UNIVERSITY OF
GOTHENBURG

Evidence based training in cross-country skiing

Predicting the force generated by the skier

Master's thesis in Mathematical Statistics

ELIJAH FERREIRA

Department of Mathematical Sciences
UNIVERSITY OF GOTHENBURG
Gothenburg, Sweden 2019

Master's thesis 2019

Evidence based training in cross-country skiing

Predicting the force generated by the skier

ELIJAH FERREIRA

UNIVERSITY OF
GOTHENBURG

Department of Mathematical Sciences
Division of Mathematical Statistics
University of Gothenburg
Gothenburg, Sweden 2019

Evidence based training in cross-country skiing
Predicting the force generated by the skier
ELIJAH FERREIRA

Supervisor: Professor Rebecka Jörnsten, Department of Mathematical Sciences
Examiner: Umberto Picchini, Department of Mathematical Sciences

Master's Thesis 2019
Department of Mathematical Sciences
Division of Mathematical Statistics
University of Gothenburg
SE-412 96 Gothenburg

Evidence based training in cross-country skiing
Predicting the force generated by the skier
ELIJAH FERREIRA
Department of Mathematical Sciences
University of Gothenburg

Abstract

Evidence based training has been around for for while, where data is collected and pre-defined measures are used to evaluate the training session. *Skisens AB* are presenting new methods to evaluate a session which can be compared between sessions in a fair way, without having outside factors influencing the results by, measuring the force generated by the skier. Measuring the force generated involves customized handles that changes the dimensions of the handles and in the extension the ergonomics of the handles. This thesis aims to try to accurately predict the force generated in each stroke from the skier, in order for *Skisens AB* not having to measure the force using the customized handles.
We propose an unsupervised detection algorithm, for detecting when a stroking motion is performed as well as a few model design for achieving the best predictive results.

Acknowledgements

I would like to thank my supervisor Prof. Rebecka Jörnsten for the guidance and advice throughout the process. I would also like thank family and friends that has supported throughout my studies. Lastly, I would also like thank my examiner Prof. Umberto Picchini for taking time to read and review my thesis.

ELIJAH FERREIRA, Gothenburg, September 2019

Contents

List of Figures

List of Tables

1
Introduction

With the technological improvements made in recent years, companies are collecting data about everything possible. Data driven decision making has increased exponentially and many call this day of age the age of data analytics.
This is true even in the world of sports where attaching sensors to athletes and their gear to collect data with the intent of further analysis of performance, risk assessments and improving their training results is something that has been around for a while. Already in about a decade ago ADDIDAS released their football shoes *adizero f50 miCoach* on the commercial market [1]. The shoes has a chip in the sole which tracks the players movement, average speed, top speed, time between steps and many other things. The player or the team could after a training session download the data, analyze it and try to adjust the training to improve the player. Another example comes from the American football league, where each of the 32 teams in the *National Football League* has an analytic department that collects data about their players physical abilities, training performance and uses in further analysis to try to increase the players abilities.

Further, with smartphones and smartwatches even amateur athletes can in real time get indications of their performance during a session. These indications could be in the form of heartbeats per minute, speed or other measurements.
Using data and measurements as evidence of the performance while training has become very wide spread with the help of technology. But the data, methods and metrics can differ between sports. In some sports it is easier to get reliable and important indications while in other it can be a harder.

1.1 Skisens

Skisens is a company founded by researcher and previous students at *Chalmers university of technology* in 2017. Their idea is a tool that will help to improve the methods in evidence based training for cross-country skiing. Today cross-country skiers have a few different options when tracking their performance such as sport clocks, chest belts and smartphone apps [2]. These tools usually keeps track of the skiers distance, time, heartbeat and from the data collected the skier can evaluate the session once finished. The downside with these measurements is that they cannot in a fair way be compared between sessions, since there are many other factors that will affect the data collected such as weather conditions, how well rested the skier is before the session and more.

Skisens tool is a power meter that in real time measures the force generated by the poles and the angle of each pole, all in order to derive the power produced. The power meter is built in to a customized handle that is then mounted on the two poles.

Figure 1.1: The customized handle made by Skisens

Using a power meter, and in the extension effect, as a way of measuring performance has been the a common method used for the past 10-15 years in competitive cycling. But in cross-country skiing there has yet not been any similar tool before, so *Skisens* saw an opportunity to improve the existing methods [3]. Effect is an absolute measurement that is directly connected with the skiers performance, making it possible to compare sessions and track development. An increased ability to generate force translates to either an increased physical ability or better technique used.
Further, with the handle not only is the skier able to get information about the angle of pole and the force but also the velocity and acceleration of the pole. This additional information opens up for the the data to give indications of reasons that could limit the skiers performance in certain situations, and in the extension offer preferable adjustments to the technique.

1.2 Objective

To be able to measure the force it requires that the custom made handle has a loading cell. To fit the loading cell in the handle the dimensions of the handle has to be changed in a way that is not ergonomic for the skier. This could in theory mean that the technique of the skier is changed due to adjustments made to the new handles. Moreover, the loading cell is an expensive sensor in comparison to the IMU sensor that measures the angle and movement of the pole.
Being able to calculate the force without measuring it is of the essence for *Skisens*. Therefor the objective of this thesis will be to try to predict the force generated by the skier from all the other information provided in the data.
This thesis in structured in the following way that in section 2 an overview of the sensor data from *Skisens* is presented. Also, the construction of the features that will be used for fitting a model and making predictions is presented and motivated. Section 3 gives an theoretical background to the statistical models considered for obtaining predictions. Section 4 and 5 presents the models that was used and the results obtained from these models. Lastly section 6 discusses the findings, possible limitations and drawbacks, and future work.

2

Data Overview

The data used for this thesis is recorded by Skisens AB during two session when *Johan Högstrand* was out skating on roller skates. The first of these two sessions is recorded when he was skating around in the city of Gothenburg, while the second one is from a skating session in the community of *Sätila* close to the northern lake of Lyngen. The data is recorded from the sensors mounted on the handles of the poles and on the skater. During the two sessions a total of four different techniques was used, `Double`, `Gear2`, `Gear3` and `Gear4`. After a session the data is cleaned and partitioned into several *CSV*-files by *Skisens*, each file contains the data from a specific session and one style executed in consecutive time. For example, during the skate session in *Sätila* on three different occasions during the session, *Johan* used the technique *Double*. Each of these three occasions are in separate files. So even though we are recording the time, the files does not indicate in which order the techniques was performed. This yields that effects like long term fatigue is something that cannot be part of the analysis.

Moreover, only the data from the techniques `Double` and `Gear3` will be used due to the extensive work of creating a general function that is able to work for all four techniques. `Double` was chosen because most of the data available is from skating using that technique, moreover `Gear3` was included due to similarities in technique as `Double` when skating.
The methodology of predicting force for the two other techniques will be the same as for the two that are included in this thesis.

2.1 Description of the data

The information recorded from each session will be described in this section. Table 2.1 gives a small overview of the all the measurements collected together with the unit they are measured in.

Measurements	Unit
`Time`	[s]
`Left force`	[N]
`Left angle`	[deg]
`Left angular velocity vector`	[rad/s]
`Left acceleration vector`	$[m/s^2]$
`Right force`	[N]
`Right angle`	[deg]
`Right angular velocity vector`	[rad/s]
`Right acceleration vector`	$[m/s^2]$
`Flat eastward position`	[m]
`Flat northward position`	[m]
`Altitude`	[m]
`Speed`	[m/s]

Table 2.1: Description of the data recorded from each session from the sensors in the handles and on the skater. The sensors records measurements each 0.2 second.

Note that the time is relative to each file, i.e it starts over from 0 in each file. Also, the sensor measures each variable every 0.2 seconds. The coordinate system for the vectors of angular velocity and acceleration are described in table 2.2.

Axis	Description
`First Axis`	Pointing right (orthogonal to pole)
`Second Axis`	Pointing down (parallel to pole)
`Third Axis`	Pointing forward (orthogonal to pole)

Table 2.2: Description of the coordinate system for the data vectors acceleration and velocity. The coordinate system for the vectors of acceleration and angular velocity is relative to the pole.

In table 2.3 the abbreviations that will be used throughout this thesis for the measured variables are presented. These abbreviations are present in figures as well as in the text.

Abbreviation	Description
fL	force from the left pole
thL	Angle of the left pole
w1L	Velocity of the left pole in the first axis
w2L	Velocity of the left pole in the second axis
w3L	Velocity of the left pole in the third axis
a1L	Acceleration of the left pole in the first axis
a2L	Acceleration of the left pole in the second axis
a3L	Acceleration of the left pole in the third axis
z	The altitude of the skier
v	The speed of the skier

Table 2.3: Abbreviation of the variables collected by *Skisens* that are used from this point forward in the report. The abbreviations of the variables related to the right pole are not present in this table with the suffix, i.e instead of ending with `L` they end with a `R`.

Below is a small sample of how each variable corresponding to the right pole looks like for the style *Double*. The corresponding figures for the left pole is shown in section *Appendix A* but in general they have the same behaviour as its right pole equivalent. The same is true for the variables when `Gear3` is used.

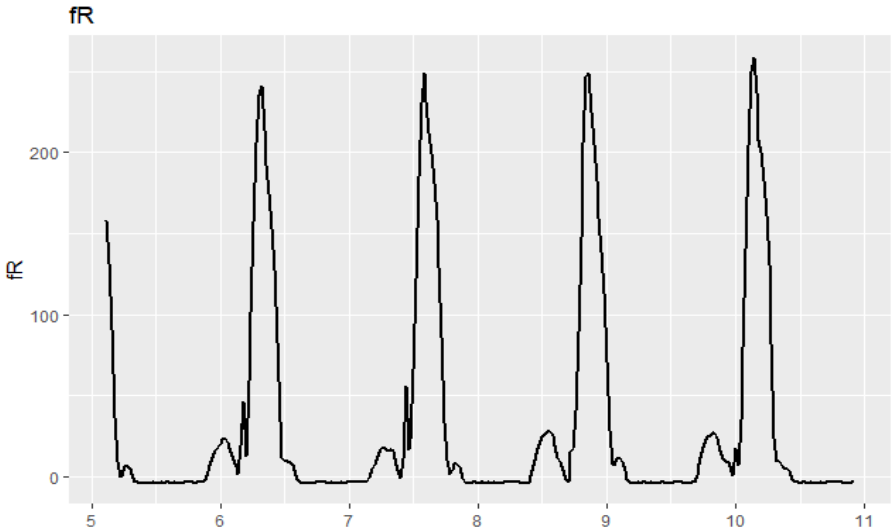

Figure 2.1: A sample of the force over time, measured in seconds, generated by the right pole during the session in *Sätila*.

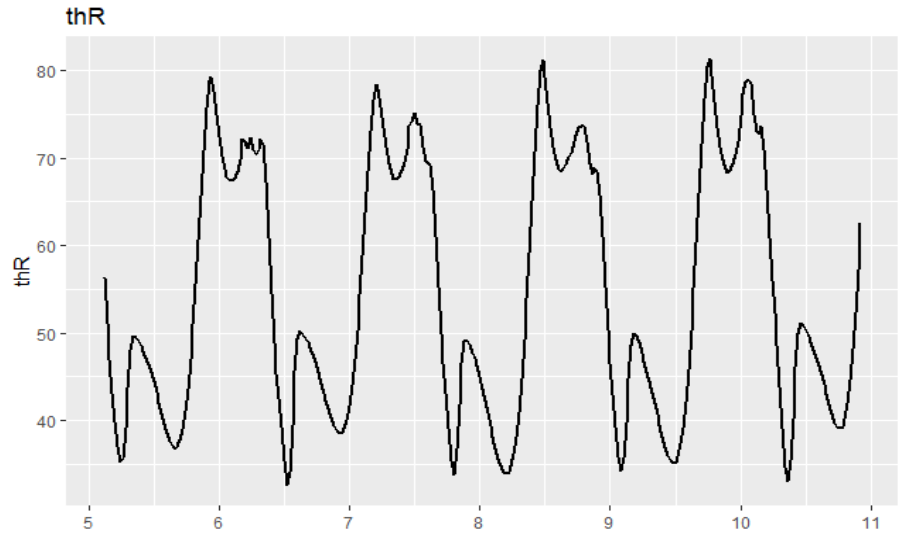

Figure 2.2: A sample of the angle of the right pole over time, measured in seconds, during the session in *Sätila*.

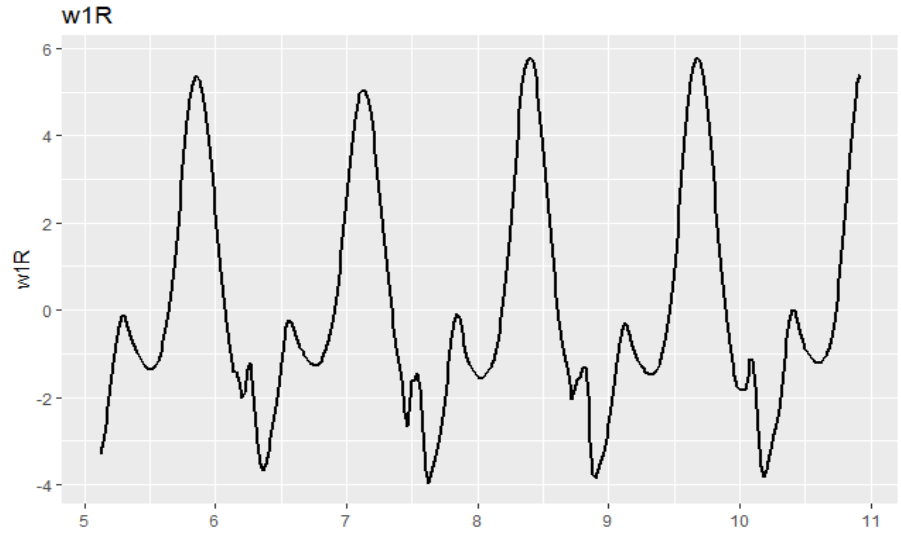

Figure 2.3: A sample of the velocity of the right pole in the first axis over time, measured in seconds, during the session in *Sätila*. The first axis of the velocity is pointing right and is orthogonal to the pole.

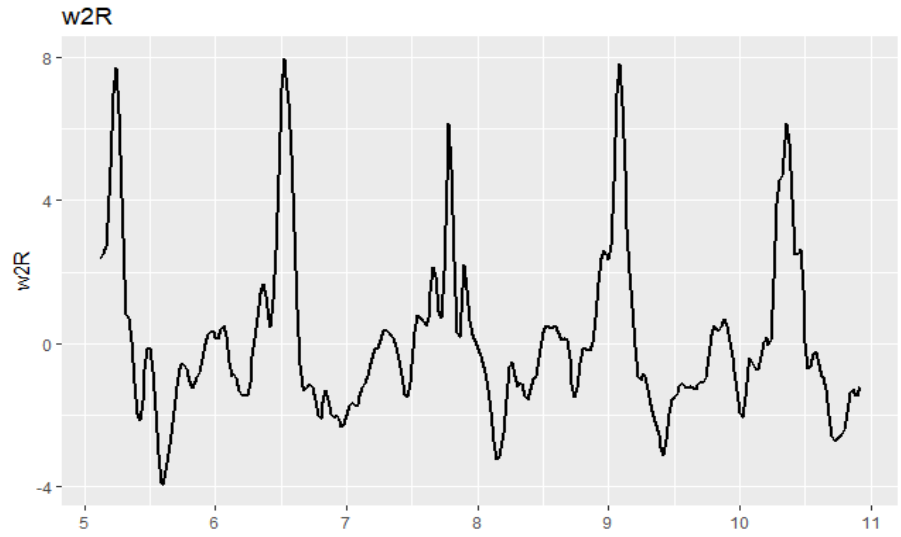

Figure 2.4: A sample of the velocity of the right pole in the second axis over time, measured in seconds, during the session in *Sätila*. The second axis of the velocity is pointing downwards and is parallel to the pole.

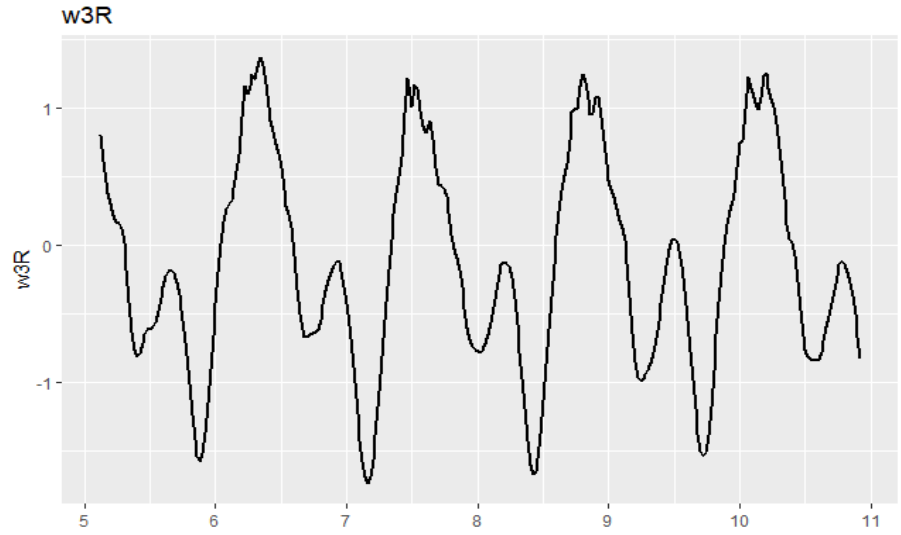

Figure 2.5: A sample of the velocity of the right pole in the third axis over time, measured in seconds, during the session in *Sätila*. The third axis of the velocity is pointing forward and is orthogonal to the pole.

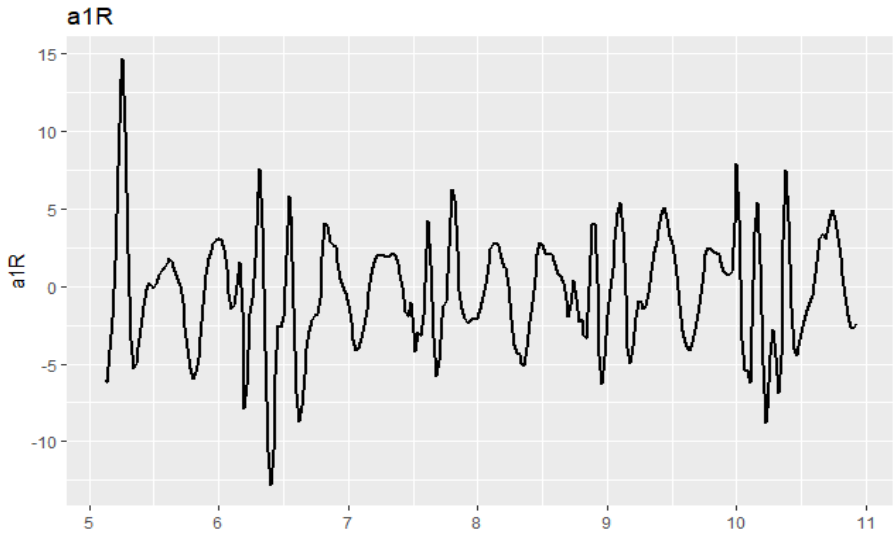

Figure 2.6: A sample of the acceleration of the right pole in the first axis over time, measured in seconds, during the session in *Sätila*. The first axis of the acceleration is pointing right and is orthogonal to the pole.

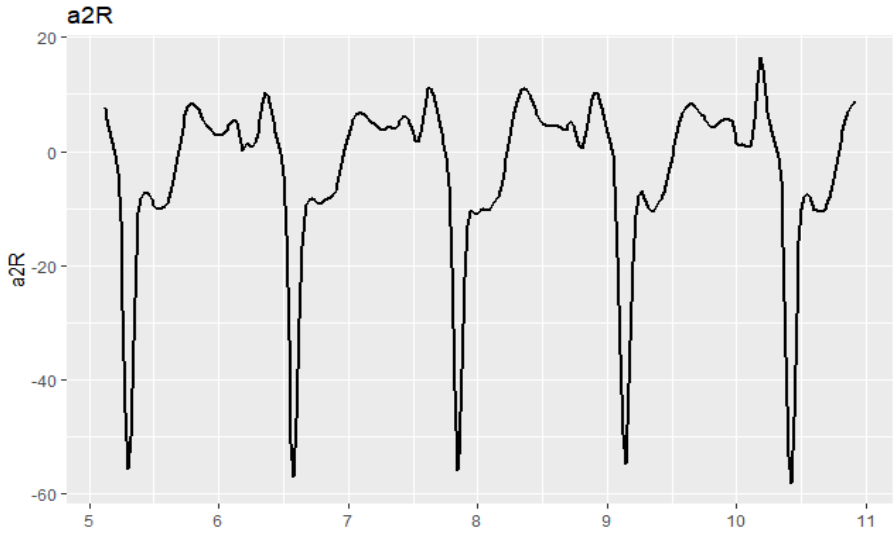

Figure 2.7: A sample acceleration of the right pole in the second axis over time, measured in seconds, during the session in *Sätila*. The second axis of the acceleration is pointing downwards and is parallel to the pole.

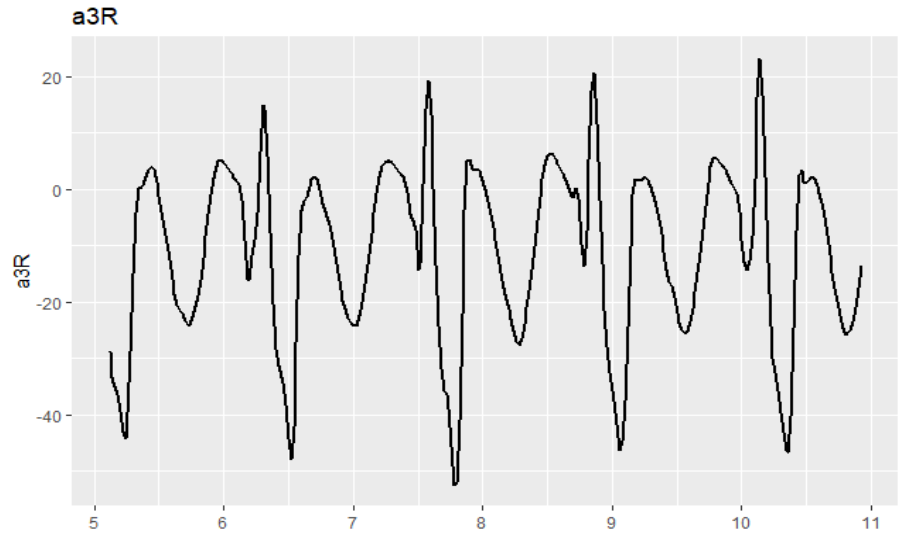

Figure 2.8: A sample of the acceleration of the right pole in the third axis over time, measured in seconds, during the session in *Sätila*. The third axis of the acceleration is pointing forward and is orthogonal to the pole.

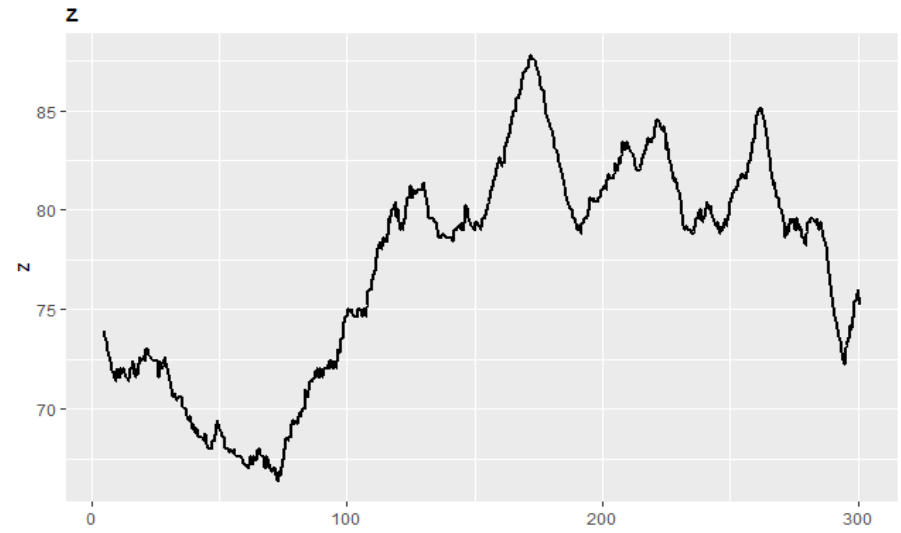

Figure 2.9: A sample of the altitude over a larger period of time, measured in seconds, during the session in *Sätila*.

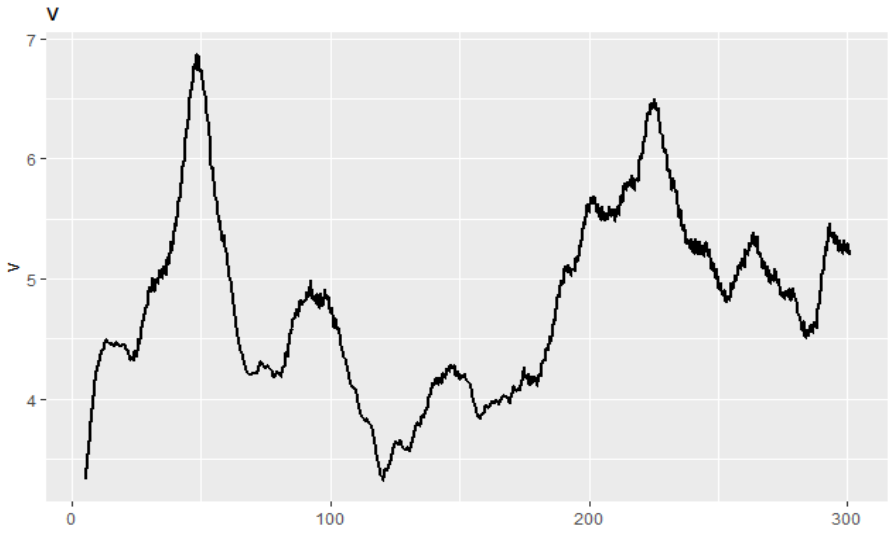

Figure 2.10: The speed of the skier over a larger period of time, measured in seconds, during the session in *Sätila*.

2.2 Unsupervised event detection

The data from each session is collected every 0.2 seconds, resulting in that we have temporal information about each record in the data. This temporal information allows for example the ability to track technique or performance changes due to fatigue. The objective of this thesis is to predict the force generated from each pole, the force is non-present (baseline is 0 [N]) most of the times, except for when the skier is performing a stroke movement. Keeping the temporal information in the data and building a statistical model for predicting the force at such a granular level as for each recorded measure would result in that for the most part the model would predict 0 force.

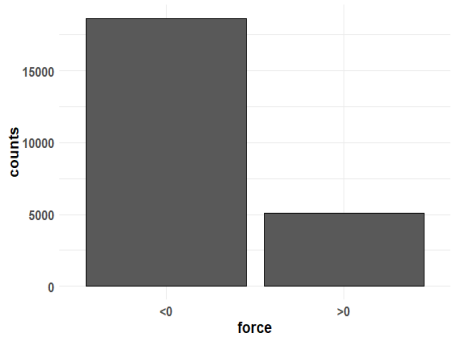

Figure 2.11: Counts of how often the force is less or equal to zero or larger than zero for one data set from the technique `Double`. From the figure it becomes obvious that the force is non-present most of the time.

Figure 2.11 displays the number of occurrences of strictly positive force vs less than zero force, and with this figure as motivation, I will consider the time period of when

the force deviating from its baseline until it comes back as an event. Meaning an event will be the time period when the skater is performing a stroking movement. So instead of predicting the force at each record, I will try to predict the overall force generated over the time period of an event. This will yield that we loose the temporal information at the most granular level but is still able to keep it on an event level.

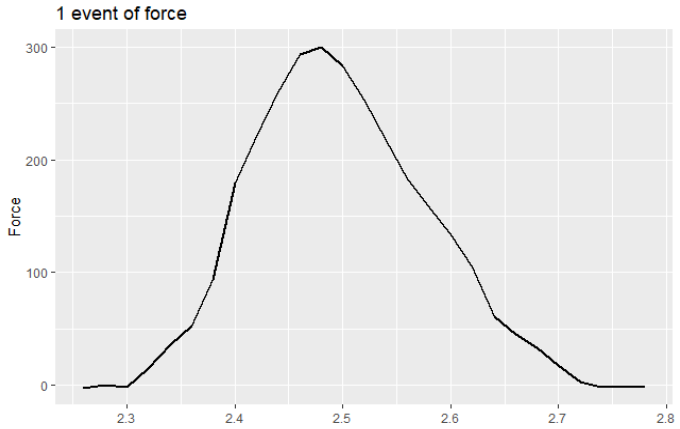

Figure 2.12: A sample of how an event in force looks like for a pole.

Next let's see how the different variables look like when we have an event in force. The red colored parts of the curves corresponds to having an event in force for the left pole. Only figures of the variables for the left pole are presented, the figures for the variables corresponding to the right pole can be found in *Appendix A* figure A.1 - A.7.

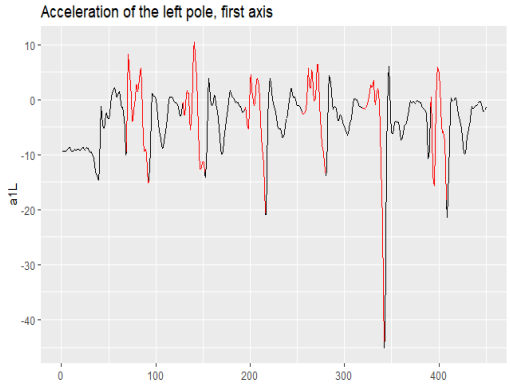

Figure 2.13: A sample of the acceleration of the left pole in the first axis. The red colored parts of the curve is the time periods when there is an event in force for the left pole, i.e the skier is performing an stroking motion and generates positive force.

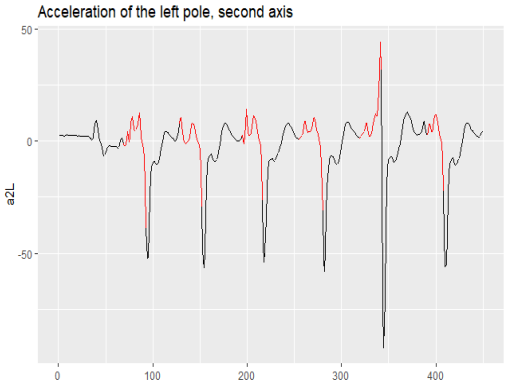

Figure 2.14: A sample of the acceleration of the left pole in the second axis. The red colored parts of the curve is the time periods when there is an event in force for the left pole, i.e the skier is performing an stroking motion and generates positive force.

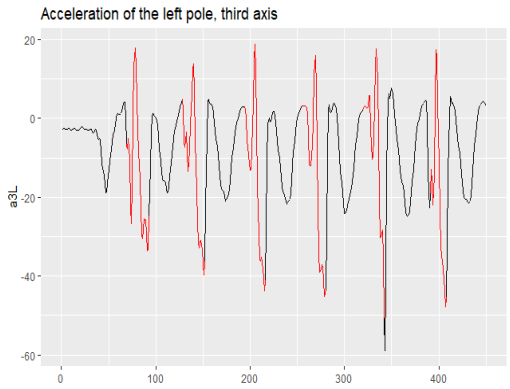

Figure 2.15: A sample of the acceleration of the left pole in the third axis. The red colored parts of the curve is the time periods when there is an event in force for the left pole, i.e the skier is performing an stroking motion and generates positive force.

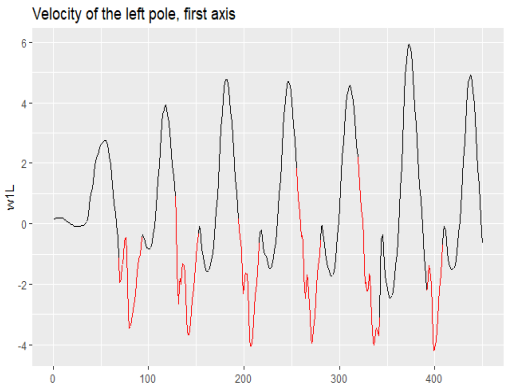

Figure 2.16: A sample of the velocity of the left pole in the first axis. The red colored parts of the curve is the time periods when there is an event in force for the left pole, i.e the skier is performing an stroking motion and generates positive force.

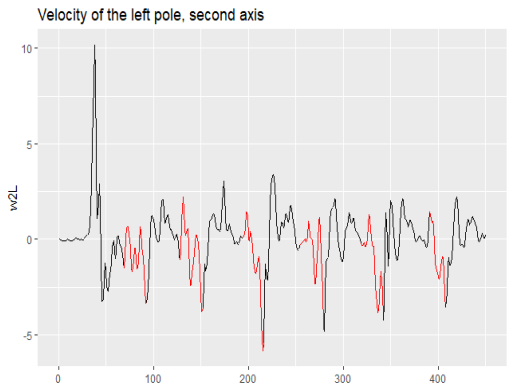

Figure 2.17: A sample of the velocity of the left pole in the second axis. The red colored parts of the curve is the time periods when there is an event in force for the left pole, i.e the skier is performing an stroking motion and generates positive force.

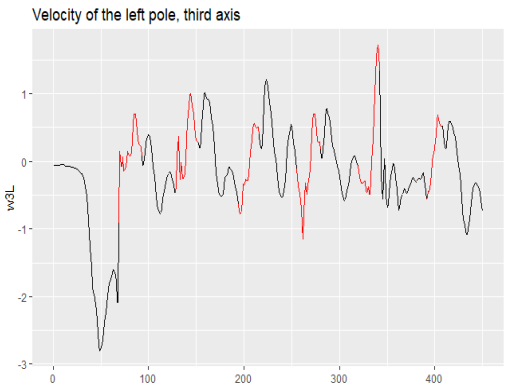

Figure 2.18: A sample of the velocity of the left pole in the third axis. The red colored parts of the curve is the time periods when there is an event in force for the left pole, i.e the skier is performing an stroking motion and generates positive force.

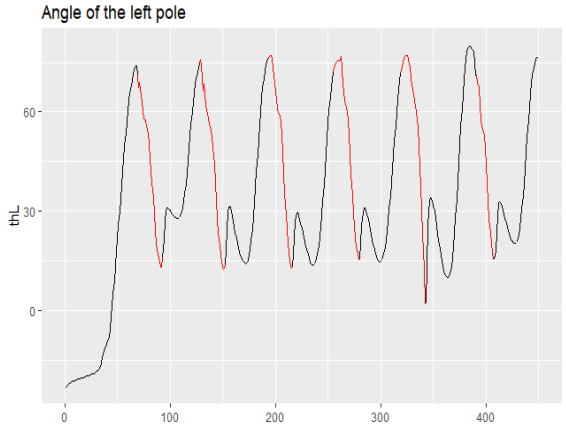

Figure 2.19: A sample of the angle of the left pole. The red colored parts of the curve is the time periods when there is an event in force for the left pole, i.e the skier is performing an stroking motion and generates positive force.

From figures 2.13 - 2.19 it becomes clear that each variable experiences the same behaviour when there is an event force, this knowledge will be the basis for construction a function that identifies an event in force given the data from the sensors.

2.2.1 Identifying an event

Having a repeated pattern in the data for when the force is positive is useful for constructing a detection algorithm to identify this time period. Some of the variables in the data has a more clear pattern than others and these variables are the ones that will be used in the algorithm. For instance, from figure 2.16 that display's the behaviour of `w1L` it clear that before an event in force this variable is experiencing some kind of event by entering a valley. These patterns in the observed data can be seen as events in those variables, and with the help of identifying events in those variables the hope is to be able to identify events in force.
We will now take a closer look at the behaviour of the force when there is an event in the variables `w1L/R, a1L/R, a2L/R, a3L/R`. It can be seen in figure 2.20 below that during an event in `w1L` the curve in force is about to peak and is going back to its baseline, i.e an event in `w1L` corresponds to capturing the end half of an event in force.

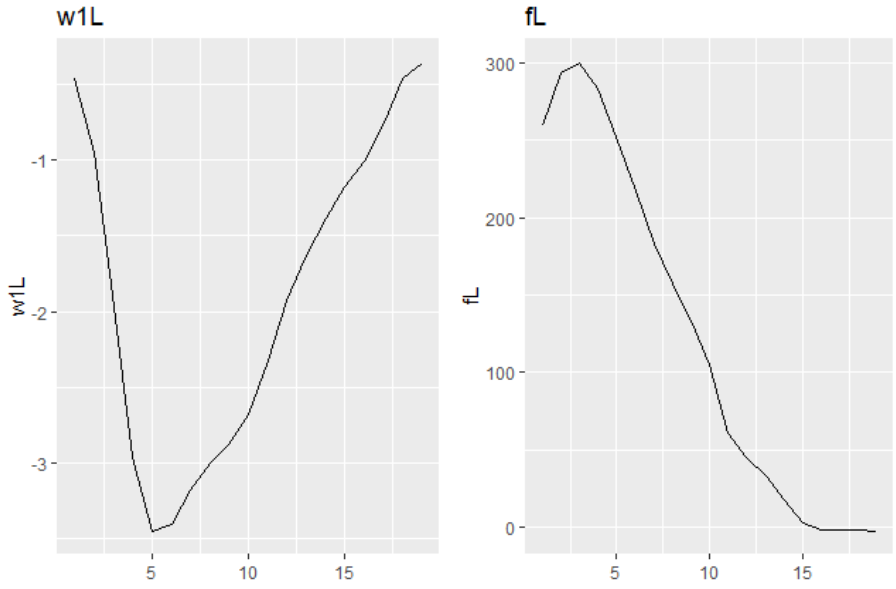

Figure 2.20: left figure: Identified event in `w1L`. Right figure: The corresponding curve for force during the same time period.

Looking at the force curve during events in `a1L` and `a2L`, figure 2.22 and 2.23, the time for the peak value of events in those two variables seems to be good indications of the end of an event in force. Moreover, an event `a3L`, like `w1L`, also seems to capture the second half of an event in force for the left pole. A stroking motion from the skier has approximately the same duration for each motion, so if the start of an event in `w1L` and `a3L` are shifted with an constant c, the idea is that this should be a good indication of the start of the event in force for the left pole.

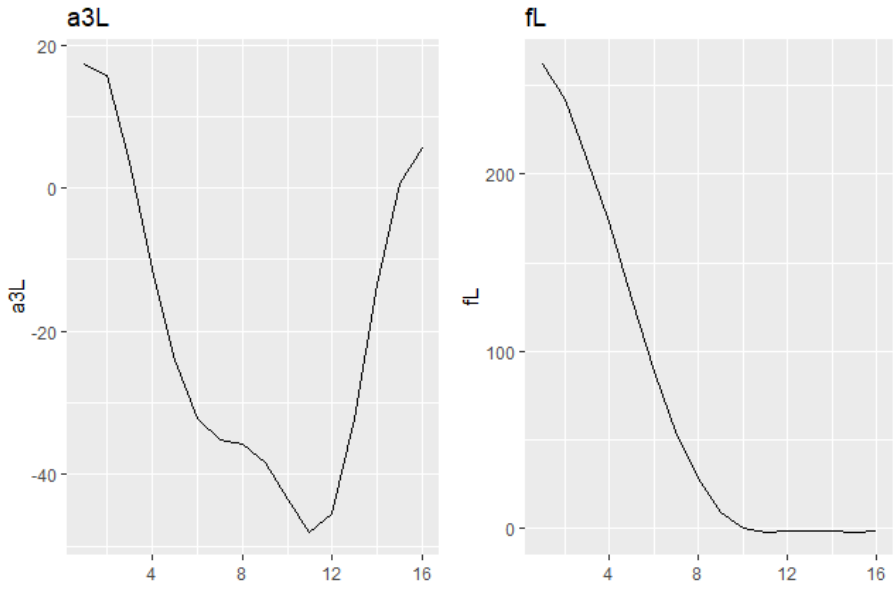

Figure 2.21: Left figure: Identified event in `a3L`. Right figure: The corresponding curve for the force left during the event

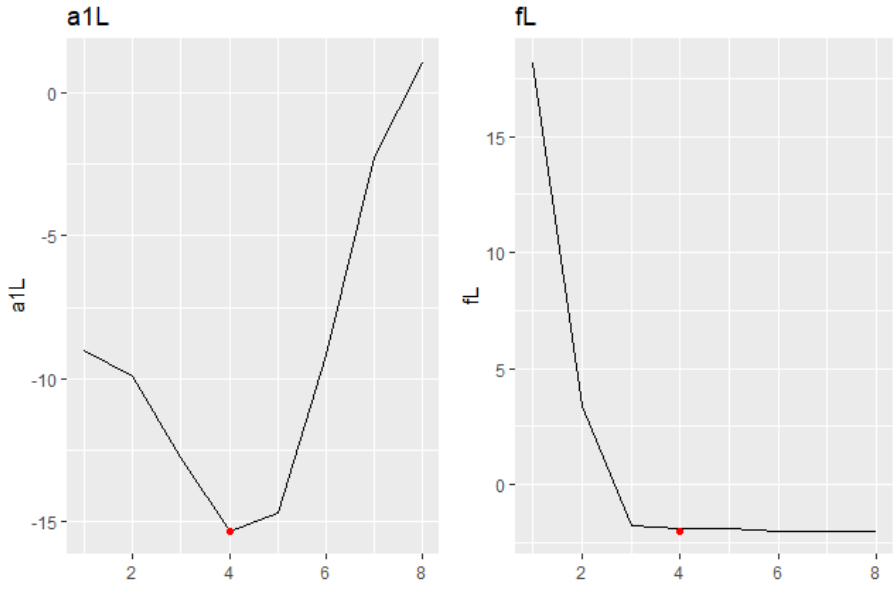

Figure 2.22: Left figure: Identified event in `a1L` where the red dot indicates the peak of the event. Right figure: The corresponding curve for the force left during the event where the red dot indicates the time of the peak value for the event in `a1L`.

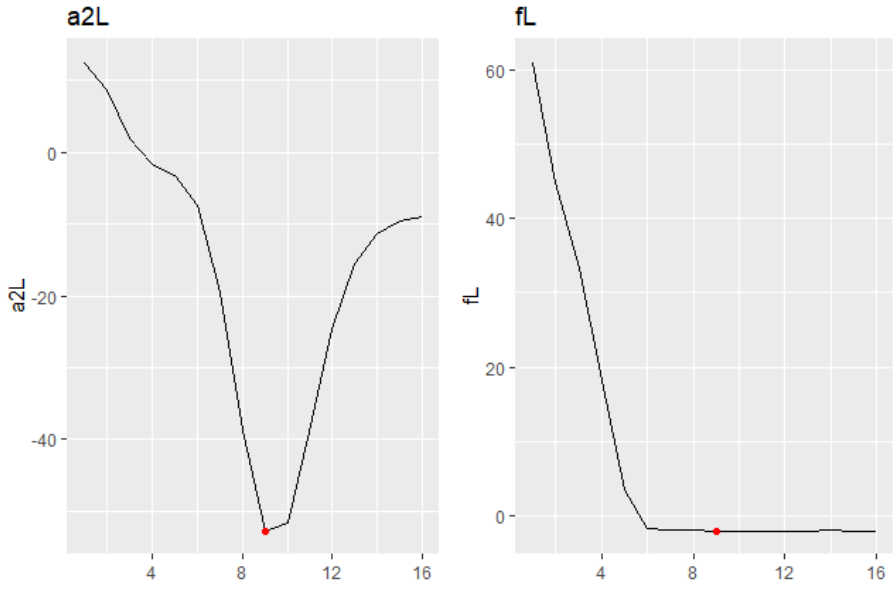

Figure 2.23: Left figure: Identified event in `a2L` where the red dot indicates the peak of the event. Right figure: The corresponding curve for the force left during the event where the red dot indicates the time of the peak value for the event in `a2L`

To my help to find events in these variables the function *findpeaks* [3] from the R-library *pracma* was used. The function took as input the data, a threshold value for what should be considered a peak and a minimum distance to the next peak. It

returned the start time, end time, index of peak value and the peak value of each event found.
Even though the data seems to be fairly regular, changes in the surroundings of the skier makes it hard to set a global value for the threshold parameter that would work over a longer period of time.
So instead of setting a global value for the threshold in the detection algorithm, the data was fed in chunks to the algorithm where a sliding window was applied and in each window the maximum value was returned. The input threshold value for the data chunk was then set to a fraction of the minimum of all the max values returned from applying the sliding window. This procedure assumes that the skier is skiing at all times throughout the data chunk that was fed. Otherwise there is a possibility of a window only containing data from when the skier was standing still, resulting in the minimum max value being the baseline value or a unusually small value for the variable, see figure 2.25.

So before identifying events in these variables, idle periods of the skier needs to be removed. An idle period was assumed to be a period of over 2 minutes without any stroking motion. To identify idle periods, a sliding window was applied over the variable `a2L` of the data chunk and if the maximum value of the variable in the window deviated enough from the global maximum of the data chunk then that indicated that we had an period of when the skier was idle. `a2L` was the most obvious variable where there was an active or idle period. Figure 2.24 illustrates `a2L` during active period and an idle period, and it becomes clear when the idle period is.

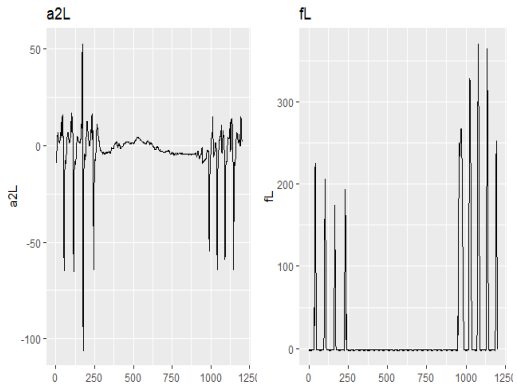

Figure 2.24: Left figure: A sample of `a2L` during active and idle period. Right figure: The corresponding `fL` in that time period. Its clear how there is hardly any activity in `a2L` when there is no force.

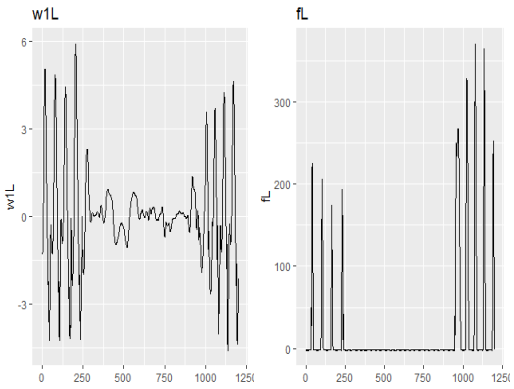

Figure 2.25: Left figure: A sample of `w1L` during active and idle period. Right figure: The corresponding `fL` in that time period. `w1L` is not experiencing the same non-activity as `a2L` when the skier is idle, see figure 2.24.

The parameter minimum distance between events for the function *findpeaks* was easier to set as a global value since it is easy to set a lower bound for the time it would physically would be possible to perform two stroke motions. After identifying events in `w1L/R, a1L/R, a2L/R, a3L/R`, the start of an event in force for the left

pole became a weighted average of the start of an event in `w1L` and `a3L`. With the end of the event was set to the weighted average of the end of an event in `w1L` and the peak for an event in `a1L` and `a2L`.
Algorithm 1 and 2 outlines how the the events in force was detected.

Algorithm 1 Pseudo-Algorithm for identifying idle periods

Input: Data chunk $\mathbf{X}$
$\omega \in (0, 1)$
$S_i := \{\mathbf{X_{i_1}}, ..., \mathbf{X_{i_m}}\}$, *i:th* disjoint slice of $\mathbf{X}$ of size m
$m_i \rightarrow$ Maximum of `a2L` in the *i:th* slice over $\mathbf{X}$

1. Compute $M_{\mathbf{X}} :=$ Global max of `a2L` for data $\mathbf{X}$

2. **for** *slices* S_i **do**
   compute m_i
   2.1 **if** $m_i < M_{\mathbf{X}} \cdot \omega$ **then**
      find $I_{start} :=$ last occurrence of `a2L` $\geq M_{\mathbf{X}} \cdot \omega$ from the start of the slice up until `a2L` is equal to m_i.

      I_{end} := first occurrence of `a2L` $\geq M_{\mathbf{X}} \cdot \omega$ from where `a2L` is equal to m_i up until the end of the slice.

      Remove the interval $[I_{start}, I_{end}]$ from S_i
end
3. **return** $\mathbf{X} = \bigcup_j S_j$

Algorithm 2 Pseudo-Algorithm for identifying events in force

Input: Data chunk X
$e_i^{(x)} \rightarrow$ *i:th* identified event for variable x
$S_i^{(x)} \rightarrow$ Start time for the *i:th* event for variable x
$E_i^{(x)} \rightarrow$ End time for the *i:th* event for variable x
$P_i^{(x)} \rightarrow$ Peak time for the *i:th* event for variable x
$D_i^{(x)} \rightarrow$ Duration of the *i:th* event for variable x
$\alpha_x \in (0, 1)$
$\beta_x \in (0, 1)$
$\theta \rightarrow$ minimum duration of event

1. Remove any possible idle periods in X

2. Identify events $e_1^{(\texttt{w1L})}, e_2^{(\texttt{w1L})}, ..., e_n^{(\texttt{w1L})}$ in `w1L` and apply possible static shifts of the start and end of each event. Do the same for the other variables `a1L`, `a2L` and `a3L` related to the left pole.

3. **for** *events i = 1,...,n* **do**
   $S_i^{(\texttt{fL})} = \alpha_{w1L} \cdot S_i^{(\texttt{w1L})} + \alpha_{a3L} \cdot S_i^{(\texttt{a3L})}$
   Set $E_i^{(\texttt{fL})} = \beta_{w1L} \cdot E_i^{(\texttt{w1L})} + \beta_{a1L} \cdot P_i^{(\texttt{a1L})} + \beta_{a2L} \cdot P_i^{(\texttt{a2L})}$
end
4. Identify events $e_1^{(\texttt{w1R})}, e_2^{(\texttt{w1R})}, ..., e_m^{(\texttt{w1R})}$ in `w1R` and the same for the other variables `a1R`, `a2R` and `a3R` related to the right pole.

5. **for** *events i = 1,...,m* **do**
   $S_i^{(\texttt{fR})} = \alpha_{w1R} \cdot S_i^{(\texttt{w1R})} + \alpha_{a3R} \cdot S_i^{(\texttt{a3R})}$
   Set $E_i^{(\texttt{fR})} = \beta_{w1R} \cdot E_i^{(\texttt{w1R})} + \beta_{a1R} \cdot P_i^{(\texttt{a1R})} + \beta_{a2R} \cdot P_i^{(\texttt{a2R})}$
end
Now if the algorithm found more events for one pole than the other, this needs to be taken care off.

6. **if** $n < m$ **then**
   6.1 Identify which of the m events for the right pole was not identified for the left pole $\{m_{k_1}, ..., m_{k_m}\}$
   for $l \in \{m_{k_1}, ..., m_{k_m}\}$ **do**
      Set $S_l^{(fL)} = S_l^{(fR)}$
      and $E_l^{(fL)} = E_l^{(fR)}$.
   end
else if $n > m$ **then**
   6.2 Identify which of the n events for the left pole was not identified for the right pole $\{n_{k_1}, ..., n_{k_n}\}$
   for $l \in \{n_{k_1}, ..., n_{k_n}\}$ **do**
      Set $S_l^{(fR)} = S_l^{(fL)}$
      and $E_l^{(fR)} = E_l^{(fL)}$.
   end
At this point, N events has been detected. This last step is to make sure an event is not unusually short.

7. **for** *events i = 1,...,N* **do**
   compute

   $D_i^{fL} = E_i^{fL} - S_i^{fL}$
   $D_i^{fR} = E_i^{fR} - S_i^{fR}$
   compute $\delta_i = \frac{|D_i^{(fL)} - D_i^{(fR)}|}{2}$

   if $D_i^{fL} < \theta$ *and* $D_i^{fR} > \theta$ **then**
      set $E_i^{(fL)} = E_i^{(fL)} + \delta_i$, $S_i^{(fL)} = S_i^{(fL)} - \delta_i$
   else if $D_i^{fL} > \theta$ *and* $D_i^{fR} < \theta$ **then**
      set $E_i^{(fL)} = E_i^{(fL)} + \delta_i$, $S_i^{(fL)} = S_{i_{-\delta_i}}^{(fL)}$
end

Note that for both techniques, *Double* and *Gear3*, the skier is stroking with both poles simultaneously making it possible to borrow information from events found for one pole to the other pole, like done in step 6 and 7.

2.3 Feature engineering

Transforming the data from containing information about the force at all times to only contain information of when somethings happens makes it convenient for the features to be summaries of the variables during an event in force. Therefore, the features in the training data will be summary statistics of each of the variables during an event in force. There are many possible summary statistics to extract from the data and I have narrowed it down to a subset that I felt would be sufficient for the objective of this thesis.
In the upcoming sections the features selected to use together with which response variable for the model are presented. A description of what information they contain, how they are constructed and small overview of each of them summary statistics of them are given.
Figure 2.26 is a schematic figure of the work flow for constructing the training data that was used for the models.

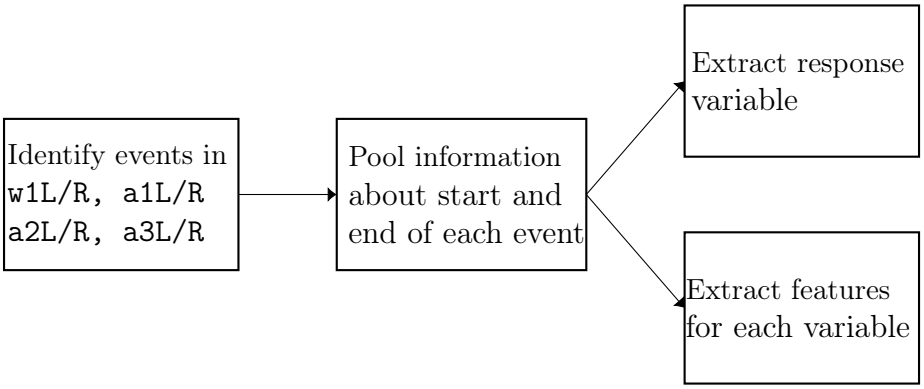

Figure 2.26: Schematic figure of the workflow for creating the training data given a data chunk **X**. Note that this workflow does not illustrate the initial step of removing any possible idle period in **X**.

2.3.1 Response variable

The response variable for the model will have to summarise the whole event in force. For each event in the data we have information about the force and the duration of the event, therefore a natural way of summarising the force over time is by the *area under the curve*. Calculating the area under the force vs time graph will result in calculating the *impulse*, which also is equal to the change in momentum. This quantity is a good summary of the force generated for each event and can be used for comparing two events.

Most humans have a preferred hand, this often leads to one arm being slightly

stronger than the other one. Moreover the arm coordination is usually also better in the preferable hand. With this in mind, it's probably not correct to assume that the force for the left pole is equal to the force for the right pole during an event that is performed simultaneously. In figure 2.27 we can see that there exists a discrepancy between the two force curves in a smaller window of time.

Figure 2.27: The force generated by the left hand side pole and the corresponding force from the right hand side pole during the same time period. From this small sample the existence of a systematical discrepancy in the force generated from the two poles is obvious.

Based on this information, there will be a need of create two separate models, one for predicting the area under the curve of the left force and one for the right force.

2.3.2 features

The features that will be extracted from the data are presented in table 2.4 and subsequently explained in a more descriptive way below.
Given the identified start and end time for an event in force, some of the variables are more active before the start of an event or after an event. For example having a look at the variable `thL` in figure 2.19, it is clear that the angle of the pole is as active before the pole has hit the ground as well as after. Only summarising `thL` during the event in force would not be fair and we would miss out on information. Therefor it is of more interest to summarising the feature in the time period before and during the event in force.
This could be seen for a few more variable, so static shifts of the start and end of an event in force was made to better fit the active period of each variable.
Given these new time periods summary statistics for each of the variables will be extracted.

Summary statistic	Number of features extracted
`Area under curve`	14
`Min value`	14
`Max value`	14
`Peak distance to center`	14
`Altitude`	1
`Velocity`	1
`Duration`	1
`Style`	1
`uphill`	1
`speed`	1
	total = 62

Table 2.4: A small overview of the summary statistics that would be extracted for each variable and be used as features for the models.

Area under curve
This feature will be extracted for each of the vector valued variables *acceleration*, *velocity* and the two variables *angle*.

Peak distance to center:
This feature will provide information of how many time units the peak in a variable is from the center of the identified event in force. The center of the identified event in force is assumed to correspond to the peak of the force curve, so this feature is a measure of distance in time units between the peaks. This feature will be extracted for each *velocity*, *acceleration* and *angle* variables .

Min value & Max value
As their names are indicating, they will be the max and in value of each variable in a given time period. The information gain these two could for example be if the skier lifted the pole too high causing a bad technique on the stroke. This feature will be extracted for each *velocity*, *acceleration* and *angle* variables .

Velocity:
This feature calculated as the maximum recorded velocity from the end an event up until the start of the next event. The reason for being calculated between events is because after a stroke is when the maximum velocity is achieved. If the event was the last event detected in the data, velocity is the maximum over that last event.
Since the start and end of events for the two poles could differ a bit, the end of the event and the start of the next upcoming event is set to maximize the window size. This is no limitation since the speed variable does not experience large fluctuations over a shorter period of time.

Duration:
This is the duration of the force event and is calculated as the mean duration of the event for the left pole and the right pole.

Style:
This feature will hold information about which technique that was used by the skier. In the collecting of the data two techniques was used, *Dubble stroking* and *Gear 3*. So this feature is a categorical feature with two levels *double* and *gear3*.

Uphill
This is a categorical feature with two levels that checks if the change in altitude between two consecutive events is positive or negative. If the change in altitude is positive then the skier is skiing upwards.

Speed
This is a categorical feature with 3 levels. *low* if `velocity` $\in [0, 2.5)$ m/s, *medium* if `velocity` $\in [2.5, 5)$ m/s and *high* if `velocity` $\in [5, \infty)$ m/s.
The cutoff points for each level is very subjective and was selected after eyeballing the velocity over time in the data.

In table 2.5 - 2.9 some descriptive stats of all the features in the data set is presented. The values for the mean and standard deviation have been truncated to one decimal.

Feature	Type	Mean	Std
AUC_thL	Float	2316.2	422.2
AUC_thR	Float	2517.2	352.3
AUC_w1L	Float	55.9	8.8
AUC_w1R	Float	55.3	8.9
AUC_w2L	Float	35.0	9.8
AUC_w2R	Float	59.3	11.7
AUC_w3L	Float	17.7	8.5
AUC_w3R	Float	14.5	6.9
AUC_a1L	Float	185.5	66.4
AUC_a1R	Float	123.5	35.3
AUC_a2L	Float	222.0	88.5
AUC_a2R	Float	251.7	82.7
AUC_a3L	Float	358.8	111.5
AUC_a3R	Float	419.4	127.7

Table 2.5: Overview of the features *area under the curve* for the measured variables

Feature	Type	Mean	Std
peak_dist_thL	Float	-0.5	0.2
peak_dist_thR	Float	-0.6	0.2
peak_dist_w1L	Float	-0.1	0.08
peak_dist_w1R	Float	-0.1	0.06
peak_dist_w2L	Float	0.3	0.1
peak_dist_w2R	Float	-0.1	0.3
peak_dist_w3L	Float	0.4	0.3
peak_dist_w3R	Float	0.3	0.2
peak_dist_a1L	Float	0.0	0.2
peak_dist_a1R	Float	0.2	0.2
peak_dist_a2L	Float	0.0	0.2
peak_dist_a2R	Float	0.0	0.2
peak_dist_a3L	Float	0.0	0.3
peak_dist_a3R	Float	0.0	0.2

Table 2.6: Overview of the feature *peak distance to center* features for the measured variables

Feature	Type	Mean	Std
MIN_thL	Float	10.0	10.9
MIN_thR	Float	13.5	10.7
MIN_w1L	Float	-0.1	0.7
MIN_w1R	Float	0.1	0.7
MIN_w2L	Float	-4.0	1.3
MIN_w2R	Float	-2.3	0.7
MIN_w3L	Float	-0.6	0.5
MIN_w3R	Float	-1.1	0.5
MIN_a1L	Float	-29.9	10.5
MIN_a1R	Float	-9.3	3.6
MIN_a2L	Float	-52.2	17.7
MIN_a2R	Float	-60.2	16.6
MIN_a3L	Float	-47.2	16.1
MIN_a3R	Float	-54.4	16.9

Table 2.7: Overview of the features *min value* for the measured variables

Feature	Type	Mean	Std
MAX_thL	Float	78.1	7.2
MAX_thR	Float	83.4	4.8
MAX_w1L	Float	3.9	0.7
MAX_w1R	Float	3.8	0.6
MAX_w2L	Float	2.1	1.0
MAX_w2R	Float	6.3	1.8
MAX_w3L	Float	1.3	0.4
MAx_w3R	Float	1.2	0.4
MAX_a1L	Float	8.2	4.4
MAX_a1R	Float	12.2	5.0
MAX_a2L	Float	16.7	8.6
MAX_a2R	Float	14.6	7.7
MAX_a3L	Float	12.9	7.8
MAX_a3R	Float	14.9	9.6

Table 2.8: Overview of the *max value* features for the measured variables

Feature	Type	Mean	Std
velocity	Float	4.8	1.1
Altitude	Float	73.0	22.6
duration	Integer	20.5	2.6
fL	Float	1531.8	756.5
fR	Float	1391.9	766.4

Table 2.9: Overview of the remaining numerical features extracted

Feature	Type	levels	proportion
style	Categorical	Double/Gear3	0.56/0.44
uphill	Categorical	Yes/No	0.49/0.51
speed	Categorical	low/medium/high	0.02/0.51/0.47

Table 2.10: Overview of the categorical features extracted

2.4 Correlations

An overview of the correlations in the data set constructed is presented in figure 2.28. Since the data collected consists of summarising similar information for both the left and the right pole, it's not very surprisingly that we can see strong correlations between features extracted. Moreover, the features are constructed from summary statistics that are closely related, so having groups of strong correlated features can be expected. And we can in the figure see a few groups of strongly correlated features.
From the figure 2.28 the block with the strongest correlations is between the some of the `Min value` features and the two response variables. Further, there is a strong negative correlation between the `Min value` features and the `Max value` features as well.

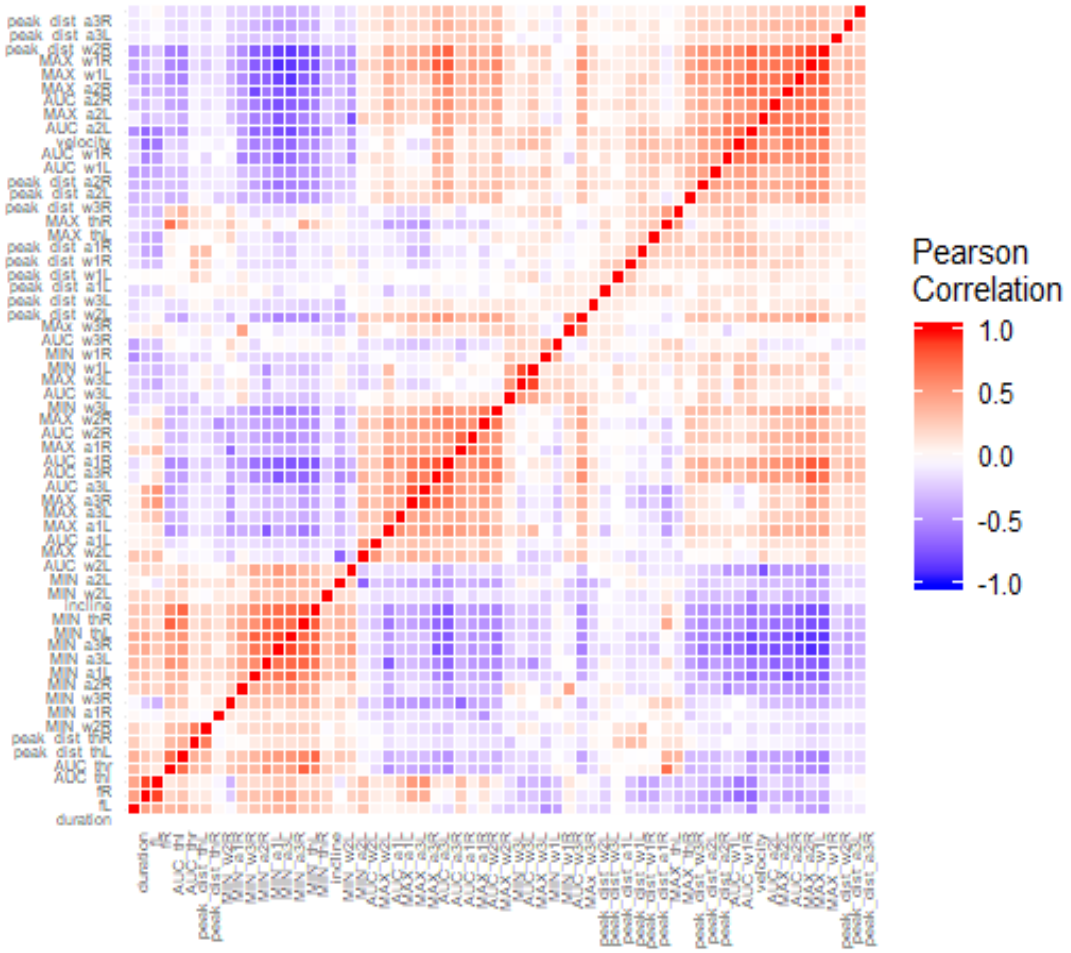

Figure 2.28: The correlation between variables in the data, including the two response variables

2.5 Assumptions

In this section the assumptions that are used in the algorithm and the thesis are shortly described and outlined.

More likely to miss events than finding ghost events:
Since skating involves using both poles simultaneous, if a event in force was identified for one of the poles but not the other one, I have assumed that we have missed to identify the event for the other pole. Consequently, the time period of the identified event will be shared for the two poles meaning that there will be the same number of events for both `fL` and `fR`.
The consequences of this assumption is that if we in fact did not have an event, then we will add noise to the data in form of an observation where the force was summarized during a time period when there was no positive force.

Roller skating equivalent to skiing on snow
All the data used throughout this analysis is collected from skiers using roller skates and skiing on the pavement, while the real application is for skiing on snow. The difference in measurements from skiing with roller skates on pavement compared to skiing on snow has not been investigated and is assumed to be zero.
The possible implications of this assumption is that the algorithm won't work on data from sessions on snow, but the methodology of how to identify events in force and how to predict will still be the same. So this assumption is not seen to have any limitations on the analysis.

Duration of events for the two poles are approximately equal
When finding events in the variables using the function *findpeaks*, there is a possibility that the function is not finding the correct start or end of the event due to noise in the data. So to handle this scenario, if an event for one pole is unusually long or short, information about the start and end of the event for the other pole is used to correct this. This is also not an limitation in any way since for both techniques the poles are used simultaneously.

3

Theory review

This section is supposed to give a theoretical background of the models that has been considered in order to achieve the aim of this thesis. The section is split into several subsection to increase readability and interpretation.

To start off lets consider a the general setting that these models will work in, namely *supervised learning*. The goal in supervised learning is given some input, $\mathbf{x}$ learn a rule to predict the outcome y for unseen data in the future. This is done by finding an approximation, $\hat{f}(\mathbf{x})$, of the function $f(\mathbf{x_j}) = y_j \ \forall j$ that underlies the predictive relationship to the outcome.
The outcome could either be quantitative or qualitative, problems with quantitative outcomes are usually refereed to as *regression problems* while problems with a qualitative outcome is refereed to as *classification problems*.
It is called *supervised* since the data used in finding the approximation are already observed input/outcome pairs $(\mathbf{x_j}, y_j)$.
The theory in the subsequent sections will only focus on the case when the outcome is quantitative due to the relevance to the aim of this thesis.

3.1 Decision trees

Decision tree methods partitions the feature space into regions by recursively making binary splits and models the response as a constant in each region. There are several different tree methods but in this thesis I will focus on *Classification and Regression Trees* ($CART$), suggested by *Breiman* in [4].
Initially *Breiman* constructed two methodologies for constructing trees dependent on the type of problem, *regression* or *classification*. This section will only deal with the *regression* case but it should be noted that the construction of a tree in either of the two cases are similar.

Let's say that we have data $(\mathbf{X}, Y)$, where $\mathbf{X}$ consists of observations $\mathbf{x}_i$ for $i = 1, ..., n$ and each $\mathbf{x_i} \in \mathbb{R}^p$, and $Y = \{y_1, ..., y_n\}$ where $y_i \in \mathbb{R}$, with the goal of constructing a predictor $f(\mathbf{X})$ to predict Y. Then a decision trees is constructed by partitioning the feature space by a sequence of binary splits into terminal nodes. The response in each terminal node t is then modeled as a constant value $y(t)$.
Figure 3.1 illustrates a toy example borrowed from [6] of this procedure in a two-dimensional case (X_1, X_2) with split-points t_i ending up with 5 terminal nodes.

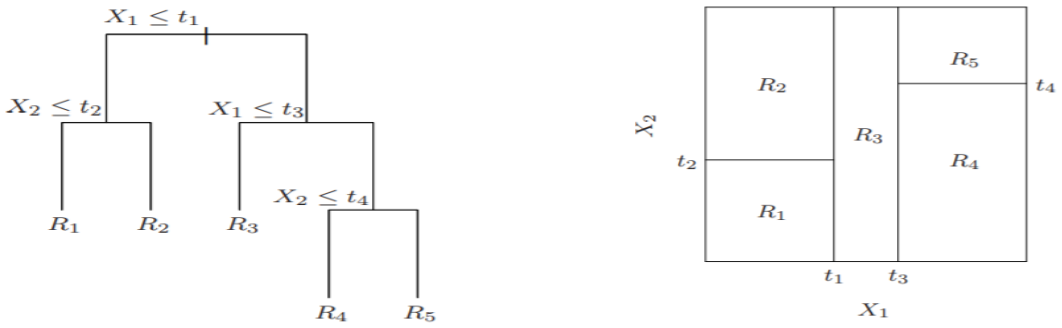

Figure 3.1: Toy example of a 2-dimensional case of how a tree based predictor is constructed. The figure is taken from [6]

To define the tree predictor, $f(\mathbf{X})$, there are a few questions that needs to be answered. *How to select the best split in each intermediate step ?*, *When to stop splitting a node ?* and *what to model the response as in each terminal node?*.
To measure the accuracy of a predictor f in regression, the *mean square error* is normally used and that is also the measurement suggested by *Breiman* in [4].
The *mean square error* $R^*(f)$ of a predictor f is defined as

$$R^*(f) = E(Y - f(\mathbf{X}))^2$$

It can then be shown that the best predictor that minimizes $R^*(f)$ is $f(\mathbf{x}) = E(Y|\mathbf{X} = \mathbf{x})$, i.e the conditional expectation of the response.

An estimate of $R^*(f)$ is $R(f) = \frac{1}{N}\Sigma_{i=1}^{N}(y_i - f(\mathbf{x}_i))^2$ and then it is straight forward to see that the best value of the constant $y(t)$ in terminal node t that minimizes $R(f)$ is the average of y_i in each node, i.e

$$\overline{y}(t) = \frac{1}{N_t}\sum_{\mathbf{x}_i \in t} y_i$$

Modeling $y(t)$ as the average in each node t and using the notation $R(T)$ for the *mean square error* we get that

$$R(T) = \frac{1}{N}\sum_{t \in \tilde{T}}\sum_{\mathbf{x}_i \in t}(y_i - \overline{y}(t))^2$$

The interpretation of the quantity $\sum_{\mathbf{x}_i \in t}(y_i - \overline{y}(t))^2$ is that it is the within node sum of squares, and summing these quantities over all terminal nodes t yields the total within sum of squares. Therefore the best value of the constant that models the response in each node t is therefor $\overline{y}(t)$.

To find best split with the intent of minimizing $R(T)$ is usually computationally infeasible since the tree is grown iterative. Instead a greedy algorithm that might not find the optimal tree is used instead. Consider splitting $\mathbf{X}$ on feature j in splitting point s into two partitions

$$P_1(j, s) = \{X|X_j \leq s\} \quad P_2(j, s) = \{X|X_j > s\}$$

Then the goal is two find the feature j and the splitting point s that minimizes

$$\min_{j,s}[\min_{a_1} \sum_{\mathbf{x}_i \in P_1(j,s)} (y_i - a_1)^2 + \min_{a_2} \sum_{\mathbf{x}_i \in P_2(j,s)} (y_i - a_2)^2]$$

The inner minimization is solved for any pair of (j, s) by setting

$$a_1 = \overline{y}(P_1(j,s)) \quad a_2 = \overline{y}(P_2(j,s))$$

For each feature, the best splitting point is easy to find therefore finding the best pair (j, s) is computationally feasible.

Now the two of the three questions in defining a tree has been answered, the last question *when to stop splitting a node* is not as easy to answer. The more splitting of nodes should decrease the mean square error but the model would eventually become to complex and not generalize so well for new unseen data.
So what was proposed was to first grow a very large tree T_{max} until that for every $t \in \tilde{T}_{max}$, $N_t \leq N_{min}$ or if all the values in the node are the same, N_{min} usually is taken as 5 [4].
Define a sub-tree $T \subset T_{max}$ to be any tree that could be obtained by collapsing the internal nodes in T_{max}.
Let

$$N_t = \#\{\mathbf{x}_i \in t\}$$

$$\overline{y}(t) = \frac{1}{N_t} \sum_{\mathbf{x}_i \in t} y_i$$

$$Q_t(T) = \frac{1}{N_t} \sum_{\mathbf{x}_i \in t} (y_i - \overline{y}(t))^2$$

Next define the *cost complex criteria* as

$$C_\alpha(T) = \sum_{t \in \tilde{T}} N_t Q_t(T) + \alpha |\tilde{T}|$$

Now for each value of α, the idea is to find the sub-tree $T_\alpha \subset T_{max}$ that minimizes $C_\alpha(T)$. The tuning parameter α controls the trade-off between *goodness of fit* and model complexity. To find the value for α, it suggested in [6] to use 10-fold cross-validation and select the value $\hat{\alpha}$ that minimizes the cross-validation sum of squares.

3.2 Random forest

Regression trees are known for having a high variance and small changes in the input data could lead to a totaly different tree model, a technique to reduce the variance is by using *bagging*. The main idea in *bagging* is to fit many trees to bootstrap samples of the data and average the predictions of all these models to obtain the final prediction. Trees are ideal to use in a *bagging* scheme since the have relative low bias and are able to capture complex structures in the data when grown deep [5].

Bagging involves repeatedly drawing a sample Z with replacement from the training data $\mathbf{X}$ and fit a tree f_b to the data where $b = 1, ..., B$. Predictions for unseen data is then given by averaging the predictions of each tree, where every tree contributes equally to the final prediction, i.e $\hat{f}^B(\mathbf{x}) = \frac{1}{B}\sum_{b=1}^{B} f_b(\mathbf{x})$.

The sub samples are likely to contain about 63% of each observations in the sample refereed to as *in-bag samples* while the rest 37% as refereed to as *out-of-bag samples* (*oob*) [7]. Sampling with replacement and fitting trees to each sample yields fitting a models to partially overlapping subsets of the data yielding the models not being uncorrelated. This causes a problem when averaging since the variance of B *i.i.d* random variables all with variance σ^2 is equal to $\frac{1}{B}\sigma^2$. Now if the variables are only *i.d* (identically distributed) but not necessarily independent with positive pairwise correlation ρ, the variance of the averaging instead becomes $\rho\sigma^2 + \frac{1-\rho}{B}\sigma^2$. It's clear that the second term goes to 0 as B increases while the first term remains.

To keep the accuracy of the ensemble model but reduce the correlation between the trees, *Breiman* introduced the *Random Forest* in [5], where a second step in the fitting of a tree was incorporated. Instead of a tree considering all p features as candidates in each split, only a sub sample of the features should be considered. More specifically before each split $m \leq p$ input features are randomly sampled as candidates for splitting, typical value for $m = p/3$ [5].
Let Θ_k characterize the *k:th* tree trained this way in terms of splitting variables, cut-points at each node and terminal node values, then a prediction for unseen data $\mathbf{x}$ is obtained from $\hat{f}^B(\mathbf{x}) = \frac{1}{B}\sum_{b=1}^{B} f_b(\mathbf{x}; \Theta_b)$.
Decreasing the number of candidate features in each split will reduce the correlation between trees, also each tree in the forest are neither *pruned* to further reduce the correlation between trees.

3.2.1 Variable Importance

Using a sub sampling scheme like *bagging* provides unseen data for each tree that can be used to provide further insights via error estimates and more. For example, the *out-of-bag-samples* for a tree could be used to estimate the *error-term* and then averaging over all these estimates to obtain an estimate of the error for the model. This estimate is similar to the estimate when using cross-validation with the difference that in *Random Forest* the estimate is unbiased [5]. The *oob-sample* could also be used for quantifying the predictive strength of each feature, this measurement is called *variable importance* and will be shortly reviewed in this section.
The variable importance for feature j is calculated in the following way. After the $b : th$ tree is grown, the *oob-samples* are passed down the tree and the *oob-error* is computed. Then the values of the $j : th$ features is permuted in the *oob-sample* and the sample is fed to the model again and the *oob-error* is computed once again. The difference between the original *oob-error* term and the one for the permuted sample averaged over all trees, normalized by the standard deviation of the differences, is a measure of the predictive strength of feature j.
In other words *variable importance* works under the assumption that if a feature is

important, permutation of the values should have a larger effect on the *oob-error* estimate while a lower difference is related to weaker predictive strength.

4
Study design

In this section of the thesis, 3 different designs of trying to predict the force are presented. The models considered are all static, meaning that they are trained on available data and as new data arrives it predicts the outcome of each new event.
The results from the investigations of the designs can be found in the section 5. As an evaluation of the accuracy from the models fitted for the regression problem in this thesis, R^2 is considered as well as the *residual mean square error*. More emphasis will be put on the R^2 since the *residual mean square error* will usually be very large due the magnitude of the response values, see table 2.9. For the classification models, the accuracy will be used as evaluation metric of performance.
Before a model was fitted to the training data the hyperparameter *mtry* for the random forest method was tuned using *K-fold* cross-validation and a random search for the best value. The choice of K differed between models depending on the sample size of the training data. Moreover, as a metric of evaluating the accuracy in the tuning, R^2 is used to obtain the best value for the hyperparameter.
Random forest also has another parameter that could be tuned namely the *ntree* parameter that controls the number or trees in the forest. This parameter has less effect on the results but a larger value is recommended to obtain a more stable model and also better results on the variable importance measures. A larger value will however increase the computational time for fitting the model, but since the training data isn't huge this parameter was not tuned and set to 2000.

4.1 Design 1

The complete data that was used in this thesis came from two different skiing sessions, so the settings for the models in this design is that the training data and the test data will be from different sessions. I.e a model will be tuned and fitted to the data from the session in *sätila* and tested on the session in the *city*, and conversely tuned and fitted on the session in the *city* and tested on the session in *sätila*.
The idea in this design is that a new customer of *Skisens* product could come in and ski a test session with the handles that measures the force to produce a training set, a model is then fitted to the training data and the customer could go home with a pair of handles that predicts the force instead of measuring it.
One cautionary remark with this design is that the sample sizes between the two sessions differs significantly. For the session in *Sätila* there is approximately 2000 events and for the session in *City* there is sligthly above 500 events for training.
A direct implication of the sample size of the session in *City* is that the number

of folds when performing tuning will be set to 5. While when a model is fitted to the data from *Sätila* the number of folds will be set to 10 instead. Moreover, the number of values tried for the parameter *mtry* will be set to 10 in both cases. Table 4.1 outlines the different models that will be trained using this design.

Model	response	data used for training
$M_{1,1}$	`fL`	*Sätila*
$M_{1,2}$	`fR`	*Sätila*
$M_{1,3}$	`fL`	*City*
$M_{1,4}$	`fR`	*City*

Table 4.1: An overview of the models that will be trained using this design.

4.2 Design 2

In this design, I will no longer distinguish between the data from the two sessions. Instead, via subsampling the data will be split into a training set, $\mathbf{X_{train}}$, and a test set, $\mathbf{X_{test}}$. On the training set an initial random forest model will be fitted with the task of predicting the `style` that was used during an event.
Then the training data will thereafter be partitioned based on the style and subsequently random forest models will be fitted to predict the force for both poles on each of the partitions. The two partitions are about the same size and large enough, therefor in parameter tuning via cross-validation the fold size was set to 10. As for the previous approach, prediction R^2 and *RMSE* will be used as evaluation metric for selecting the best value on `mtry`.
Since discrepancies in the features and the response variables can be seen when conditioned on style, Appendix A figures A.8 - A.13, the hope is having specific models for each style and pole results in better predictions.
As a result, in this approach there will be a total of 5 models, that are outlined in table 4.2 below.

Model	Task	response	data used for training
$M_{2,1}$	Classification	`style`	$\mathbf{X_{train}}$
$M_{2,2}$	Regression	`fL`	$\mathbf{X_{train\|style=Double}}$
$M_{2,3}$	Regression	`fR`	$\mathbf{X_{train\|style=Double}}$
$M_{2,4}$	Regression	`fL`	$\mathbf{X_{train\|style=Gear3}}$
$M_{2,5}$	Regression	`fR`	$\mathbf{X_{train\|style=Gear3}}$

Table 4.2: An overview of the models that will be trained using design2 together with their task and what data was used for fitting the model.

When all the models have been fitted to its respective data and its time to predict the force for the two poles. For each observation in the test data, the style will first be predicted and based on the outcome of that prediction a model will be selected to predict the force generated. Figure 4.1 is a schematic figure of how the predictions is made in this approach.

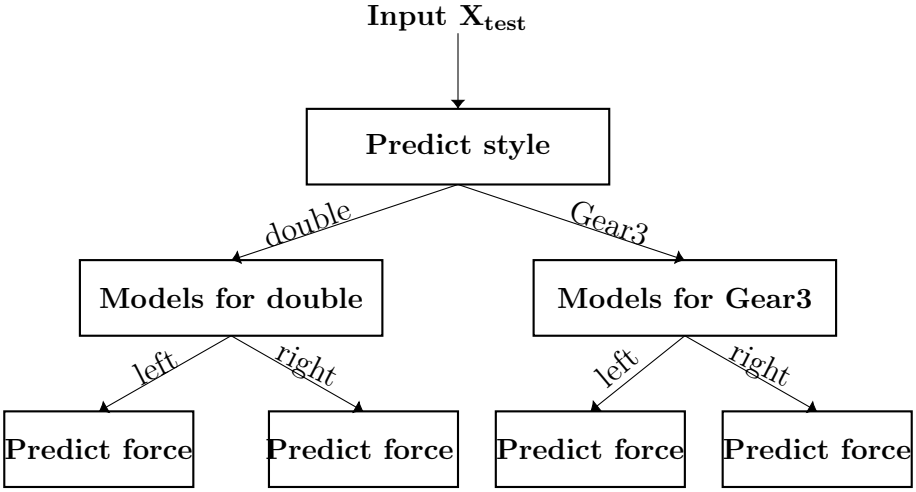

Figure 4.1: Schematic figure of how predictions are obtained for force left an force right in design 2.

4.3 Design 3

The third and last design investigated in this thesis was to incorporate the two previous ideas into one single design. Besides from training on data from one session and trying to predict the outcomes for the another session, the models will also be style specific. I.e their will be two models for predicting the force for the left pole depending on the style used during the event. In design 2, I tried to train a classification model with the task of classifying the *style* use. That model was trained on a sufficiently large training data, but in this design the *City* session in quite small in sample size. So for not to have faulty classifications affect the end result in this design, the previous model will be used in the initial classification step of predicting new data.
As a result a total of 8 models will be trained and evaluated in this design. In table 4.3 there is an overview of the models and their task in this design.

Model	Task	response	data used for training
$M_{3,1}$	Regression	`fL`	$\mathbf{X_{Satila\|style=Double}}$
$M_{3,2}$	Regression	`fR`	$\mathbf{X_{Satila\|style=Double}}$
$M_{3,3}$	Regression	`fL`	$\mathbf{X_{Satila\|style=Gear3}}$
$M_{3,4}$	Regression	`fR`	$\mathbf{X_{Satila\|style=Gear3}}$
$M_{3,5}$	Regression	`fL`	$\mathbf{X_{City\|style=Double}}$
$M_{3,6}$	Regression	`fR`	$\mathbf{X_{City\|style=Double}}$
$M_{3,7}$	Regression	`fL`	$\mathbf{X_{City\|style=Gear3}}$
$M_{3,8}$	Regression	`fR`	$\mathbf{X_{City\|style=Gear3}}$

Table 4.3: An overview of the models that will be trained using design 3 together with their task and what data was used for fitting the model.

As mentioned in the description of design 1, the data set from the session in *City*

is not very large, and in this design we are partitioning that data based on style yielding an even smaller sample size's for the models to train on.

5

Results

In this section the results from all the models considered in the thesis is presented. The results are presented with figures and tables that are considered to be relevant for the aim outlined in section 1. Moreover all the models are outlined and described in section 4. For starters the results from the parameter tuning for each model is presented followed by the performance of the models for each design. Lastly, which features are the most important for predictions are evaluated with the variable importance measures from each model.

5.1 Parameter tuning

As described in section 4 the parameter tuning was made using the `R` function *train* that is available in the library *Caret*. The parameter to tune is *mtry* and it was conducted using cross-validation where the number of folds differs between models due to the sample size of the training data in hand. Moreover, as mentioned with the models being static and the run time is not a concern, the parameter *ntree* is not tuned and set to 2000.
The evaluation metric for selecting the best value of *mtry* was a combination of both *RMSE* and R^2, but only the results of R^2 will be presented in the tables since that metric was subjectively of most interest. Since for each value value of *mtry* we are performing *K-fold* cross-validation, we will get K estimates of R^2. This allows for estimating the error, $\hat{\sigma}$, of the point estimate of R^2 and this quantity is also presented in the tables below. Lastly, the different values of *mtry* are sampled uniformly between 1 and the dimension of the input feature space.

5.1.1 Result of tuning for the models in design 1

mtry	R^2	$\hat{\sigma}$
1	0.852	0.019
5	0.884	0.032
10	0.889	0.033
14	0.890	0.034
19	0.891	0.034
28	0.892	0.033
29	0.891	0.034
36	0.890	0.035
43	0.889	0.035
56	0.887	0.036

mtry	R^2	$\hat{\sigma}$
2	0.869	0.044
13	0.887	0.043
14	0.885	0.044
18	0.886	0.041
19	0.886	0.041
29	0.886	0.039
39	0.886	0.035
42	0.885	0.035
58	0.885	0.030

Table 5.1: The results of tuning the parameter *mtry*, the first column contains the values tested. The following two columns contains the R^2 value and the standard deviation of the estimate. The models was fitted to the data from the *City* session and the left figure is the outcome from having `fL` as response variable and the right figure is for the model having `fR` as respons. Cross-validation was used as method of tuning with effective sample size of 519 therefor the number of folds was set to 5.

mtry	R^2	$\hat{\sigma}$
1	0.801	0.016
25	0.855	0.014
31	0.854	0.016
42	0.849	0.017
44	0.850	0.017
55	0.846	0.019
56	0.845	0.017
59	0.846	0.017
62	0.847	0.017

mtry	R^2	$\hat{\sigma}$
1	0.805	0.022
4	0.844	0.016
8	0.852	0.014
16	0.855	0.012
44	0.855	0.013
21	0.851	0.014
46	0.850	0.013
54	0.849	0.013
58	0.849	0.015

Table 5.2: The results of tuning the parameter *mtry*, the first column contains the values tested. The following two columns contains the R^2 value and the standard deviation of the estimate. The models was fitted to the data from the *Sätila* session and the left figure is the outcome from having `fL` as response variable and the right figure is for the model having `fR` as response. Cross-validation was used as method of tuning with effective sample size of 2074 therefor the number of folds was set to 10.

Table 5.1 and 5.2 displays the results of the tuning done on each respectively data set. The results are similar to each other with *mtry* not being so influential to the fit of the model. Only for very low values *mtry* could there be seen a little drop

in the fit. With many of the features being closely related and highly correlated is not very surprisingly that there wouldn't be a large discrepancy between lower and higher values of *mtry*.

5.1.2 Result of tuning for the models in design 2

mtry	R^2	$\hat{\sigma}$
9	0.908	0.023
10	0.908	0.024
19	0.912	0.022
20	0.911	0.023
21	0.910	0.023
23	0.910	0.024
33	0.911	0.025
40	0.911	0.024
44	0.912	0.025
50	0.911	0.025

mtry	R^2	$\hat{\sigma}$
2	0.767	0.036
8	0.792	0.041
17	0.792	0.052
22	0.790	0.050
33	0.785	0.056
39	0.782	0.060
47	0.783	0.062
49	0.781	0.061
55	0.781	0.063
61	0.777	0.064

Table 5.3: The results of tuning the parameter *mtry*, in the first column contains the values tested. The following two columns contains the R^2 value and the standard deviation of the estimate. Both models had `fL` as response variable where the results in the left table are from the model trained with data from the style *Double*. The right table contains the results from the model trained on data from the style *Gear3*. The effective sample sizes where 1098 and 846 respectively K in cross-validation was set to 10.

mtry	R^2	$\hat{\sigma}$
2	0.851	0.026
8	0.873	0.019
13	0.875	0.017
14	0.875	0.016
19	0.875	0.015
35	0.874	0.014
36	0.874	0.013
37	0.874	0.015
38	0.873	0.013
42	0.872	0.014

mtry	R^2	$\hat{\sigma}$
2	0.801	0.042
14	0.834	0.030
30	0.834	0.030
33	0.833	0.028
38	0.832	0.028
55	0.828	0.028
57	0.827	0.027
60	0.826	0.027
61	0.826	0.028

Table 5.4: The results of tuning the parameter *mtry*, in the first column contains the values tested. The following two columns contains the R^2 value and the standard deviation of the estimate. Both models had `fR` as response variable where the results in the left table are from the model trained with data from the style *Double*. The right table contains the results from the model trained on data from the style *Gear3*. The effective sample sizes where 1098 and 846 respectively K in cross-validation was set to 10.

mtry	Accuracy	$\hat{\sigma}$
9	0.987	0.007
19	0.984	0.008
25	0.983	0.008
30	0.982	0.008
41	0.980	0.008
43	0.982	0.008
46	0.980	0.008
51	0.980	0.008
56	0.980	0.007
57	0.980	0.008

Table 5.5: The results of tuning the parameter *mtry*, in the first column contains the values tested. The following two columns contains the Accuracy and the standard deviation of the estimate. The task of this model was to predict the style used during an event. It is clear that *mtry* is not influential to the fit of the model.

From figure 5.3 and 5.4 it is clear that the models perform about the same as long as the value of *mtry* is not to low. When the parameter value there was a drop in performance. A slight discrepancy between the models in predicting `fL` and `fR`, where the model for predicting `fL` achieves the best results.

The results from performing tuning on the models in design 3 are not present in this section of the thesis due to the similarities in results to the models in design 1 and 2. Where the value of *mtry* did not have a huge effect on the fit of the models. Instead those results could be found in *Appendix A* at the end of this thesis.

As mentioned in the theory review, section 3, a way to reduce the correlation between the trees and consequently the variance in the prediction was to select a lower value of *mtry*. So models was trained using one of its respective lowest tested value of *mtry* where the R^2 value still was high.

5.2 Performance of models

5.2.1 Performance of the models in design 1

Training data	response	R^2
sätila	`fL`	0.787
Sätila	`fR`	0.759
City	`fL`	0.639
City	`fR`	0.692

Table 5.6: The results of the performance for the models trained in design. In design 1 a model was trained on data from one session only to predict the outcome from another session. When there is more training data available, i.e *sätila* session, the models are able to perform a little bit better.

There seems not be be any larger difference between predicting left force or right force. Both those models are achieving almost the same result. The models trained on the data from the session *Sätila* are performing a bit better than the models trained on *City*. This is probably due to the fact that the sample size in *Sätila* is larger which helps to predictive relationship.

5.2.2 Performance of the models in design 2

predicted \real	*double*	*Gear3*	%
double	354	2	
Gear3	1	292	
			99.54

Table 5.7: Confusion matrix for predicting the style used in the test set.

The model for predicting the style used by the skier is performing very well with an accuracy of over 99%. This model is key for predicting the force in this design, so these results are very satisfactory.

Style	response	R^2
Double	`fL`	0.875
Gear3	`fL`	0.767
Double	`fR`	0.808
Gear3	`fR`	0.796

Table 5.8: The results of the evaluation metrics for the models trained in design 2. The overall R^2 when predicting `fL` came out at 0.828 and for predicting `fR` was 0.815

Compared to the models in design 1 there was a slight increase in performance from the models in this design. The model for predicting `fL` that was trained on the data from the style *Double* is performing very well. Both models trained on data from the style *Gear3* seems to perform worse than for the style *Double*.

5.2.3 Performance of the models in design 3

Training data	Style	response	R^2
Sätila	*Double*	`fL`	0.822
Sätila	*Gear3*	`fL`	0.610
Sätila	*Double*	`fR`	0.758
Sätila	*Gear3*	`fR`	0.606
City	*Double*	`fL`	0.663
City	*Gear3*	`fL`	0.424
City	*Double*	`fR`	-0.446
City	*Gear3*	`fR`	0.478

Table 5.9: The results of the evaluation metrics for the models trained in design 3. Only two of the models seemed to perform fairly well, all the other models performed worse than in previous designs. Note that the model for predicting `fR` trained on the *Double* data from the *City* session has a negative R^2. The model seems to have induced a huge bias making predictions lousy.

The models in design 3 are not performing so well, it is most obvious looking at the models trained on the *City* data. One reason for it being like this is that the sample sizes of the training sets are so small and no model is able to accurately find the predictive relationship between the features and the response variables.

5.3 Feature importance

There where many models trained throughout the investigation performed and in this section we will have a look at which features where important in predicting the force for each of the models.
For each of the models, the 5 most influential features will be listed and compared between models and response variables. Table 5.10 displays the most influential

features for the models with the task of predicting fL and 5.11 is the corresponding table for the models with the response fR. The model abbreviations used in these two tables are described in section 4 under each of the designs, next the each model abbreviation the individual ranking of the importance of the feature for that model is displayed.

MAX_a3R	MAX_a3L	MAX_a1L	MIN_a3R	MIN_thR	MIN_a3L	AUC_w1R	peak_dist_a3L	duration	Altitude	velocity
$M_{1,1}$ (1)	$M_{1,1}$ (2)					$M_{1,1}$ (3)		$M_{1,1}$ (4)	$M_{1,1}$ (5)	
$M_{1,3}$ (4)	$M_{1,3}$ (1)	$M_{1,3}$ (5)				$M_{1,3}$ (3)				$M_{1,3}$ (2)
$M_{2,2}$ (3)	$M_{2,2}$ (1)		$M_{2,2}$ (2)				$M_{2,2}$ (4)			$M_{2,2}$ (5)
$M_{2,4}$ (1)	$M_{2,4}$ (3)				$M_{2,4}$ (2)	$M_{2,4}$ (4)				$M_{2,4}$ (5)
$M_{3,1}$ (3)	$M_{3,1}$ (1)		$M_{3,1}$ (4)	$M_{3,1}$ (5)			$M_{3,1}$ (2)			
$M_{3,3}$ (1)	$M_{3,3}$ (3)				$M_{3,3}$ (2)	$M_{3,3}$ (4)			$M_{3,3}$ (5)	
	$M_{3,5}$ (1)		$M_{3,5}$ (2)		$M_{3,5}$ (4)	$M_{3,5}$ (3)				$M_{3,5}$ (5)
$M_{3,7}$ (1)	$M_{3,7}$ (3)					$M_{3,7}$ (2)	$M_{3,7}$ (5)		$M_{3,7}$ (4)	

Table 5.10: The table displays the top 5 features for each model trained throughout this thesis with the task of predicting fL. The model abbreviation and feature abbreviations are described in section 4 and 2 respectively. The table does not give any indication on the magnitude of the importance for each model but their ranking within each model is presented next to each model abbreviation. Noticeable is that even though the models are supposed to predict fL many of the features are related to the right pole of the skier.

MAX_a3R	MAX_a3L	MAX_a1L	MIN_a3R	MIN_a3L	MIN_w3L	AUC_w1R	AUC_w3L	AUC_a1R	peak_dist_w1R	peak_dist_a2R	altitude	velocity
$M_{1,2}$ (1)	$M_{1,2}$ (5)					$M_{1,2}$ (4)	$M_{1,2}$ (3)				$M_{1,2}$ (2)	
$M_{1,4}$ (1)	$M_{1,4}$ (3)	$M_{1,4}$ (5)				$M_{1,4}$ (4)						$M_{1,4}$ (2)
$M_{2,3}$ (1)	$M_{2,3}$ (5)					$M_{2,3}$ (4)	$M_{2,3}$ (2)				$M_{2,3}$ (3)	
$M_{2,5}$ (1)			$M_{2,5}$ (4)			$M_{2,5}$ (5)		$M_{2,5}$ (2)				$M_{2,5}$ (3)
$M_{3,2}$ (1)					$M_{3,2}$ (4)		$M_{3,2}$ (3)		$M_{3,2}$ (5)		$M_{3,2}$ (2)	
$M_{3,4}$ (1)				$M_{3,4}$ (3)		$M_{3,4}$ (5)		$M_{3,4}$ (2)				$M_{3,4}$ (4)
$M_{3,6}$ (1)	$M_{3,6}$ (3)		$M_{3,6}$ (2)	$M_{3,6}$ (5)		$M_{3,6}$ (4)						
$M_{3,8}$ (1)	$M_{3,8}$ (4)							$M_{3,8}$ (3)		$M_{3,8}$ (5)	$M_{3,8}$ (2)	

Table 5.11: The table displays the top 5 features for each model trained throughout this thesis with the task of predicting fR. The model and feature abbreviations are described in section 4 and 2 respectively. The table does not give any indication on the magnitude of the importance for each model but their individual ranking is displayed next to each model abbreviation. As in table 5.10 it is noticeable that even though the models are supposed to predict fR many of the features are related to the left pole of the skier.

From the two tables with the variable importance results from the models trained it seems that many of the same features are important for predicting either fL or fR. For most models MAX_a3R, MAX_a3L and AUC_w1R seems to be important, even though the magnitude and of the importance of the features is not displayed in the tables, from their ranking it is obvious that MAX_a3R is important for predicting either fL or fR.
It is also a little interesting that features related to the pole on one side is important for predicting the force generated from the pole on the other side. One reason for it being this way could be that even though the sensors are mounted on different poles making the measurements independent in that sense, the motion of the a pole is dependent on the motion of the opposite side pole. For example, in the style *Double*

where the two poles are supposed to follow the same movement. So in the stroking motion, if the right hand side pole is raised to a certain height the left side pole will approximately be raised to the same height.

6

Conclusions

The goal with this thesis was to try to predict the force generated by the skier when performing cross-country skiing. Being able to accurately predict the force generated will help the company *Skisens* in their strive of incorporating new methods for evidence based training for cross-country skiing. The result would be methods where performance could be comparable between sessions without outside factors influencing the results.
At my disposal, I had data collected by *Skisens* from two sessions where each measurement was recorded every $0.2s$. Predicting the force at that granular level seemed superfluous since the force is non-present except for when a stroking motion is performed and the poles hits the ground. So, the methodology I chose to work with was to view each stroke as an event of some kind and then try to summarize the force generated throughout that event.
An unsupervised algorithm for detecting these events was created by trial and error which also was tuned in an *ad hoc* manner for the data from the two sessions, with the hope that the algorithm would work for unseen data.
As an event was detected the variables in the data was summarized into several features that was thought to contain the necessary information to be able to predict the force during an event.

Random forest was the only method used for building the predictive models in this thesis. One reason for that is that the features extracted from the data where highly correlated and where some methods breaks down from having correlated features, *Random forest* does not break down from the features being correlated. Also utilizing the strength of an ensemble of predictors was preferable given that there might be many outside factors affecting the response variable yielding that the relationship between the features and the response being hard to capture. Moreover, as is mentioned in *Future work* with the existence of a nice extension of *Random forest* in an online setting available, solidified the reason for going with tree based ensemble models.

A few different designs of models for predicting the force was tried in this thesis. Each of the designs had in mind how *Skisens* could build sufficient models for new customers of their product. For example, in design 1 the idea was to see if a model trained on the data from one session could predict the outcome of the response from another session. That way, a new costumer could initially do a training session using the handles where force is measured to create enough data for the model to pick up the signal, and afterwards use handles where a model predicts the force in future

sessions. Similarly, the key idea in design 2 was to see if having separate models for each *technique* used could enhance the predictive performance of the models.
The results was that having separate models for each *technique* improved the performance compared to the results in design 1, where the results was not very satisfactory. I would say that it is hard to evaluate the results in design 1 in a fair way because the two session are quite different. In one session the skier is skiing in *Gothenburg* city on the streets while the other session is around a lake close to the more calmer place *Sätila*. The terrain in the second session also allowed for longer sections of uninterrupted skating. Moreover, I would also consider the unbalance in sample sizes a problem for achieving great results with this design.
The last and third design is meant to be a mixture of the two first designs, with fairly good results in performance from design 2 the idea was to investigate if the I could improve the performance compared to design 1. In this design, the limited samples at hand was a huge pitfall for the models, in the *City* session a total of 519 events was detected and used for creating the training data. These 519 events was then partitioned into two partitions based on the *technique* used yielding even smaller sample size's for the models to train on.
Only one model in this design preformed well, and that was the model trained on data from the *Sätila* session with the task of predicting `fL` for the technique *Double*. This small result makes me hopeful that if more data was available, the models in this design would perform well.
As a proof of concept investigation, I would say that even though that the results are not fantastic, there is a slight indication in design 2 that it would be possible of predicting the force accurately.

6.1 Future work

One of the problems encountered throughout this thesis was the lack of more data and more diverse data. With more data at hand the models would have an easier time to pick up the predictive signal. However, there are a lot of factors influencing the skier that is not measured, so it would be beneficial to have more data where the surroundings differs. It would also be interesting to investigate if there is a need for having a personalized model, i.e would a model trained on the data from one skier be able to accurately predict the outcome from another skier?

A lot of work was put into detecting events, In the construction of the training data there is one simplification made that could have a larger effect on the results. That was that once an event in force was identified we had the start time and end time of that event, call those two time periods t_1 and t_2. Given the interval $[t_1, t_2]$ information about the other variables was extracted. For some variables the interval needed to be shifted in order to overlap with something happening in the variable. Constant shifts was applied to the variables that it was found needed for, for example for the variable `thL` the interval as shifted to $[t_1 - c, t_2]$, for some value of c , since half of work in `thL` is conducted before an even in force.
The downfall with this is that if t_1 isn't corresponding to the start of an event in force but rather a few time steps before the start, then $t_1 - c$ will neither correspond

to the start of work in `thL` yielding that we would add noise into the features constructed from `thL`. A better way would to from the interval $[t_1 - c, t_2]$ use a function to identify an even in `thL`. This would limit the amount of uninformative data added into the construction of the features.
Further, the features for the training data could and probably should be extended and overview by someone with more insights in cross-country skiing than me. The features constructed in this thesis seemed logical from a physical point of view, but since I have never cross-country skied there are probably many aspects in the data that I am missing.

When transforming the data to contain events instead of records measured each 0.2 second, we lose a bit of information that could be useful. Since the skier is human it is natural that he/she is not able to perform with the same intensity over a long time making the measurements time dependent. It would be good to incorporate this dependency in the model in order to increase the accuracy of the predictions. When I got the data, the data was partitioned into several *csv-files* where the time was local to each file. And without the possibility of knowing in which order the data was collected made it almost impossibly to model this time dependency. With the events there is still information about time since we know in which order the events occurred, but what makes it hard to model the time dependency is that the time lives in a different scale and would still only be local to each file. So as part of the next step in this investigation, I propose that the time should not be local to each file for the possibility to include this dependency in the model.

Lastly, as mentioned before *Random forest* has an extension to an online learning scheme where the data arrives to the model in a stream and the model is able to replace bad performing learners and update it self on the fly. I think that this kind of model would be able to perform much better due to the possible changes in surroundings for the skier. A models like this would be more locally anchored be able to adjust to changes, while the models trained in this thesis are more globally anchored and are assumed to work over time.

Bibliography

[1] B.Byrne, *"Adidas F50 adiZero miCoach Released"*, *soccercleats101.com*, 29 sept 2011. [Online]. Available *https://www.soccercleats101.com/2011/09/29/adidas-f50-adizero-micoach-released/*. [Accessed 25, Aug. 2019].

[2] Skisens AB, Personal email (30 April 2019).

[3] Andrew R. Coggan, *"A brief history of training and racing with a power meter"*, *http://www.trainingandracingwithapowermeter.com*, 10 April 2010. [Online]. Available *http://www.trainingandracingwithapowermeter.com/2010/04/brief-history-of-training-and-racing_1025.html*. [Accessed 4, June 20190].

[4] L.Breiman, J. H.Friedman, R. A.Olshen, C.J.Stone, *"Classification and Regression Trees*, Taylor Francis Group, Boca Raton, Fl, 2017.
ISBN 13:978-1-1384-6952-5

[5] L.Breiman, "Random Forests", *Machine learning*, vol 45, 2001, p.5-32
https://doi.org/10.1023/A:1010933404324

[6] T.Hastie, R.Tibshirani, J.Friedman, *"The elements of statistical learning"*, Springer Science+Business Media, LLC, 2009.
ISSN: 0172-7397

[7] D. R.Cutler, T. C.Edwards, K. H.Beard, A.Cutler, "Random Forests for Classification in Ecology", *Ecology*, vol 88, Dec 2007, p.2883-92,
https://doi.org/10.1890/07-0539.1

A
Appendix

A.1 Right side variables when positive force

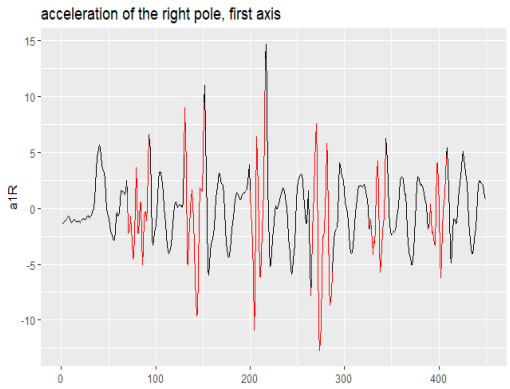

Figure A.1: Acceleration of the right pole in the first axis, the red parts of the line is when we have a positive force.

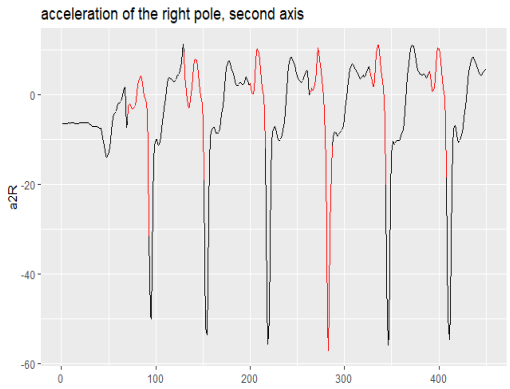

Figure A.2: Acceleration of the right pole in the second axis, the red parts of the line is when we have a positive force.

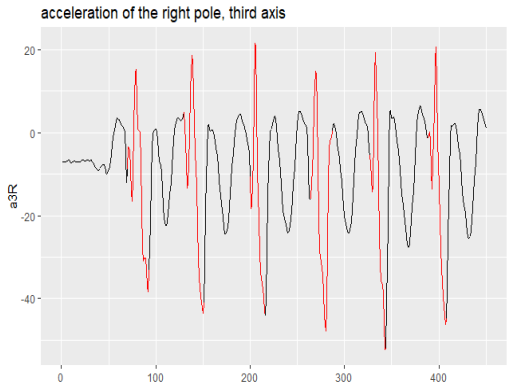

Figure A.3: Acceleration of the right pole in the third axis, the red parts of the line is when we have a positive force.

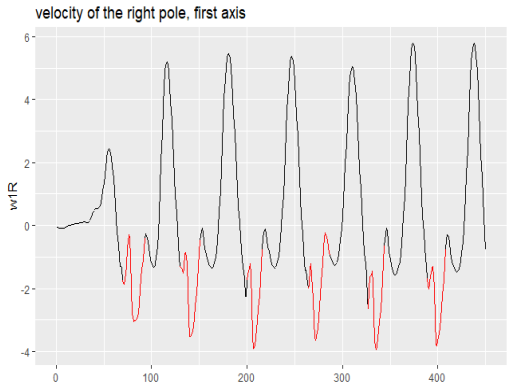

Figure A.4: Velocity of the right pole in the first axis, the red parts of the line is when we have a positive force.

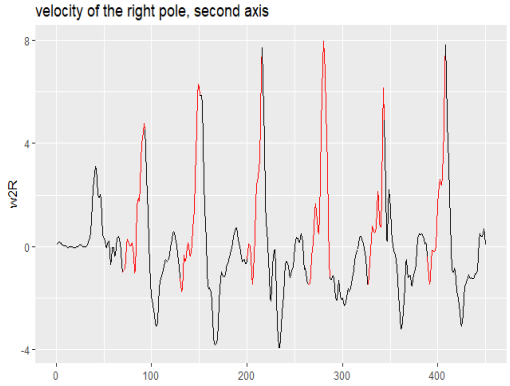

Figure A.5: Velocity of the right pole in the second axis, the red parts of the line is when we have a positive force.

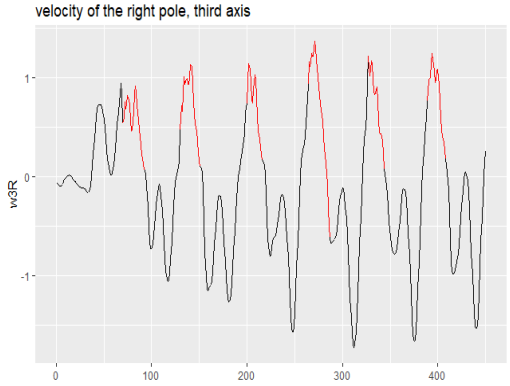

Figure A.6: Velocity of the right pole in the third axis, the red parts of the line is when we have a positive force.

Figure A.7: Angle of the right pole, the red parts of the line is when we have a positive force.

A.2 Distribution of a few features conditioned of style

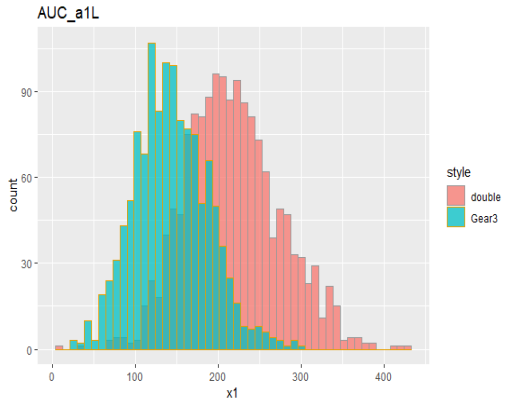

Figure A.8: Empirical distribution of the feature area under curve of a1L

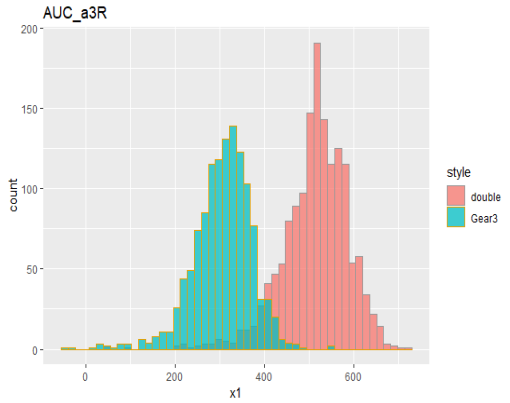

Figure A.9: Empirical distribution of the feature area under curve of a3R.

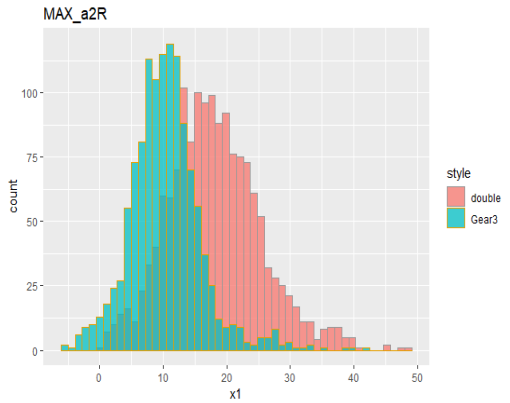

Figure A.10: Empirical distribution of the feature max a2R

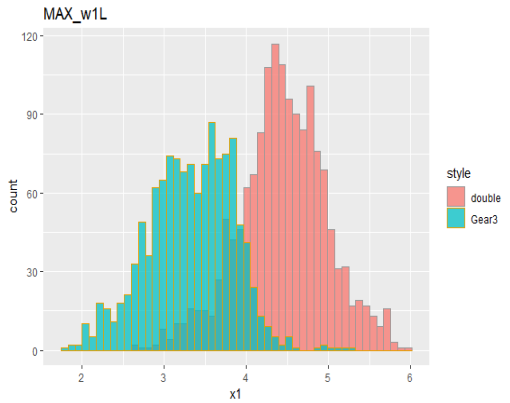

Figure A.11: Empirical distribution of the feature max w1L

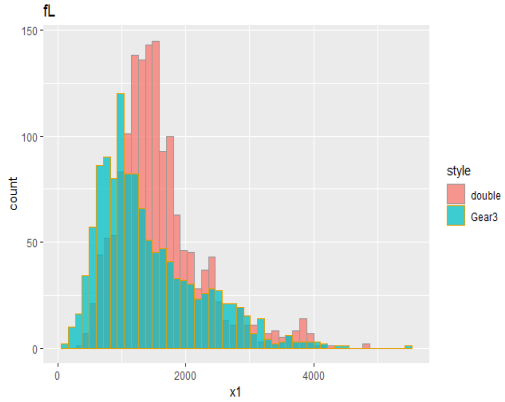

Figure A.12: Empirical distribution of the response variable force left when conditioned on style

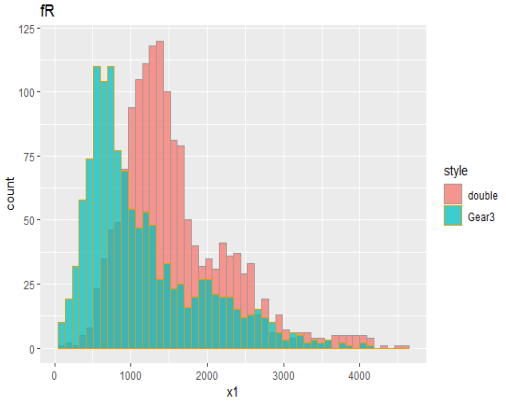

Figure A.13: Empirical distribution of the response variable force right when conditioned on style

A.3 Result of tuning for the models in design 3

mtry	R^2	$\hat{\sigma}$
2	0.934	0.007
4	0.935	0.008
5	0.937	0.009
6	0.937	0.009
8	0.938	0.010
25	0.934	0.015
30	0.933	0.017
38	0.930	0.020
60	0.924	0.029
61	0.924	0.029

mtry	R^2	$\hat{\sigma}$
15	0.708	0.056
17	0.707	0.058
23	0.694	0.065
26	0.692	0.063
29	0.688	0.065
35	0.677	0.066
36	0.678	0.069
46	0.667	0.072
55	0.662	0.073
60	0.662	0.071

Table A.1: The results of tuning the parameter *mtry*, the first column contains the value tested. The following two columns contains the prediction R^2 value and the standard deviation. The models was trained on data from the *Ciry* session, where the left table is for the style *Double* and the right table corresponds to the style *Gear3*. Both models had `fL` as response variable. The sample sizes for fitting the models where 338 and 181 respectively.

mtry	R^2	$\hat{\sigma}$
29	0.928	0.005
33	0.927	0.005
39	0.925	0.006
43	0.925	0.005
48	0.924	0.007
49	0.924	0.007
53	0.923	0.007
54	0.923	0.007
55	0.923	0.007
58	0.922	0.007

mtry	R^2	$\hat{\sigma}$
7	0.722	0.076
18	0.719	0.091
21	0.718	0.091
27	0.712	0.097
32	0.710	0.101
33	0.710	0.097
35	0.703	0.106
38	0.707	0.101
53	0.694	0.111

Table A.2: The results of tuning the parameter *mtry*, the first column contains the value tested. The following two columns contains the R^2 value and the standard deviation. The models was trained on data from the *Ciry* session, where the left table is for the style *Double* and the right table corresponds to the style *Gear3*. Both models had `fR` as response variable. The sample sizes for fitting the models where 338 and 181 respectively.

mtry	R^2	$\hat{\sigma}$
34	0.876	0.030
40	0.876	0.030
44	0.875	0.031
46	0.875	0.031
49	0.874	0.031
50	0.875	0.031
53	0.873	0.031
58	0.873	0.032
61	0.873	0.032

mtry	R^2	$\hat{\sigma}$
1	0.786	0.040
6	0.829	0.031
23	0.833	0.031
24	0.832	0.030
28	0.832	0.030
31	0.830	0.032
42	0.828	0.032
53	0.826	0.033
59	0.825	0.033

Table A.3: The results of tuning the parameter *mtry*, the first column contains the value tested. The following two columns contains the R^2 value and the standard deviation. The models was trained on data from the *Sätila* session, where the left table is for the style *Double* and the right table corresponds to the style *Gear3*. Both models had `fL` as response variable. The sample sizes for fitting the models where 1115 and 959 respectively.

mtry	R^2	$\hat{\sigma}$
3	0.798	0.042
7	0.812	0.040
8	0.812	0.039
9	0.814	0.040
25	0.813	0.039
36	0.809	0.040
46	0.808	0.040
56	0.807	0.040
60	0.807	0.040

mtry	R^2	$\hat{\sigma}$
1	0.815	0.029
4	0.852	0.026
9	0.864	0.025
36	0.869	0.025
41	0.868	0.026
44	0.868	0.026
55	0.867	0.027
60	0.8685	0.029
61	0.864	0.028

Table A.4: The results of tuning the parameter *mtry*, the first column contains the value tested. The following two columns contains the R^2 value and the standard deviation. The models was trained on data from the *Ciry* session, where the left table is for the style *Double* and the right table corresponds to the style *Gear3*. Both models had `fR` as response variable. The sample sizes for fitting the models where 1115 and 959 respectively.

A.4 Diagnostic plots

A.4.1 Models design 1

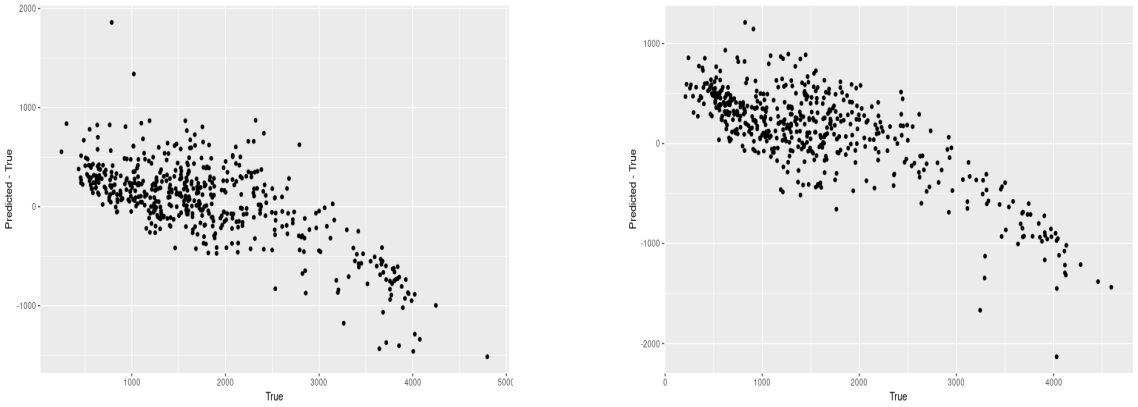

Figure A.14: Left: The residuals vs the true outcome from the model trained on the events from the sätila session with the task of predicting the force from the left pole. Right: The residuals vs the true outcome from the model trained on the events from the sätila session with the task of predicting the force from the right pole. Both figures shows that there is a trend in the residuals that the model is over estimating the force for low values of the true outcome and underestimating the force for larger values.

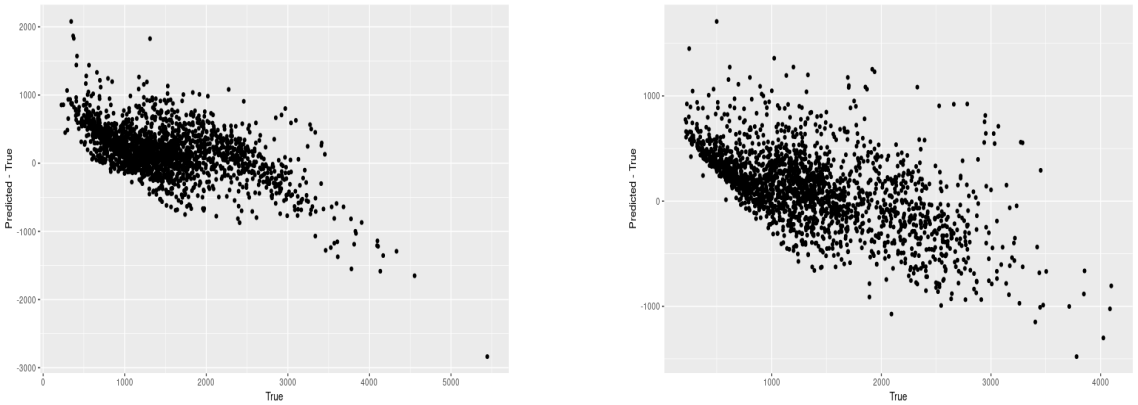

Figure A.15: Left: The residuals vs the true outcome from the model trained on the events from the City session with the task of predicting the force from the left pole. Right: The residuals vs the true outcome from the model trained on the events from the City session with the task of predicting the force from the right pole. Both figures shows that there is a trend in the residuals that the model is over estimating the force for low values of the true outcome and underestimating the force for larger values.

A.4.2 Models design 2

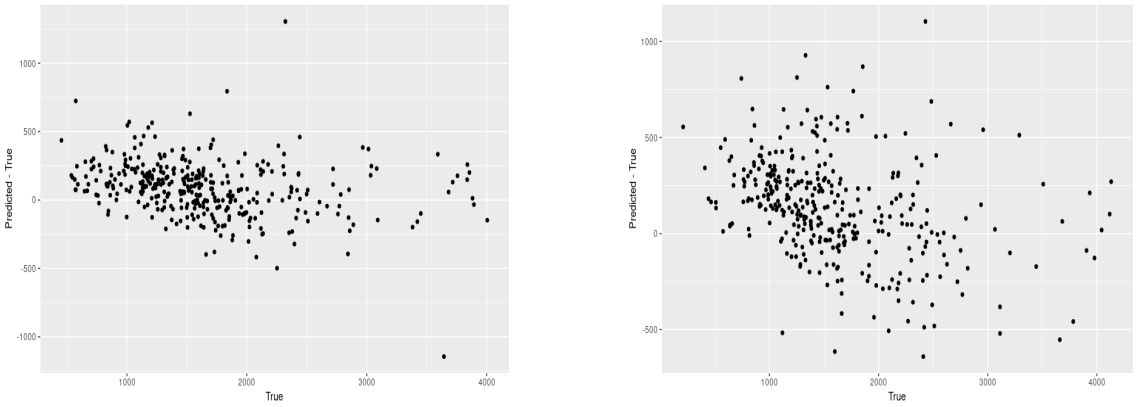

Figure A.16: Left: The residuals vs the true outcome from the model trained on events when the style *Double* was used with the task of predicting the force for the left pole. Right: The residuals vs the true outcome from the model trained on events when the style *Double* was used with the task of predicting the force for the right pole. Looking at the figures, it becomes clear that the model for predicting the left force is achieving better results, which also was shown in the table 5.8. The model for predicting the force from the right pole is over estimating the force for low values while for larger values it is both under and over estimating the force.

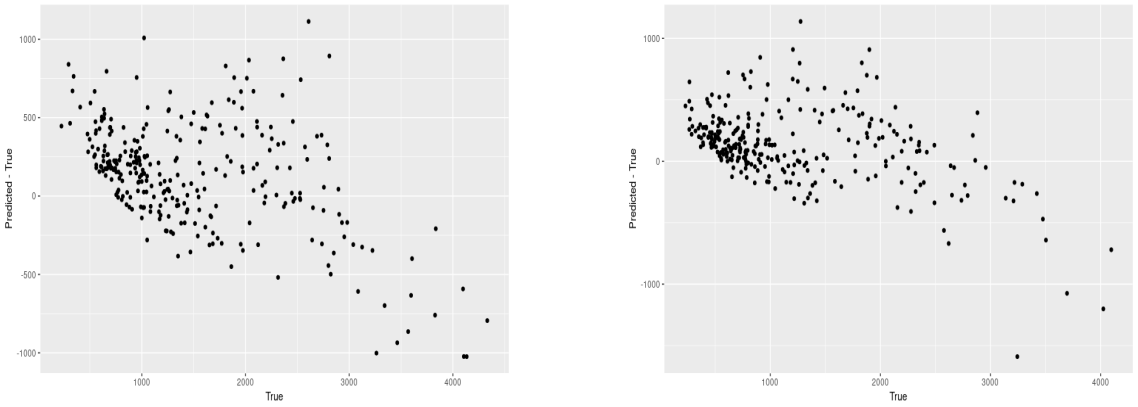

Figure A.17: Left: The residuals vs the true outcome from the model trained on events when the style *Gear3* was used with the task of predicting the force for the left pole. Right: The residuals vs the true outcome from the model trained on events when the style *Gear3* was used with the task of predicting the force for the right pole. As in the previous figures in this section, the models seems to overestimate the force for lower values and underestimating the force for larger values.

A.4.3 Models design 3

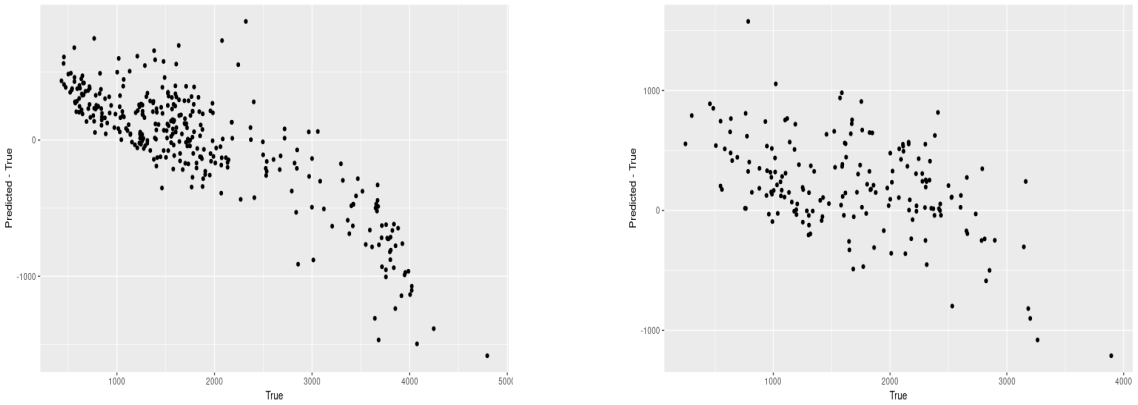

Figure A.18: Residuals vs true force for the left pole. Left figure is the model with trained on events from the *Sätila* session when the style *Double* was used. Right figure is the model with trained on events from the *Sätila* session when the style *Gear3* was used.

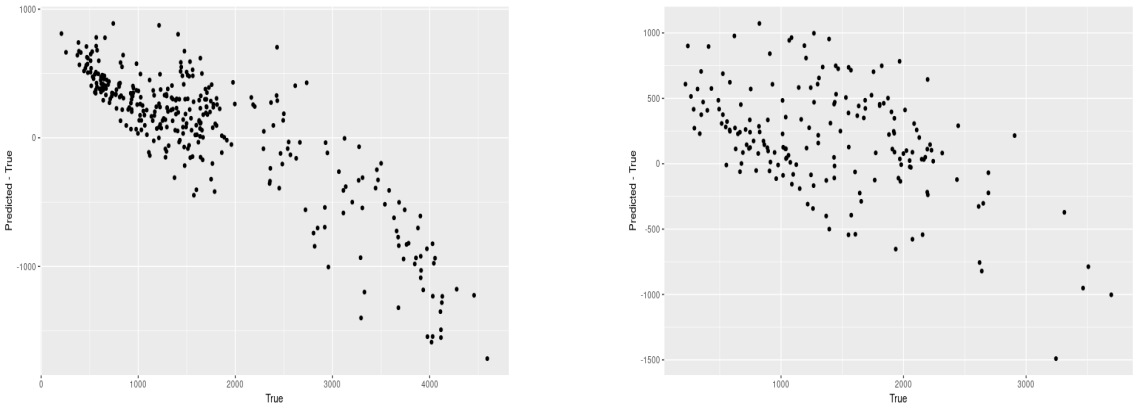

Figure A.19: Residuals vs true force for the right pole. Left figure is the model with trained on events from the *Sätila* session when the style *Double* was used. Right figure is the model with trained on events from the *Sätila* session when the style *Gear3* was used.

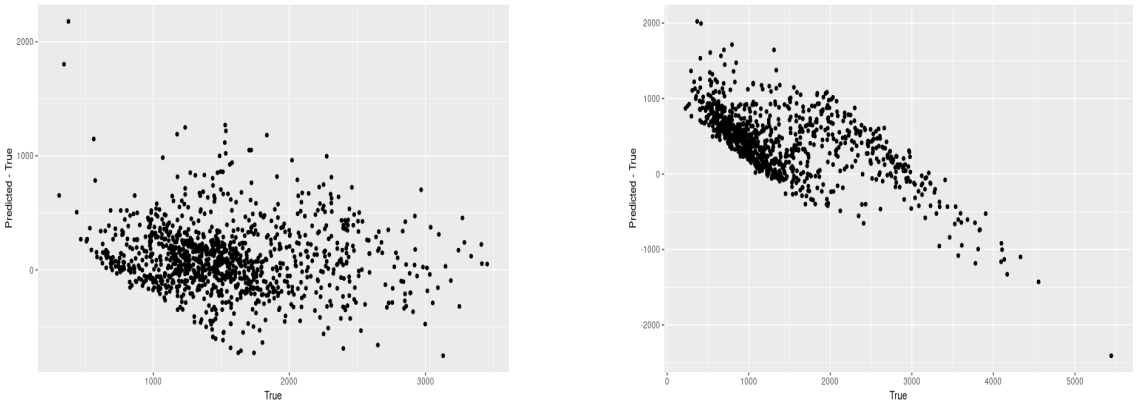

Figure A.20: Residuals vs true force for the left pole. Left figure is the model with trained on events from the *City* session when the style *Double* was used. Right figure is the model with trained on events from the *City* session when the style *Gear3* was used.

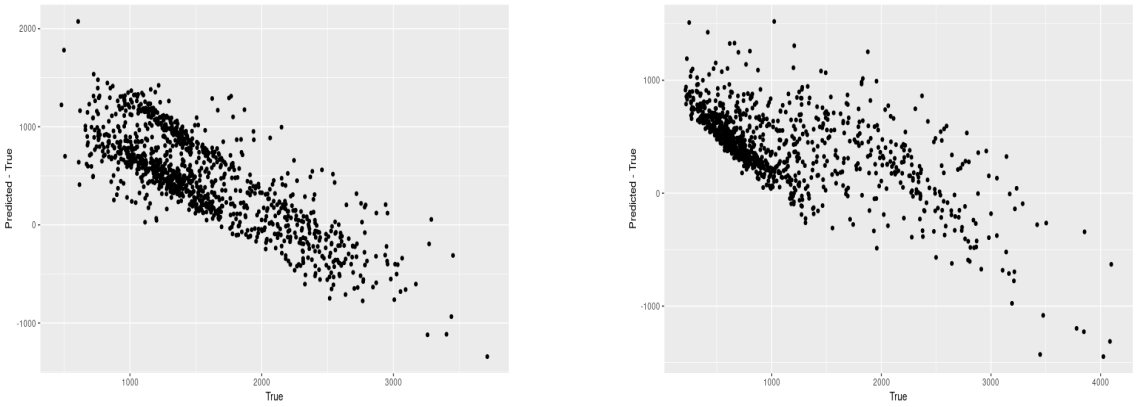

Figure A.21: Residuals vs true force for the right pole. Left figure is the model with trained on events from the *City* session when the style *Double* was used. Right figure is the model with trained on events from the *City* session when the style *Gear3* was used.