1) is fitted to the conditioned TYDEX data using a fitting tool.

The first five steps are usually performed by a tire manufacturer, who produces a tire property file (.tir) with values for all the coefficients of the MF (see Appendix A). It is also possible to obtain .tir files from a measurement institute. This is generally more costly, but provides complete insight into the methods used to measure the tire. As described by Oosten and Bakker [4], Alagappan, Rao, and Kumar [5], [6], Farroni, Lamberti, Mancinelli, et al. [7], and Feng, Zhao, Deng, et al. [8], the problem of fitting measured data to the MF tire model is established and is usually done with a fitting software.

[1] A Flat-Trac machine is a tire force and moment measurement system designed to evaluate tire performance and handling by applying various inputs to a spinning tire on a flat surface [2].

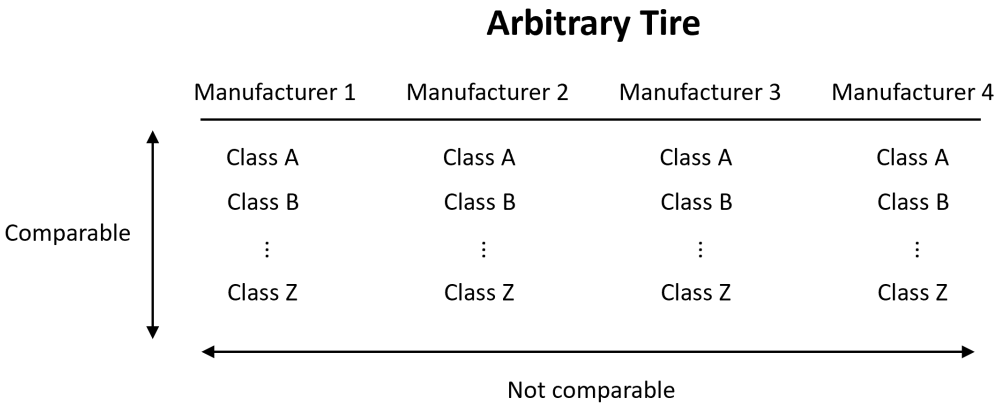


Figure 1.1: Illustration of an arbitrary tire measured by different tire manufacturer. The tire models of the same class between different manufacturers differ from each other in terms of behavior when they should be the same.

Due to the competitive nature of the tire manufacturing market, the exact tools and methods used by tire manufacturers are not disclosed. As such, differences in measurement, conditioning, and fitting methods can cause significant discrepancies between the MF coefficient values of the same physical tire. Moreover, as described by Alagappan, Rao, and Kumar [5], the choice of fitting algorithm used in the fitting software can also produce different sets of MF coefficients for the same raw data.

This problem is illustrated in Figure 1.1. For any tire produced by a single manufacturer, the differences between tire classes (e.g., Class A, Class B, etc.) are well-defined and consistent, since they use the same tools and methods to measure their tires. This consistency allows for reliable comparisons across different classes within the same manufacturer. However, due to biases introduced by different measurement regimes and fitting routines, tires of the same class from different manufacturers may not be directly comparable in terms of behavior. This effectively yields different tire models with different driving perceptions.

Interestingly, analyzing the available .tir files, it can be observed that the differences between the manufacturers' tire models' MF coefficients seem to differ in the same way, meaning that each manufacturer exhibits some kind of hidden bias. It can also be noted that some tire models differ from their real-life behavior and can overpromise their tire performance. Consequently, this complicates vehicle development.

Given that each tire manufacturer appears to exhibit their own hidden biases, it should be possible to identify these patterns. By leveraging the understanding of these biases, this thesis will explore a methodology using machine learning (ML) to determine whether it is possible to convert a biased tire model into an unbiased one.

1.1 Purpose of the Thesis

This thesis aims to use machine learning techniques, such as supervised learning and reinforcement learning, in the domain of tire modeling in order to improve its

quality. The objective is to use a combination of ML techniques to translate biased tire property files into unbiased ones, thereby aligning the real-world behavior more closely with the output of the MF tire model. The results of this thesis can thus be used to improve simulation correlation and enable more precise vehicle tuning.

The following research questions (RQs) will be studied:

- RQ1: Can supervised learning be used to classify tire property files?
- RQ2: Can a deep reinforcement learning agent learn a policy that optimizes the output probabilities of a classifier?
- RQ3: Can reinforcement learning or other machine learning techniques be used to translate biased tire models into unbiased models ?
- RQ4: What learnings about the data can be made from the reinforcement learning policies?

RQ1 will be answered by comparing the accuracy and weighted average F1 score[2] of different classifiers in classifying .tir files.

To answer RQ2, the behavior of the reinforcement learning model during training will be used. If the reinforcement learning model shows signs of learning a policy and improving over time until convergence of training rewards, RQ2 will be considered answered.

For answering RQ3, an arbitrary .tir file will be translated to match a reference .tir file, which will serve as the unbiased .tir file. The force curves from the two .tir files will be compared to assess how closely they match.

RQ4 will be assessed by examining the actions performed by the reinforcement learning model. Both in terms of distributions of actions as well as the effect of the most frequent actions.

1.2 Originality and Significance

There exists a lot of research on how to parameterize raw tire data into MF coefficients [4], [5], [6], [7], [8]. However, there is a clear gap in the research on how to validate and set said coefficients without access to raw data. The better aligned with reality the tire model is, the more information it can provide.

There exists one obvious solution to the problem, namely collect all data and brute-force a solution to the problem. The problem with this approach is that it is prohibitively expensive and time-consuming. Therefore, an approach that can increase data accuracy without elevating the data needs could provide better tire models for much cheaper than alternative approaches.

Another approach that might be feasible for the given problem is to define an optimization problem using a classifier, and then use techniques for optimizing non-

[2]F1 score is a metric used to evaluate the performance of a classifier, and is the harmonic mean of precision and recall [9].

convex problems instead of reinforcement learning. However, due to the existence of similar approaches for problems in other domains [10], [11], reinforcement learning seems to be promising.

1.3 Thesis Outline

The thesis will begin in Chapter 2 by presenting the relevant theory. This chapter is divided into three parts: the first part provides a general understanding of the MF tire model, the second part explains the machine learning classifiers used and gives an overview of reinforcement learning, and the third part details the data preprocessing methods employed.

Next, in Chapter 3, a detailed mathematical description of the problem, the available data, and the proposed solution is provided. This chapter offers a comprehensive understanding of the feasibility of our approach.

In Chapter 4, the details regarding how the dataset was preprocessed are described, as well as four different ways of representing the data. We also show how the classifiers were trained and set up, and how the reinforcement learning model was developed. The metrics used to evaluate the model are also explained.

Moreover, in Chapter 5, the results of the classifier and RL model are presented, followed by a discussion in Chapter 6. Finally, in Chapter 7, a conclusion of our work is presented along with opportunities for future research.

2

Theory

In this chapter, the relevant theory needed to understand the topic of the thesis is provided. It will first describe the MF tire model in detail, and then explain different types of classifiers and the fundamentals of reinforcement learning. The chapter will also introduce the data preprocessing tools used in this thesis.

2.1 Tire Modeling

To capture the behavior of the physical tire in a simulator, the MF tire model can be used. It was developed in the mid-1980s in a collaboration between Delft University of Technology and Volvo Cars [1]. The first software product, titled *MF-Tire*, implemented the MF Tire Model in various multi-body software packages. In the late 1990s, version 5.2 of MF-Tire achieved industry standard status for modeling passenger-car tires. Over time, additions and updates have been made to MF-Tire, resulting in newer versions. In this thesis, the equations are in line with the description of MF-Tire 5.2 described in [12].

2.1.1 Mathematical Description of the Magic Formula

The general form of the MF is expressed as follows [1]:

$$y = D \sin\left(C \arctan\left(Bx - E\left(Bx - \arctan(Bx)\right)\right)\right) \tag{2.1}$$

with

$$Y(X) = y(x) + S_V$$
$$x = X + S_H$$

where Y is the output of the longitudinal force, F_x, or lateral force, F_y. X is the input variable of either longitudinal slip, κ, for F_x, or slip angle, α, for F_y. To obtain the overturning torque, M_x, rolling resistance, M_y, or aligning torque, M_z, the sine function is replaced by a cosine function, where the input can be α or camber angle, γ.

The variables B, C, D, E, S_H, and S_V are constants that depend on further underlying equations (see Appendix A). In these equations, camber angle, γ, and normal load, F_z, of the vehicle are used. The inputs and outputs of the MF are summarized in Table 2.1. Additionally, Table 2.2 indicates which inputs are used to calculate each output of the MF. For further reference, see Appendix A.

Table 2.1: This presents the inputs and outputs of the MF as described in Eq. 2.1. Different combination of inputs are used for the different outputs.

Inputs	**Outputs**
Slip Angle (α)	Longitudinal Force (F_x)
Longitudinal Slip (κ)	Lateral Force (F_y)
Camber Angle (γ)	Overturning Torque (M_x)
Normal Load (F_z)	Rolling Resistance (M_y)
	Aligning Torque (M_z)

Table 2.2: Mapping of inputs to each output of the MF. A "×" symbol denotes that the input is utilized in the computation of the corresponding MF output.

		Inputs			
		$\boldsymbol{\alpha}$	$\boldsymbol{\kappa}$	$\boldsymbol{\gamma}$	$\boldsymbol{F_z}$
Outputs	F_x		×	×	×
	F_y	×		×	×
	M_x			×	×
	M_y			×	×
	M_z	×		×	×

2.1.2 Shape of the Magic Formula Equation

The curve generated by Eq. 2.1 is illustrated in Figure 2.1. As can be seen, the curve typically intersects the origin ($x = y = 0$), obtains a maximum, and approaches a horizontal asymptote [1]. For different values of the variables B, C, D, and E, the curve can be asymmetric with respect to the origin. To account for this, the variables S_H and S_V are added to shift the curve from the origin and are called horizontal and vertical shift factors, respectively.

Furthermore, the variable C, known as the *shape factor*, limits the range of the sine function, thus determining the horizontal asymptote of the curve. The variable D represents the *peak value* of the curve, while B controls the angle of the slope at the origin and is called the *stiffness factor*. The product of these variables, BCD, represents the slope at the origin. Finally, E, the *curvature factor*, influences the curvature and the horizontal position of the peak.

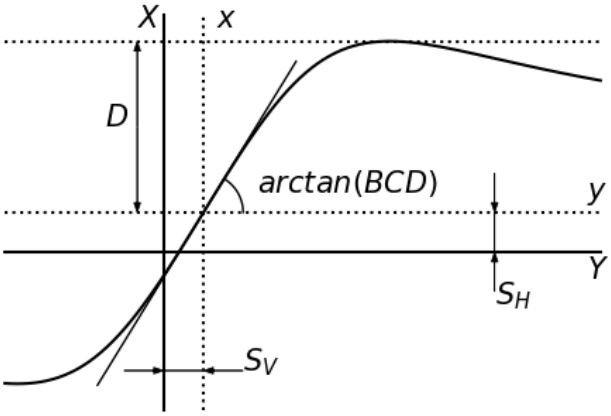


Figure 2.1: The MF equation illustrated with slip angle (α) on the X-axis and lateral force (F_y) on the Y-axis.

2.1.3 Cornering Stiffness

When a vehicle travels on a curve, the lateral force (F_y) pushes each wheel sideways against the ground [13]. This causes the tire to slightly deform, making the path the tire follows on the ground different from the direction the wheel is pointing, creating a slip angle. The resistance of the tire to this deformation is called cornering stiffness. When cornering, the load transfers from the inner to the outer wheel, thereby affecting the effective cornering stiffness of the tire. An important relationship is the variation of the cornering stiffness, BCD_y, with F_z and γ [1]. The relation can be expressed as:

$$BCD_y = p_1 \sin[2 \arctan(F_z/p_2)]/(1 + p_3\gamma^2) \tag{2.2}$$

where $p_1 = F_{z0}p_{Ky1}$, $p_2 = F_{z0}p_{Ky2}$ and $p_3 = p_{Ky3}$ are non-dimensional parameters with F_{z0} being the nominal wheel load. When the camber angle (γ) is zero, Eq. 2.2 states that the cornering stiffness reaches its maximum p_1 at $F_z = p_2$, see Figure 2.2.

2.1.4 Load Sensitivity of Peak Friction Coefficient

The load sensitivity μ defined as:

$$\mu = (p_1 + p_2) \cdot (1 - p_3\gamma),$$

where $p_1 = p_{Dy1}$, $p_2 = p_{Dy2}df_z$ and $p_3 = p_{Dy3}$, is a factor that decides how much of the normal load can be converted into friction, i.e. $F_z \cdot \mu$ is the peak of the MF curve, denoted as D in Eq. 2.1.

2.2 Supervised Learning

Supervised learning is a machine learning (ML) technique that makes it possible for a system to learn the input-output relationship based on a given set of paired training samples [14]. In essence, given an input, the system learns to predict the

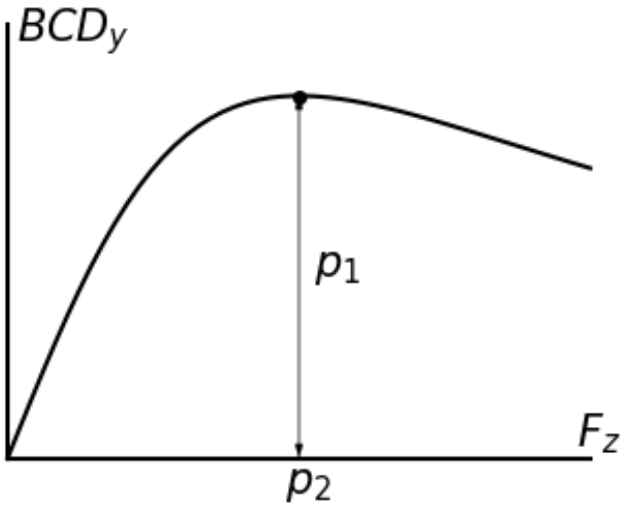


Figure 2.2: Cornering stiffness (BCD_y) as a function of normal load (F_z) with $\gamma = 0$.

correct output. If the output is a finite set of discrete values that indicate the class labels of the input, the systems are called classifiers.

In this section, the different types of classifiers evaluated are described. The selected classifiers to investigate were Artificial Neural Network, Naive Bayes, and Random Forest. These classifiers use different methodologies to achieve their classification [15] and were chosen to provide a diverse range of approaches for comparison.

2.2.1 Artificial Neural Network

Artificial neural networks (ANN) are models that convert an input vector into an output [16]. They are constructed as a chain of different layers, where each layer consists of different nodes that act in parallel. The first layer of the network is called the input layer, whereas the final layer is called the output layer. The number of nodes in the input layer depends on the dimension of the input data. For instance, if the input data has x parameters, then the input layer will have x nodes. Similarly, the number of nodes in the output layer will depend on the model's task. If the goal of the model is to classify N categories based on arbitrary data, the output layer will have N nodes, each representing the probability that the input data belongs to one of the categories. In Figure 2.3, a simple ANN is depicted.

2.2.1.1 Activation Functions

The layers between the input and output layers are known as hidden layers [16]. Between the hidden layers, an activation function is applied to the outputs of each layer. With the use of a non-linear activation function, the model will learn non-linear relationships in the data [17]. A commonly used non-linear activation function is the

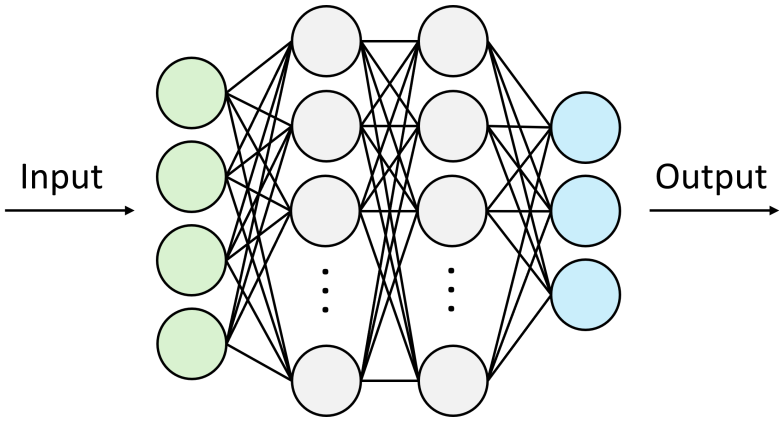


Figure 2.3: ANN with four input nodes, two hidden layers, and three output nodes.

Rectified Linear Unit (ReLU), defined as

$$\mathrm{ReLU}(x) = \max(0, x) = \begin{cases} x, & \text{if } x \geq 0. \\ 0, & \text{otherwise.} \end{cases},$$

where x is the output of the previous hidden layer. This output is then passed on to the next layer as a new input.

For classification tasks, the Softmax function, denoted as $s(x)$, is a common choice in the last layer of an ANN to convert the output into a discrete probability distribution over the N target classes [18]. Given the output vector $\mathbf{o} = [o_1, o_2, \ldots, o_N]$ from the last hidden layer, the Softmax function assigns each class i its own probability p_i as follows:

$$p_i = s(o_i) = \frac{\exp(o_i)}{\sum_{n=1}^{N} \exp(o_n)}.$$

Furthermore, to train the ANN, a backpropagation algorithm must be used [19].

2.2.1.2 Overfitting and Weight Decay

An ANN can be overfitted if the network is very complex and there is little information in the training data [20]. One way to limit a model's complexity and thereby improve its generalization is weight decay.

For adaptive gradient algorithms, such as the AdamW optimizer, weight decay, λ, is induced directly to the weight parameter [21], defined as

$$w_t = w_{t-1} - \gamma\lambda w_{t-1},$$

where w are the current weights, and γ is the learning rate. As motivated by Loshchilov and Hutter [21], this leads to better generalization for adaptive gradient algorithms compared to the method described by Rumelhart, Hinton, and Williams [19], where weight decay is applied to the loss function. In Figure 2.3, the weights of the ANN are represented by the lines connecting the nodes, which are updated as the model trains.

2.2.2 Naive Bayes

Naive Bayes (NB) classifiers are classifiers that are based on Bayes theorem [22],

$$P(y|x_1,\ldots,x_n) = \frac{P(y)P(x_1,\ldots,x_n|y)}{P(x_1,\ldots,x_n)}.$$

The method assumes naive conditional independence, i.e.,

$$P(x_i|y,x_1,\ldots,x_{i-1},x_{i+1},\ldots,x_n) = P(x_i|y).$$

Since $P(x_1,\ldots,x_n)$ is constant,

$$P(y|x_1,\ldots,x_n) \propto P(y)\prod_{i=1}^{n} P(x_i|y).$$

Then the prediction $\hat{y}$ is computed as

$$\hat{y} = \arg\max_y P(y)\prod_{i=1}^{n} P(x_i|y).$$

A Gaussian NB (GNB) classifier is a NB classifier that assumes a Gaussian likelihood for the features [22], i.e.,

$$P(x_i|y) = \frac{1}{\sqrt{2\pi\sigma_y^2}}\exp\left(-\frac{(x_i-\mu_y)^2}{2\sigma_y^2}\right),$$

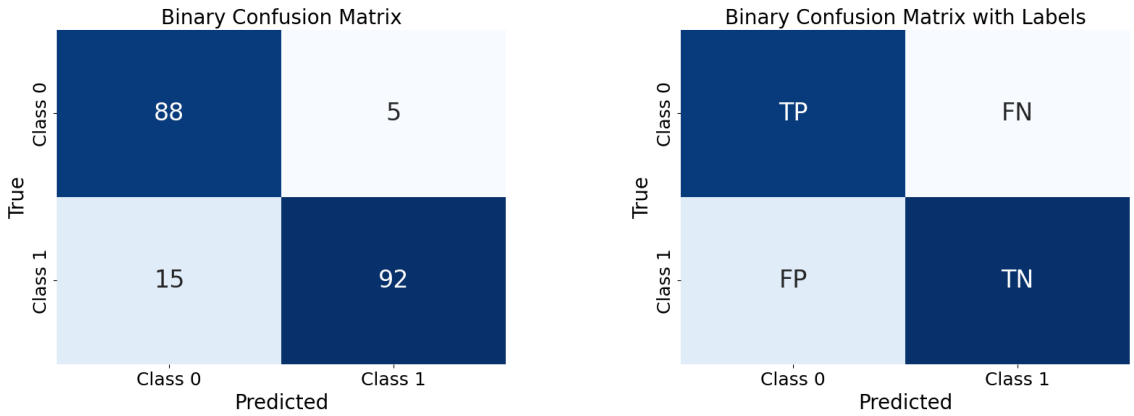
where σ_y and μ_y are estimated using maximum likelihood.

2.2.3 Random Forest

The Random Forest (RF) classifier was first developed in 1995 and is based on multiple decision tree classifiers [23]. It is an ensemble learning method, which means that it uses many classifiers at the same time and aggregates their results [24]. The method used in the Random Forest classifier is bagging of classification trees. In bagging, successive trees do not depend on earlier trees. Instead, each tree is constructed independently using a bootstrap sample of the data. At each node, the data is split using the best among a subset of randomly chosen predictors. In a classification task, the prediction becomes the class that gets the majority vote among all trees [24].

2.2.4 Metrics for Evaluation

To evaluate a classifier, several metrics can be used. One way is to present the result with a confusion matrix. This is a table that summarizes the output of the classifier [25]. An example of a confusion matrix for a binary classification task is seen in Figure 2.4a.



(a) Example of binary confusion matrix.

(b) Binary confusion matrix with labels.

Figure 2.4: Examples of binary confusion matrices.

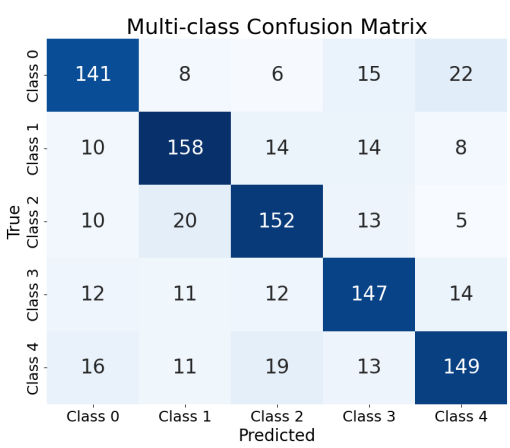
Figure 2.4a shows the classifier's predictions, where the values on the diagonal from the upper left corner to the lower right corner represent the correct predictions. From the perspective of Class 0, the cells can be labeled in four different ways, see Figure 2.4b:

- True Positive (TP): Represents correctly classified points for Class 0.
- True Negative (TN): Represents correctly classified points not being Class 0.
- False Positive (FP): Represents instances where the classifier wrongly predicts Class 0 when it is actually another class.
- False Negative (FN): Represents instances where the classifier mistakes Class 0 for another class.

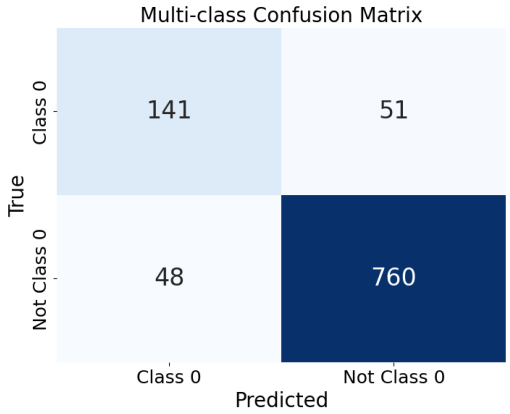
For a multiclass classification problem, a similar confusion matrix can be generated, see Figure 2.5a. As in the binary case, the diagonal represents the correctly predicted classes. To convert this confusion matrix to a binary one, the numbers need to be aggregated. From the perspective of Class 0:

- TP is the value (141) in the upper left corner.
- TN is the sum of the correctly predicted points (760) that are not Class 0.
- FP is the sum of the values (48) in Class 0 column.
- FN is the sum of the values (51) in Class 0 row.

This results in the confusion matrix shown in Figure 2.5b. The same logic can be applied to find TP, TN, FP, FN for the other classes. With these values, metrics such as accuracy, precision, recall, and F1 score can be calculated [25].



(a) Example of multiclass confusion matrix.



(b) Multiclass confusion matrix converted to a binary confusion matrix.

Figure 2.5: Examples of multiclass confusion matrices.

Accuracy This metric is simply defined as the fraction of correctly classified points [25]:

$$\text{Accuracy} = \frac{\text{Correct Predictions}}{\text{Total Predictions}} = \frac{TP + TN}{TP + TN + FP + FN}$$

Precision The precision of a classifier is the ratio of all samples actually belonging to the target class by the number of samples predicted to be the target [25]:

$$\text{Precision} = \frac{\text{Correct Predictions of Target}}{\text{Total Predictions of Target}} = \frac{TP}{TP + FP}$$

This metric is effectively a measure of how well the classifier can classify its target correctly.

Recall Recall is a similar metric to precision, but instead of dividing by the total number of samples predicted to be the target, it divides by the total number of samples that were of the target class [25].

$$\text{Precision} = \frac{\text{Correct Predictions of Target}}{\text{Total Samples of Target}} = \frac{TP}{TP + FN}$$

This metric measures the effectiveness of the classifier in labeling samples as the target when they are the target.

F1 score A combined measure of precision and recall is the F1 score. This is the harmonic mean between precision and recall [25]:

$$\text{F1 score} = \frac{2 \cdot \text{Precision} \cdot \text{Recall}}{\text{Precision} + \text{Recall}}$$

There are different ways of calculating the F1 score for multiclass classification tasks [9]. One way is to calculate the F1 score of each individual class based on its precision and recall scores. Then, a weighted average of each class's F1 score is calculated based on the number of samples of each class. This is called a weighted average F1 score and is a suitable metric when the data contains class imbalance [9].

2.3 Reinforcement Learning

Reinforcement learning (RL) is a method of learning what to do in order to maximize a numerical reward signal [26]. It does not explicitly state which actions the learner should take; instead, the learner must train and discover which actions yield the most reward. The actions affect not only the immediate reward, but the future situations as well. Essentially, RL uses training to iteratively evaluate the actions the learner takes, with the goal of maximizing its reward over time through their choice of actions. Eventually, this guides the learner to the optimal solution.

The key features of RL are [26]:

- Agent: The learner/decision maker.
- Environment: The space which the agent interacts with.
- State (S): A representation of the environment and agent at a time step t.
- Action (A): A choice of action moving the agent to another state given the current state.
- Reward (R): A numerical value the agent seeks to maximize over time, usually a function depending on the current and next state.
- Policy (π): A mapping from states to probabilities of selecting each possible action.
- Value function ($v_\pi(s)$): The expected return when starting in S_t and applying policy π, see Eq. 2.4.
- Action-value function ($q_\pi(s, a)$): The expected return when starting in S_t, taking action A_t, and following policy π, see Eq. 2.5.

2.3.1 Agent-Environment Interaction

The agent interacts with its environment in a sequence of discrete time steps t [26]. At each t, the agent enters one of the environment's states $S_t \in \mathcal{S}$, whereby it selects an action $A_t \in \mathcal{A}(s)$. One time step later, $t + 1$, the agent receives a reward $R_{t+1} \in \mathcal{R} \subset \mathbb{R}$ and enters a new state S_{t+1}. The sequence of these steps from start to finish is called an episode or trajectory if there is no pre-defined stop. These steps repeat in an RL model and can be visualized as:

$$S_0 \to A_0 \to R_1 \to S_1 \to A_1 \to R_2 \to ...$$

In Figure 2.6, a schematic visualization of the RL loop is illustrated.

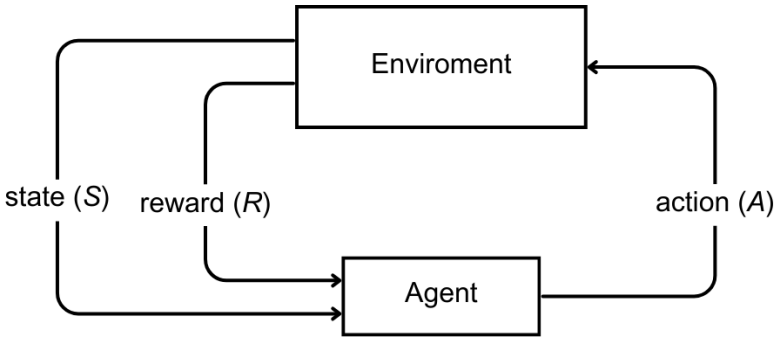


Figure 2.6: Simple RL loop.

2.3.2 Reward and Expected Return

The ultimate end goal of an agent is to maximize the cumulative reward over time. Since future rewards are less valuable than what they would be if they were received immediately, a discount rate, γ, is introduced. The discount rate is determined by the specific problem context and reflects the relative importance of future rewards compared to immediate ones. The agent therefore chooses an action A_t that maximizes the expected discounted return, G_t [26]:

$$G_t = R_{t+1} + \gamma R_{t+2} + \gamma^2 R_{t+3} + ... = \sum_{k=0}^{\infty} \gamma^k R_{t+k+1}, \tag{2.3}$$

where γ can take values $0 \leq \gamma < 1$.

2.3.3 Policy and Value Functions

An agent's policy is its probability of selecting an action given its current state [26]. This probability can be expressed as: $\pi(a|s)$. When an agent applies a given policy, the value function can determine the expected future return in a state. Using Eq. 2.3, this function can be written as:

$$v_\pi(s) = \mathbb{E}_\pi[G_t \mid S_t = s] = \mathbb{E}_\pi \left[\sum_{k=0}^{\infty} \gamma^k R_{t+k+1} \,\middle|\, S_t = s \right], \forall s \in \mathcal{S}, \tag{2.4}$$

where $\mathbb{E}_\pi$ is the expected value of a random variable given that the agent follows policy π.

In addition, it is also possible to describe the value of taking action a in state s under policy π. This is called the action-value function, $q_\pi(s, a)$, and can be expressed as follows:

$$q_\pi(s, a) = \mathbb{E}_\pi[G_t \mid S_t = s, A_t = a] = \mathbb{E}_\pi \left[\sum_{k=0}^{\infty} \gamma^k R_{t+k+1} \,\middle|\, S_t = s, A_t = a \right]. \tag{2.5}$$

The *optimal value functions*, v^* and q^*, are those that give the highest expected return by any policy, and these functions are unique. However, there could be several different *optimal policies* that satisfy the optimal value functions.

2.3.4 Reinforcement Learning Algorithms

For the agent to learn, different algorithms can be used. Two algorithms for reinforcement learning are Advantage Actor Critic and Proximal Policy Optimization, which are described in this section. Both algorithms are based on the Actor-Critic framework [26], which uses an actor who explores actions randomly and a critic who gives feedback on the actions taken. In essence, the actor learns to improve its policy (π) for taking actions, while the critic learns to update its action-value function ($q_\pi(s, a)$) to assess how good the actions taken are in a given state. These algorithms enable the agent to learn and find its solution in different ways.

2.3.4.1 Advantage Actor Critic

Advantage Actor Critic (A2C) is a synchronous and deterministic variant of Asynchronous Advantage Actor Critic (A3C) proposed by Mnih, Badia, Mirza, et al. [27]. As the name implies, both algorithms use the Actor-Critic framework. However, instead of using $q_\pi(s, a)$ to evaluate the actor, the critic uses the advantage function:

$$A(s, a) = q(s, a) - v(s),$$

where $v(s)$ is the average value of the state s. This function calculates the extra reward for taking the action in a state compared to the mean reward when staying in the state. In [27], an estimate of the advantage function is used:

$$A(s, a) \approx \sum_{i=0}^{k-1} \gamma^i r_{t+i} + \gamma^k v(s_{t+1}; \theta_v) - v(s_t; \theta_v),$$

where k varies for different states and is bounded by t_{max} (arbitrarily chosen) , γ^i, γ^k are discount factors, r_{t+1} is the return at $t + 1$, and θ_v are parameter values to the value functions.

A2C is implemented in the Python library `stable_baselines3` [28]. In Table 2.3, a subset of the tunable parameters is presented.

Table 2.3: Subset of parameters for A2C in `stable_baselines3` [28].

Parameter	**Default Value**	**Allowed Range**
Policy (π)	N/A	N/A
Learning rate	0.0007	$[0, 1]$
Number of roll-out steps (T)	5	$[1, \infty)$
Gamma (γ)	0.99	$[0, 1]$
Number of parallel actors (N)	1	$[1, \infty)$

2.3.4.2 Proximal Policy Optimization

Proximal Policy Optimization (PPO) is a popular family of policy gradient methods announced in 2017 [29]. They are policy-based RL algorithms, which means that they optimize the agent's policy instead of its value function.

In essence, policy gradient methods compute an estimator of the policy with the goal of maximizing it using stochastic gradient ascent [29]. The result is an updated policy. However, optimizing multiple times using the same trajectory shows that it often leads to undesirably large policy updates

This issue can be mitigated with Trust Region Policy Optimization methods [29]. These methods use a "surrogate" objective function that is maximized subject to a constraint on the size of the policy update:

$$\underset{\theta}{\text{maximize}} \quad \mathbb{E}_t \left[\frac{\pi_\theta(a_t \mid s_t)}{\pi_{\theta_{\text{old}}}(a_t \mid s_t)} A_t \right]$$

$$\text{subject to} \quad \mathbb{E}_t[\text{KL}[\pi_{\theta_{\text{old}}}(\cdot|s_t), \pi_{old}(\cdot|s_t)]] \leq \delta$$

In this optimization problem, θ_{old} represents the policy parameters before the update, and KL refers to the Kullback-Leibler divergence [30], which measures the difference between two probability distributions.

This objective function can be written as [29]:

$$L^{CPI}(\theta) = \mathbb{E}_t[r_t(\theta)A_t] \tag{2.6}$$

where

$$r_t(\theta) = \frac{\pi_\theta(a_t \mid s_t)}{\pi_{\theta_{\text{old}}}(a_t \mid s_t)}$$

is the probability ratio between the new and old policies, so $r(\theta_{old}) = 1$.

Without a constraint on L^{CLI}, maximizing it would lead to unreasonably large updates to the policy. Therefore, a penalty is added to changes that move $r_t(\theta)$ away from 1:

$$L^{CPIP}(\theta) = \mathbb{E}[\min(r_t(\theta)\hat{A}_t,\ \text{clip}(r_t(\theta), 1-\epsilon, 1+\epsilon)\hat{A}_t)] \tag{2.7}$$

where ϵ is a tunable hyperparameter, which is set to 0.2. The first term inside the min is L^{CLI}. The second term inside min clips the probability ratio and incentivizes the objective function to be within $[1-\epsilon, 1+\epsilon]$. The minimum of the two terms creates a lower bound on the unclipped objective.

An implementation of the PPO algorithm, Actor-Critic Style, is used in the Python library `stable_baselines3` [28]. At each iteration, N parallel actors collect data in the environment using the old policy for T timesteps [29]. Afterward, the loss of a "surrogate" objective function, for instance $L^{CPIP}(\theta)$, on these TN timesteps of data is optimized using the Adam optimizer [31] for K epochs. In Table 2.4, a subset of the parameters of the PPO agent used in `stable_baselines3` [28] are described.

Table 2.4: Subset of parameters for PPO in `stable_baselines3` [28].

Parameter	**Default Value**	**Allowed Range**
Policy (π)	N/A	N/A
Learning rate	0.0003	$[0, 1]$
Number of roll-out steps (T)	2,048	$[1, \infty)$
Batch Size	64	$[1, \infty)$
Epochs (K)	10	$[1, \infty)$
Gamma (γ)	0.99	$[0, 1]$
Clip range (ϵ)	0.2	$[0, 1]$
Number of parallel actors (N)	1	$[1, \infty)$

2.4 Data Preprocessing Tools

This section outlines the tools used to handle imbalanced data and data discretization.

2.4.1 SMOTE

Synthetic Minority Over-sampling TEchnique (SMOTE) and is a method to balance an imbalanced dataset [32]. An imbalanced dataset is one with unequal representation of each class. In Figure 2.8, an example of an imbalanced dataset is illustrated, where it is evident that Class D and G have much more data than the other classes. SMOTE works by oversampling the minority classes by creating "synthetic" datapoints that mimic the existing class.

In Figure 2.7, the process of creating the synthetic points (gray squares) is illustrated graphically. The process begins by selecting a starting sample from the minority class (orange squares). In Figure 2.7, the starting sample is the orange square with the black border labeled "Original sample". Next, k (default $k = 5$) randomly selected nearest neighbors to the starting sample are chosen. n of these k are then selected to generate the new samples, where n depends on how much the minority class needs to be oversampled. In Figure 2.7, $n = 3$.

A new sample is generated by calculating the Euclidean distance between the original sample and its neighbor. This distance is then multiplied by a random number between 0 and 1 and added to the original sample. This creates a synthetic sample that is close to the other class samples in feature space. In the end, this results in an equal number of data points for each class, matching the size of the majority class, as shown in Figure 2.9.

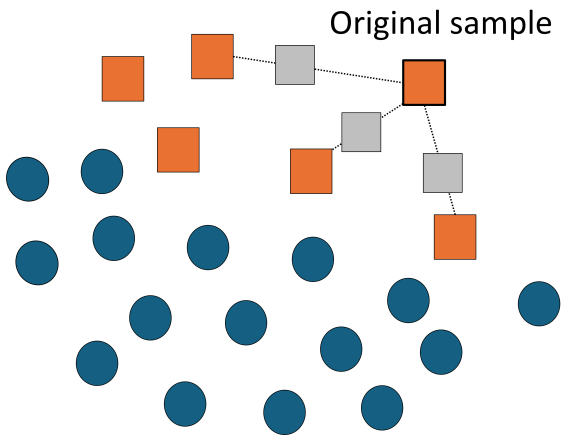


Figure 2.7: The process of creating new synthetic samples with SMOTE. The minority class that is oversampled is the orange squares. The chosen sample is the top right square with the black border. In this example, $n = 3$ out of the $k = 5$ nearest neighbors are chosen, generating three new synthetic gray squares.

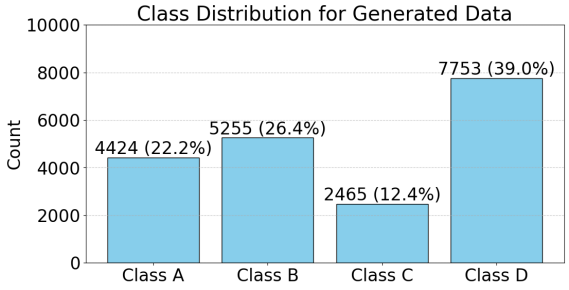


Figure 2.8: Imbalanced dataset

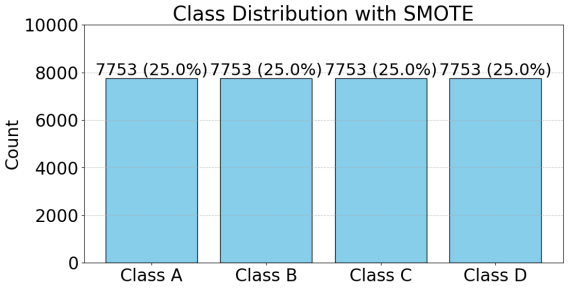


Figure 2.9: SMOTE balanced dataset

2.4.2 Data Binning

Data binning is a process of converting a continuous range to a series of discrete values [33]. This is done by dividing the range into different "bins", where each value is assigned a bin value. For instance, if the range is the numbers between 1-100, the bins could have the ranges 1-10, 11-20, . . . , 91-100, see Table 2.5. This means that the values of 1 and 9 would be placed in the same bin. One reason for binning the data is to reduce noise, as it protects the data from outliers and smooths out fluctuations.

There are different methods of binning numbers, and two ways are by dividing the range into equally spaced intervals (see Table 2.5) or by dividing the bins into equally large sizes (see Table 2.6). The advantage of binning the data as in Table 2.6 could be when the data distribution is skewed. If the data is binned as in Table 2.5, the number of points within each bin could be unequal [34], making the model unable to train on some of the bins. In contrast, binning the values as in Table 2.6 would naturally create equal-sized intervals, giving the model the same amount of training data for each bin. However, in some cases, it might be beneficial to have a fixed

interval size, as this attribute is important. The choice of binning method ultimately depends on the use case.

Table 2.5: Binning values into equally spaced bins.

Bin	Range	Number of data points
0	$[1, 10]$	12
1	$[11, 20]$	4
⋮	⋮	⋮
9	$[91, 100]$	3

Table 2.6: Binning values into equally large bins. Note: 100 data points are binned.

Bin	Range	Number of data points
0	$[1, 32]$	10
1	$[33, 45]$	10
⋮	⋮	⋮
9	$[84, 100]$	10

2.4.3 Autoencoder

An autoencoder is an ANN-based model that consists of an encoder and a decoder and is trained to map an input x to itself [16]. The autoencoder works by training the encoder to learn a mapping function f to a new latent space such that:

$$h = f(x)$$

where x is in the original feature space and h is in a new latent space, and by training a decoder to map from the latent space back to the original space such that:

$$r = g(h)$$

where r is the reconstructed sample. An autoencoder is trained similarly to any other ANN, with the difference being the utilization of a loss function that encourages $x = r$, such as mean squared error, i.e., minimize the following [16]:

$$\frac{1}{n}\sum_{i=1}^{n}(x_i - g(f(x_i)))^2$$

The usefulness of an autoencoder is not strictly defined by how closely it may match the original data. An autoencoder that can perfectly recreate the original data is not necessarily useful, since then the original data may be used instead [16]. In order for an autoencoder to be useful, it needs to accomplish some other benefit or extract useful information. One such useful benefit might be dimensionality reduction. By training the encoder to encode into a smaller latent space than the

original space, a reduced dimensionality representation may be learned. This is called an undercomplete autoencoder [16] and can be defined as:

$$x \in \mathbb{R}^n, h \in \mathbb{R}^m, n > m, h = f(x)$$

An undercomplete autoencoder can be used to compress data into a fixed size while preserving as much data as possible.

3

Problem Formulation

This chapter provides a comprehensive description of the problem, the dataset, and the proposed solution.

3.1 Mathematical Description

In mathematical terms, the problem can be written as follows:

- $P \in \mathbb{R}^d$: A vector of d measured coefficients that together with the MF tire model describes the behavior of an arbitrary tire.
- x: The inputs to the MF tire model, including normal load, slip angle, camber angle and longitudinal slip (see Table 2.1).
- y: The outputs of the MF tire model, including forces and moments in all three axes (see Table 2.1). For an in-depth description of the model, see Appendix A.
- MF: The MF tire model, mathematically treated as a non-convex differentiable function.

This yields the following equation:

$$y = MF(x; P) \tag{3.1}$$

The problem that this thesis is examining is that each P is not necessarily a correct coefficient vector for any given tire, due to variations of the measuring techniques, which introduce both noise and bias. Therefore, the following variables are introduced:

- y^*: The real-world behavior of a tire for a given x.
- $P^* \in \mathbb{R}^d$: which solves $y^* = MF(x; P^*), \forall x$.

Note that multiple P^* may exist, but they all have the same y^*. y^* is not known but is rather the theoretical true output (expected value), meaning that in theory, this is how the tire would behave in a real-world scenario.

3.2 Dataset

In order to mathematically describe the dataset consisting of .tir files, the following notation is introduced:

- S: The set of all data, i.e., the set containing all .tir file coefficients P for all tires.
- S_i: A subset of data where all data have the same origin, i.e., belong to the same category. A category is defined as all data entries measured by the same actor. Note, $S_i \subset S$.
- S_U: The set of unbiased data, defined as, $P \in S_U \iff P = P^*$. Note, $S_U \subset S$.

Note that all $P \in S_i$ have the same origin and are therefore assumed to come from the same distribution. However, the distributions (noise and bias) are not known for any S_i except S_U. Any given P belongs to exactly one S_i or S_U.

P^* is known for all tires in S_U, however, it is also only known for these tires. There exists the possibility to collect P^* for a very limited number of tires, likely only one or two.

Note that there is nothing inherently special about the set S_U. For the purpose of evaluating the system, any S_i could be used instead as the set of "correct" data.

It is also important to note that all P contain two parts, one part metadata about the physical attributes of the tires and one part coefficients. The differences between the two are that the metadata is known to be correct for all tires and contains information such as radius, weight, width, etc. The coefficients are influenced by some bias and noise, which depend on which S_i that P belongs to.

The problem can be approached in a multitude of ways. The most straightforward method would probably be a supervised model that takes P as input and outputs P^*. However, this would require significant amounts of data. Therefore, we propose another method, based on RL and a classifier as a proxy for P^*. This proposed method is described below.

3.3 Classifier as a Proxy

Since the amount of data available in S_U is a small subset of the data, a supervised approach is unlikely to yield satisfactory results. Therefore, we will opt to use a proxy for P^*, such that if we optimize towards the proxy, then we likely also optimize towards P^*. One such proxy that could plausibly work is to train a classifier on P, in order to predict which category, S_i, it belongs to.

- CLF: The classifier that is able to predict the category of any arbitrary P.

The classifier would work as follows:

$$P \in S_i \iff CLF(P) = S_i \tag{3.2}$$

Further,

$$CLF(P) = S_U \iff P = P^* \tag{3.3}$$

If the classifier can accurately predict the category, then both equations hold true. In particular, Eq. 3.3 allows the use of the classifier as a proxy to find P^*. If some P can be found such that $CLF(P) = S_U$, then $P = P^*$. Therefore, if it is possible to create an accurate classifier, according to evaluation scores such as F1 or accuracy, etc., we should be able to use this instead of the hard-to-come-by P^*.

In practice, the classifier would output the probability that $P \in S_U$. If this probability is above some arbitrary threshold, then we assume $P \approx P^*$. Another benefit of this approach is that it confirms or rejects the assumption that each category contains a measurable bias compared to each other and belongs to the same distribution.

3.4 Formulation of Reinforcement Learning Problem

This part of the proposed solution could be approached using different techniques other than RL. In essence, an optimization problem is formulated as a reward function and from that tries to learn a policy for translating P into P^*. Other techniques could be used to solve this optimization problem for each tire. However, using a classifier in the training of an RL algorithm has been shown to provide promising results in problems in other domains [35].

By using the classifier described previously as a proxy for finding P^*, it is possible to formulate an RL problem to optimize for[1]. The state space of the problem is defined by all combinations of values that each coefficient can take. That is, for each coefficient in P, there exists a maximum and a minimum value, and the coefficient can take any value on this continuous interval. The action space would be defined as a range of values per coefficient that can be added to the current space. The transition probabilities are deterministic, which means that given a new state s', an action a, and a state s, $s' = s + a$.

The reward function can be formulated as follows:

$$R(s, a) = Pr(CLF(P'(s')) = S_U) \tag{3.4}$$

where CLF is the classifier, P' denotes the current state of the coefficient vector, and S_U is the set of unbiased data. The state, s, of the RL algorithm is naturally the current $P'(s)$, and the reward for moving to a new state, $P'(s')$, is determined by the reward function.

[1]This step assumes the existence of an accurate classifier. If this assumption proves to be false, then it is likely that the data do not contain any detectable bias.

Depending on the quality of the outputs, it might be necessary to introduce penalties for solutions that are physically impossible, as well as for behaviors where the output is always the same. Further, if the existing datasets do not contain enough data in S_U, some other S_i might be used instead for evaluation purposes. The quality of the solution will be evaluated by hold-out data where P^* is known. By comparing a translated P' for a given tire with the corresponding P^*, the similarity of the two would be used as an evaluation metric.

4

Methods

This chapter explains the methodology used to develop the AI model. It includes details on the available data and how it was preprocessed, four different ways of representing the data, as well as which classifiers were tested and how the RL model framework was set up.

4.1 Dataset and Preprocessing

The data used to train the model was obtained from existing .tir files at the company. In total, 3,648 labeled .tir files were available, each belonging to a unique category, i.e., had the same origin. However, to ensure that each category contained a sufficient amount of data, a threshold of at least 20 .tir files per category was used. There were also multiple files with identical values for all coefficients. After removing these .tir files, as well as files that had features with no values, the dataset was reduced to approximately 56% (2,061) of the original .tir files. In total, six unique categories were found.

4.1.1 Normalization and Binning

Some of the coefficients of the MF tire model depend on the nominal load, F_{z0}, used to measure the tire. To improve comparability between tire models, these coefficients were normalized to the same nominal load (6000 N). A detailed description of this process is presented in Appendix B.

After normalization, each feature (input of the classifier) of the dataset underwent binning, which assigned each value a specific bin number. To bin the dataset, each feature was first scaled to be within the range of 0 and 1. Then, the values were distributed into equally spaced bins (see Table 2.5). This method of binning was preferred because it allowed the RL environments to use equal step sizes.

4.1.2 Train and Test Dataset

The train and test sets were split using stratified sampling with a ratio of 80/20. Stratified sampling [36] ensures that the class distribution is the same in both the train and test sets. In addition, the reason for the given split ratio is that there is not a large amount of data available. Therefore, it was deemed appropriate to allocate a larger proportion of the dataset for testing the classifier.

4.1.3 Balancing of the Dataset

In Figure 4.1, the class distribution of the data is visualized. As can be seen, the initial dataset is imbalanced since Category D and Category E represent around 70% of all data. This imbalance can be an issue, since the classifiers will be able to train more on the majority classes and less on the minority classes, potentially making them less successful in classifying these classes. To prevent this, there was an option to apply the balancing method SMOTE to the training dataset, adding synthetic samples to the minority classes. This removes the class imbalance on the training set, making each class the same size. In Figure 4.2, the new training set with SMOTE applied is visualized.

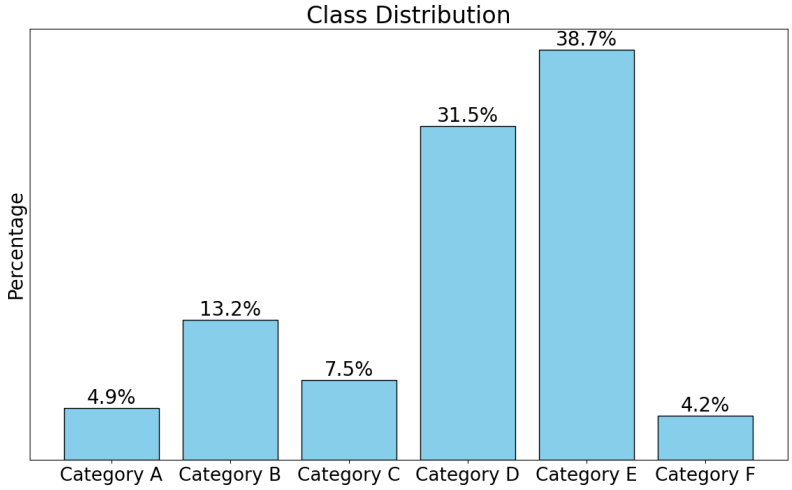


Figure 4.1: Class distribution of the dataset. The dataset is imbalanced as Category D and Category E represents around 70% of the data.

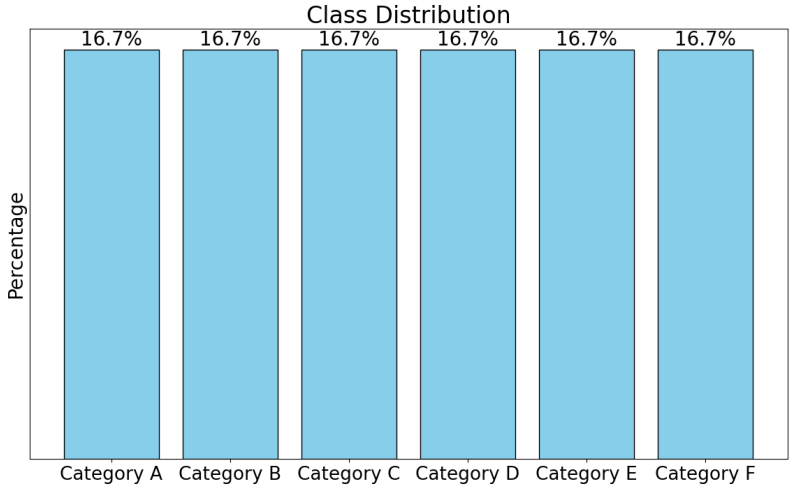


Figure 4.2: Class distribution of the training dataset after applying SMOTE.

4.2 Data Representations

In this section, four different ways of representing the available data are described. The first way was to simply represent the data with the MF coefficients. These can be obtained directly from the .tir files. Another way was to use the key attributes of the force/moment curve generated by the different .tir files. A third way was to use an autoencoder to create a latent space representation of the force curve. The final way was to express the MF as a power series and use the derivatives of this function as input.

The reason for the different data representations was to expose the model to different ways of interpreting the data and observe how the results differ. The state representations can then be converted to force curves that are comparable across representations.

4.2.1 Magic Formula Coefficients

This way of representing the data consisted of the coefficients that determine the static behavior of the vehicle, namely the coefficients p and q in the MF-Tire 5.2 version (see Appendix A). The total number of p and q coefficients in this version is 58 [12].

The coefficients influence different outputs of the MF. As such, the data was divided into four datasets, one for each MF output, and Table 4.1 describes the different datasets. Each dataset contains different numbers of coefficients depending on the static forces/moments they generate. In this thesis, the chosen dataset for this data representation was "Lateral Force" [1] with all coefficients that do not depend on the camber angle (γ), see Appendix A. This constitutes 11 of the 18 lateral coefficients.

Table 4.1: The datasets and corresponding parameters.

Type of Dataset	**Parameters**	**No. of MF Coefficients**
Full	p and q	58
Longitudinal Force	p used to calculate F_x	15
Lateral Force	p used in calculate F_y	18
Aligning Torque	q used in calculate M_z	25

4.2.2 Key Attributes of Force Curve

This dataset contained information on three key attributes of the MF lateral force curve for normal loads (F_z) of 4000 N, 6000 N, and 8000 N. These key attributes were the following:

- Cornering stiffness (derivative at the origin)

[1] Each output of the MF tire model is computed similarly, making the methodology agnostic to the specific output used. The handling and "feel" of a car depend mainly on the lateral load, so this thesis focuses solely on lateral loads for scope limitation reasons.

- Peak force
- Slip angle at peak force

These attributes were selected based on their importance for the behavior of a tire. In Figure 4.3, the MF lateral force curve is plotted, with cornering stiffnesses indicated by a black circle, peak forces highlighted by red circles, and slip angles at peak force marked by a purple circle. In total, each .tir file got three values (one for each F_z) for each attribute, totaling nine features.

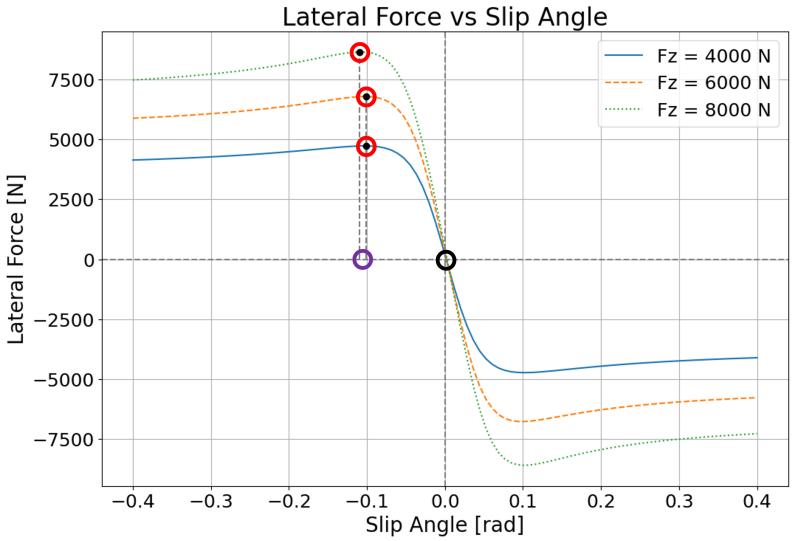


Figure 4.3: Lateral force curve graph of an arbitrary .tir file with normal load of 4000 N, 6000 N, and 8000 N. Cornering stiffnesses, peak forces, and slip angles at peak force are indicated by black, red, and purple circles, respectively.

4.2.3 Latent Space of Force Curve

In this dataset, the MF lateral force curves for three different normal loads (F_z) of each .tir file were converted into n latent features using an autoencoder. 100 evenly spaced force values for slip angles between [-0.25, 0.25] were selected for normal loads of 4000 N, 6000 N, and 8000 N, resulting in 300 values in total. These values were used as input for the autoencoder, which outputted n latent features for each .tir file. The value of n could be varied.

The autoencoder consisted of two ANNs: one encoder and one decoder. Both ANNs had two hidden layers with 128 nodes each and used the ReLU activation function between their hidden layers. The loss function was MSELoss [37], and they used the Adam [31] optimizer.

4.2.4 Magic Formula as a Power Series

This data representation used the derivatives of a power series that is equal to the MF equation described in Eq. 2.1. A power series is an infinite series that can be

used to approximate functions. As an example, the sine function can be expressed as follows:

$$\sin x = x - \frac{x^3}{3!} + \frac{x^5}{5!} + \mathcal{O}(7)$$

where $\mathcal{O}(7)$ denotes all terms with an exponent of 7 or higher. The MF equation can be expressed as the following power series:

$$CBx - \frac{C(C^2 + 2E + 2)(Bx)^3}{6} + \frac{C(C^4 + C^2(20E + 20) + 64E + 24)(Bx)^5}{120} + \mathcal{O}(7)$$

The details of converting Eq. 2.1 into a power series can be found in Appendix C.

From this power series, the first, third, and fifth derivatives, k_1, k_2, k_3, are obtained, which contain physical information about the tire:

- $k_1 = BC$
- $k_2 = -B^3C(C^2 + 2E + 2)$
- $k_3 = B^5C(C^4 + C^2(20E + 20) + 64E + 24)$

The variables B, C, and E come from Eq. 2.1. k_1, k_2 and k_3 were then calculated using the MF coefficients for normal loads of 4000 N, 6000 N, and 8000 N. Together with variable D in Eq. 2.1, this dataset contained 4 variables (k_1, k_2, k_3, D) times 3 normal loads, resulting in 12 features for each .tir file.

4.3 Overview of Reinforcement Learning Framework

In Figure 4.4, a schematic overview of the final model is presented. It consists of an ML classifier and a translator in the form of an RL model. Initially, various classifiers were developed and evaluated to find the best-performing classifier, based on accuracy and weighted average F1 score, in the Magic Formula Coefficients data representation.

The notion behind only evaluating the classifiers in this data representation was that it contained the most comprehensive information about the tire models. Since the MF coefficients directly define the MF itself, this representation inherently captures its characteristics the most. Furthermore, the other data representations are derived from the Magic Formula Coefficients representation. As such, performance in this representation was expected to be indicative of the performance in the others, as they essentially capture the same information relevant for classification.

Having obtained the optimal classifier, the RL agent/policy was trained using this classifier as a component of its reward function. The reward function also contained increasing penalties for states further away from the origin, see Section 4.5.4. Each data representation was modeled as its own environment with a varying number of bins, i.e., a different number of possible discrete values. Different RL algorithms and reward functions were compared.

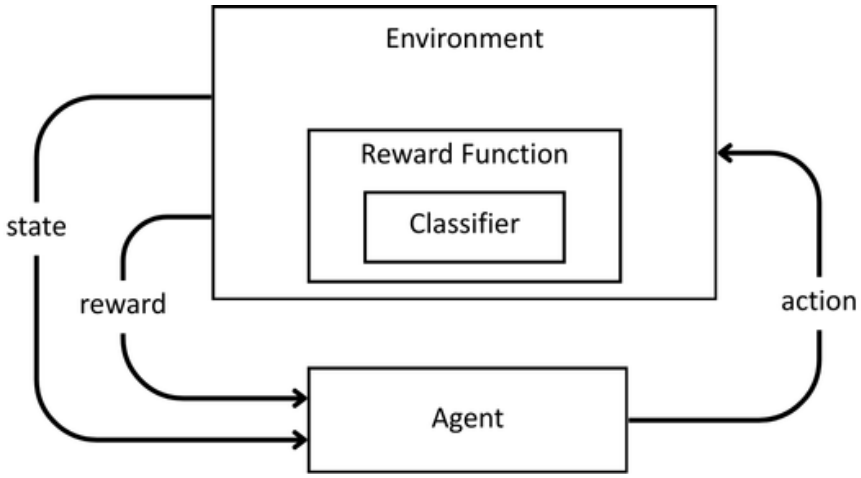


Figure 4.4: Overview of the RL framework.

4.4 Selection of Classifier

Three different types of classifiers were tested, and these were chosen because they use different methods to make their classification. The types of classifiers that were tested consisted of: ANN, GNB, and RF.

Before evaluating the classifiers on the test set, the hyperparameters of each classifier were determined by comparing the accuracy and F1 score of the different combinations. Afterward, the best-performing combination of hyperparameters for each classifier was trained on a dataset with and without SMOTE. SMOTE was introduced because the dataset was imbalanced (see Figure 4.1). As such, it was interesting to see whether the performance could increase even further for the tuned classifiers. In Figure 4.5, the process of selecting the classifier is illustrated.

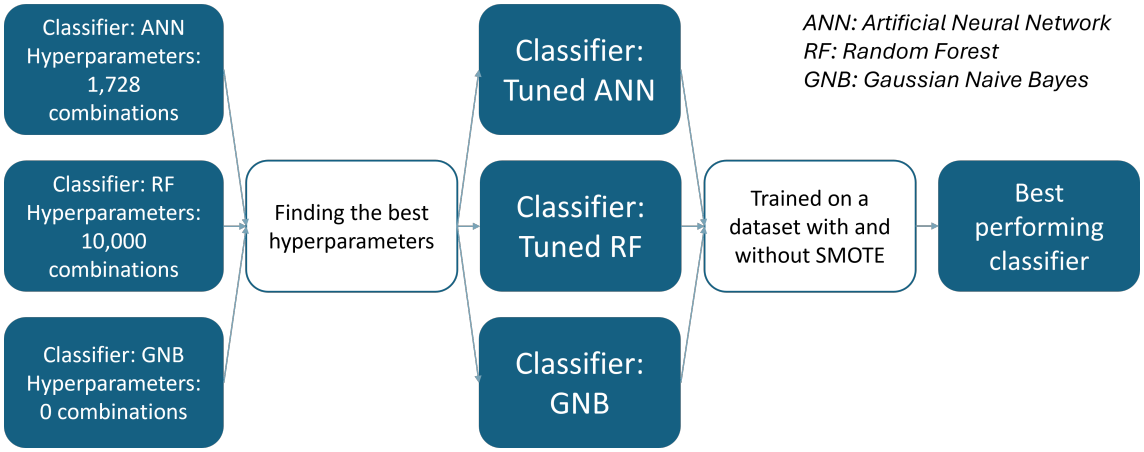


Figure 4.5: The workflow of choosing the best-performing classifier to use as part of the RL framework. At the start, each of the classifiers is tuned to their best set of hyperparameters. Next, the tuned classifiers are trained on a datasets with and without SMOTE, and then evaluated on a test set. In the end, the best-performing classifier is obtained.

The ANN consisted of one input layer, three hidden layers, and one output layer. The input layer had a varying number of input nodes depending on the number of inputs used. The hidden layers had 128, 128, and 64 nodes respectively, while the output had 6 nodes (the number of categories, see Figure 4.1). Between each hidden layer, the ReLU activation function was used, and in the last layer, the Softmax function was applied. The GNB and RF classifiers used were implementations from the Python library `scikit-learn` [9].

4.4.1 Tuning of Classifier Hyperparameters

For the ANN, the tested hyperparameters are presented in Table 4.2. A thorough analysis for tuning the number of layers and nodes of the ANN was not performed due to time constraints. However, experiments were performed early on using a larger network, but this did not show a notable increase in performance.

In total, the number of combinations of the ANN's parameters was 1728 ($4 \cdot 4 \cdot 4 \cdot 3 \cdot 3 \cdot 3 = 1728$), which is a large number. Each combination was evaluated with stratified k-fold cross-validation [38] with $k = 4$ folds, which preserves the class distribution within each fold. Before splitting the data into the different folds, the original dataset was shuffled.

Since the number of combinations was large and each combination was evaluated with cross-validation, it was not feasible to perform a full grid search [39] to find the best-performing configuration. Instead, a random grid search [39] was performed first to obtain an indicator of good parameter values. These were selected by observing the most frequent parameter values among the top-performing configurations. Using these selected parameter values, a full grid search was then performed to determine the best configuration in terms of accuracy and F1 score.

Table 4.2: The evaluated hyperparameters of the ANN.

Hyperparameter	**Values**
Learning Rate	$\{0.0001, 0.001, 0.01, 0.1\}$
Weight Decay	$\{0.001, 0.01, 0.1, 0\}$
Epochs	$\{200, 250, 300, 350\}$
Batch Size	$\{32, 64, 128\}$
Optimizer	{AdamW [21], SGD [40], RMSprop [40]}
Loss Function	{CrossEntropyLoss [41], MSELoss [37], NLLLoss [42]}

For the RF classifier, the hyperparameters evaluated are presented in Table 4.3. Initially, the accuracy and F1 score were recorded based on the number of trees, without setting any limits on tree depth. After identifying the optimal number of trees, the tree depth was varied to find which gave the highest accuracy and F1 score.

Table 4.3: The evaluated hyperparameters of the RF.

Hyperparameter	Values
Number of Trees	$\{1, 2, \ldots, 99, 100\}$
Tree Depth	$\{1, 2, \ldots, 99, 100\}$

For the GNB classifier, there are no specific hyperparameters to tune [9]. An uninformative prior was used, and the features are assumed to follow a Gaussian distribution.

As a final step, the tuned classifiers were evaluated with and without SMOTE applied to the training set. To evaluate the classifiers, 10 separate instances of each classifier were individually trained and tested on the test dataset, resulting in an average accuracy and F1 score. The best-performing classifier based on the test runs was chosen as the classifier to be used in the RL model.

4.5 Reinforcement Learning Model Framework

The RL environment was created using the open-source Python library `gymnasium` [43]. The agent was trained using Advantage Actor Critic (A2C) and Proximal Policy Optimization (PPO) algorithms from the Python library `stable_baseline3` [28]. All parameters of both algorithms were set to their default value except for the number of roll-out steps (T) and the number of environments (N). To increase training speed and balance training stability, these parameters were set to 320 and 4, respectively (see Table 2.3 and Table 2.4).

During the development of the RL model, the different data representations described in Section 4.2 were modeled as different state spaces, resulting in four types of environments that could be used. However, in all environment types, the action space remained the same, see Section 4.5.3. Moreover, the agent was trained in each environment using different reward functions. A summary of the RL model configurations can be seen in Table 4.4.

Table 4.4: The parameters of the RL model.

Parameter	Values
RL Algorithm	{PPO, A2C}
Reward Function	{Maximize Reward, Reach Target Fast}
Data Representation	{See Section 4.2}

4.5.1 Optimal Number of Bins and Latent Features

Before evaluating the agent in each environment, the optimal classifier's performance was assessed when the number of bins varied for each data representation. In the Latent Space of Force Curve data representation, the number of latent features was varied first before varying the number of bins. The number of bins and latent

features naturally determined the number of parameters in the state space for each environment.

4.5.2 State Space

In this section, the state spaces of each environment are described. The state space was discrete in all environments, and each dimension had a maximum and minimum value beyond which the agent should not move. The initial state was a randomly chosen .tir file.

4.5.2.1 MF Coefficients

- Description: The state space was defined as the allowed bin ranges for all MF coefficients in a .tir file.
- Dimension: n-dimensional vector, where n depended on the number of coefficients used.
- Range: $[0, k]$. k was the maximum bin value for each coefficient.
- Type: Discrete.

4.5.2.2 Key Attributes

- Description: The state space was defined as the allowed bin ranges for the three key attributes for the three normal loads.
- Dimension: 9-dimensional vector, where each value corresponds to one of the key attributes for one normal load.
- Range: $[0, k]$. k was the maximum bin value of each attribute.
- Type: Discrete.

The output of the RL model in this state space was the bin values of each attribute. To generate a continuous curve based on these values, a fitting method was developed to obtain approximate curves. This method is described in Appendix D.

4.5.2.3 Latent Space

- Description: The state space was defined as the allowed bin ranges for the latent features of the autoencoder.
- Dimension: n-dimensional vector, where n depended on the number of latent features used.
- Range: $[0, k]$. k was the maximum bin value of each latent feature.
- Type: Discrete.

4.5.2.4 Power Series

- Description: The state space was defined as the allowed bin ranges for each derivative and D variable for the three normal loads.
- Dimension: 12-dimensional vector, where each value corresponds to one of the derivatives or D variable for one normal load.
- Range: $[0, k]$. k was the maximum bin value of each feature.
- Type: Discrete.

4.5.3 Action Space

The action space was the same in all environments. The actions were to either increment or decrement one feature, i.e., bin value, of the state space by 1. It was an n-dimensional vector, where n was the same size as the state space vector. Each entry was either 0 or 1, where 0 indicated a decrease and 1 indicated an increase. Its shape was therefore $(n, 2)$.

4.5.4 Reward Functions

This section describes the different reward functions used, one aimed at reaching the best possible target and one aimed at finding a target as fast as possible . Whenever an agent moves to a new state, the classifier is fed the agent's state as input and outputs its probability distribution among the categories. To prevent the agent from moving outside its boundaries, a penalty of -1 was used in all reward functions whenever the agent made such a move.

For the Power Series environment, there exists the possibility of obtaining imaginary solutions to the power series for some combination of k_1, k_2, and k_3 that lead to negative discriminants in the variable C in Appendix C. Whenever the agent ends up in these states, the reward is set to -0.01. The penalty is less for this compared to an out-of-bounds move since the state may lie in the path to a better state.

4.5.4.1 Maximize Reward

In this reward function, the agent seeks to maximize its total reward. The agent's valuation $V(s)$ of a state is calculated as the reward in Eq. 3.4:

$$V(s) = Pr(CLF(P'(s')) = S_U)$$

In other words, the current valuation (V) is simply the classifier's (CLF) probability of the state (s') belonging to the target category (S_U). To obtain the reward at each step, the difference between the agent's current valuation and the highest valuation it has seen with a small penalty is calculated:

$$R(s, a) = -0.01 + \max(0, \text{Current valuation} - \text{Max valuation})$$

The -0.01 term encourages the agent to increase its reward quickly, while the difference makes increasing the target probability more rewarding. If no improvements are made to the total reward in five steps, the episode is terminated[2].

4.5.4.2 Reach Target Fast

In this reward function, the agent is encouraged to reach the target state as quickly as possible. The agent reaches the target state whenever the probability that the state being the target is the highest among all categories. Whenever the agent moves, a small penalty of -0.01 is added to the reward to encourage moving to the target category as quickly as possible. The reward for reaching the target is 1. Formally, this reward function is defined by:

$$R(s,a) = \begin{cases} 1 & \text{for } CLF(s) = S_U, \\ -0.01 & \text{else.} \end{cases}$$

4.5.5 Evaluation Metric

The RL model's performance was assessed on a dataset of 20 .tir files from four different tires. In this dataset, four different tires have been measured by five different manufacturers (corresponding to five different categories), resulting in five .tir files of the same tire but from different categories, see Figure 4.6. Each of the tires was physically identical, but the .tir file from each category was different.

Figure 4.6: The four different tires that were measured, resulting in .tir files from different categories and tires.

The process for assessing the RL model's performance can be seen in Figure 4.7. Here, an arbitrary category (e.g., Category A) is set as the target category. The .tir files of this category are labeled as reference .tir files. Then, the other categories' .tir files are translated to the target category (Category A). This is repeated for the four tires, resulting in (4 translations/tire $\times$ 4 tires) 16 translations per category.

[2]The RL model tended to reach its best target in fewer steps, and by choosing to limit the number of steps at the end of each run, training could speed up substantially.

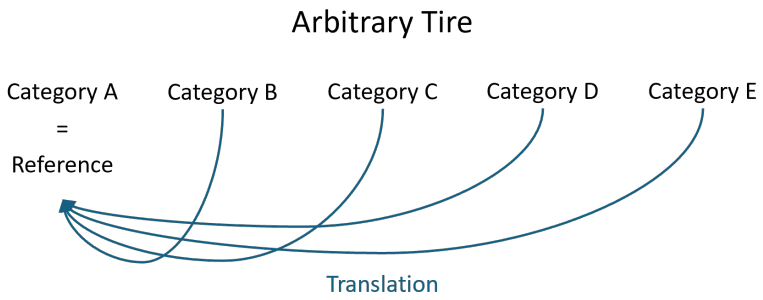


Figure 4.7: In this Figure, Category A has been set as the target category. Each of the other Categories' .tir files are translated to the target category. This is repeated for all four tires.

The performance was evaluated by calculating the mean squared error (MSE) between the curves of the translated .tir file and the target .tir file, Δ_T. MSE was chosen as the metric because it was deemed important to penalize large deviations from the target curve. Δ_T was then compared with the MSE of the original .tir before translation and the target .tir file, Δ_O:

$$\Delta_T = MSE(\text{Translation of Arbitrary Category; Target})$$
$$\Delta_O = MSE(\text{Arbitrary Category; Target})$$

Specifically, the differences in force/moment, cornering stiffness, and load sensitivity of peak friction curves were compared between the .tir files. If $\Delta_T < \Delta_O$, then the RL model could find a translation that was better aligned with the target than the original.

5

Results

In this chapter, the results of the thesis are presented. As described in Section 4.5.5, the evaluation data of the RL models are 20 .tir files from 5 manufacturers and 4 tires. The results will be presented in the order of the RQ they are associated with. Below, the RQs are repeated for convenience.

- RQ1: Can supervised learning be used to classify tire property files?
- RQ2: Can a deep RL agent learn a policy that optimizes the output probabilities of a classifier?
- RQ3: Can RL or other ML techniques be used to translate biased tire models into unbiased models?
- RQ4: What learnings about the data can be made from the RL policies?

5.1 Result of Classifiers

In this section, the optimal hyperparameters for each classifier are presented. Then, the results of the evaluation of the tuned classifiers with and without SMOTE are presented. At the end of the section, the best-performing classifier is evaluated on the different data representations. The section contains the results associated with RQ1: "Can supervised learning be used to classify tire property files?".

5.1.1 Optimal Hyperparameters of Classifier

This part provides insight into the hyperparameters selected for the ANN and RF classifiers.

5.1.1.1 Tuning of ANN

The evaluated hyperparameters can be seen in Table 4.2. Due to the many combinations of hyperparameters, a random grid search was performed first. The five best combinations in terms of cross-validation accuracy and F1 score using four folds are presented in Table 5.1. Given the results in Table 5.1, it was possible to narrow down the search space by only using parameter values that appeared to result in the best performance. The values selected for the full grid search are presented in Table 5.2.

Table 5.1: Top five performing combinations from the random grid search.

Configuration	**Accuracy**	**F1 score**
{'weight_decay': 0.01, 'optimizer': 'AdamW', 'loss_function': 'CrossEntropyLoss', 'learning_rate': 0.001, 'epochs': 250, 'batch_size': 32}	0.933	0.932
{'weight_decay': 0, 'optimizer': 'RMSprop', 'loss_function': 'NLLLoss', 'learning_rate': 0.001, 'epochs': 250, 'batch_size': 64}	0.925	0.925
{'weight_decay': 0.001, 'optimizer': 'RMSprop', 'loss_function': 'CrossEntropyLoss', 'learning_rate': 0.001, 'epochs': 350, 'batch_size': 64}	0.924	0.925
{'weight_decay': 0, 'optimizer': 'SGD', 'loss_function': 'CrossEntropyLoss', 'learning_rate': 0.001, 'epochs': 300, 'batch_size': 64}	0.923	0.922
{'weight_decay': 0, 'optimizer': 'AdamW', 'loss_function': 'MSELoss', 'learning_rate': 0.001, 'epochs': 300, 'batch_size': 64}	0.921	0.920

Table 5.2: Hyperparameter values selected for the full grid search.

Hyperparameter	**Values**
Learning rate	$\{0.001\}$
Weight decay	$\{0.001, 0.01, 0\}$
Epochs	$\{250, 300\}$
Batch size	$\{32, 64\}$
Optimizer	{AdamW, SGD, RMSprop}
Loss Function	{CrossEntropyLoss, MSELoss, NLLoss}

Table 5.3: The three top-performing configurations from the full grid search.

Configuration	**Accuracy**	**F1 score**
{'batch_size': 32, 'epochs': 300, 'learning_rate': 0.001, 'loss_function': 'CrossEntropyLoss', 'optimizer': 'AdamW', 'weight_decay': 0.01}	0.938	0.937
{'batch_size': 32, 'epochs': 300, 'learning_rate': 0.001, 'loss_function': 'NLLLoss', 'optimizer': 'AdamW', 'weight_decay': 0.001}	0.936	0.935
{'batch_size': 32, 'epochs': 300, 'learning_rate': 0.001, 'loss_function': 'MSELoss', 'optimizer': 'AdamW', 'weight_decay': 0}	0.935	0.935

For the optimizer and loss function, it was difficult to choose a subset of the evaluated parameters, since all appeared to lead to high accuracy and F1 score. As such, all combinations were used again. Using these values, a full grid search was performed and the result can be seen in Table 5.3. The best-performing configuration of hyperparameters is presented in Table 5.4 in column "Selected Value".

Table 5.4: The evaluated hyperparameters and the selected values for the final ANN.

Hyperparameter	**Evaluated Values**	**Selected Value**
Learning Rate	$\{0.0001, 0.001, 0.01, 0.1\}$	0.001
Weight Decay	$\{0.001, 0.01, 0.1, 0\}$	0.01
Epochs	$\{200, 250, 300, 350\}$	300
Batch Size	$\{32, 64, 128\}$	32
Optimizer	{AdamW, SGD, RMSprop}	AdamW
Loss Function	{CrossEntropyLoss, MSELoss, NLLoss}	CrossEntropyLoss

5.1.1.2 Tuning of RF

For the RF classifier, the accuracy and F1 score were obtained as a function of increasing trees with no restriction on the depth of the trees. The result can be seen in Figure 5.1. It is evident that increasing the number of trees results in higher accuracies. However, the increase seems to slow down after around 30 trees. Given that the computational time for the RF classifier increases with increasing trees, 30 trees were considered the optimal choice.

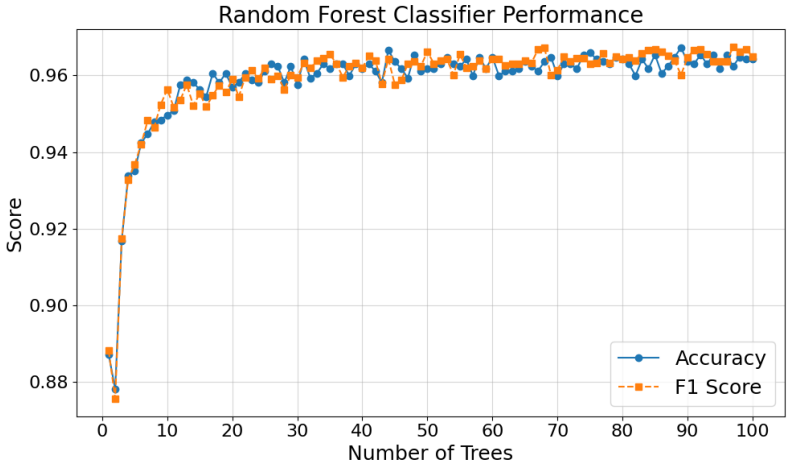


Figure 5.1: The accuracy and F1 score as a function of the number of trees.

Next, the accuracy and F1 score were plotted as a function of tree depth when 30 trees were used. In Figure 5.2, it can be noted that a minor increase is recorded for both metrics with a tree depth of 10 or more. As such, a tree depth of 10 was chosen. The selected values for the RF classifier are summarized in Table 5.5.

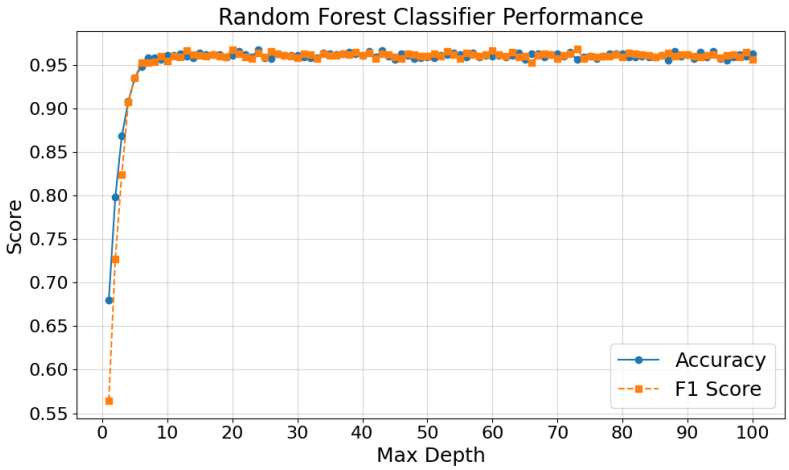


Figure 5.2: The accuracy and F1 score as a function of max depth with 30 trees.

Table 5.5: The evaluated hyperparameters and the selected values for the final RF.

Hyperparameter	**Evaluated Values**	**Selected Value**
Number of Trees	$\{1, 2, \ldots, 99, 100\}$	30
Tree Depth	$\{1, 2, \ldots, 99, 100\}$	10

5.1.2 Best-Performing Classifier

Next, the tuned ANN and RF classifiers, together with the GNB classifier, were evaluated when trained on a dataset with and without SMOTE applied. Each classifier was trained and evaluated ten times on the test data, yielding an average accuracy and F1 score. The results of the test runs are presented in Table 5.6.

Table 5.6: Accuracy and F1 score of the classifiers on the test dataset when trained on a dataset with and without SMOTE. The best-performing classifier is in **bold**.

Classifier	**SMOTE Applied**	**Test Accuracy**	**Test F1 score**
ANN	No	0.951	0.951
ANN	Yes	0.954	0.954
GNB	No	0.801	0.813
GNB	Yes	0.765	0.779
RF	No	0.964	0.965
RF	**Yes**	**0.970**	**0.971**

As can be seen in Table 5.6, the RF classifier achieves better accuracy and F1 score on the test sets compared to the other classifiers. Furthermore, adding SMOTE to the training set while training gives higher scores for both metrics, achieving an average accuracy of 97.0% and an F1 score of 97.1%. As such, this classifier and configuration, summarized in Table 5.6, was used in the RL model.

Table 5.7: Configuration of the best-performing classifier.

Parameter	Value
Classifier	Random Forest
Number of Trees	30
Tree Depth	10
SMOTE Applied	Yes

In Figure 5.3, the confusion matrix resulting from the test run of the best RF classifier is illustrated. The confusion matrix has been normalized by dividing each entry by the sum of all other entries. Furthermore, the precision, recall, and F1 score for each of the classes are described in Table 5.8.

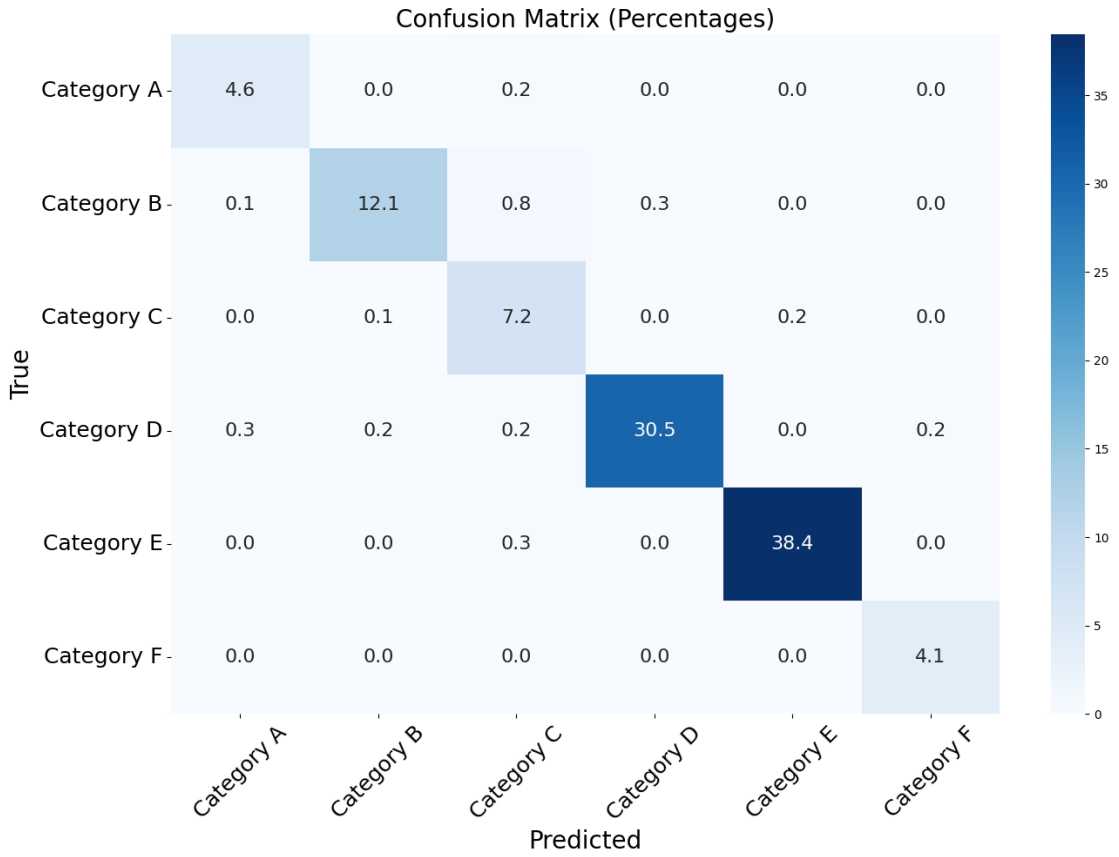


Figure 5.3: Confusion matrix from using RF classifier with SMOTE applied. In the optimal case, the entries on the diagonal should equal the class distribution in Table 4.1.

5.1.3 Result of Classifier on Different Data Representations

After obtaining the best-performing classifier (see Table 5.7), it was evaluated on each data representation described in Section 4.2. For the Latent Space of Force Curve data representation, the number of latent features used was first determined.

Table 5.8: Precision, recall, and F1 score of the RF classifier for each category from the test run.

Class	Precision	Recall	F1 score
Category A	0.955	0.965	0.960
Category B	0.973	0.905	0.938
Category C	0.832	0.974	0.897
Category D	0.991	0.970	0.980
Category E	0.992	0.991	0.991
Category F	0.934	1.000	0.966

This was done by comparing the classifier's performance when the number of latent features varied. The metrics used were accuracy and F1 score. Afterward, the same metrics were used to evaluate the classifier's performance when the number of bins varied in each data representation.

In Figure 5.4, the performance of the classifier is shown when the number of latent features increases. As can be seen, the performance peaks at around 8 latent features and remains the same after. Hence, 8 latent features were chosen.

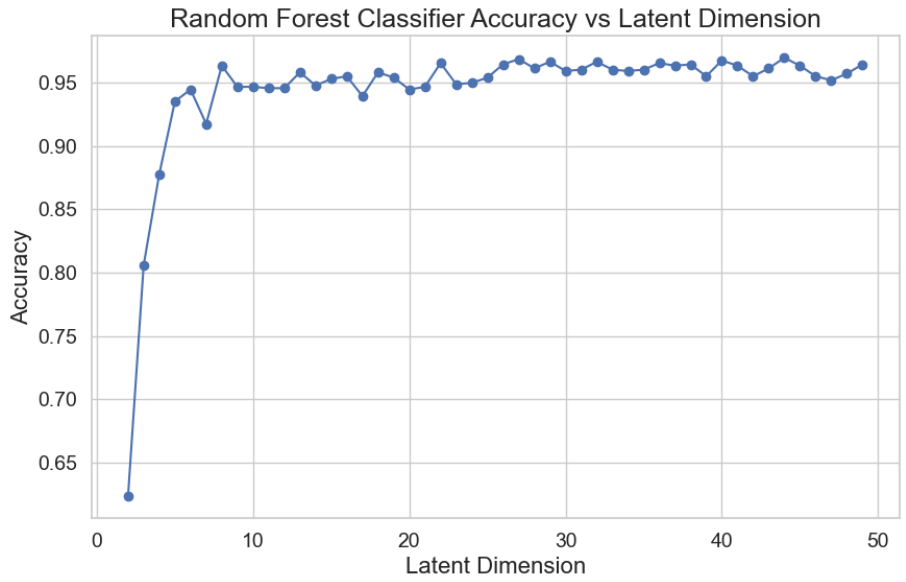


Figure 5.4: Accuracy as function of increasing number of latent features in the Latent Space of Force Curve data representation.

Next, the number of bins was varied. The performance of the classifier is illustrated in Figure 5.5, 5.6, and 5.7. As the figures show, the performance plateaus after around 10 bins in all data representations. As such, the number of bins was decided to be 10. The final accuracy and F1 score of the RF classifier in each data representation with 10 bins is presented in Table 5.9.

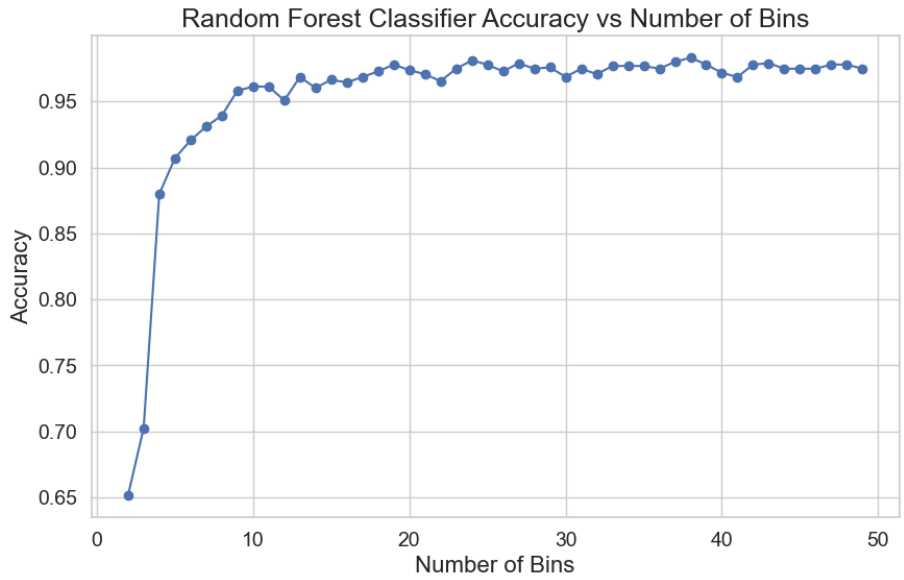


Figure 5.5: Accuracy as a function of increasing number of bins in the Magic Formula Coefficients data representation.

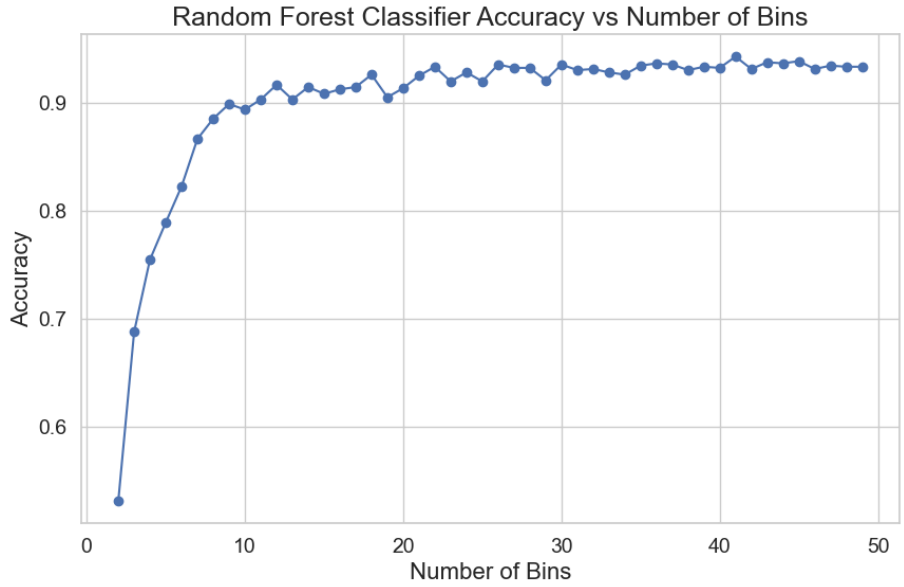


Figure 5.6: Accuracy as a function of increasing number of bins in the Key Attributes of Force Curve data representation.

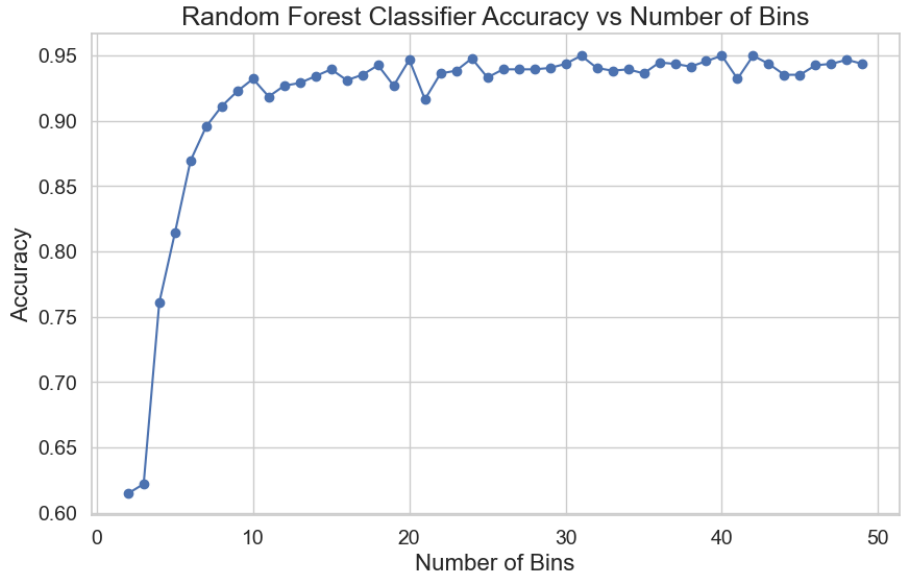


Figure 5.7: Accuracy as a function of increasing number of bins in the Latent Space of Force Curve data representation.

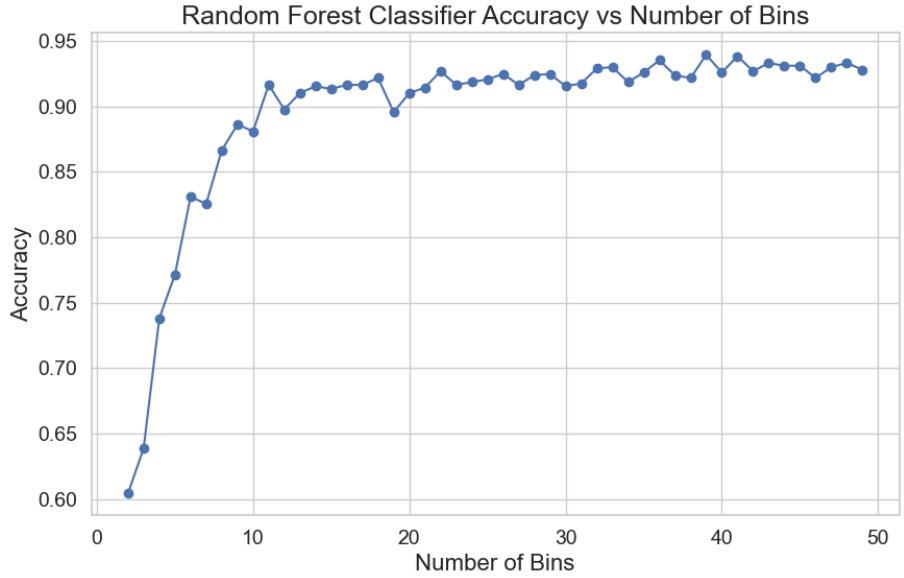


Figure 5.8: Accuracy as a function of increasing number of bins in the Magic Formula as a Power Series data representation.

Table 5.9: Accuracy (ACC) and F1 score (F1) for the best classifier on each data representation.

Data Representation	ACC	F1	Bins	Features
Magic Formula Coefficients	0.961	0.961	10	11
Key Attributes of Force Curve	0.894	0.893	10	9
Latent Space of Force Curve	0.942	0.942	10	8
Magic Formula as a Power Series	0.881	0.881	10	12

5.2 Training of Reinforcement Learning Models

In this section, results used to answer RQ2: "Can a deep RL agent learn a policy that optimizes the output probabilities of a classifier?" will be presented.

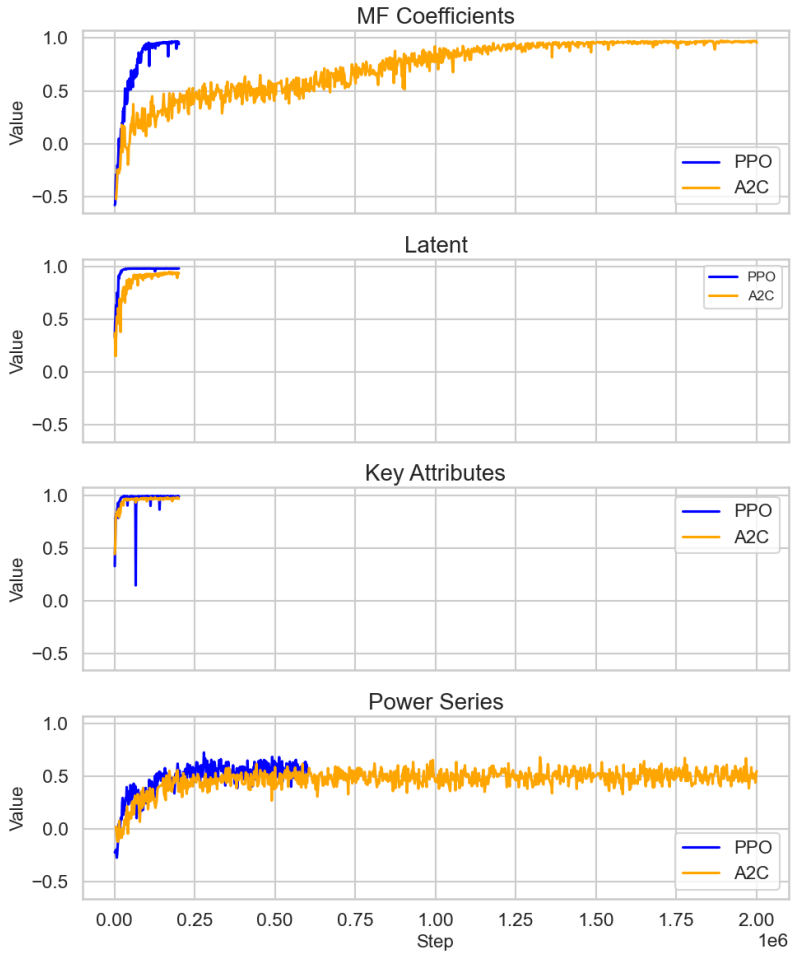


Figure 5.9: Illustration of episodic reward (y-axis) during training for all models trained on environments with 20 bins and reward function: Reach Target Fast.

5.3 Result of Reinforcement Learning Model

In this section, the results of the reinforcement learning model are presented, aimed at answering RQ3: "Can RL or other ML techniques be used to translate biased tire models into unbiased models?".

The best-performing classifier (see Table 5.7) was used as part of the reward functions. As the result shows in Section 5.1.3, 10 bins was a sufficiently large value for achieving good classification performance. However, as the bin values define the state space in each data representation's environment, it was deemed important to vary this to check whether the result from the RL model differed. The bin values used were 10, 20, and 40. Then, the RL agent was trained in each environment using two different reward functions, resulting in several different outputs.

5.3.1 Mean Squared Error for Varying Bin Number and Reward Function

In this section, the MSE results of the RL model are presented for varying numbers of bins and reward functions. The values in the tables are aggregated MSE for all translated curves to the target, and have been normalized by the MSE of the original data to the target. The best value in each column is marked in **bold**, while the second-best value is marked in **blue bold**.

5.3.1.1 Reward Function: Reach Target Fast

Table 5.10: Average MSE of translated models, normalized by original data MSE to target, with *10 bins*. Reward function: Reach Target Fast.

Model	Forces	Slope	Peak Friction Coeff.	Average
Original Data	**1.000**	**1.000**	**1.000**	**1.000**
MF Coefficients PPO	1.338	2.643	1.631	1.871
MF Coefficients A2C	1.218	2.643	1.482	1.781
Latent PPO	1.242	**1.977**	1.483	**1.567**
Latent A2C	**1.068**	3.231	**1.197**	1.832
Power Series PPO	86.649	24.909	50.250	53.936
Power Series A2C	71.663	8.671	32.135	37.490
Key Attributes PPO	2.047	6.863	1.424	3.445
Key Attributes A2C	2.094	11.417	1.424	4.978

Table 5.11: Average MSE of translated models, normalized by original data MSE to target, with *20 bins*. Reward function: Reach Target Fast.

Model	Forces	Slope	Peak Friction Coeff.	Average
Original Data	**1.000**	**1.000**	**1.000**	**1.000**
MF Coefficients PPO	3.978	5.432	7.407	5.606
MF Coefficients A2C	2.363	2.387	3.342	2.697
Latent PPO	**0.986**	**1.889**	1.340	**1.405**
Latent A2C	1.340	16.170	5.910	7.807
Power Series PPO	80.298	50.351	59.918	63.522
Power Series A2C	56.884	50.708	41.286	49.626
Key Attributes PPO	2.625	4.285	**0.942**	2.617
Key Attributes A2C	2.670	30.477	1.286	11.478

Table 5.12: Average MSE of translated models, normalized by original data MSE to target, with *40 bins*. Reward function: Reach Target Fast.

Model	Forces	Slope	Peak Friction Coeff.	Average
Original Data	**1.000**	**1.000**	**1.000**	**1.000**
MF Coefficients PPO	1.306	**1.481**	2.047	1.611
MF Coefficients A2C	**1.159**	1.489	**1.094**	**1.247**
Latent PPO	1.357	1.918	1.659	1.645
Latent A2C	2.879	6.749	4.108	4.579
Power Series PPO	96.394	19.403	47.801	54.533
Power Series A2C	223.263	365.097	196.405	261.588
Key Attributes PPO	2.243	3.668	1.234	2.382
Key Attributes A2C	2.464	2.983	1.234	2.227

5.3.1.2 Reward Function: Maximize Reward

Table 5.13: Average MSE of translated models, normalized by original data MSE to target, with *10 bins*. Reward function: Maximize Reward.

Model	Forces	Slope	Peak Friction Coeff.	Average
Original Data	**1.000**	**1.000**	**1.000**	**1.000**
MF Coefficients PPO	1.698	2.629	1.507	1.945
MF Coefficients A2C	1.698	2.629	1.507	1.945
Latent PPO	**1.228**	**1.564**	**1.086**	**1.293**
Latent A2C	**1.228**	**1.564**	**1.086**	**1.293**
Power Series PPO	157.216	280.081	178.592	205.296
Power Series A2C	157.216	280.081	178.592	205.296
Key Attributes PPO	2.124	2.326	1.424	1.958
Key Attributes A2C	2.124	2.326	1.424	1.958

Table 5.14: Average MSE of translated models, normalized by original data MSE to target, with *20 bins*. Reward function: Maximize Reward.

Model	Forces	Slope	Peak Friction Coeff.	Average
Original Data	**1.000**	**1.000**	**1.000**	**1.000**
MF Coefficients PPO	**1.251**	1.924	1.155	1.443
MF Coefficients A2C	**1.251**	1.924	1.155	1.443
Latent PPO	1.253	1.680	**1.140**	**1.358**
Latent A2C	1.253	1.680	**1.140**	**1.358**
Power Series PPO	108.561	190.141	142.967	147.223
Power Series A2C	108.561	190.141	142.967	147.223
Key Attributes PPO	2.744	**1.659**	1.286	1.896
Key Attributes A2C	2.744	**1.659**	1.286	1.896

Table 5.15: Average MSE of translated models, normalized by original data MSE to target, with *40 bins*. Reward function: Maximize Reward.

Model	Forces	Slope	Peak Friction Coeff.	Average
Original Data	**1.000**	**1.000**	**1.000**	**1.000**
MF Coefficients PPO	1.252	**1.463**	1.245	**1.320**
MF Coefficients A2C	1.252	**1.463**	1.245	**1.320**
Latent PPO	**1.234**	1.973	**1.210**	1.472
Latent A2C	**1.234**	1.973	**1.210**	1.472
Power Series PPO	79.139	99.090	112.545	96.925
Power Series A2C	79.139	99.090	112.545	96.925
Key Attributes PPO	2.497	1.483	1.234	1.738
Key Attributes A2C	2.497	1.483	1.234	1.738

5.3.2 Improvement Fraction for Varying Bin Number and Reward Function

This section illustrates the fraction of improved force curves based on Δ_T and Δ_O in Section 4.5.5 for the different algorithms:

$$\Delta_T = MSE(\text{Translation of Arbitrary Category; Target})$$
$$\Delta_O = MSE(\text{Arbitrary Category; Target})$$

This fraction was calculated by summing all instances where $\Delta_T < \Delta_O$ and dividing it by the total number of translated .tir files. The values in **bold** indicate the best performer in each metric for each table.

5.3.2.1 Reward Function: Reach Target Fast

Table 5.16: Fraction of improved curves of translated models with *10 bins*. Reward function: Reach Target Fast.

Model	Forces	Slope	Peak Friction Coeff.	Average
MF Coefficients PPO	**0.562**	**0.312**	0.375	**0.416**
MF Coefficients A2C	**0.562**	**0.312**	0.312	0.395
Latent PPO	0.375	0.125	**0.438**	0.313
Latent A2C	0.438	0.000	0.250	0.229
Power Series PPO	0.000	0.125	0.062	0.062
Power Series A2C	0.000	0.250	0.000	0.083
Key Attributes PPO	0.062	0.125	0.250	0.146
Key Attributes A2C	0.188	0.000	0.312	0.167

Table 5.17: Fraction of improved curves of translated models with *20 bins*. Reward function: Reach Target Fast.

Model	Forces	Slope	Peak Friction Coeff.	Average
MF Coefficients PPO	0.375	0.188	0.375	0.313
MF Coefficients A2C	0.500	0.312	0.312	0.375
Latent PPO	**0.750**	**0.438**	0.438	**0.542**
Latent A2C	0.688	0.062	**0.500**	0.417
Power Series PPO	0.125	0.000	0.000	0.042
Power Series A2C	0.062	0.062	0.125	0.083
Key Attributes PPO	0.312	0.125	**0.500**	0.312
Key Attributes A2C	0.188	0.000	0.375	0.188

Table 5.18: Fraction of improved curves of translated models with *40 bins*. Reward function: Reach Target Fast.

Model	Forces	Slope	Peak Friction Coeff.	Average
MF Coefficients PPO	0.250	0.250	0.062	0.187
MF Coefficients A2C	**0.438**	0.250	0.188	0.292
Latent PPO	0.375	**0.375**	0.250	**0.333**
Latent A2C	0.188	0.250	0.312	0.250
Power Series PPO	0.000	0.062	0.125	0.062
Power Series A2C	0.000	0.000	0.000	0.000
Key Attributes PPO	0.250	0.125	0.375	0.250
Key Attributes A2C	0.250	0.188	**0.562**	**0.333**

5.3.2.2 Reward Function: Maximize Reward

Table 5.19: Fraction of improved curves of translated models with *10 bins*. Reward function: Maximize Reward.

Model	Forces	Slope	Peak Friction Coeff.	Average
MF Coefficients PPO	**0.375**	**0.312**	0.312	0.333
MF Coefficients A2C	**0.375**	**0.312**	0.312	0.333
Latent PPO	**0.375**	0.188	**0.562**	**0.375**
Latent A2C	**0.375**	0.188	**0.562**	**0.375**
Power Series PPO	0.000	0.062	0.062	0.041
Power Series A2C	0.000	0.062	0.062	0.041
Key Attributes PPO	0.250	0.125	0.312	0.229
Key Attributes A2C	0.250	0.125	0.312	0.229

Table 5.20: Fraction of improved curves of translated models with *20 bins*. Reward function: Maximize Reward.

Model	Forces	Slope	Peak Friction Coeff.	Average
MF Coefficients PPO	**0.500**	0.250	**0.625**	0.458
MF Coefficients A2C	**0.500**	0.250	**0.625**	0.458
Latent PPO	0.438	**0.438**	**0.625**	**0.500**
Latent A2C	0.438	**0.438**	**0.625**	**0.500**
Power Series PPO	0.062	0.062	0.188	0.104
Power Series A2C	0.062	0.062	0.188	0.104
Key Attributes PPO	0.188	0.125	0.375	0.229
Key Attributes A2C	0.188	0.125	0.375	0.229

Table 5.21: Fraction of improved curves of translated models with *40 bins*. Reward function: Maximize Reward.

Model	Forces	Slope	Peak Friction Coeff.	Average
MF Coefficients PPO	**0.562**	**0.250**	0.438	0.417
MF Coefficients A2C	**0.562**	**0.250**	0.438	0.417
Latent PPO	**0.562**	**0.250**	0.500	**0.437**
Latent A2C	**0.562**	**0.250**	0.500	**0.437**
Power Series PPO	0.125	0.062	0.312	0.166
Power Series A2C	0.125	0.062	0.312	0.166
Key Attributes PPO	0.188	0.188	**0.562**	0.313
Key Attributes A2C	0.188	0.188	**0.562**	0.313

5.3.3 Force Curve Plots

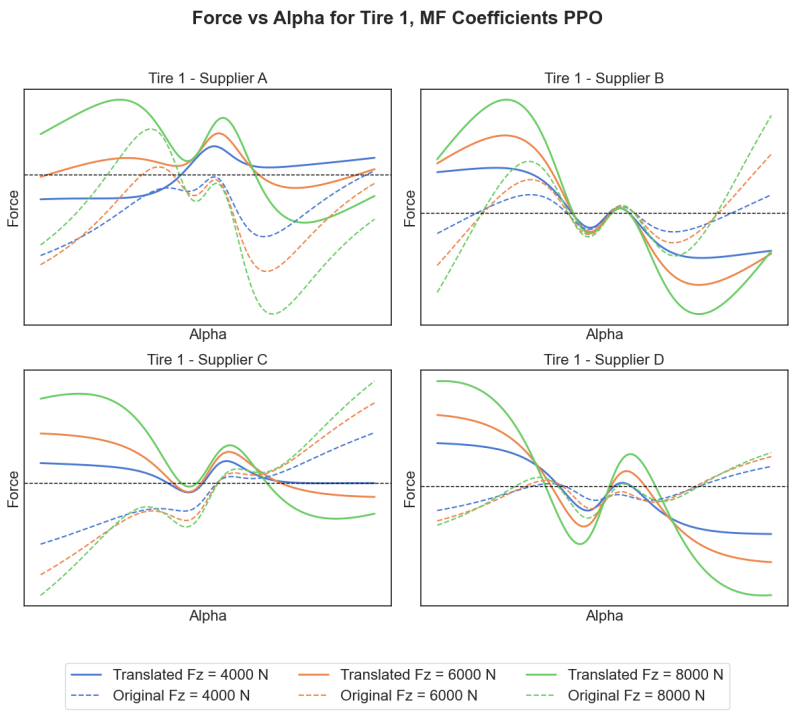


Figure 5.10: Lateral force curve vs. slip angle for the MF Coefficients environment using PPO, Reach Target Fast, and *20 bins*.

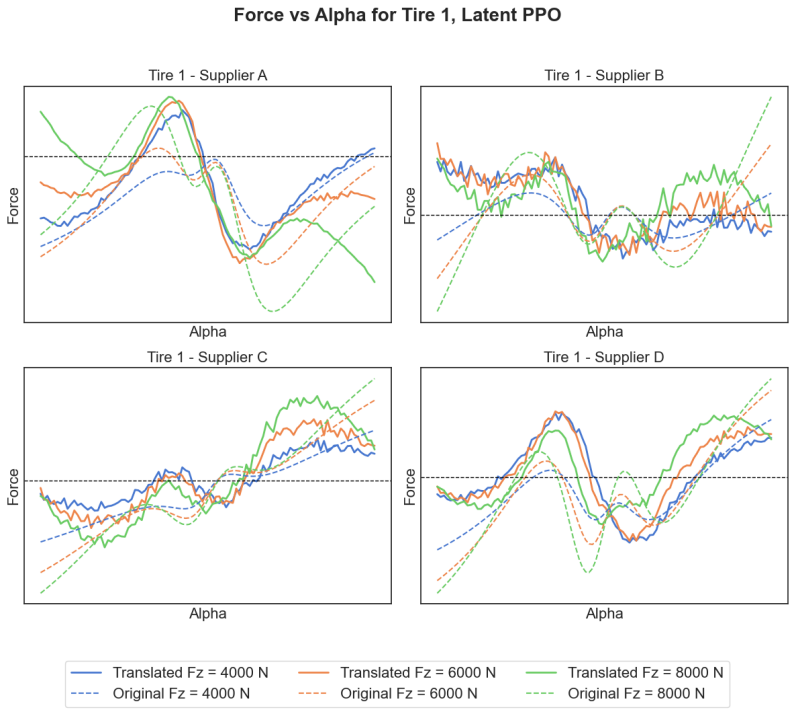


Figure 5.11: Lateral force curve vs. slip angle for the Latent Space environment using PPO, Reach Target Fast, and *20 bins*.

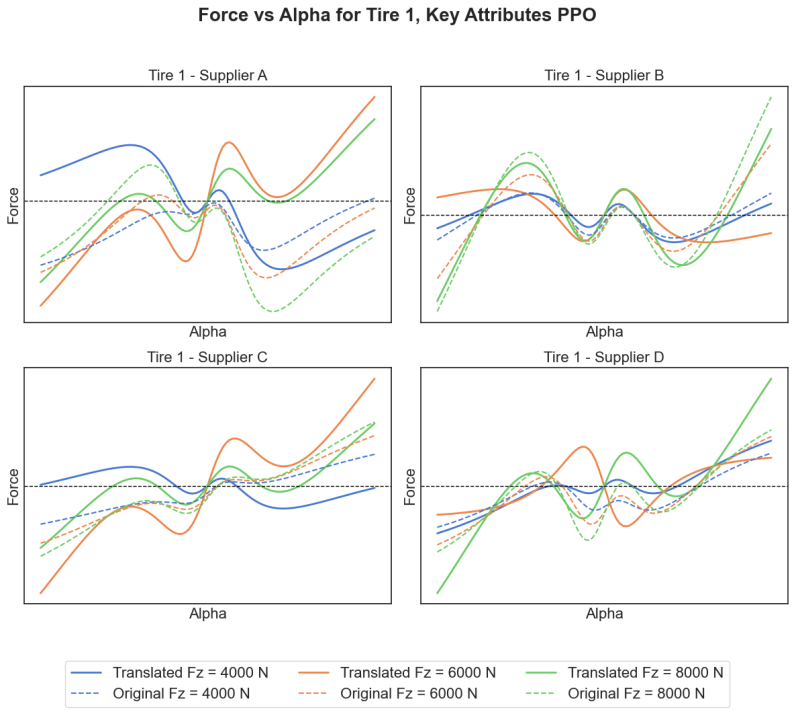


Figure 5.12: Lateral force curve vs. slip angle for the Key Attributes environment using PPO, Reach Target Fast, and *20 bins*.

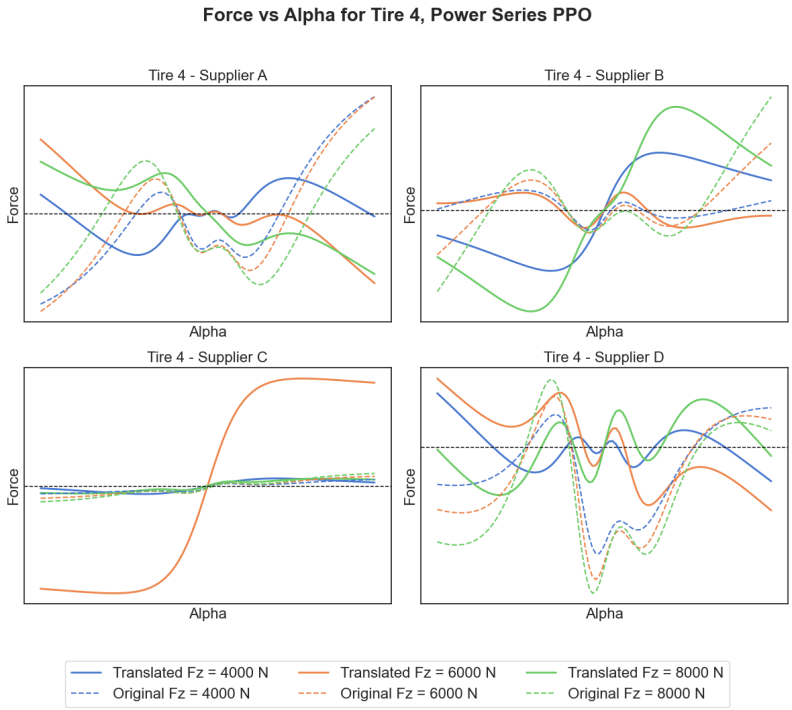


Figure 5.13: Lateral force curve vs. slip angle for the Power Series environment using PPO, Reach Target Fast, and *20 bins*.

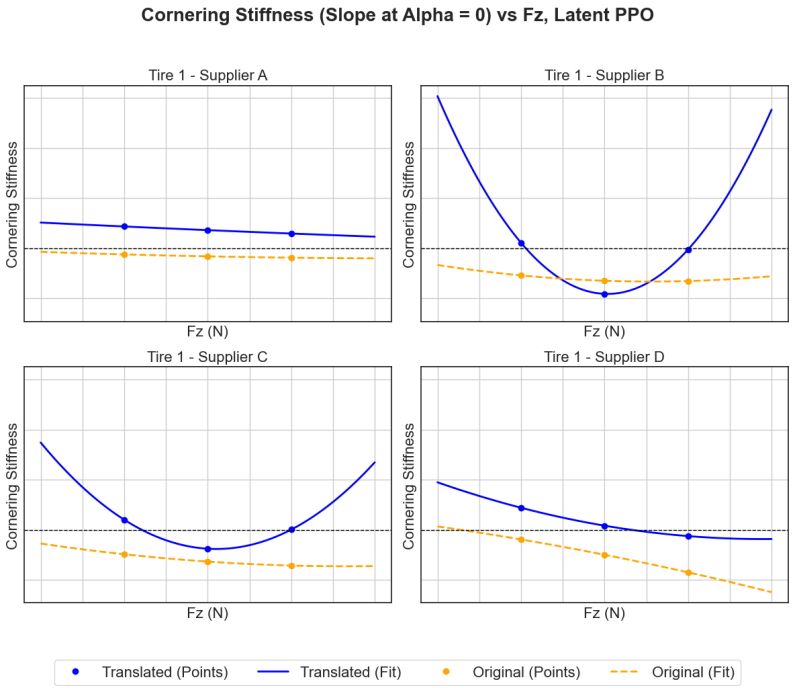


Figure 5.14: Cornering stiffness vs. normal load (F_z) for the Latent Space environment using PPO, Reach Target Fast, and *20 bins*.

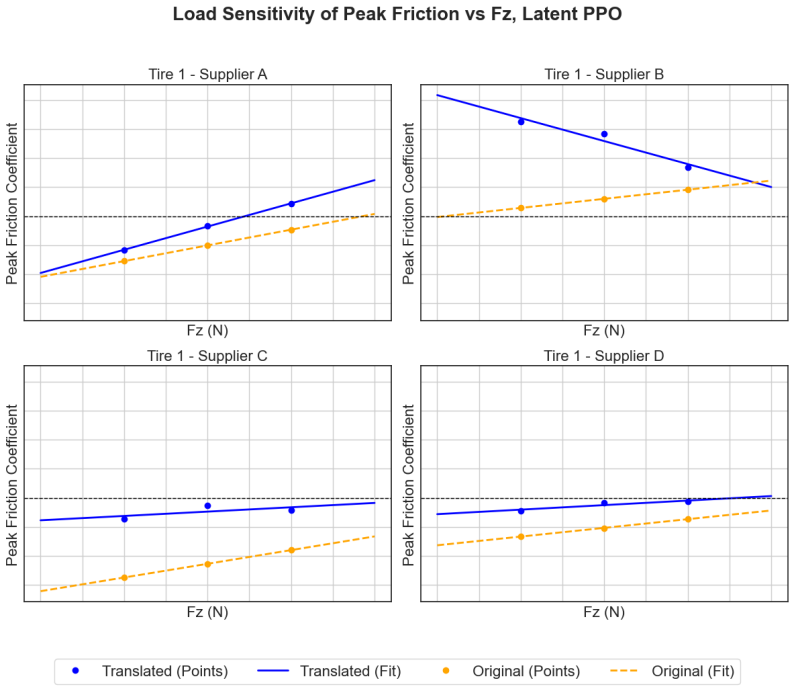
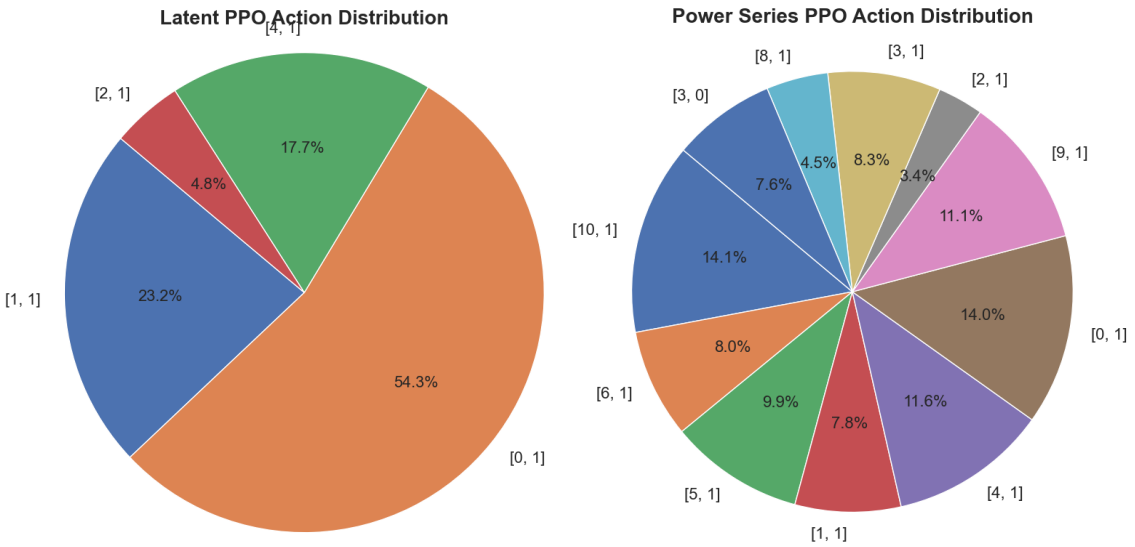


Figure 5.15: Load sensitivity of peak friction vs. normal load (F_z) for the Latent Space environment using PPO, Reach Target Fast, and *20 bins*.

5.3.4 Action Distributions

In this section, the action distributions of the agents' learned policies are presented in a subset of environments. In Figure 5.16a, the action distribution for the best-performing configuration of RL parameters is visualized. Meanwhile, in Figure 5.16b, the action distribution of a less optimal policy is shown. This section presents results that will be used to answer RQ4: "What learnings about the data can be made from the RL policies?".



(a) Latent Space environment.

(b) Power Series environment.

Figure 5.16: Action distribution for the PPO algorithm when using 20 bins. The reward function was Reach Target Fast.

6

Discussion

In this chapter, a discussion of the results will be provided related to the RQs stated in Section 1.1:

- RQ1: Can supervised learning be used to classify tire property files?
- RQ2: Can a deep RL agent learn a policy that optimizes the output probabilities of a classifier?
- RQ3: Can RL or other ML techniques be used to translate biased tire models into unbiased models?
- RQ4: What learnings about the data can be made from the RL policies?

RQ1 is addressed in Section 6.1, while RQ2 is analyzed in Section 6.2. The model's performance in relation to RQ3 is discussed in Section 6.3. Finally, insights related to RQ4 are covered in Section 6.5.

6.1 Evaluating the Optimal Machine Learning Classifier for Tire Property File Classification

This section discusses the credibility of the classifier selection process as well as what implications the classifier's result has on the available data and the RL reward function. It also analyzes the reasons for the varying performance in each data representation.

6.1.1 Hyperparameter Identification

When selecting the optimal hyperparameters of the classifiers, a full grid search was not performed on all possible combinations (1,728 for the ANN and 10,000 for the RF). Consequently, there exists a possibility of better performing combinations than the ones found. However, due to time constraints, it was not feasible to evaluate every single combination of hyperparameters. Instead, our approach weighed the trade-off between optimal performance and tuning time. Using a random grid search first still yielded a good indication of the suitable parameters for the ANN, removing parameter values that would lead to bad performance. By limiting the number of combinations to contain only suitable options and performing a full grid search on these, a good trade-off was achieved.

An indicator that this method worked can be noticed when comparing the results between the random and full grid search. The best-performing configurations of the full grid search have higher scores for both metrics, which means that the selected values are indeed values that yield high performance.

For the RF, evaluating it based on the number of trees first and then varying the maximum depth seemed intuitive, as the number of trees generally influences the accuracy of the RF the most. Having determined this, the maximum tree depth was the logical step to determine next.

6.1.2 Impact of SMOTE

When comparing the performance between the models with the selected hyperparameters, the RF classifier achieved the highest score, both when SMOTE was used and when not, see Table 5.6. Interestingly, while training on a dataset with SMOTE applied yielded a performance boost to the RF model, the increases were not significant. For the ANN, the performance increase was only minor, while for the NB, it even decreased. In general, it seems that the effect of SMOTE was not that significant.

One possible explanation for this might be that although some categories of .tir files contained many fewer samples compared to others, the .tir files between each category were still sufficiently separated. This meant that the classifiers did not need to train on additional synthetic examples of the minority classes to perform well. As a result, the models were generally able to classify the .tir files accurately even without the use of SMOTE.

Another explanation might be that SMOTE simply was not the most suitable method for addressing class imbalance in our dataset. Generating new synthetic samples based on Euclidean distance might not accurately capture the underlying data distribution of the minority classes.

6.1.3 Comparison of Classifier Models

The highest performing classifier was a fairly small RF classifier with 30 trees, each with a maximum tree depth of 10. However, the performance of the ANNs did not lag too far behind. It is worth noting that the number of nodes and layers of the ANN were not tuned in this thesis. The ANNs had three hidden layers with 128, 128, and 64 nodes each, which is a relatively small network.

One could argue that varying the number of nodes and layers might lead to a better performing ANN, potentially better than the RF. Although this may be true, some experiments were performed at the beginning using a larger network, but this did not improve the performance significantly. Nevertheless, a thorough investigation was not conducted as this would require additional time. Given that the primary focus of the thesis was the development of the RL part of the model, further tuning of the ANN was not prioritized. Ultimately, the performance of the final RF classifier was considered sufficiently strong for the classification task, making further optimization

of the ANN unnecessary in the context of this thesis.

6.1.4 Deeper Analysis of Best Classifier's Performance

Analyzing the precision, recall, and F1 scores of the best-performing classifier in Table 5.8 reveals that the classifier struggles to achieve a high precision for Category C. While precision exceeds 0.93 for the other categories, it is comparatively lower at 0.832 for Category C.

The confusion matrix in Figure 5.3 shows that the classifier often misclassifies instances from other categories (except Category F) as Category C. This suggests that the information within .tir files of Category C is similar to that of the other categories, making them harder to distinguish. However, the recall of Category C remains high (0.974), which indicates that the classifier most of the time predicts Category C correctly. In other words, the model is reliable when predicting Category C for a true Category C instance, but it often mislabels samples from other categories as Category C.

In contrast, the opposite is true to some extent for Category B. In this case, the recall score is the lowest among the categories (0.905), while the precision is high (0.973). Examining the confusion matrix, the classifier sometimes confuses Category B as Category A, C, or D. However, as the high precision implies, the classifier does not tend to confuse the other categories as Category B. This implies that when the classifier predicts a sample as Category B, it is usually correct. However, it sometimes fails to recognize true Category B samples, instead classifying them as other categories.

6.1.5 Potential Implication of Classifier in Reward Function

The low recall for Category B and the low precision for Category C may have direct implications for the RL model. This was not investigated in-depth, but some potential implications are described below.

For example, when using Category B as the target category, the agent might reach valid Category B states but receive no reward due to the classifier incorrectly classifying these states as other categories. This might make the agent avoid certain states that are actually beneficial but not rewarded properly. In turn, the agent might take longer to learn effective policies as it does not receive consistent feedback for correct actions.

Conversely, if Category C is used as the target, the agent might receive rewards for reaching states that are not truly Category C, but are misclassified as such. This might make the agent prioritize exploring states that are not genuinely aligned with the target category. As such, the agent's policy might converge fast (due to high recall) to strategies that favor incorrect states, leading to suboptimal policies.

6.1.6 Classifier Performance on Data Representation

As illustrated in Table 5.9, the performance of the RF classifier varied for the different data representations. In particular, the classifier received lower accuracy and F1 scores for the Key Attributes of Force Curve and Magic Formula as a Power Series data representations. This had direct implications for the RL model, as it consistently performed worse in these data representations, see Section 5.3.1 and 5.3.2.

One reason for the classifier's poorer performance might be the potential loss of information in these representations. In the case of Key Attributes of Force Curve, only three distinct points on the force curve were used per normal load. Although these points were selected to capture the most important aspects of tire behavior, they did not represent the full force curve. As a result, it may struggle to differentiate between categories if the categories are similar in these key areas. In contrast, other data representations might allow the classifier to separate categories based on less meaningful information, which might not be preferable but would explain their higher performance.

Similar logic can be applied for the Magic Formula as a Power Series data representation. In this case, the MF had been converted to a power series, and its derivatives, k_1, k_2, and k_3, were used as input for each normal load along with the variable D. The three derivatives contain the variables B, C, and E, and carry physical meaning behind them. For instance, $k_1 \cdot D = BCD$ represents the cornering stiffness, while k_2 and k_3 are derivatives of this. However, although the derivatives come from the complete MF itself, they do not capture all the information of the force curve. Since these derivatives mainly focus on cornering stiffness, which may be quite similar across categories, the classifier might again struggle to distinguish between them.

6.2 Classifier as Part of a Reward Function

Figure 5.9 shows the mean episode reward for all state representations and RL algorithms for the 20 bin and reward function: Reach Target Fast. It can be seen that all models increased their reward over time. Consequently, it can be stated that all models learned to improve their reward.

The various models had different learning speeds, which correlate to the difficulties of the different environments. Since the mean reward improves over time, it can be concluded that the RL agents learned how to reach states that improved the reward. Further, the reward was a function of the probability assessed by the classifier, therefore, it shows that an RL agent indeed can learn a policy that increases the assessed probability of a classifier. However, this does not necessarily mean that the outputs of the RL model are good translations. The quality of the outputs of the RL model will be discussed in depth in the next section.

6.3 Mitigating Bias in Tire Property Files

Tables 5.10–5.15, which display the average MSE of the translated tire models compared to the original data, show that only two configurations (see Table 6.1), improve some metric. No configuration improves on all three scores simultaneously. There are several reasons for this, which will be discussed in the next subsections.

Table 6.1: The two configurations that improved one MSE metric compared to the original data.

RL algo.	State Repr.	Bins	Reward	Metric
PPO	Latent Space	20	Reach Target Fast	Force
PPO	Key Attributes	20	Reach Target Fast	Peak Friction

6.3.1 Improvement vs. Deterioration

One aspect that plagued the models was that it was only possible to improve the translations by a fixed magnitude per test case, while it was possible to perform infinitely worse compared to the original data. Combining this with the fact that MSE squares the errors, each test case that was worse affects the average much more than an improvement does. Therefore, Tables 5.16–5.21 were used to illustrate the fraction of test cases that were improved. From these tables, it is apparent that few configurations managed to improve a majority of test cases on one of the metrics. However, even with this more generous metric, no model improved on all three metrics.

6.3.2 Jaggedness of Translated Tire Models With Latent Space State Representation

As can be seen in Figure 5.11, the output of the RL model that uses the Latent Space state representation has very jagged force curves. This was due to the way the latent space representation works. The intended use of an autoencoder is to decode a force curve into a latent representation, not to decode a latent representation to a force curve. This meant that some combinations of latent features generated by the RL model might not have smooth representations of force curves, since these combinations were never part of the original training data of the autoencoder.

Notably, the jaggedness was worsening the performance in terms of MSE. This is because the points farther from the reference affect the MSE more than the points closer. This meant that all the benefit the jaggedness gets from being slightly closer to the reference at some points was outweighed by the cost of being slightly further away at other points.

The jaggedness could likely be alleviated by adding a penalty term to the reward function that punishes this behavior. However, this would likely lead to translations with a lower probability of being the target class compared to the current solution. Another remedy would be to smooth the force curves using either a rolling average

or curve fitting. The latter of the approaches likely yields force curves that are of a higher probability of being classified as being from the target category compared to using penalties in the reward function.

6.3.3 Reinforcement Learning Model Abusing Weaknesses in Classifier

The RL agents/policies stated goal was to increase its reward by any means necessary, i.e., it did not necessarily behave in the way the authors intended. The state with the highest reward might not have been the state that a perfect translator would reach. The RL was incentivized to abuse any weaknesses that the classifier might have possessed. Such a weakness could, as an example, be: a higher value of the first feature indicates an increasing likelihood of the state belonging to the target category. In that case, the RL model would be incentivized to increase this feature. Formally,

$$x > \varepsilon \implies CLF(s|x) = S_U$$

where x is a feature of the state s and ε is some threshold value. Although x being above a certain threshold implies that the state is classified as the target category, the implication does not necessarily hold for the reverse. This weakness is potentially abused by the RL model in order to reach highly rewarded states that might not necessarily be states seen in the data, and the RL model might incorrectly assume that the implication holds both ways:

$$x > \varepsilon \iff CLF(s|x) = S_U$$

6.3.4 Numerical Instability of Power Series

Tables 5.10–5.15 show that the average MSE for all models that used the Power Series state representation were orders of magnitude worse than for all other models. Meanwhile, the Tables 5.16–5.21 show that for some of the data representations, these models managed to improve the similarity compared to the original data. Furthermore, in Figure 5.13 it is clear that the models tended to either marginally improve the similarity or drastically worsen it (see Tire 4 – Supplier C). Small changes in the features might radically alter the output force curve. The state representation shows great potential but struggles greatly with numerical stability. If this instability could be alleviated, then there is likely potential when using this type of state representation.

6.4 Environment Tuning

The results of the RL model depended on the reward function, the RL algorithm, and the number of bins used. The effect of each will be discussed in this section.

6.4.1 Effects of Reward Function

The two different reward functions used yielded different results. Despite this, no clear winner can be stated between the two reward functions. In some cases, using Reach Target Fast led to a lower MSE and a higher improved fraction, while at other times Maximize Reward was the better option. In general, it seems like the other RL parameters affected the results equally as much, since the results fluctuated a lot depending on the number of bins, the environment, and the RL algorithm. As such, it is difficult to conclude which one is the best.

However, notably, the only time the agent was able to obtain a lower MSE compared to the reference was when using Reach Target Fast, see Table 5.11. Moreover, using this reward function was also the only time the agent was able to improve a majority of all their curves, see Table 5.17. This might indicate that Reach Target Fast is the superior reward function, but it could also be a coincidence. To firmly state this, further investigations would be required.

Interestingly, Maximize Reward seems to yield the same results in each data representation regardless of the RL algorithm. This was, however, not that surprising, since the goal of this reward was to find the state that maximizes the reward. This just indicates that both RL algorithms managed to identify the best possible state.

6.4.2 Effects of Reinforcement Learning Algorithm

Figure 5.9 shows the average episodic reward during training. The data clearly shows that PPO was faster to converge to a fixed policy compared to A2C. Further, the average training reward was larger for PPO compared to A2C. However, there is no real conclusion to be drawn from this, since it might just mean that the A2C algorithm was executing the exploitation and exploration trade-off differently.

6.4.3 Effect of Binning

By binning the input features, the state space of the RL environment became discrete rather than continuous. The effect was that the RL environment contained a finite number of states as opposed to an infinite amount[1]. The environment was simplified by discretization, and it was, from preliminary results, easier for the RL models to improve on.

A drawback of using binning was that some amount of information about the data was lost. This can be seen in Figures 5.5–5.8, where fewer number of bins leads to the classifier struggling to differentiate between categories. However, the data that was lost was not necessarily useful for translation purposes. The information lost was mainly the specific digits of the MF coefficients, while the approximate magnitude remains. Essentially, by binning the data, the overall force curve remained very similar, but the details were removed. While these details might have included a lot of information about the category, they were not useful for translations of the magnitude observed in the test data.

[1]In theory, the number of states was finite due to the precision of floating point numbers.

6.5 Mapping Data Characteristics to Reinforcement Learning Policies

Figure 5.16a shows the action distribution for the policy of a model with the reward function: Reach Target Fast trained in the Latent Space environment with 20 bins. The policy was among the top performers for several of the metrics used in Tables 5.11 and 5.17. From the distribution, it is apparent that the policy did not consist of random actions. The model has access to 16 actions, but only four were used. This type of distribution was consistent for all top performers; the best policies tended to prefer 2-4 actions, which indicates that the original data differed in only some key attributes between the different categories. Figure 5.16b shows the distribution of a policy that, according to the metrics, struggled quite a lot compared to the previous one. Note that this distribution is also much more uniform. This could indicate either of two things:

- The environment of the Power Series state representation was much less predictable and thus the agent struggled to find a good policy.
- The environment of the Power Series state representation required a much more diverse policy, since there was a greater variability of the optimal action for each state.

Due to the overall bad performance of the model in the Power Series environment, the former seems more likely to be true. In general, the trend seems to be that the policies with worse MSE and lower fraction of improvements seem to have more diverse policies. This supports the notion that a diverse action distribution is indicative of a struggling policy due to a difficult environment.

In terms of the action taken, for the Latent Space state representation, it is difficult to interpret what the consequence of each action was. However, for all other state representations, frequent increases of features indicate that the reference tends to have larger values for that particular feature, and vice versa for decreases. The action distributions could thus be used to identify features that tended to be over- or underestimated in the different categories.

6.6 Limitations

In this section, limitations of the thesis will be discussed. In particular, limitations related to the methodology and the available data.

6.6.1 Data

For any ML task, the quantity and quality of available data is of utmost importance. For this thesis, the available data could be categorized into two types:

- Labeled data: The data used as test data for the RL model, where each category had measurements from the same tire.

- Unlabeled data: .tir files without direct references in the other categories. Used in order to train classifier.

The first type is generally the ideal type of data for a supervised learning approach. Since each category has .tir files of the same tire, it is possible to compare these and use one as a reference output. With sufficiently many (on the order of thousands) of these .tir files of the same tire in different categories, it would be possible to train a supervised translator model. However, for this thesis, there were only 4 tires and 5 categories, totaling 20 .tir files of this kind. By using one category as a reference, only 16 files could be used as test cases.

Due to the expensive nature of data acquisition for this kind of data, a supervised approach is infeasible. However, even a smaller amount of tires measured by different manufacturers (categories) could yield great insight into the variance, mean, and bias of tire measurement.

The second type of data is what was used to train the translator model of the thesis, this was done by first training a classifier to identify the manufacturer (category) and then using that as the reward function . Increases in data quantity would certainly yield a higher performance in all metrics. However, the difference would likely be less than even a small increase in the quantity of the first type of data.

6.6.2 Misalignment of Evaluation Metrics and Reward Functions

Due to the limitations of the data, it was not possible to use the evaluation metric as a reward function of the RL model. The effect of this was that the RL model was optimizing a reward function that acted as a proxy for the evaluation metric. Depending on which reward function was used, the quality of the mapping of reward function and evaluation metrics varied, as a high average reward did not necessarily lead to a successful translation.

6.6.3 Finetuning of RL Hyperparameters

The RL algorithms used in this thesis, from `stable_baselines3` [28], contains a number of different hyperparameters, such as:

- Learning rate
- Discount factor
- Clip range
- etc.

Due to time constraints, optimization of these hyperparameters was not performed. It is likely that with more optimal hyperparameters, the performance of the translation system could be further improved.

6.6.4 Focus on Lateral Force

The MF tire model outputs forces and moments in all three axes. However, in this thesis, the only output considered was the lateral force. The reason for this was that the methodology is independent of the output chosen, and the lateral force was deemed the most important phenomenon to model. This meant that all results are only verified for the lateral load, although the expected result and behavior on models for longitudinal load and moments would likely behave similarly as in the lateral load case.

6.6.5 Other Potential Methods

Without access to a larger quantity of higher-quality data, the potential methods that could be used instead are somewhat limited. The field of image processing contains some methods for unsupervised translation tasks, such as "Cycle-GAN" [44]. There might be some potential in adapting these techniques for tabular data. Further, the field of natural language processing often deals with translation tasks, there might exist some techniques in that field that might be applicable to the problem of this thesis.

Furthermore, other reward function techniques could also be considered. For example, "Random Network Distillation" [45] is a technique that could encourage the RL agent to explore more and find other optimal states. Few experiments with this method were performed in this thesis, but if investigated further, it could lead to better-performing models.

7

Conclusion

Accurate and reliable tire models are essential in the development of new vehicles. This thesis explores a new approach that leverages supervised learning to discriminate different tire models, and RL to translate biased tire models into unbiased ones.

The results show that while a classifier can accurately distinguish between different categories of tire models (see Figure 5.3), the RL model struggles to achieve reliable translations. Regardless of the RL algorithm, the data representation, or the reward function, successful translations, i.e. lower MSE than the original data, were only achieved in a few cases (see Tables 5.16–5.21). However, it should be noted that some RL models did improve several of their translated force curves, even if the overall MSE difference was higher (see Tables 5.10–5.21), indicating partial success in learning meaningful translations.

Although the performance of the RL models was limited, the results still indicate potential in the proposed methodology. In the cases where the model did in fact improve, it is clear that the RL agent had learned an effective policy rather than achieving improvements by chance. This is supported by the fact that some model configurations improved a majority of their translated curves and that their action distributions were uniform (see Figure 5.16). Furthermore, there exist clear problems in both the latent and power series state representations, the latent state representation produced jagged force curves and the power series state representation struggled with numerical stability. However, when successful, these representations actually yield the most promising translations (see Figures 5.11 and 5.13). As such, addressing these issues could lead to even greater potential for this approach.

Future research using this methodology should focus on obtaining more high-quality labeled data in order to estimate the variance of tire models from different categories. Knowing this would help establish whether translation is feasible in the first place. In addition, a thorough investigation of the optimal RL parameters, such as learning rate, roll-out length, number of trajectories, etc., should be performed to establish better models. Finally, applying alternative techniques such as “Cycle-GAN”, which have been proven to work with unsupervised image translations, and “Random Network Distillation”, which increases agent exploration, could lead to better results.

Bibliography

[1] H. Pacejka and I. J. M. Besselink, *Tire and Vehicle Dynamics*. Chantilly, UNITED KINGDOM: Elsevier Science & Technology, 2012, ISBN: 9780080970172. [Online]. Available: http://ebookcentral.proquest.com/lib/chalmers/detail.action?docID=892225.

[2] MTS, *Flat-Trac® Tire Force & Moment Measurement Systems*. [Online]. Available: https://www.mts.com/en/products/automotive/tire-test-systems/flat-trac-tire-system.

[3] D.-I. J. Zamow, "TYDEX-Format Reference Manual, Release 1.3 TYDEX-Format Description and Reference Manual Release 1.3 Initiated by the TYDEX Workshop worked out by," Tech. Rep., 1997.

[4] J. J. van Oosten and E. Bakker, "Determination of magic tyre model parameters," *Vehicle System Dynamics*, vol. 21, no. sup1, pp. 19–29, Jan. 1992, ISSN: 17445159. DOI: 10.1080/00423119208969995.

[5] A. V. Alagappan, K. V. Rao, and R. K. Kumar, "A comparison of various algorithms to extract Magic Formula tyre model coefficients for vehicle dynamics simulations," *Vehicle System Dynamics*, vol. 53, no. 2, pp. 154–178, Feb. 2015, ISSN: 17445159. DOI: 10.1080/00423114.2014.984727.

[6] Z. Cheng and Z. Lu, "Nonlinear Research and Efficient Parameter Identification of Magic Formula Tire Model," *Mathematical Problems in Engineering*, vol. 2017, 2017, ISSN: 15635147. DOI: 10.1155/2017/6924506.

[7] F. Farroni, R. Lamberti, N. Mancinelli, and F. Timpone, "TRIP-ID: A tool for a smart and interactive identification of Magic Formula tyre model parameters from experimental data acquired on track or test rig," *Mechanical Systems and Signal Processing*, vol. 102, pp. 1–22, Mar. 2018, ISSN: 10961216. DOI: 10.1016/j.ymssp.2017.07.025.

[8] S. Feng, Y. Zhao, H. Deng, Q. Wang, and T. Chen, "Parameter Identification of Magic Formula Tire Model Based on Fibonacci Tree Optimization Algorithm," *Journal of Shanghai Jiaotong University (Science)*, vol. 26, no. 5, pp. 647–657, Oct. 2021, ISSN: 19958188. DOI: 10.1007/s12204-021-2354-9.

[9] F. Pedregosa, V. Michel, O. Grisel, et al., "Scikit-learn: Machine Learning in Python Gaël Varoquaux Bertrand Thirion Vincent Dubourg Alexandre Passos PEDREGOSA, VAROQUAUX, GRAMFORT ET AL. Matthieu Perrot," Tech. Rep., 2011, pp. 2825–2830. [Online]. Available: http://scikit-learn.sourceforge.net..

[10] M. Lagoudakis and R. Parr, "Reinforcement Learning as Classification: Leveraging Modern Classifiers.," May 2003, pp. 424–431.

[11] M. A. Wiering, H. van Hasselt, A.-D. Pietersma, and L. Schomaker, "Reinforcement learning algorithms for solving classification problems," in *2011 IEEE Symposium on Adaptive Dynamic Programming and Reinforcement Learning (ADPRL)*, 2011, pp. 91–96. DOI: 10.1109/ADPRL.2011.5967372.

[12] "Using the MF-Tyre Model Overview," Tech. Rep., 2001. [Online]. Available: www.delft-tyre.com.

[13] H. Heisler, *Advanced Vehicle Technology*, 2nd ed. Elsevier, 2002, ISBN: 9780750651318. DOI: 10.1016/B978-0-7506-5131-8.X5000-3.

[14] Q. Liu and Y. Wu, "Supervised Learning," in *Encyclopedia of the Sciences of Learning*, Springer US, 2012, pp. 3243–3245. DOI: 10.1007/978-1-4419-1428-6{_}451.

[15] A. F. Alnuaimi and T. H. Albaldawi, "An overview of machine learning classification techniques," in *BIO Web of Conferences*, vol. 97, EDP Sciences, Apr. 2024. DOI: 10.1051/bioconf/20249700133.

[16] I. Goodfellow, Y. Bengio, and A. Courville, *Deep Learning*. MIT Press, 2016.

[17] J. Lederer, "Activation Functions in Artificial Neural Networks: A Systematic Overview," Jan. 2021. [Online]. Available: http://arxiv.org/abs/2101.09957.

[18] A. F. Agarap, "Deep Learning using Rectified Linear Units (ReLU)," Mar. 2018. [Online]. Available: http://arxiv.org/abs/1803.08375.

[19] D. E. Rumelhart, G. E. Hinton, and R. J. Williams, "Learning representations by back-propagating errors," *Nature*, vol. 323, no. 6088, pp. 533–536, Oct. 1986, ISSN: 0028-0836. DOI: 10.1038/323533a0.

[20] A. Krogh and J. A. Hertz, "A Simple Weight Decay Can Improve Generalization," J. Moody, S. Hanson, and R. P. Lippmann, Eds. [Online]. Available: https://proceedings.neurips.cc/paper_files/paper/1991/file/8eefcfdf5990e441f0fb6f3fad709e21-Paper.pdf.

[21] I. Loshchilov and F. Hutter, "Decoupled Weight Decay Regularization," Nov. 2017. [Online]. Available: http://arxiv.org/abs/1711.05101.

[22] H. Zhang, "The Optimality of Naive Bayes." [Online]. Available: https://aaai.org/papers/flairs-2004-097/.

[23] Tin Kam Ho, "Random decision forests," in *Proceedings of 3rd International Conference on Document Analysis and Recognition*, IEEE Comput. Soc. Press, 1995, pp. 278–282, ISBN: 0-8186-7128-9. DOI: 10.1109/ICDAR.1995.598994.

[24] A. Liaw and M. Wiener, "Classification and Regression by randomForest," Tech. Rep. 3, 2002. [Online]. Available: https://journal.r-project.org/articles/RN-2002-022/RN-2002-022.pdf.

[25] P. Singh, N. Singh, K. K. Singh, and A. Singh, "Diagnosing of disease using machine learning," in *Machine Learning and the Internet of Medical Things in Healthcare*, Elsevier, 2021, pp. 89–111. DOI: 10.1016/B978-0-12-821229-5.00003-3.

[26] R. S. Sutton and A. G. Barto, *Reinforcement learning : an introduction*. The MIT Press, 2020, ISBN: 9780262039246. [Online]. Available: https://web.stanford.edu/class/psych209/Readings/SuttonBartoIPRLBook2ndEd.pdf.

[27] V. Mnih, A. P. Badia, M. Mirza, et al., "Asynchronous Methods for Deep Reinforcement Learning," Feb. 2016. [Online]. Available: http://arxiv.org/abs/1602.01783.

[28] A. Raffin, A. Hill, A. Gleave, A. Kanervisto, M. Ernestus, and N. Dormann, "Stable-Baselines3: Reliable Reinforcement Learning Implementations," Tech. Rep., 2021, pp. 1–8. [Online]. Available: https://github.com/DLR-RM/stable-baselines3.

[29] J. Schulman, F. Wolski, P. Dhariwal, A. Radford, and O. Klimov, "Proximal Policy Optimization Algorithms," Jul. 2017. [Online]. Available: http://arxiv.org/abs/1707.06347.

[30] I. Csiszar, "I-Divergence Geometry of Probability Distributions and Minimization Problems," *The Annals of Probability*, vol. 3, no. 1, Feb. 1975, ISSN: 0091-1798. DOI: 10.1214/aop/1176996454.

[31] D. P. Kingma and J. Ba, "Adam: A Method for Stochastic Optimization," Dec. 2014. [Online]. Available: http://arxiv.org/abs/1412.6980.

[32] N. V. Chawla, K. W. Bowyer, L. O. Hall, and W. P. Kegelmeyer, "SMOTE: Synthetic Minority Over-sampling Technique," Tech. Rep., 2002, pp. 321–357. [Online]. Available: https://arxiv.org/abs/1106.1813.

[33] R. Nisbet, G. Miner, and K. Yale, "Data Understanding and Preparation," in *Handbook of Statistical Analysis and Data Mining Applications*, Elsevier, 2018, pp. 55–82. DOI: 10.1016/B978-0-12-416632-5.00004-9. [Online]. Available: https://doi.org/10.1016/B978-0-12-416632-5.00004-9.

[34] Google for Developers, *Numerical data: Binning*, Oct. 2024. [Online]. Available: https://developers.google.com/machine-learning/crash-course/numerical-data/binning.

[35] Kevin Li, Abhishek Gupta, Vitchyr H. Pong, et al., "REINFORCEMENT LEARNING WITH BAYESIAN CLASSI-FIERS: EFFICIENT SKILL LEARNING FROM OUTCOME EXAMPLES," Tech. Rep., 2001. [Online]. Available: https://openreview.net/forum?id=0ZgVHzdKicb.

[36] V. L. Parsons, "Stratified Sampling," in *Wiley StatsRef: Statistics Reference Online*, Wiley, Feb. 2017, pp. 1–11. DOI: 10.1002/9781118445112.stat05999.pub2.

[37] J. Qi, J. Du, S. M. Siniscalchi, X. Ma, and C.-H. Lee, "On Mean Absolute Error for Deep Neural Network Based Vector-to-Vector Regression," Aug. 2020. DOI: 10.1109/LSP.2020.3016837. [Online]. Available: http://arxiv.org/abs/2008.07281%20http://dx.doi.org/10.1109/LSP.2020.3016837.

[38] P. Refaeilzadeh, L. Tang, and H. Liu, "Cross-Validation," in *Encyclopedia of Database Systems*, Boston, MA: Springer US, 2009, pp. 532–538. DOI: 10.1007/978-0-387-39940-9{_}565.

[39] P. Liashchynskyi and P. Liashchynskyi, "Grid Search, Random Search, Genetic Algorithm: A Big Comparison for NAS," Dec. 2019. [Online]. Available: http://arxiv.org/abs/1912.06059.

[40] S. Ruder, "An overview of gradient descent optimization algorithms," Sep. 2016. [Online]. Available: http://arxiv.org/abs/1609.04747.

[41] A. Mao, M. Mohri, and Y. Zhong, "Cross-Entropy Loss Functions: Theoretical Analysis and Applications," Apr. 2023. [Online]. Available: http://arxiv.org/abs/2304.07288.

[42] J. Terven, D. M. Cordova-Esparza, A. Ramirez-Pedraza, E. A. Chavez-Urbiola, and J. A. Romero-Gonzalez, "Loss Functions and Metrics in Deep Learning," Jul. 2023. [Online]. Available: http://arxiv.org/abs/2307.02694.

[43] M. Towers, A. Kwiatkowski, J. Terry, et al., "Gymnasium: A Standard Interface for Reinforcement Learning Environments," Jul. 2024. [Online]. Available: http://arxiv.org/abs/2407.17032.

[44] J.-Y. Zhu, T. Park, P. Isola, and A. A. Efros, "Unpaired Image-to-Image Translation using Cycle-Consistent Adversarial Networks," Mar. 2017. [Online]. Available: http://arxiv.org/abs/1703.10593.

[45] Y. Burda, H. Edwards, A. Storkey, and O. Klimov, "Exploration by Random Network Distillation," Oct. 2018. [Online]. Available: http://arxiv.org/abs/1810.12894.

[46] Hans Pacejka, *Tire and Vehicle Dynamics*, 2nd ed. Elsevier Science & Technology, Jan. 2006.

A

Magic Formula Equations

The following equations are an excerpt from [12]. For a more in depth description of the MF 5.2 model see: [12], [46] or for a newer version of the model (6.2) see: [1].

Input variables:

κ - Longitudinal slip

α - Slip angle

γ - Camber angle

F_z - Normal wheel load

Output variables:

F_x - Longitudinal force

F_y - Lateral force

M_x - Overturning couple

M_y - Rolling resistance torque

M_z - Aligning torque

Basic tire parameters variables:

F_{z0} - Nominal load

R_0 - Unloaded tire radius

m_{belt} - Tire belt mass

$$df_z = \frac{F_z - F'_{z0}}{F'_{z0}} \tag{A.1}$$

$$F'_{z0} = F_{z0}\lambda_{F_{z0}} \tag{A.2}$$

Longitudinal Force ($\alpha = 0$):

$$F_x = F_{x0}(\kappa, F_z) \tag{A.3}$$

$$F_{x0} = D_x \sin[C_x \arctan\{B_x\kappa_x - E_x(B_x\kappa_x - \arctan(B_x\kappa_x))\}] + S_{Vx} \tag{A.4}$$

$$\kappa_x = \kappa + S_{Hx} \tag{A.5}$$

$$\gamma_x = \gamma\lambda_{\gamma x} \tag{A.6}$$

$$C_x = p_{Cx1}\lambda_{Cx} \tag{A.7}$$

$$D_x = \mu_x F_z \tag{A.8}$$

$$\mu_x = (p_{Dx1} + p_{Dx2}df_z)(1 - p_{Dx3}\gamma_x^2)\lambda_{\mu x} \tag{A.9}$$

$$E_x = (p_{Ex1} + p_{Ex2}df_z + p_{Ex3}df_z^2)\{1 - p_{Ex4}sgn(\kappa_x)\}\lambda_{Ex} \quad (\leq 1) \tag{A.10}$$

$$K_x = F_z(p_{Kx1} + p_{Kx2}df_z)\exp(p_{kx3}df_z)\lambda_{Kx} \tag{A.11}$$

$$B_x = K_x/(C_x D_x) \tag{A.12}$$

$$S_{Hx} = (p_{Hx1} + p_{Hx2}df_z)\lambda_{Hx} \tag{A.13}$$

$$S_{Vx} = F_z(p_{Vx1} + p_{Vx2}df_z)\lambda_{Vx}\lambda_{\mu x} \tag{A.14}$$

Lateral Force ($\kappa = 0$):

$$F_y = F_{y0}(\alpha, \gamma, F_z) \tag{A.15}$$

$$F_{y0} = D_y \sin[C_y \arctan\{B_y\alpha_y - E_y(B_y\alpha_y - \arctan(B_y\alpha_y))\}] + S_{Vy} \tag{A.16}$$

$$\alpha_y = \alpha + S_{Hy} \tag{A.17}$$

$$\gamma_y = \gamma\lambda_{\gamma y} \tag{A.18}$$

$$C_y = p_{Cy1}\lambda_{Cy} \tag{A.19}$$

$$D_y = \mu_y F_z \tag{A.20}$$

$$\mu_y = (p_{Dy1} + p_{Dy2}df_z)(1 - p_{Dy3}\gamma_y^2)\lambda_{\mu y} \tag{A.21}$$

$$E_y = (p_{Ey1} + p_{Ey2}df_z)\{1 - (p_{Ey3} + p_{Ey4}\gamma_y)sgn(\alpha_y)\}\lambda_{Ey} \quad (\leq 1) \tag{A.22}$$

$$K_y = p_{Ky1}F_{z0}\sin[2\arctan\{F_z/(p_{Ky2}F_{z0}\lambda_{F_{z0}})\}](1 - p_{Ky3}|\gamma_y|)\lambda_{F_{z0}}\lambda_{Ky} \tag{A.23}$$

$$B_y = K_y/(C_y D_y) \tag{A.24}$$

$$S_{Hy} = (p_{Hy1} + p_{Hy2}df_z)\lambda_{Hy} + p_{Hy3}\gamma_y \tag{A.25}$$

$$S_{Vy} = F_z\{(p_{Vy1} + p_{Vy2}df_z)\lambda_{Vy} + (p_{Vy3} + p_{Vy4}df_z)\gamma_y\}\lambda_{\mu y} \tag{A.26}$$

Aligning Torque ($\kappa = 0$):

$$M_z' = M_{z0}(\alpha, \gamma, F_z) \tag{A.27}$$

$$M_{z0} = -tF_{y0} + M_{zr} \tag{A.28}$$

$$t(\alpha_t) = D_t\cos[C_t\arctan\{B_t\alpha_t - E_t(B_t\alpha_t - \arctan(B_t\alpha_t))\}]\cos\alpha \tag{A.29}$$

$$\alpha_t = \alpha + S_{Ht} \tag{A.30}$$

$$M_{zr}(\alpha_r) = D_r\cos[C_r\arctan(B_r\alpha_r)]\cos\alpha \tag{A.31}$$

$$\alpha_r = \alpha + S_{Hf} \tag{A.32}$$

$$S_{Hf} = S_{Hy} + S_{Vy}/K_y \tag{A.33}$$

$$\gamma_z = \gamma\lambda_{\gamma z} \tag{A.34}$$

$$B_t = (q_{Bz1} + q_{Bz2}df_z + q_{Bz3}df_z^2)(1 + q_{Bz4}\gamma_z + q_{Bz5}|\gamma_z|)\lambda_{Ky}/\lambda_{\mu y} \tag{A.35}$$

$$C_t = q_{Cz1} \tag{A.36}$$

$$D_t = F_z(q_{Dz1} + q_{Dz2}df_z)(1 + q_{Dz3}\gamma_z + q_{Dz4}\gamma_z^2)(R_0/F_{z0})\lambda_t \tag{A.37}$$

$$\begin{aligned} E_t =&(q_{Ez1} + q_{Ez2}df_z + q_{Ez3}df_z^2) \\ &\left\{1 + (q_{Ez4} + q_{Ez5}\gamma_z)\frac{2}{\pi}\arctan(B_tC_t\alpha_t)\right\} \quad (\leq 1) \end{aligned} \tag{A.38}$$

$$S_{Ht} = q_{Hz1} + q_{Hz2}dfz + (q_{Hz3} + q_{Hz4}df_z)\gamma_z \tag{A.39}$$

$$B_r = q_{Bz9}\lambda_{Ky}/\lambda_{\mu y} + q_{Bz10}B_yC_y \tag{A.40}$$

$$D_r = F_z((q_{Dz6} + q_{Dz7}df_z)\lambda_{Mr} + (q_{Dz8} + q_{Dz9}df_z)\gamma_z)R_0\lambda_{\mu\gamma} \tag{A.41}$$

$$K_z = -tK_y \tag{A.42}$$

Longitudinal Force (Combined Slip):

$$F_x = F_{x0}G_{x\alpha}(\alpha, \kappa, F_z) \tag{A.43}$$

$$F_x = D_{x\alpha}\cos[C_{x\alpha}\arctan\{B_{x\alpha}\alpha_s - E_{x\alpha}(B_{x\alpha}\alpha_s - arctan(B_{x\alpha}\alpha_s))\}] \tag{A.44}$$

$$\alpha_s = \alpha + S_{Hx\alpha} \tag{A.45}$$

$$B_{x\alpha} = r_{Bx1}\cos[\arctan(r_{Bx2}\kappa)]\lambda_{x\alpha} \tag{A.46}$$

$$C_{x\alpha} = r_{Cx1} \tag{A.47}$$

$$D_{x\alpha} = \frac{F_{x0}}{cos[C_{x\alpha}\arctan\{B_{x\alpha}S_{Hx\alpha} - E_{x\alpha}(B_{x\alpha}S_{Hx\alpha} - \arctan(B_{x\alpha}S_{Hx\alpha}))\}]} \tag{A.48}$$

$$E_{x\alpha} = r_{Ex1} + r_{Ex2} df_z \tag{A.49}$$

$$S_{Hx\alpha} = r_{Hx1} \tag{A.50}$$

$$G_{x\alpha} = \frac{\cos[C_{x\alpha} \arctan\{B_{x\alpha}\alpha_s - E_{x\alpha}(B_{x\alpha}\alpha_s - \arctan(B_{x\alpha}\alpha_s))\}]}{\cos[C_{x\alpha} \arctan\{B_{x\alpha}S_{Hx\alpha} - E_{x\alpha}(B_{x\alpha}S_{Hx\alpha} - \arctan(B_{x\alpha}S_{Hx\alpha}))\}]} \tag{A.51}$$

Lateral Force (Combined Slip):

$$F_y = F_{y0} G_{y\kappa}(\alpha, \kappa, F_z) + S_{Vy\kappa} \tag{A.52}$$

$$F_y = D_{y\kappa} \cos[C_{y\kappa} \arctan\{B_{y\kappa}\kappa_S - E_{y\kappa}(B_{y\kappa}\kappa_S - \arctan(B_{y\kappa}\kappa_S))\}] + S_{Vy\kappa} \tag{A.53}$$

$$\kappa_S = \kappa + S_{Hy\kappa} \tag{A.54}$$

$$B_{y\kappa} = r_{By1} \cos[\arctan\{r_{By2}(\alpha - r_{By3})\}]\lambda_{y\kappa} \tag{A.55}$$

$$C_{y\kappa} = r_{Cy1} \tag{A.56}$$

$$D_{y\kappa} = \frac{F_{y0}}{\cos[C_{y\kappa} \arctan\{B_{y\kappa}S_{Hy\kappa} - E_{y\kappa}(B_{y\kappa}S_{Hy\kappa} - \arctan(B_{y\kappa}S_{Hy\kappa}))\}]} \tag{A.57}$$

$$E_{y\kappa} = r_{Ey1} + r_{Ey2} df_z \tag{A.58}$$

$$S_{Hy\kappa} = r_{Hy1} + r_{Hy2} df_z \tag{A.59}$$

$$S_{Vy\kappa} = D_{Vy\kappa} \sin[r_{Vy5} \arctan(r_{Vy6}\kappa)]\lambda_{Vy\kappa} \tag{A.60}$$

$$D_{Vy\kappa} = \mu_y F_z (r_{Vy1} + r_{Vy2} df_z + r_{Vy3}\gamma) \cos[\arctan(r_{Vy4}\alpha)] \tag{A.61}$$

$$G_{y\kappa} = \frac{\cos[C_{y\kappa} \arctan\{B_{y\kappa}\kappa_S - E_{y\kappa}(B_{y\kappa}\kappa_S - \arctan(B_{y\kappa}\kappa_S))\}]}{\cos[C_{y\kappa} \arctan\{B_{y\kappa}S_{Hy\kappa} - E_{y\kappa}(B_{y\kappa}S_{Hy\kappa} - \arctan(B_{y\kappa}S_{Hy\kappa}))\}]} \tag{A.62}$$

Aligning torque combined slip:

$$M_z' = -tF_y' + M_{zr} + sF_x \tag{A.63}$$

$$t = t(\alpha_{t,eq})) = D_t \cos[C_t \arctan\{B_t\alpha_{t,eq} - E_t(B_t\alpha_{t,eq} - \arctan(B_t\alpha_{t,eq}))\}] \cos\alpha \tag{A.64}$$

$$F'_{y,\gamma=0} = F_y - S_{Vy\kappa} \tag{A.65}$$

$$M_{zr} = M_{zr}(\alpha_{r,eq}) = D_r \cos[\arctan(B_r\alpha_{r,eq})] \cos\alpha \tag{A.66}$$

$$s = \{s_{sz1} + s_{sz2}(F_y/F_{z0}) + (s_{sz3} + s_{sz4}df_z)\gamma\}R_0\lambda_s \tag{A.67}$$

$$\alpha_{t,eq} = \arctan\sqrt{\tan^2\alpha_t + \left(\frac{K_x}{K_y}\right)^2\kappa^2} \cdot sgn(\alpha_t) \tag{A.68}$$

$$\alpha_{r,eq} = \arctan\sqrt{\tan^2\alpha_r + \left(\frac{K_x}{K_y}\right)^2\kappa^2} \cdot sgn(\alpha_r) \tag{A.69}$$

Overturning moment:

$$M_x = R_0F_z\{q_{Sx1}\lambda_{Vmx} + (-q_{Sx2}\gamma + q_{Sx3}F_y/F_{z0})\lambda_{Mx}\} \tag{A.70}$$

Rolling resistance torque:

$$M_y = R_0F_z\{q_{Sy1} + q_{Sy2}F_x/F_{z0} + q_{Sy3}|V_x/V_{ref}| + q_{Sy4}(V_x/V_{ref})^4\} \tag{A.71}$$

$$(q_{Sy1} = 0 \wedge q_{Sy2} = 0) \implies M_y = R_0(S_{Vx} + K_xS_{Hx}) \tag{A.72}$$

B

Normalizing the Magic Formula with Regard to Nominal Load

The outputs of the MF tire model depend on the nominal load F_{z0}, however, the choice of nominal load affects the values of several of the underlying coefficients. In order to improve the ability to compare tire models it is beneficial to normalize the nominal load such that all tire models have the same one. How this is done is described mathematically below.

The underlying equations that depend on the nominal load are structured as follows:

$$y = g(F; F_{z0}) \cdot f(F)$$

where f is a function of F that does not depend on F_{z0} and g is a function of F and F_{z0}.

In order to normalize the MF model, we have to find h such that:

$$h(F; F'_{z0}) = g(F; F_{z0}), \quad \forall F$$

where F'_{z0} is the new nominal load.

There are a few different cases of g that appear in the MF model.

$$g = p_1 + p_2 \frac{F - F_{z0}}{F_{z0}} + p_3 \left(\frac{F - F_{z0}}{F_{z0}} \right)^2 \quad [46]$$

Assume,

$$h = p'_1 + p'_2 \frac{F - F'_{z0}}{F'_{z0}} + p'_3 \left(\frac{F - F'_{z0}}{F'_{z0}} \right)^2$$

then

$$p'_1 + p'_2 \frac{F - F'_{z0}}{F'_{z0}} + p'_3 \left(\frac{F - F'_{z0}}{F'_{z0}} \right)^2 = p_1 + p_2 \frac{F - F_{z0}}{F_{z0}} + p_3 \left(\frac{F - F_{z0}}{F_{z0}} \right)^2$$

Let $F = F'_{z0}$, then

$$p'_1 = p_1 + p_2\frac{F'_{z0} - F_{z0}}{F_{z0}} + p_3\left(\frac{F'_{z0} - F_{z0}}{F_{z0}}\right)^2$$

Taking the first order derivatives with regard to F:

$$\frac{p_2}{F_{z0}} + p_3\frac{2F - 2F_{z0}}{F_{z0}^2} = \frac{p'_2}{F'_{z0}} + p'_3\frac{2F - 2F'_{z0}}{F'^2_{z0}}$$

Let $F = F'_{z0}$, then

$$p'_2 = p_2\frac{F'_{z0}}{F_{z0}} + 2p_3\frac{F'_{z0}(F'_{z0} - F_{z0})}{F_{z0}^2}$$

Second order derivative:

$$\frac{p_3}{F_{z0}^2} = \frac{p'_3}{F'^2_{z0}} \iff p'_3 = p_3\left(\frac{F'_{z0}}{F_{z0}}\right)^2$$

Thus,

$$p'_1 = p_1 + p_2\frac{F'_{z0} - F_{z0}}{F_{z0}} + p_3\left(\frac{F'_{z0} - F_{z0}}{F_{z0}}\right)^2$$

$$p'_2 = p_2\frac{F'_{z0}}{F_{z0}} + 2p_3\frac{F'_{z0}(F'_{z0} - F_{z0})}{F_{z0}^2}$$

$$p'_3 = p_3\left(\frac{F'_{z0}}{F_{z0}}\right)^2$$

Consider the following:

$$K_x = F_z(p_{Kx1} + p_{Kx2}df_z)\exp(p_{kx3}df_z)\lambda_{Kx} \quad [46]$$

then,

$$g = (p_1 + p_2df_z)exp(p_3df_z) = (p_1 + p_2\frac{F - F_{z0}}{F_{z0}})exp(p_3\frac{F - F_{z0}}{F_{z0}}) =$$

$$\frac{p_1 + p_2\frac{F - F_{z0}}{F_{z0}}}{exp(p_3)}exp(p_3\frac{F}{F_{z0}})$$

Set $g = h$:

$$\frac{p'_1 + p'_2\frac{F-F'_{z0}}{F'_{z0}}}{exp(p'_3)}exp(p'_3\frac{F}{F'_{z0}}) = \frac{p_1 + p_2\frac{F-F_{z0}}{F_{z0}}}{exp(p_3)}exp(p_3\frac{F}{F_{z0}})$$

Exponents needs to be equal:

$$exp(p_3\frac{F}{F_{z0}}) = exp(p'_3\frac{F}{F'_{z0}}) \iff p_3\frac{F}{F_{z0}} = p'_3\frac{F}{F'_{z0}} \iff p'_3 = p_3\frac{F'_{z0}}{F_{z0}}$$

This means that

$$\frac{p_1 + p_2\frac{F-F_{z0}}{F_{z0}}}{exp(p_3)} = \frac{p'_1 + p'_2\frac{F-F'_{z0}}{F'_{z0}}}{exp(p'_3)}$$

Let $F = F'_{z0}$, then

$$\frac{p_1 + p_2\frac{F'_{z0}-F_{z0}}{F_{z0}}}{exp(p_3)} = \frac{p'_1}{exp(p'_3)} \iff p'_1 = (p_1 + p_2\frac{F'_{z0} - F_{z0}}{F_{z0}})\frac{exp(p'_3)}{exp(p_3)} =$$

$$(p_1 + p_2\frac{F'_{z0} - F_{z0}}{F_{z0}})exp(p_3\frac{F'_{z0} - F_{z0}}{F_{z0}})$$

Solving for p_2

$$\frac{p_1 + p_2\frac{F-F_{z0}}{F_{z0}}}{exp(p_3)} = \frac{p'_1 + p'_2\frac{F-F'_{z0}}{F'_{z0}}}{exp(p'_3)}$$

$$\iff$$

$$p'_2 = (p_2\frac{F'_{z0}}{F_{z0}})exp(p_3\frac{F'_{z0} - F_{z0}}{F_{z0}})$$

Thus

$$p'_1 = (p_1 + p_2\frac{F'_{z0} - F_{z0}}{F_{z0}})exp(p_3\frac{F'_{z0} - F_{z0}}{F_{z0}})$$

$$p'_2 = (p_2\frac{F'_{z0}}{F_{z0}})exp(p_3\frac{F'_{z0} - F_{z0}}{F_{z0}})$$

$$p'_3 = p_3\frac{F'_{z0}}{F_{z0}}$$

In some cases g can be as simple as:

$$g = pF_{z0} \quad [46]$$

then,

$$F_{z0}p = F'_{z0}p' \iff p' = p\frac{F_{z0}}{F'_{z0}}$$

Consider

$$D = F_z(q_1 + q_2 df_z)(1 + q_3\gamma + q_4\gamma^2)(R_0/F_{z0}) \quad [46]$$

then,

$$g = \frac{q_1 + q_2\frac{F-F_{z0}}{F_{z0}}}{F_{z0}}$$

Let $h = g$

$$\frac{q'_1 + q'_2\frac{F-F'_{z0}}{F'_{z0}}}{F'_{z0}} = \frac{q_1 + q_2\frac{F-F_{z0}}{F_{z0}}}{F_{z0}}$$

Let $F = F'_{z0}$, then

$$\frac{q_1 + q_2\frac{F'_{z0}-F_{z0}}{F_{z0}}}{F_{z0}} = \frac{q'_1}{F'_{z0}}$$

$$\iff$$

$$q'_1 = F'_{z0}\frac{q_1 + q_2\frac{F'_{z0}-F_{z0}}{F_{z0}}}{F_{z0}} = q_1\frac{F'_{z0}}{F_{z0}} + q_2\frac{F'_{z0}(F'_{z0} - F_{z0})}{F^2_{z0}}$$

Solve for q_2

$$\frac{q_1\frac{F'_{z0}}{F_{z0}} + q_2\frac{F'_{z0}(F'_{z0}-F_{z0})}{F^2_{z0}} + q'_2\frac{F-F'_{z0}}{F'_{z0}}}{F'_{z0}} = \frac{q_1 + q_2\frac{F-F_{z0}}{F_{z0}}}{F_{z0}}$$

$$\iff$$

$$q'_2 = q_2\left(\frac{F'_{z0}}{F_{z0}}\right)^2$$

Thus

X

$$q_1' = q_1 \frac{F_{z0}'}{F_{z0}} + q_2 \frac{F_{z0}'(F_{z0}' - F_{z0})}{F_{z0}^2}$$

$$q_2' = q_2 \left(\frac{F_{z0}'}{F_{z0}} \right)^2$$

C

Expressing the Magic Formula as a Power Series

The Maclaurin series of sin and arctan are expressed as follows:

$$\sin z = z - \frac{z^3}{3!} + \frac{z^5}{5!} + \mathcal{O}(7)$$

$$\arctan z = z - \frac{z^3}{3} + \frac{z^5}{5} + \mathcal{O}(7)$$

Let

$$f(x) = \frac{MF(x)}{D}$$

such that,

$$f(x) = \sin(C \arctan(Bx - E(Bx - \arctan(Bx)))) \quad [46]$$

Express the innermost function of $f(x)$ as a Maclaurin series:

$$\arctan(Bx) = Bx - \frac{(Bx)^3}{3} + \frac{(Bx)^5}{5} + \mathcal{O}(7)$$

Further,

$$Bx - E(Bx - \arctan(Bx)) = Bx - \frac{E(Bx)^3}{3} + \frac{E(Bx)^5}{5} + \mathcal{O}(7)$$

Express the outer arctan as a Maclaurin series:

$$\arctan(Bx - E(Bx - \arctan(Bx))) = Bx - \frac{(E+1)(Bx)^3}{3} + \frac{(8E+3)(Bx)^5}{15} + \mathcal{O}(7)$$

Then,

$$f(x) =$$

$$CBx - \frac{C(C^2 + 2E + 2)(Bx)^3}{6} + \frac{C(C^4 + C^2(20E + 20) + 64E + 24)(Bx)^5}{120} + \mathcal{O}(7)$$

By using the first, third and fifth derivative of the Maclaurin series, it is possible to approximate the B, C and E coefficients of the MF tire model. Since D is the

amplitude, it is possible to characterize the complete behavior of the model with knowing the forces in a small interval close to 0, as well as the amplitude at the peak. This is done as follows:

$$k_1 = f'(0) = BC$$
$$k_2 = f'''(0) = -B^3C(C^2 + 2E + 2)$$
$$k_3 = f^{(5)}(0) = B^5C(C^4 + C^2(20E + 20) + 64E + 24)$$

$$\begin{cases} k_1 = BC \\ k_2 = -B^3C(C^2 + 2E + 2) \\ k_3 = B^5C(C^4 + C^2(20E + 20) + 64E + 24) \end{cases}$$

$$B = \frac{k_1}{C}$$

$$k_2 = -\frac{k_1^3(C^2 + 2E + 2)}{C^2}$$
$$k_3 = \frac{k_1^5(C^4 + C^2(20E + 20) + 64E + 24)}{C^4}$$

$$k_2 = -\frac{k_1^3(C^2 + 2E + 2)}{C^2} \iff C^2k_2 = -k_1^3C^2 - 2k_1^3E - 2k_1^3 \iff$$
$$E = -\frac{C^2k_2 + k_1^3C^2 + 2k_1^3}{2k_1^3}$$

$$k_3 = \frac{k_1^5(C^4 + C^2(20(-\frac{C^2k_2+k_1^3C^2+2k_1^3}{2k_1^3}) + 20) + 64(-\frac{C^2k_2+k_1^3C^2+2k_1^3}{2k_1^3}) + 24)}{C^4}$$
$$\iff C^4(k_3 + 10k_2k_1^2 + 9k_1^5) + C^2(32k_1^2k_2 + 32k_1^5) + 40k_1^5 = 0$$

Let

$$p := \frac{32k_1^2k_2 + 32k_1^5}{k_3 + 10k_2k_1^2 + 9k_1^5}$$

and

$$q := \frac{40k_1^5}{k_3 + 10k_2k_1^2 + 9k_1^5}$$

then

$$C = \begin{cases} \pm\sqrt{-\frac{p}{2} + \frac{\sqrt{p^2-4q}}{2}} \\ \pm\sqrt{-\frac{p}{2} - \frac{\sqrt{p^2-4q}}{2}} \end{cases}$$

However, C is defined to be positive. Thus,

$$C = \begin{cases} \sqrt{-\frac{p}{2} + \frac{\sqrt{p^2-4q}}{2}} \\ \sqrt{-\frac{p}{2} - \frac{\sqrt{p^2-4q}}{2}} \end{cases}$$

The value of C can then be used to calculate B and E.

k_1, k_2 and k_3 can be calculated using finite difference.

D

Approximating the Magic Formula from Key Attributes

The method for approximating the Magic Formula from some key attributes was done as follows:

The key attributes were defined as follows:

$$\begin{aligned}\hat{K} &:= MF'(0)\\ \hat{D} &:= \max_x MF(x)\\ \hat{x} &:= \arg\max_x MF(x)\end{aligned}$$

Since there are not enough data points to fully fit all the coefficients of the MF model, E is assumed to be 0. This yields the following modified version of the MF tire model:

$$f(x) = D\sin(C\arctan(Bx))$$

Then, let

$$D = \hat{D}$$

and

$$B = \frac{\hat{K}}{C\hat{D}}$$

Thus,

$$f(x) = \hat{D}\sin\left(C\arctan\left(\frac{\hat{K}}{C\hat{D}}x\right)\right)$$

Then,

$$\hat{D} = f(\hat{x}) = \hat{D}\sin\left(C\arctan\left(\frac{\hat{K}}{C\hat{D}}\hat{x}\right)\right) \iff 1 = \sin\left(C\arctan\left(\frac{\hat{K}}{C\hat{D}}\hat{x}\right)\right)$$

Solve for C, which then can be used to calculate B.