Wind inference from GPS tracks of sailboats

Vladislav Chumak

Master's thesis

Software Engineering degree course

Faculty of Computer Science

Hochschule Mannheim

28.09.2018

Conducted in cooperation with SAP SE, Walldorf

University supervisor: Prof. Dr. Ivo Wolf, Hochschule Mannheim

Company supervisor: Dr. Axel Uhl, SAP SE

Declaration

This thesis is the result of my own work and includes nothing, which is the outcome of work done in collaboration except where specifically indicated in the text. It has not been previously submitted, in part or whole, to any university of institution for any degree, diploma, or other qualification.

I agree that my work is going to be published, which means that the work gets electronically stored, converted into other formats, made publicly accessible on the servers of Hochschule Mannheim and distributed over the Internet.

Mannheim, 28.09.2018

Vladislav Chumak

Abstract

Performance sailing is a popular competitive sport where sailors compete against each other by sailing to a destination in the fastest way possible. The main challenge in this activity is to predict the fastest route to destination considering the current sea state and wind conditions. While professional sailors are able to perceive these circumstances and derive the optimal route on the fly without any usage of measurement instruments, the audience of sailing competitions is still in demand for that kind of information in order to understand what is going on in the sailing area. The information is also required by various sailing software in order to provide performance analysis and sailing assistance features for sailors of any level.

This thesis is dealing with determination of true wind data from given GPS-tracks and characteristics of sailboats. More precisely, a technical component is being constructed which estimates the direction and the speed of true wind by means of provided GPS-tracks and polars of sailboats. The component is meant to be used in sailing software as default wind data in cases when there is no other wind information available. The derived wind information might be not as accurate as provided by external wind measurement systems, but it could be still accurate enough to serve as guidance, as well as the input for various analytics to facilitate maneuver type and point of sail determination.

Within this work, efforts were made to investigate the solution space for wind inference considering multiple approaches from data and computer science. There have been five wind estimator concepts designed and implemented in Java. Four of the concepts rely on maneuver classification which was leveraged by application of supervised machine learning methods. Thanks to SAP, more than ten million of sailing maneuvers with known labels could be analyzed and employed in the learning process of classification models. All steps of data analysis and data preparation for machine learning is covered in detail by this thesis. To determine the most appropriate machine learning method for maneuver classification, a benchmark was conducted with nine classification algorithms. The benchmark revealed that the most accurate maneuver classification is leveraged with neural network-based classifier. Hence, the classifier was adapted accordingly to be compatible with Java-based wind estimation models. Finally, all engineered wind estimators were evaluated in terms of its estimation accuracy. The best wind estimation results were achieved with Hidden Markov Model-based wind estimator.

Acknowledgements

I would like to acknowledge and thank the following important people who have supported me, not only during the course of this work, but throughout my Master's degree.

Firstly, I would like to express my gratitude to my company supervisor Dr. Axel Uhl from SAP SE. He provided me the opportunity to work on the most exciting topic of computer and data science that I could ever dream of. Thanks to you Axel, I could acquire solid knowledge about Data Science. Its combination with my existing Software-Engineering skills opens a new horizon of possibilities which allows me to proceed my career with joy and excitement. Thank you for this opportunity. I would also like to thank you for your overwhelming ability to criticize and question everything what does not feel right. Your critics were all constructive and beneficial for me and this work. It helped me tremendously to improve my way of thinking and to develop new ideas. Last but not least, I would also like to thank for the permitted Balinese Home Office opportunity.

Secondly, I would like to thank to my university supervisor Prof. Dr. Ivo Wolf from Hochschule Mannheim for his willingness to supervise my work. All your feedback regarding thesis was beneficial and highly appreciated. I would also like to thank you for your permission to extend the time frame of this thesis, as well as to break the conventional size limit of a master thesis and write as many pagers as necessary in order to present its subject in most understandable and transparent manner.

Throughout this work, I was visiting various meetups which were organized for data science enthusiast free of charge. The most meaningful insights I have collected within a workshop conducted by Venkatesh Tadinada. He showed me the well-developed data science ecosystem of Python language. Thanks to his workshop I could make one of the most significant decisions for this work which was to perform the whole data analysis and machine learning method evaluation in Python instead of Java. It allowed me to save time and to employ with ease various complex machine learning techniques to collect additional useful insights concerning feature engineering and hyperparameter tuning. I am also highly appreciating his literature recommendations which supported me vastly throughout this work.

Furthermore, I would like to thank Dirk Sohn from OIO who hired me to work for SAP SE with Dr. Axel Uhl. I also appreciate his choice, hiring me as working student during my Bachelor study. Additionally, I acknowledge with thank all my former OIO colleagues, who helped me to acquire my Java expertise. In particular, I would like to

thank Steffen Schäfer for his valuable know-how sharing and initial support in SAP Sailing Analytics project.

I am very much thankful to my family and my girlfriend, who supported me mentally throughout the whole work. You give me the inspiration and power to get through all obstacles I am facing. I am glad to have you.

And finally, I would like to finish my acknowledgements considering all further committers to this work:

- RaceQs for its prompt and friendly answers on interview questions regarding its own implemented wind estimation.
- Members of Forums sailinganarchy.com and sailing.net for doubting the feasibility and the use of my work. It inspired to successfully complete my tasks in order to sort out your doubts and underpin the importance of will for innovation.

Brief Table of Contents

Detailed Table of Contents

1 Introduction

This chapter provides an overview about the whole thesis regarding its context, goals and structure.

1.1 Motivation

Sailing vessels have been of great importance for trade, transport, warfare and fishing for many centuries. However, in the 19th century engine-driven ships evolved and consequently, sailing vessels got displaced. Nowadays, sailing is mainly regarded as a leisure activity and competition sport. The competitions are conducted, as well as broadcasted on TV worldwide. It is also no longer rare that regular newspapers write about it.

Overall, sailing sport has become popular. It is sponsored by many prominent companies, including **SAP**[1]. Besides financial help, SAP supports sailing sport with its technological commitment. It provides an online platform for sailing events called **SAP Sailing Analytics**[2] where users get informed about upcoming, currently running and already finished sailing events. Moreover, the platform provides to its audience a graphical view on races by visualizing the competing sailboats on a map with a configurable overlay of interesting analytical information, various charts and leaderboard. The software is actively used by sailing fans, sailing clubs, event organizers, commentators and reporters.

In order to provide its features, the SAP Sailing Analytics platform interoperates with various external systems to retrieve its required data. Such data includes GPS-fixes of sailboats, wind tracking information within the competition area, position of buoys and many more. Additionally, SAP provides multiple smartphone and tablet apps for tracking and event organization. The software system of the platform is huge and continues to grow due to its active development. It is part of the marketing strategy of SAP which intends to demonstrate superior technological skills of the company. This work has been initiated on behalf of SAP to investigate the solution space for the next advanced

[1] *"SAP is a German multinational software corporation that makes enterprise software to manage business operations and customer relations."* – see http://www.sap.com

[2] SAP Sailing Analytics is publicly available at http://www.sapsailing.com

feature for its sailing platform. The results of this work provide a positive example for bridging the worlds of theoretical and practical data and computer science.

1.2 Goals

The SAP Sailing Analytics platform requires wind tracking information within the sailing area in order to provide its full functional spectrum. If the wind information is not available, the platform will not be capable of estimating optimal sailing courses, showing performed maneuver types and computing important measures for performance analytics.

The main goal of this work was to prevent the limited operation of SAP Sailing Analytics by introducing default wind data for sail races. The default wind data is meant to be used for cases, when there is no wind data provided by external measurement systems. In such cases, only GPS-tracking data of participating sailboats and the names of its boat classes are available.

The GPS data allows derivation of various other measures, such as speeds and courses which might correlate with the wind. By means of boat class, various boat specific characteristics may be obtained. For this work, sailing polar performance is of special relevance because it describes target boat speeds by a given true wind angle and true wind speed[3].

One task of this work requires identification of relevant features within GPS-track and its correlations with the true wind. Then, appropriate concepts for estimation of true wind speed and true wind direction must be elaborated. The most promising concepts must be devised in detail and implemented in a compatible way to the backend of SAP Sailing Analytics which is Java-based[4]. Additionally, each implementation shall be evaluated against the huge sailing database of SAP Sailing Analytics. Finally, all achieved wind estimator implementations must be evaluated and compared.

1.3 Contributions

The principal contributions of this thesis are the following:

[3] Sailing terms, such as true wind, are introduced in chapter *2 Sailing basics.*

[4] Java is a popular object-oriented platform-independent programming language, see https://www.oracle.com/uk/java/index.html

- Working maneuver detection operating on GPS-tracks of sailboats
- Elaborated correlation concepts concerning true wind estimation and its confounders
- Determination of maneuver features to be used for maneuver classification in terms of its type
- Determination of the most appropriate machine learning algorithm for maneuver classification
- Machine learning library evaluation for Java
- Elaboration of five wind estimator concepts and its evaluation

1.4 Outline

In order to reflect the course of this thesis, the outline of this work has been designed in accordance with the applied process model which leans heavily on **Cross-industry standard process for data mining** (CRISP-DM)[5]. The outline consists of the following chapters:

- **Business understanding:** Focuses on understanding of the subject, context and objectives of this thesis. At first, foundations of sailing theory and SAP Sailing Analytics project are conveyed which are required for understanding of this work. Then, a market research is conducted to gain a picture about the current state of the art in terms of existing wind estimators. Afterwards, the problem of this thesis gets precisely defined and mapped to a machine learning problem with possible proposals for solution. Then, machine learning basics are conveyed on which this thesis builds. Finally, the appropriate technologies for machine learning are introduced which will be employed throughout this work.
- **Data understanding:** Focuses on understanding of data involved in this work. Firstly, the data gets examined to obtain its meaning, format and access options. Based on the available data, first generic approaches for the input-output mapping with its disruptive factors are elaborated.
- **Data preparation:** The data of SAP Sailing Analytics platform gets analyzed and prepared for wind estimation models. Firstly, the data quality is improved by enhancement of SAP Sailing Analytics maneuver detection algorithm. Then, the demanded data is identified and loaded into local environment. Afterwards,

[5] CRISP-DM is the de facto standard process model for data mining and knowledge discovery projects, which was developed by a consortium of many companies involved in data mining [17, p. 17].

data cleansing is performed to sort out the data which could disrupt the wind estimation. Finally, the data gets analyzed and the final feature set for maneuver classification is engineered.

- **Modelling:** Five concepts for wind estimation are devised and implemented. Since four of the concepts rely on maneuver classification, the modelling of maneuver classification is also introduced.
- **Evaluation:** All engineered models including maneuver classification get evaluated and compared regarding its estimation performance.
- **Ending:** The achieved results are summarized. Possible improvements and further ideas are presented, followed by a look into the future.

1.5 Conventions

To facilitate the legibility of this thesis, the following conventions were elaborated for this document:

- Quotations are highlighted with *cursive font*, followed by source number in square brackets, e.g. [1]. Direct quotations are additionally enclosed with quotation marks. An example of a direct quotation is provided as follows: "*Analysis of variance (ANOVA) can determine whether the means of three or more groups are different.*" [1].
- Important terms are highlighted in **bold**. Bold font is also used to emphasize a particular important word.
- Chapters and sections are referred by its number and name which are highlighted with *cursive font*.
- All non-obvious acronyms and abbreviations are written in full by its first occurrence. The full term of an abbreviations can be looked up in *List of Abbreviations and Acronyms* which is located at the end of this thesis.
- Mathematical notations, formula and variables are highlighted with a $special\ cursive\ font$.
- Code specific names of interfaces, classes, variables and methods are highlighted with a `monospace font`. The reference of methods by its name is omitted. Instead, the methods are introduced as input-output relations in the following format:
  (input parameter 1, …, input parameter n) → (output)
- Within listings with pseudocode, the following syntax is used:

- `functionName(param)`: call of a function/method with name `functionName`, providing it `param` as parameter
- `variableName := value`: New variable with name `variableName` is declared and initialized with `value`
- `variableName[1]`: Index-based access of the first element of `variableName`, assuming that `variableName` is a list or an array
- `variableName[symbol]`: Symbol-based access of the element contained in `variableName` with key `symbol`, assuming that `variableName` is a map or an associative array
- Unintentional line break within a code statement is denoted with ellipses before the line break and after the line break in the following manner:

```
variableName := initialize variable by describing its …
  … very long initialization strategy
```

- Nested blocks are indicated by space indents in front of each code statement. Each new code block starts after an expression which ends with a colon, e.g.:

```
global code block
if statement:
  nested code block of the if statement
global code block again
```

2 Business understanding

This chapter provides the whole background knowledge required to understand the course of this thesis, including sailing fundamentals and SAP Sailing Analytics platform. Additionally, a market research is conducted to get familiar with the current state of the art and to infer the key concepts of existing solutions for a GPS-based wind estimation. Afterwards, the problem of this work gets precisely defined and mapped to the fields of data science and machine learning. Then, machine learning basics are conveyed which are required for understanding of this work. And finally, the most suitable machine learning technologies are selected.

2.1 Sailing basics

This section conveys the sailing foundations which are relevant for the context, as well as for the further course of this thesis. It presents navigation tricks for getting around by a sailboat. Further details are given about the different wind types and their meaning for the context of this thesis. Additionally, important sailing terminology is introduced.

2.1.1 Navigation

Due to sailing physics it is impossible to sail directly toward the wind. Nevertheless, most sailboats are still capable of sailing 45° upwind. Consequently, in order to reach a target destination which is directed toward the wind, a zigzag course must be sailed.

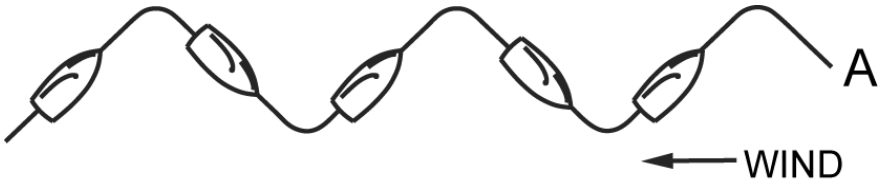

Figure 1: Sailing against the wind in a zigzag course [2]

In sailing terms this tactic is called **tacking**. The maneuver, which is performed in order to get from one side of the wind to another by maneuvering the boat's bow[6] through the **dead wind zone**, is called a **tack** [3, p. 102].

In contrast to upwind, there is no dead wind zone when sailing downwind. However, sailing straight away from wind is considered as dangerous due to inadvertent shifts of

[6] Bow is defined as the front end of the boat [3, p. 33]

the boom[7] between both wind sides which can result in boat damages and injuries of the sailing crew. Moreover, due to its characteristics, many sailboats perform slower when sailing straight away from wind than when sailing downwind in a zigzag course. Thus, the zigzag strategy is also applied when the direction to destination is toward the wind flow is required.

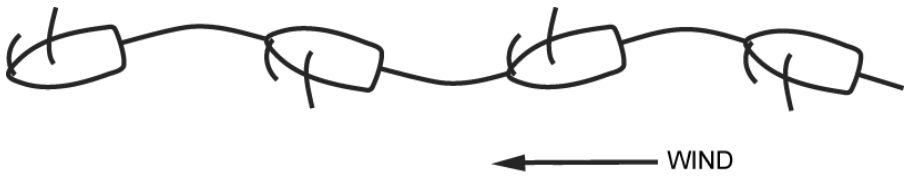

Figure 2: Sailing with the wind in a zigzag course [2]

In terms of sailing, this tactic is called **jibing**, and the corresponding maneuvers are **jibes**. Within a jibe, the bow of the boat is turned through the wind which is coming from behind [3, p. 105].

There are no limitations for any other wind dependent sailing courses. A sailboat is capable of going upwind, downwind, and even perpendicular to the wind. Therefore, any target from any direction can be reached.

2.1.2 Sailing maneuvers

There are various maneuvers, including the already mentioned tack and jibe, which can be performed by a sailboat in order to adapt its course. This section introduces the terminology for sailing maneuvers which will be used throughout this thesis.

Tack: The boat's bow is steered through the wind blowing directly from ahead [3, p. 102].

Jibe: The boat's bow is steered through the wind which is coming directly from behind [3, p. 105].

Head-up: Change of boats course toward the wind [4].

Bear-away: Change of boats course away from the wind [4].

Each of the mentioned maneuvers may be performed in two different directions. Similar to direction indication by means of "left" and "right", the terms "port" and "starboard" are applied in nautical language which are defined as follows [3, p. 34].

[7] Boom is *"the horizontal spar which is attached to the mast to support the bottom part of the mainsail"* [3, p. 33ff.]

Port: left side of the boat when facing the boat's bow

Starboard: right side of the boat when facing the boat's bow

The following figure exemplifies these terms.

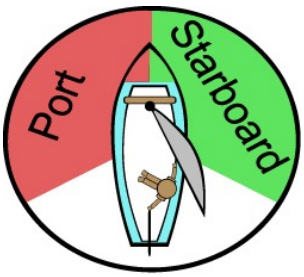

Figure 3: Starboard vs. port [5]

2.1.3 Course, heading, bearing, track, route

In contrast to a sailboat, a car is always driven to the direction, where the nose of the car is pointing to. Therefore, in context of cars, no distinction is being made between its course and heading, as they appear identical. In sailing, however, multiple forces are acting on sailboat, which can result in a drift between heading and course. Therefore, the two terms must be defined separately, as they vastly affect this thesis. In nautical context, the **heading** of a boat expresses the direction, where the boat's bow is pointing to, whereas the **course** of a boat represents the direction, where the boat is currently moving to in relation to the ground [6, p. 125f.]. The former can be obtained by using compass, whereas the latter from GPS. In order to express the direction from one point to another point, the term **bearing** is used [6, p. 126].

It is usual to express heading, course and bearing in **degrees**, starting from the north counting clockwise. However, the north can be referred either as **magnetic** or **geographic** north. Throughout this thesis, all directions indicated in degrees will refer the geographic north as its null value.

In case of a strong current hitting the boat sideways, the drift between course and heading gets severe. A notable drift can even occur in an area free of currents because there are multiple aerodynamic forces acting on sail including **lift** and **drag**. While the lift force helps to push a sailboat toward its heading, the drag force drags the boat to the direction of the wind flow.

Besides current and wind, there are many other factors affecting the drift between course and heading, including air pressure, sail shape, boat shape, sail trim and course. Due to instability of influencing factors, the drift is continuously changing. Considering

this, sailing navigation becomes complex. In order to reach a certain destination, a **route** needs to be planed considering prevailing wind, sea state, as well as the map and boat characteristics. Then, the sailboat must be steered accordingly to follow the planed route. The aim for the skipper is to sail a **track** which matches the route as far as possible.

2.1.4 Sailing wind

During sailing, the perceived wind on boat is constantly changing. The reason for this effect is the boat's movement [7, p. 31]. For instance, if a motorboat is traveling in a wind-free zone, e.g. to the north with 10 knot speed over ground (SOG), the perceived wind on the boat will be about 10 knots coming from the opposite direction, in this case – the south. If the boat stops in a no-wind area, no wind might be perceived if the waters in the area are calm and free of currents. However, if the boat catches a current in a no-wind area during its halt without anchor, there will be still some perceptible wind because the boat gets moved by the current.

As already stated, this thesis is dealing with wind determination. And as the upper section preluded, there are multiple winds which may be perceived and determined. This thesis targets the **true wind**. True wind is the wind which is perceived when standing still on a non-moving ground [7, p. 31]. The wind which is perceived during movement is called **apparent wind** [7, p. 31]. **Current adjusted wind** is a special case of apparent wind which is encountered on a halted boat when it is not set on anchor and is moved by prevailing current. The relationship between boat b, true wind w and apparent wind w' may be expressed by an equation of its motion vectors as follows:

$$w' = w - b \quad \Big| \quad w, w', b \in \mathbb{Q}^2$$

The equation allows to deduce that the apparent wind gets stronger when sailing upwind and weakens by sailing downwind. Furthermore, if the boat is not going directly against the true wind or straight away from the true wind, the direction of the apparent wind will differ from the direction of the true wind.

To describe the wind conditions, measurement of **wind direction** and **wind speed** is required. In context of true wind, the terms **true wind direction (TWD)** and **true wind speed (TWS)** are used. In sailing world, the wind speed is often expressed in **knots** (**kn**), whereas the direction is either represented in a coarse-grained manner by means of **cardinal directions**, such as north, south, southwest and etc., or in a fine-grained manner by means of degrees. Analogously to heading and course, the wind direction in

this thesis will always refer the geographic north as its null value. The wind direction indicates the direction from which the wind is blowing.

The true wind does not remain same over time and can also vary between different areas. Thus, the true wind is regarded as a time and position dependent variable.

2.1.5 True wind angle

The **true wind angle (TWA)** measure is used to describe the angle between boat's heading and TWD. It is expressed in degrees within the interval $(-180; 180]$ where the null value is referring the TWD. A negative TWA value means that the wind is approaching the sailboat from its port side, whereas a positive TWA value implies the wind coming from the boat's starboard side. The relationship between sailboat's heading, TWD, TWA, apparent wind angle (AWA) and apparent wind speed (AWS) is exemplified by the following figure:

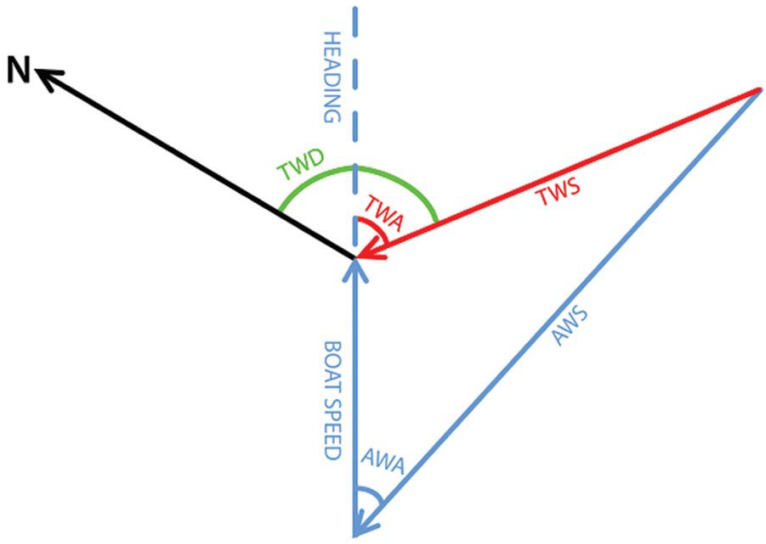

Figure 4: Relationship between TWA, TWD, boat's heading and etc.[8]

2.1.6 Tack

To express the side from which the wind is approaching the sailboat, the term tack is used. Starboard tack implies a positive TWA and thus, the wind coming the boat's starboard side, whereas port tack implies the opposite. It should be noted that the term

[8] Source: *URL <https://www.sailingworld.com/how-to/instruments-and-real-angles#page-2>, Accessed 10.07.2018*

tack is context sensitive and thus, can also refer to the maneuver type with equal designation.

2.1.7 Polars

The sailing performance of a sailboat depends highly on the characteristics of its corresponding **boat class**. A polar chart is meant to describe this performance considering the TWA [8, p. 35].

Besides TWD, TWS affects the boat's performance as well. Therefore, polar charts are usually composed of multiple lines, where each line is reflecting the target boat speed per TWA considering either a particular TWS value, or a TWS value range. A polar chart is usually drawn in a circular shape to improve its readability. An example of a polar chart related to multiple wind speeds is illustrated below.

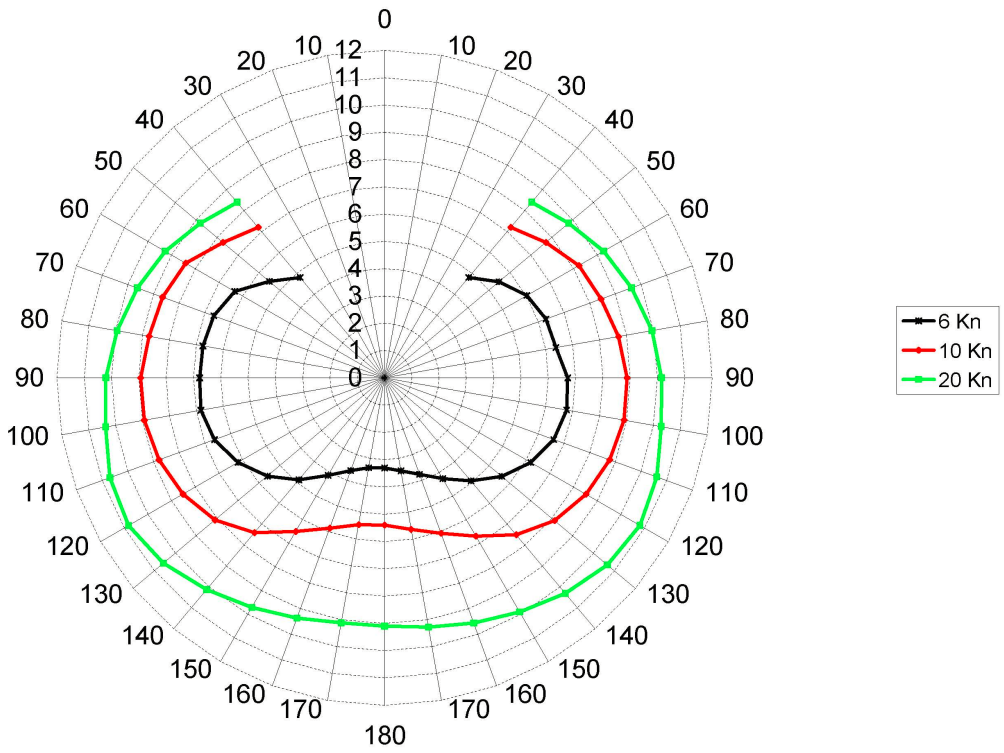

Figure 5: Polar chart example[9]

[9] Source: *URL <http://hanse470.com/polar-chart/>, Accessed 14.03.2018*

The TWA labels shown in the chart above do not contain negative values due to legibility. The values in [-40°; 40°] are not displayed because it represents the non-sailable TWA-range zone of the corresponding sailboat.

The polars for a certain sailboat are usually provided by its manufacturers. However, one can also generate the polars itself. This can be achieved by sailing different TWAs at a stable speed in a best way possible. However, the recording process of polars can get distorted by prevailing sea state. Therefore, ideal polars require the recording process to be performed within an area with stable wind and minimal sea state activities so that the shape of the recorded polars preserves a symmetry axis crossing the TWA values 0° and 180°.

2.1.8 Point of Sail

TWA allows a precise description of boat's heading in relation to TWD. However, to express the heading without degrees in a more coarse-grained manner, the names of points of sail are used. There are three basic points of sail [3, p. 94]:

- Beating: wind is from ahead
- Reaching: wind is from the side
- Running: wind is from behind

All of those point of sails can either be hold on starboard tack, or on port tack. The complete terminology for all points of sail is provided by the following illustration.

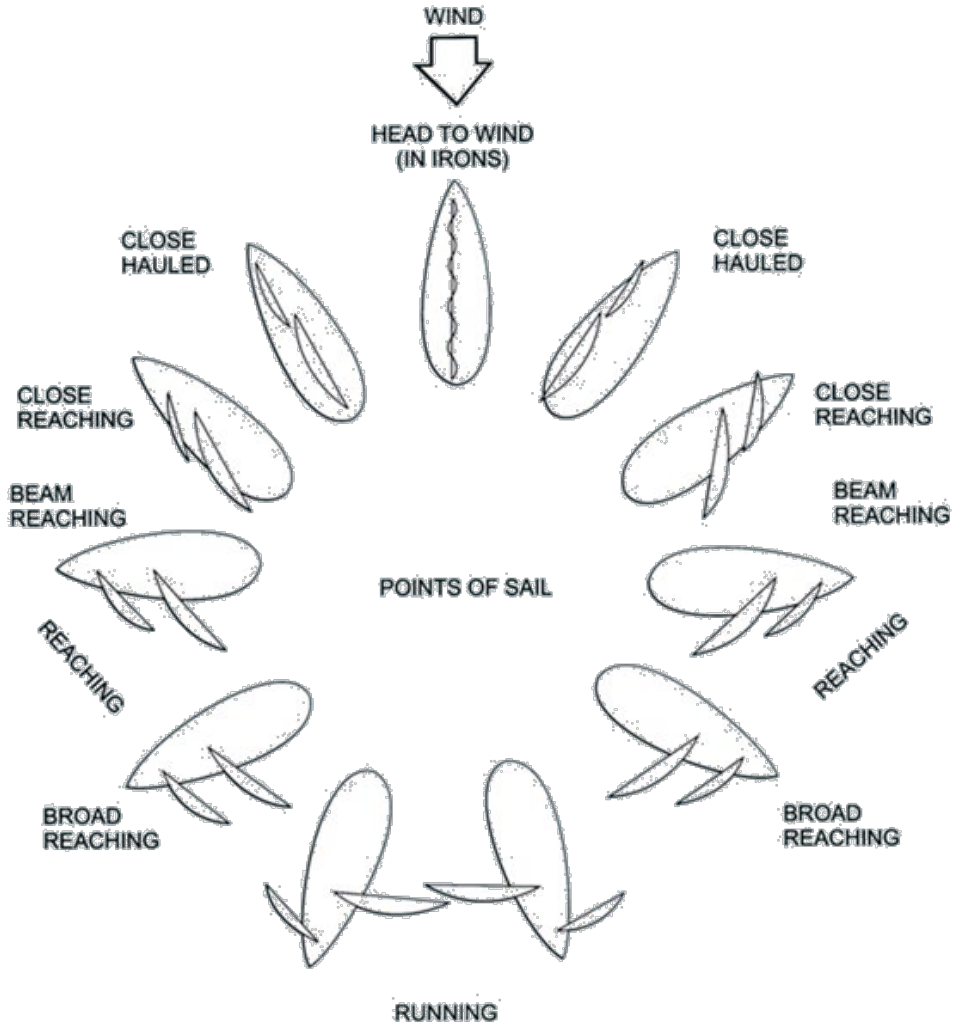

Figure 6: Points of sail [3, p. 93]

2.2 SAP Sailing Analytics project

This work has been initiated on behalf of SAP in order to enhance its online platform for sailing events, which is called SAP Sailing Analytics. It is part of SAP sponsorships program, in which SAP is demonstrating its technological skills by managing the whole development of the platform. The platform has been already successfully employed in thousands of sailing events, and its demand is still rising. And this work deals with the privilege, to extended this platform by another innovative feature – GPS-track based wind estimation. Since this work has a strong overlap with SAP Sailing Analytics, this section provides a concise explanation about the relevant technical parts and its context. However, due to the size and complexity of the platform, a complete description of all its features with its architecture and technological stack are omitted.

2.2.1 Sail racing and terminology

The target audience of SAP Sailing Analytics are events with racing sailboats. Within a race, multiple sailors are competing against each other by sailing a predefined path as fast as possible. The path gets defined by a **start-line**, zero or more **waypoints**, and a **finish-line**.

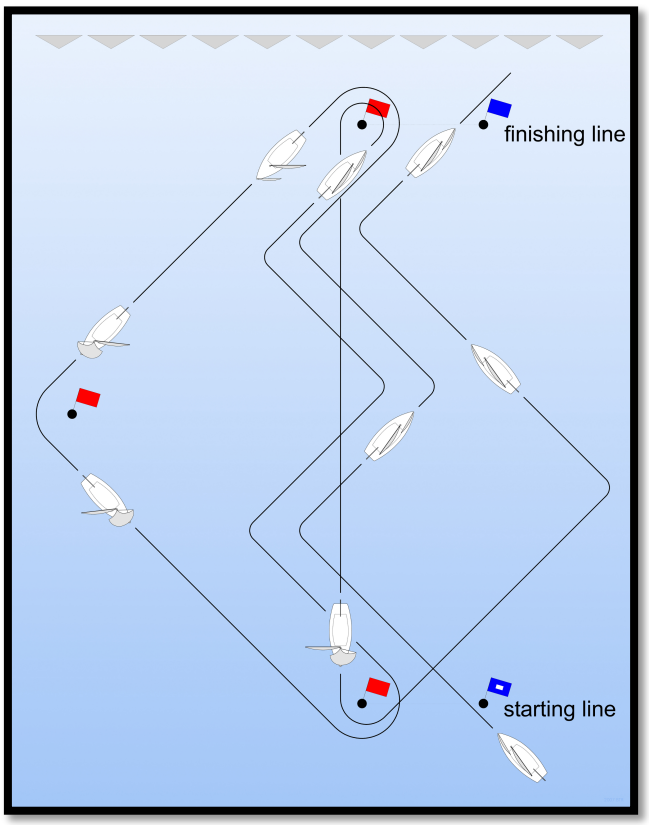

Figure 7: An example of predefined race path[10]

All three components are represented by **buoys** which are also referred as **marks**. In order to represent a start-line or a finish-line, a mark for each end of the line is required. A waypoint, however, may be designed in various ways, depending on its demanded passing strategy. There are two possible mark passing strategies:

- Rounding a buoy from a predefined side, such as port, or starboard. Its representation requires only one buoy.
- Sailing between two buoys, analogously to start- and finish-line.

Each section between waypoints is called **leg**. Thus, the race plan illustrated above is composed of six legs.

[10] Source: *URL <https://commons.wikimedia.org/wiki/File:Olympisches_Dreieck_en.jpg>, Accessed 14.03.2018*

2.2.2 Additional maneuver types

Besides the maneuvers described in *2.1.2 Sailing maneuvers*, in the context of SAP Sailing Analytics, there are following additional maneuver types defined:

Mark passing: The boat is steered to pass a buoy and to rearrange its course to the next buoy. Usually, a mark passing maneuver is performed by starting from a downwind course and ending in an upwind course, or vice versa.

Penalty circle: Special maneuver which gets imposed by judge due to a penalty. A penalty circle gets acknowledged when the maneuver arc contains a tack and a jibe. Hence, the maneuver angle of a penalty circle gets ca. 360 degrees.

2.2.3 SAP Sailing Analytics data

The first release of SAP Sailing Analytics was in 2011. Since that time it was used in a large number of sailing events including the prominent ones such as *Extreme Sailing Series*[11] and *Kieler Woche*[12]. Thanks to its active usage and wide portfolio, SAP Sailing Analytics platform has collected a huge amount of sailing data. The sailing data includes:

- Event information, e.g. name, date, races, competitors and boat class of sailed sailboats
- GPS-fixes of competing sailboats
- GPS-fixes of buoys which define the start-line, finish-line and waypoints to pass
- Wind measurement records from various wind measurement devices at various positions

The raw GPS data is processed by SAP Sailing Analytics to deduce further data:

- Maneuvers with its measures
- Polar data for each boat class

The following statistics about SAP Sailing Analytics data shall give an impression about the amount of data made available for this work:

- 928 regattas
- 12.151 races

[11] https://www.extremesailingseries.com/about

[12] http://www.kieler-woche.de

- 162.451 boat tracks
- 448.634.264 GPS-fixes
- 11.130.685 maneuvers
- 1.609.502 total sailed distance in km

2.2.4 Boat tracking

The boats get tracked by various GPS tracking devices. Depending on the event budget, the variety of used GPS-trackers ranges from smartphones to high-quality trackers with costs of multiple thousands of euros. Therefore, the quality of GPS-tracks can vary from race to race.

2.2.5 True wind measurement

Depending on the event budget, the options for wind measurement can vary as well. Within high-budget events, multiple advanced wind measurement systems on different locations are employed. These systems are capable of transmitting the wind measurement records to SAP Sailing Analytics platform automatically, without the need of human intervention. Within the platform, the wind measurement records get aggregated by advanced algorithms to infer the wind for all positions within the race area. SAP Sailing Analytics also provides the option for manual input of wind records. This case is particularly suitable for low-budget events where organizers watch the wind with inexpensive devices such as wind vane and anemometer and manually enter the records via UI. Therefore, the wind measurement quality can also vary from race to race.

2.2.6 Polar data collection

The GPS-data collected by SAP Sailing Analytics platform is employed to induce sailboat polars for all boat classes which have participated in races. The boat class for each competitor of a race is known because it is a mandatory input for event organization with SAP Sailing Analytics. Based on the measured wind, and information about the participating sailboats including its boat classes and GPS-tracks, the aggregation of the data is performed by a complex proprietary solution to induce the final boat class polars.

2.2.7 SAP Sailing Analytics code

The SAP Sailing Analytics platform is based on Java programming language. The whole code of the platform has been made accessible for this work. Thus, appropriate adjustments to the platform can be made in order to increase the compatibility level with the wind estimation components. The code can be also used to increase the understanding about the data generated by SAP Sailing Analytics. Furthermore, the code allows execution of the platform on a local machine.

2.3 Market research

The topic of this thesis appears to be not new to the sailing app market. There are already several apps for smartphones and tablets which pretend to master the wind estimation based on GPS by means of own proprietary solutions. Some of the apps are even backed by an online platform. Although the solutions with its concepts are closed-source for public and science, simple user tests still reveal few basic ideas about their input and output, as well interesting clues during runtime behavior. Therefore, several apps with wind estimation functionality have been located and reviewed in order to gain a coarse-grained picture about the current state of the art.

2.3.1 List of evaluated apps

To discover existing solutions with wind estimation functionality, experienced sailors, as well as sailing communities were questioned. Additionally, the "*Similar apps*" proposal feature of Google Play[13] and Apple App Store[14] has been used. The following list contains the names of apps which were evaluated regarding wind estimation.

- *RaceQs*
- *SailRacer*
- *Sailing Tack-Tastic*
- *TheFastTrack*
- *iRegatta*
- *Sailtracker*
- *StartLine 5 with Race Box*

[13] Google Play is the online app market platform for Android devices developed by Google. It is available online at http://play.google.com

[14] Apple App Store is an app market platform provided by Apple for its iOS, tvOS and watchOS. It is accessible through special client applications provided by Apple, such as iTunes.

All of the listed apps are available for Android OS. The evaluation revealed that RaceQs and Sailing Tack-Tastic are the only apps from the list which truly include the GPS-based wind estimation. However, even without wind estimation, the other apps still support this work with interesting ideas regarding feature engineering. Therefore, the three most relevant and representative apps have been picked for a detailed review, whereas the remainder is omitted because there are no new ideas introduced concerning wind estimation.

2.3.2 RaceQs

RaceQs[15] is the most advanced sailing platform from the whole list, and surprisingly, its use is completely free of charge. While the app for smartphones acts as a simple tracker for sailboats, the RaceQs website provides a wide variety of sailing analytics for tracked races, including a 3D-replay of the whole race with the whole fleet and a configurable overlay of various performance figures. Furthermore, the website appears as a social network where sailors can get in touch with each other and share sailing events, as well as own sailing records.

According to RaceQs makers, the GPS tracks of the whole fleet are analyzed and averaged by the app to determine the wind. More precisely, upwind sailing patterns are examined with a special attention to SOG. As RaceQs team stated, instead of true wind, current adjusted wind is estimated by the platform. Only the wind direction is estimated. RaceQs team claims to master the estimation with an accuracy of ±5°.

The user tests showed that the app requires the user to set up his race with at least a start-line. If the start-line is not set, the estimated wind is not shown. The race setup also allows to set waypoints, as well as a finish-line.

A tutorial video published by RaceQs in YouTube reveals that their proprietary wind determination algorithm uses meteorological data as initial wind direction which gets refined throughout the race by the algorithm [9]. It is also possible to adjust the wind manually. The manual wind adjustment will still get refined throughout the race.

One remarkable feature of RaceQs is the option to track roll, pitch and yaw by means of smartphone's gyroscope. This requires the smartphone to be mounted horizontally parallel to the boat's center line. The tracked roll, pitch and yaw of boats are visualized within 3D-rendering of corresponding boat. In terms of wind estimation, this additional tracking data could provide further correlated predictors. However, utilization of this

[15] http://raceqs.com/

data can also introduce problems when the tracking smartphone gets mounted incorrectly.

Considering all information gained from this review, the following assumptions about the operation of RaceQs wind estimation algorithm were met:

Algorithm input:

- Initial wind direction which comes either from user's input, or a query of a meteorological third-party service
- GPS-tracks of the whole fleet
- Race setup containing at least a start-line, optional waypoints, and an optional finish line

Processing:

- Upwind sailing patterns of each boat are analyzed considering SOG
- The initial wind direction gets refined by results of upwind sailing pattern analysis

Output:

- Direction of the current adjusted wind

2.3.3 Sailing Tack-Tastic

Sailing Tack-Tastic[16] is free and appears to be the most basic app from the list. But it is still worth noting due to its simple but particular different strategy for guessing the TWD. The app provides a very basic virtual instrument with a display of course over ground (COG) and SOG. According to the description of the app in Google Play, Kalman-Filter is used to smoothen the changes of COG and SOG in order to make it appear more yacht-like. However, the user tests showed that the utilized Kalman-Filter might be suitable for large sail vessels, but inadequate for small and maneuverable racing sailboats. The app description also says that each time when a tack is performed, TWD gets estimated. Conducted user tests showed, that the app expects from user to set an optimal tack angle for his boat which is initially set to 90°. Whenever a maneuver is performed with an absolute course change not lower than the defined optimal tack angle, the wind estimation assumes that the performed maneuver is a tack, and the wind

[16] https://play.google.com/store/apps/details?id=au.point4.tacktastic

direction gets estimated as the course recorded during a point when the recognized tack maneuver was completed by half.

Considering all the information gathered from this review, the following assumptions about the operation of Sailing Tack-Tastic wind estimation algorithm were met:

Algorithm input:

- GPS-track of a single boat
- Optimal tack angle

Processing:

- Default TWD is geographic north
- If a maneuver with maneuver angle equal or higher than the defined optimal tack angle is detected, it gets classified as tack
- COG in the middle of the last recognized tack is regarded as TWD

Output:

- TWD

2.3.4 SailRacer

SailRacer[17] is promoted as an additional tool for assistance in tactical sailing decision making. It provides virtual instruments with various sailing information, including SOG, COG, target speed according to boat polars, direction and speed of apparent wind, true wind, current, and more. The wind information is either retrieved from supported measurement systems over NMEA[18] protocol or gets set manually by user. In contrast to wind, the app does not allow the information about the current to be set manually. Thus, the app must be capable of deducing speed and direction of the current by means of data provided by NMEA. The user tests showed, that the default direction of the current is the inverted TWD.

The app expects the user to provide boat polars. From these polars optimal tack angle and jibe angle are derived. Initially, the app comes with default boat polars which define the optimal tack angle as 84° and the jibe angle as 60°. The optimal tack and jibe angles are only changeable by providing new boat polars.

[17] http://sailracer.net

[18] NMEA is a communication protocol which is usually used by digital measurement devices in a nautical context

If the user sets up a race with a start line, waypoint buoys and a finish line, SailRacer will provide assistance in tactical decision making by showing turning instructions to follow the optimal sailing route toward the destination. It is highly likely that besides wind and current information, the derived optimal tack and jibe angles are considered for optimal route determination considering the race setup.

2.3.5 Conclusion

Considering the market, the GPS-based wind estimation appears to be in its beginning state. RaceQs introduces a solid working solution which makes use of meteorological data and GPS tracks of sailboats. The review of the app indicated that there are certain patterns concerning SOG which characterize upwind sailing courses. Another idea gathered from the review is the utilization of meteorological data which might contribute a high value, when multiple choices for a possible wind direction are given. However, the potential of RaceQs wind estimation algorithm can still be increased by incorporation of further correlated features introduced by other apps, as well as from further feature engineering. One of the most important data, which is ignored by RaceQs, are polars. The polars allow derivation of optimal tacking and jibing angles, as well as a coarse-grained determination of wind speed. The app Sailing Tack-Tastic makes use of optimal tacking angle to deduce the wind. The wind estimation strategy of the app also allows to conclude that the jibing angle is usually smaller than the tacking angle, otherwise the actual jibes would be also recognized by the app as tacks. The polars are used by SailRacer to determine the optimal tacking and jibing angles automatically. The default polars of SailRacer underpin the credibility of Sailing Tack-Tastic hypothesis about tacking angle being usually higher than jibing angle. SailRacer is capable of calculating the optimal route considering the prevailing wind, current and polars, which induces a correlation of total sailed route with true wind, current and polars. The current seems to be an important variable which can deform the shape of actual polars making it deviating from the target polars.

Overall, the market research revealed few ideas and some correlated features which could be reused in this work. It has also showed that none of the apps utilize the full feature spectrum available for the wind estimation. Thus, the focus of this thesis is to develop a solution for true wind estimation considering all useful features which correlating with the true wind to achieve the most accurate estimation results.

2.4 Problem definition

The wind estimation shall deliver TWD, as well as TWS by processing the GPS records jointly with available target polars. In addition to these estimates, the estimation confidence must be provided.

The target audience for the wind estimation cannot not be specified. Therefore, it is unknown whether the wind estimation component is planned to be used in the context of racing, or cruising. Consequently, it should be designed in a generic way to handle both types of sailing activities. However, the main focus of this work is set on racing because it represents the principal context of SAP Sailing Analytics platform.

2.4.1 Input

Boat polars can be regarded as a dataset with fixed length which remains static throughout the whole race. However, these properties do not apply on GPS tracks because each GPS-track consists of a sequence of GPS fixes ordered by time. The length of a GPS-fix sequence can vary. Depending on GPS device, the time interval between fixes can vary as well.

The order within a GPS fix sequence allows deduction of various additional features, including maneuver spots, maneuver angle, velocity curve, turning rate and etc. Furthermore, the GPS fixes can be aggregated arbitrarily to produce additional features such as actual polars.

2.4.2 Output

TWD will be indicated in degrees, and TWS in knots. The estimation confidence level will be provided as a floating-point value between 0 and 1 which is meant to reflect the percentage. Since true wind is position and time dependent, multiple wind fixes shall be produced and related to a time point and a GPS position. This shapes the output to a sequence of wind fixes.

2.4.3 Input-output mapping

The sailing domain experts assume that GPS data contains various features which correlate with TWD. The assumption comes from the ability of sailing experts to guess the TWD by taking a look on a visual representation of a sailed GPS track. The

interviews with domain experts revealed few criteria which had been subconsciously used for TWD guessing:

- Dead-wind zone, which is approximately 90°
- Optimal tacking and jibing angles, which are derivable from polars
- Upwind and downwind speed ratio, which is derivable from polars

These are the first clues for the possible features correlating with the true wind. There are more such features expected. Thus, the goal of the wind estimation component is to map the input with the output considering the correlation of features with TWD.

2.4.4 Constraints

The wind estimation must be implemented in a compatible way to SAP Sailing Analytics backend, which is Java based. Hence, it must provide a Java API for interoperability with existing backend components. The classes and interfaces of the platform, which semantically overlap with the wind estimation implementation must be reused. The existing code of SAP Sailing Analytics is allowed to be extended and improved, but it must not be redundantly reproduced with additional classes and interfaces. Thus, lose coupling of wind estimation to SAP Sailing Analytics backend is not an issue of this work. Therefore, the implemented wind estimators of this work cannot be embedded into another project if dependencies to appropriate platform modules are not satisfied. However, this document provides concepts with its reasoning which can serve as a guidance to reproduce the achieved implementation in a standalone manner.

Furthermore, there were no further specifications given for this work regarding estimation accuracy and etc. because the feasibility of the given task as well as the concrete utilization strategy of results of this work are currently unknown. Therefore, this work will be treated more like a research, than a software engineering project.

2.5 Problem mapping

After the problem is known, it will be mapped to the fields of science in order to identify appropriate solution concepts. The purpose of this section is to provide a big-picture perspective for all possible solutions. To keep the focus on the problem within a high-level perspective, all technical terms occurring in this section will be covered in *2.6 Machine learning basics*.

2.5.1 Problem category

The primary task of the wind estimation component is the mapping of independent numerical values to dependent numerical values. To achieve this, the relationship between input and output must be induced and represented within an estimation model accordingly. At first glance, the mapping between numerical input and output vectors is pointing to **regression analysis**, and thus, the primary problem which is targeted by this work may be categorized as a **regression problem**. Since this work is dealing with GPS tracks which are composed of sequential data ordered by time, one part of the problem can also be stated more precisely as a **sequence regression problem**. However, regression analysis field has a substantial overlap with **machine learning (ML)** which addresses same problem categories and appears even more suitable field for this work because it provides algorithms for automated generation of estimation models based on available data. This work is supported by a huge amount of sailing data provided by SAP Sailing Analytics, including the input and target output data for wind estimation and thus, allows utilization of **supervised ML** techniques for extraction of relationship patterns between input and output.

2.5.2 Sequence regression with ML

As the chapter *2.4.3 Input-output mapping* already preluded, the nature of possible features correlating with the true wind is expected to be diverse. For sequence regression, the most promising ML method category is **Deep Learning (DL)** which introduces interesting neural network architectures such as Long Short-Term Memory (LSTM) which are capable of learning relevant features with its complex patterns automatically. But the demit of neural networks is its black-box which makes debuggability and thus, the understanding about its input-output mapping nearly impossible. Therefore, instead of Deep Learning multiple white-box models will be engineered which provide a transparent view on its utilized patterns. It is expected that multiple statistical and ML models can be combined to achieve a synergy effect. In context of ML, the technique for combination of multiple **sub-models** to produce one high-level model is known as **ensemble learning**. The purpose of ensemble learning is the aggregation of results of multiple sub-models to an optimal result. It builds on the **wisdom of the crowd** principle which propagates that the aggregation of many answers is better than an expert's answer [10, p. 183]. The task for this work is to devise the most suitable **ensemble method** which is the algorithm for aggregation of results produced by each sub-model.

2.5.3 Probabilistic maneuver classification

In context of sailing data, the sub-models can be maneuver classifiers classifying maneuver by its type. This adds **classification** as an additional buzzword to this thesis. For instance, if a tack maneuver is known, the only possible TWD can be between its COG before and after maneuver, ideally at the COG at maneuver middle c_{middle} which is defined as follows:

$$c_{middle} = c_{before} + \frac{\alpha}{2}$$

$\{c \in \mathbb{Z}_{360} | \, c \geq 0\}$, c_{before} represents course in degrees before maneuver started, c_{middle} represents the **maneuver middle course**
$\alpha \in \mathbb{Q}$ represents the performed direction change within maneuver in degrees. The value gets positive if the maneuver was performed toward starboard, and negative if toward port.

Analogously, TWD can be also extracted from jibes. But in addition, its middle course must be flipped by 180°. From GPS data, various maneuver features can be derived, e.g. SOG and COG before and after maneuver, speed trend within maneuver curve, turning rate and many more. Since the wind, and therefore, the maneuver type of each maneuver contained in SAP Sailing Analytics data store is already known, a classification algorithm can be trained with maneuver data with supervised learning methods. To allow best possible aggregation of classification results for each maneuver, only **probabilistic classifier** algorithms will be used. The merit of probabilistic classifiers is that its output is composed of probabilities for each possible discrete output value instead of a simple vote. This allows aggregation of ensemble results considering the confidence of each vote.

2.5.4 Feature engineering

The raw input of the component might not express the relationship as well as certain high-level features extracted from GPS tracks such as maneuver angle. Therefore, in order to construct precise but also simple, comprehensible and debuggable models, it is necessary to pre-process the raw data accordingly and extract the features which represent the underlying problem in a most suitable manner. In terms of ML, this task is called **feature engineering**.

2.6 Machine learning basics

This section presents all the theory and fundamental concepts of ML which are required for understanding of this work. It introduces the common workflows in ML projects and various data pre-processing and exploration techniques which are employed in chapter *4 Data preparation*. Furthermore, theoretical concepts of estimation models are introduced which will be employed in chapter *5 Modelling*, and evaluation strategies which will be used in *6 Evaluation*.

2.6.1 Action plan

In ML „*the two things that can go wrong are "bad algorithm" and "bad data."*" [10, p. 22]. Thus, efforts must be made to acquire high quality data with useful features and to determine the most suitable classification algorithm with its optimal setup. For this, an action plan was elaborated considering the best practices of ML introduced in [10]. As preluded in *1.4 Outline*, the action plan leans heavily on CRISP-DM. Its phases and transitions are depicted below:

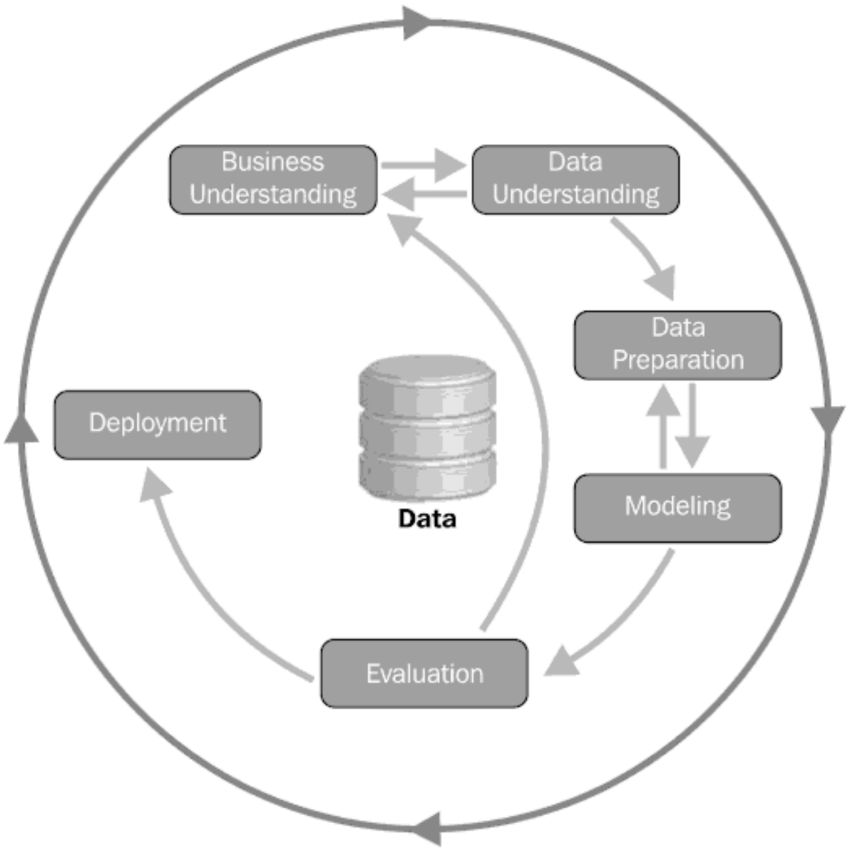

Figure 8: CRISP-DM

As the deployment phase is outside the scope of this thesis, it will not be discussed further. The following subsections summarize the tasks of each phase of this work.

Data understanding

Besides problem understanding, the acquisition of the right data requires its understanding and relevance assessment for the given problem. Otherwise, irrelevant or even wrong data can be obtained which results in ML models with low accuracy or even random models. In current context, wrong data is the data which does not represent the information it is meant to represent. An example of such data is provided in section *3.4 Maneuver detection* which demonstrates the difference between assumptions and reality.

Data preparation

When the data is understood, it must be prepared for ML algorithms. For this, the right data must be selected and loaded into an environment where data analysis and evaluation of ML algorithms can be performed. Mostly, such environment is set up locally. Afterwards, the feature set is engineered which is meant to be used as input for ML models. Its task is to capture the relevant features which contribute positively to model scoring. Some of the features can be used as given without any transformations, while other features might need a transformation or even aggregation to a new high-level feature. The task is also known as feature engineering. Furthermore, the data must be cleaned from noise which is usually occurring due to measurement errors. Then, the data is explored considering its distributions, correlations and influence on the target variable. The exploration is meant to collect additional insights, underpin the importance of each feature and uncover redundant and useless features. Finally, the final feature set is selected and further pre-processing such as feature scaling and dimensionality reduction is performed. The theory of the latter pre-processing tasks is conveyed in sections *2.6.3 Feature scaling* and *2.6.4 Dimensionality reduction and PCA*.

Modelling

After data preparation, appropriate models for estimation are engineered considering the known ML algorithms. The engineered models are implemented and prepared for its evaluation.

Evaluation

Lastly, the models get evaluated in terms of its accuracy and runtime performance. For this, it is important to consider the most optimal setup of each evaluated model. Otherwise the evaluation results can get skewed. The optimal setup for a model depends on its preferable pre-processing pipeline and optimal hyperparameters for the given data. The pre-processing pipeline is composed of transformation steps applied to the input data before it reaches the ML model, e.g. feature scaling, dimensionality reduction and etc. A hyperparameter is a parameter of ML model which is required to be set before the learning process. An example is given by the *k* value of k-Means clustering algorithm which is introduced in *2.6.6 Clustering with k-Means*.

2.6.2 Feature importance tests with ANOVA and Random-Forest

In machine learning **Analysis of Variance (ANOVA)** F-test is commonly used for univariate feature selection, as it ranks the importance of each feature by its F-value. It is a statistical method which internally splits the dataset by its discrete target variable and compares the mean and variance across the groups for each feature individually [1].

Another method for determination of feature importance is leveraged by training of a tree-based model [11], e.g. **Random-Forest** which allows derivation of feature ranking from its node hierarchy. Random-Forest is an ensemble model composed of multiple decision trees which are trained independently. During model training, each decision tree gets a random subset of the training data. For scoring, the votes of all decision trees get aggregated to produce a final vote. Despite its simplicity, Random-Forest is considered as one of the most advanced ML algorithms today [10, p. 183], and thus, it was employed in feature importance evaluation of this work. However, for result interpretation one must be aware that in contrast to ANOVA, the feature evaluation of Random-Forest is based on a multivariate method rather than on a univariate. The difference between both methods is that in univariate method each feature is evaluated independently while within a multivariate method all features are evaluated together [12, p. 169].

2.6.3 Feature scaling

Most of the machine learning algorithms do not perform well, when its numerical input features have different scales [10, p. 66]. For distance-based ML-algorithms such as

SVM and K-Nearest-Neighbor, the distances of features with large scale can overweigh other features and result in bad accuracy. For other ML-algorithms such as neural networks, different scales deteriorate the rate of convergence which results in longer training times and local optima findings. Therefore, it is considered as a good practice to scale all the input features to a similar value range. There are two common approaches to make all the features having the same scale: min-max scaling and standardization.

Min-max scaling

Min-max scaling operates as follows: the min value gets subtracted from all the values. All the yielded values get divided by difference between max and min. The biggest merit of min-max scaling is that it is simple and fits all the values in the range [0; 1] which is required by some ML-libraries. However, to maintain this value range, the knowledge about min and max values of future data is required which is not always given. Furthermore, it is sensitive to outliers which can lead to considerably smaller scales of non-outlier values than intended.

Standardization

In standardization, the mean gets subtracted from all the values. Afterwards, all the new values get divided by its variance. In contrast to min-max scaling, standardization does not guarantee all the scaled values being contained within a fixed interval. However, most of the values will fit in [-1; 1]. Unlike min-max scaling, standardization is more robust against outliers and does not require knowledge about the min and max of future data.

2.6.4 Dimensionality reduction and PCA

Each added input feature adds a dimension to the vector space containing all possible values of the feature vector. The larger the vector space gets, the harder it will be for ML algorithms to find optimal solutions [10, p. 207]. The problem is known as **curse of dimensionality** and results in longer training times and reduced accuracy mainly due to lack of data covering the feature space. The impact of the problem can be mitigated by various dimensionality reduction methods which purpose is to reduce the number of features without considerable information loss regarding its patterns.

The most common algorithm for dimensionality reduction is **Principal Component Analysis (PCA)**. It is a projection-based dimensionality reduction algorithm which

transforms features into a lower vector space considering eigenvalues and eigenvectors of original vector space. By means of such transformations, correlated features can be represented with less features. Furthermore, it can help to reduce noise and eliminate unnecessary details in data [10, p. 207]. The transformation can also be seen as a lossy compression. Most of PCA implementations require target number of target dimensions to be set which are called **PCA components**. However, the target number of features can also be derived from the demanded percentile of variance to preserve in the transformed features.

Before application of PCA it is crucial to normalize the features to a common scale, otherwise the variances of dimensions with different scales will not be reasonably comparable which yields in elimination of wrong dimensions with low scales.

2.6.5 Hidden Markov Model

Hidden Markov Model (HMM) is a tool for modelling of time series data [13, p. 1]. Within this work, a custom HMM has been employed to develop the most successful wind estimation algorithm. To understand the algorithm with its HMM modifications, the concepts of a conventional HMM will be concisely conveyed.

Definition

HMM is composed of the following components:

- Time points of subject observations starting from 1 and ending at n.
- Observations $o_1..o_n$: The sequence of given observations about the modelled subject. An observation can be discrete, a continuous value, or even a vector. A particular observation observed at time point t is denoted as o_t.
- Hidden states $s_1..s_n$: The sequence of states which are hidden from the observer. A hidden state s_t is represented by a discrete value which corresponds to the observation o_t at time point t.
- Set of hidden state values $S_1..S_m$: All possible values which can occur within a hidden state variable s.

The following figure depicts HMM components with its conditional independence:

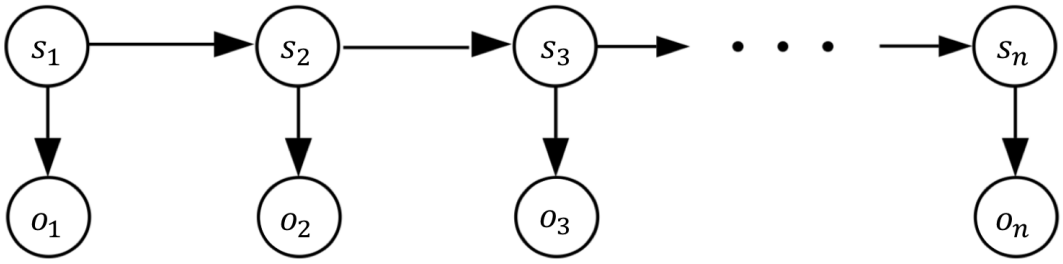

Figure 9: HMM components with its conditional independence

The figure above allows derivation of the following dependence properties:

- The hidden state s_t is only dependent on its preceding state s_{t-1}
- An observation o_t is only dependent on the hidden state s_t

To perform various estimation tasks, HMM encompasses further components:

- Initial probabilities: Contain probabilities $P(s_1 = S_x)$ for each possible value S_x of the hidden state variable.
- Transition probabilities: Contain conditional probabilities $P(s_t = S_x \mid s_{t-1} = S_y)$ for all possible transitions between hidden state values
- Emission probabilities: Contain conditional probabilities $P(o_t \mid s_t = S_x)$ for all observations o_t conditioned with all possible hidden state values S_x of s_t.

In context of this work, an observation o_t can contain maneuver features derived from GPS data such as manoeuvre angle, while the hidden state could be the combination of maneuver type and tack after maneuver, e.g. jibe at starboard, tack at port and etc.

Solving problems and algorithms

There are various tasks which can be solved by HMM:

- Determination of the most likely sequence of hidden states by means of Viterbi algorithm
- Determination of probability for hidden state s_t being value S_x at time point t by means of Forward-Backward algorithm
- Learning of initial, transition and emission probabilities from sequences with observations and its known hidden states by means of Baum-Welch algorithm

2.6.6 Clustering with k-Means

Clustering is used to determine groups of similar instances. Each group of elements is called a **cluster**, and the process of finding the groups – **clustering** [14, p. 283]. Clustering belongs to unsupervised learning category of ML. In contrast to supervised learning, unsupervised learning is dealing with unlabeled data which means that there is no given target variable. One of the most popular clustering algorithms is **k-Means**. The **k** is a hyperparameter of the algorithm which defines the fixed number of clusters to be found in a set of data. The merit of k-Means is that it is simple and fast. Its operation is defined by the following steps:

1. Select randomly k instances from data and define them as k centroids
2. Assign each instance in the data to the closest centroid considering the Euclidian distance of instance features and the features of centroid
3. For each centroid, calculate the mean of the values of all instances belonging to a centroid. The new mean value defines the value of the corresponding centroid.
4. Repeat steps 2 and 3 as long as the centroid values change

A proprietary implementation of k-Means is already contained within SAP Sailing Analytics code. It will be used to develop maneuver clustering-based wind estimation.

2.6.7 Evaluation metrics

For model evaluation, various scoring metrics are used. The reason is that the conventional accuracy measure based on $\frac{number\ of\ correct\ predictions}{number\ of\ all\ predictions}$ is insufficient for many cases, especially if the data distribution is skewed. This case is exemplified by the following classification result:

Table 1: Example of a confusion matrix

	Predicted: True	Predicted: False
Actual: True	1 True Positives (TP)	9 False Negatives (FN)
Actual: False	0 False Positives (FP)	90 True Negatives (TN)

The provided table is known as **confusion matrix**. Assuming that the classifier was predicting whether the maneuver is a jibe (True) or not (False), the confusion matrix is read as follows:

- 1 jibe maneuver was predicted correctly as jibe (True Positive)

- 9 jibes in the dataset were predicted falsely as non-jibes (False Negatives)
- There was no non-jibe maneuver in the dataset classified as jibe (False Positives)
- 90 non-jibe maneuvers in the dataset were correctly classified as non-jibes (True Negatives)

The accuracy of the classifier is calculated as follows:

$$\frac{TP+TN}{TP+FP+TN+FN} = \frac{1+90}{1+0+90+9} = 91\ \%$$

The accuracy of the classifier is high, although it predicted only $^1/_{10}$ of jibe maneuvers correctly. The classifier performance is as similar as claiming that all performed maneuvers are non-jibes which yields in 90 % accuracy. To provide more credible classifier performance measures, further evaluation metrics are introduced:

- **Precision**: $\frac{\mathrm{TP}}{\mathrm{TP+FP}} = \frac{1}{1+0} = 100\ \%$
- **Recall**: $\frac{\mathrm{TP}}{\mathrm{TP+FN}} = \frac{1}{1+9} = 10\ \%$
- **F1-score**: $\frac{2 \times \mathrm{Recall} \times \mathrm{Precision}}{\mathrm{Recall+Precision}} = \frac{2 \times 0.1 \times 1.0}{0.1 + 1.0} = 18\ \%$

Each of the measures allows to draw conclusions about the error characteristics of the classifier. Optimistic classifiers are penalized by precision, while pessimistic classifiers by recall. F1-score is considered as the metric balancing precision and recall.

2.6.8 Datasets and cross-validation

Throughout the phases of projects dealing with ML the data is usually split in multiple datasets:

- **Training set:** is used to train ML algorithm
- **Validation set:** is used to evaluate the scoring of the algorithm during algorithm selection and elaboration of its optimal setup such as hyperparameter search
- **Test set:** is used to evaluate and report the final score of the fine-tuned ML model by simulating new data

The aim is to test the model on different data than it was trained on in order to verify its generalization ability. A model that perfectly fits the training data, but fails in predicting unseen data is **overfitting** due to lack of **generalization** ability. Such models are considered as useless. Therefore, to prove the generalization ability of a model, the data is split in multiple datasets. However, the less training data the model gets, the harder it will be for models to generalize. To increase the amount of training data by

removing the need for a separate validation set, **cross-validation** technique is used. Within this work, **k-fold cross-validation** is employed which is defined as follows:

1) Training set is divided into k folds
2) The following procedure is performed k times within an iteration loop:
    a. Model is trained on $k - 1$ folds
    b. Model is tested on the remaining fold which is always unique within the iteration loop. Test score is recorded.
3) The final test score is averaged over the k test scores produced within the aforementioned iterations

2.7 Tools and libraries

There were various ML tools and libraries evaluated for this work. This section presents the selected tools for each ML task.

2.7.1 Programming languages

The whole SAP Sailing Analytics backend is developed in Java. Therefore, the decision for the implementation language of wind estimator was already set. The encapsulation of other programming languages by means of Java Native Interface[19] (JNI) and other interfaces provided by Java Virtual Machine[20] (JVM) was avoided because it introduces additional complexity and maintenance overhead for SAP Sailing Analytics administrators.

However, in terms of ML the ecosystem of Java is less developed than of other languages such as R and Python[21]. Furthermore, Java is regarded as too explicit and cumbersome in terms of code length which is required to be written in order to solve a task. Therefore, due to productivity reasons a decision was made to split the work in two parts:

- Implementation of wind estimators
- Data analysis and ML algorithm evaluation

[19] JNI is used to embed compilations from other languages such as C, C++, FORTRAN and etc. [21, p. 590]

[20] JVM is used to execute bytecode which is the compilation result of JVM-compatible programming languages such as Java. It manages system memory and provides an execution environment supported by many operation systems such as Windows, Linux and Mac OS.

[21] "*Python is an interpreted high-level programming language for general-purpose programming*" [19]. It excels with its minimalistic syntax and its outstanding ecosystem with ML and data science packages.

While the former part will be implemented using Java, within the latter part, Python will be used to facilitate the processes of data analysis, hyperparameter tuning and performance evaluation of ML algorithms. Python was preferred over R due to own existing skills. In contrast to Java ML library selection, there was no comprehensive evaluation of ML libraries performed because the whole Python code is not meant for production use.

2.7.2 Wind estimator implementations

For Java, various ML libraries were evaluated considering the following criteria:

- Presence of various probabilistic classification algorithms:
  - SVM, Logistic Regression, Naive Bayes, k-Nearest neighbor, RandomForest, Neural networks
- Pre-processing features:
  - Data scaling, PCA
- License:
  - Free and corporate friendly
- Simplicity:
  - Easy to learn and easy to use
- Documentation:
  - Literature, tutorials, examples for library understanding
- Lightweight:
  - Easy to embed into Java application, small artifact size, the less dependencies the better
- Active development
  - The library gets continuously improved and will be supported in the future

The following libraries were sorted out quickly due to its GPL license:

WEKA, MEKA, MOA, ADAMS, ELKI, JSAT, Java-ML

The remaining candidates and its fulfilment of the evaluation criteria are listed in the following table:

Table 2: Comparison of Java libraries for ML

Library	ML features	License	Simplicity	Documentation	Lightweight	Last commit
Spark with MLlib and/or H2O	Classic ML algorithms, neural networks, pre-processing, distribution	Apache	Low	Low/Medium, Spark is popular with Scala, H2O with Python/R	Fat (141 MB)	2 days ago
H2O	Classic ML-algorithms, neural networks	Apache	Low	Low, no documentation for Java except JavaDoc	Fat (400 MB zip)	1 day ago
Encog	SVM, Bayesian Nets, HMM, neural networks	BSD	Medium	Low	Light (< 2MB)	1 year ago (one man project)
TensorFlow	Advanced neural networks, GPU-support, distribution	Apache License	Medium	Comprehensive	Medium (56 MB)	< 1 day ago
Deeplearning4j	Advanced neural networks, pre-processing, GPU-support, distribution, Spark, automated hyperparameter search	Apache	High	Comprehensive	Fat with 22 maven dependencies (353 MB)	5 days ago
SMILE	Many classic ML-algorithms, neural networks	Apache	High	Comprehensive, but for Scala, Java only with Javadoc	Light (< 1 MB)	17 days ago

TensorFlow, Deeplearning4j and SMILE were selected for a detailed evaluation.

TensorFlow

Despite the absence of classic ML algorithm, TensorFlow had been regarded as a promising library for the case if neural network algorithms are demanded. It is a Python-based project which comes with a Java-Wrapper API for JVM. Unfortunately, the following statement was captured on its website[22]:

[22] Official website of TensorFlow for Java: https://www.tensorflow.org/install/lang_java

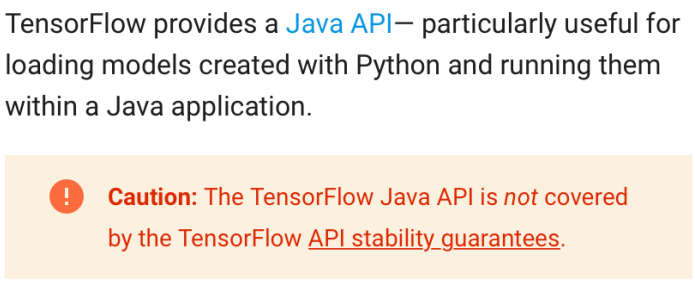

Figure 10: Screenshot of TensorFlow for Java official website (23th September, 2018)

Thus, a conclusion can be drawn that TensorFlow for Java is considered as immature. Considering the presence of JNI, official instability hint and promotion of Python for model training, the decision was made to refrain from usage of TensorFlow in context of Java.

Deeplearning4j

Deeplearning4j[23] is another option for the case if neural network algorithms are demanded. Its demerit is its huge dependency on other libraries which totally sums up to 364 MB. However, the investigation of dependencies showed that 267 MB is constituted by *OpenCV*[24] library which is meant to provide GPU-support for all prominent operations systems including Android. Thus, it is assumed that it is possible to reduce the footprint of the whole deeplearning4j library artifacts to 100 MB. Nevertheless, it is still considered as oversized for the purpose it is meant to be used. The artifact size for SAP Sailing Analytics is regarded as important factor due to its operation in the cloud. However, it is considered as the only eligible library for Java if advanced neural network architectures such as LSTM are required.

SMILE

SMILE[25] stands for Statistical Machine Intelligence and Learning Engine and is considered as an extremely efficient and lightweight library for ML. It provides various ML algorithms and pre-processing features, including feature scaling, PCA, various classic classification algorithms and even neural networks. However, in terms of neural networks, the feature scope is vastly limited in comparison to deeplearning4j. There

[23] https://deeplearning4j.org

[24] https://opencv.org

[25] https://haifengl.github.io/smile/

are only multi-layer feedforward neural network architecture supported. While the number of hidden layers and its neurons is configurable, there is only one activation function supported for all hidden layers, which is logistic sigmoid. Fortunately, softmax activation function is supported within the output layer which allows probability output for each possible value of the output variable. The documentation for SMILE is only available for Scala programming language. However, a detailed investigation of the source code showed that the library is simple and easy to use. Code examples for Java can be obtained from unit tests.

Despite its limitations in neural networks, SMILE is considered as the most appropriate ML library for implementation of wind estimation. It fulfils all the defined evaluation criteria and will be used throughout this work in context of Java.

2.7.3 Data analysis

For data analysis and its visualization described in *4.7 Data exploration*, Python programming language was used with the following packages:

- **Pymongo**[26]: For loading of data from MongoDB
- **pandas**[27]: For management and manipulation of data by means of table-like data structures such as `DataFrame`
- **Matplotlib**[28]: For visualization of data and other statistics by means of charts
- **Scikit-learn**[29]: For feature importance tests with ANOVA and Random-Forest

2.7.4 ML algorithm evaluation

The classification of maneuver types requires appropriate classification models which are generated by means of ML algorithms. To elaborate the most efficient classification algorithm with its optimal setup, Python language was used with pymongo, pandas, matplotlib and scikit-learn as well. While pymongo, pandas and matplotlib were used for same purposes as introduced in *2.7.3 Data analysis*, scikit-learn was employed as ML infrastructure which provides various probabilistic classification algorithm

[26] "*PyMongo is a Python distribution containing tools for working with MongoDB, and is the recommended way to work with MongoDB from Python.*" – see https://api.mongodb.com/python/current/

[27] "*pandas is an open source, BSD-licensed library providing high-performance, easy-to-use data structures and data analysis tools for the Python programming language*" – see https://pandas.pydata.org

[28] "*Matplotlib is a Python 2D plotting library which produces publication quality figures in a variety of hardcopy formats and interactive environments across platforms*" – see https://matplotlib.org

[29] Scikit-learn is an efficient and simple to use ML package for Python, see http://scikit-learn.org/

implementations, data pre-processing components, scoring evaluation components and components for automated hyperparameter search.

3 Data understanding

To identify correlations regarding true wind, the data of SAP Sailing Analytics must be analyzed. The prerequisite for this step is, that the data is understood, including its meaning, format, quality and access options. According to [15, p. 17] and [16, p. 65], within ML-projects 80% - 90% of the time is accounted to data understanding and data preparation because the data is regarded as a vital element which vastly affects the quality of learned models. The experience gathered from this work confirms this statement.

3.1 Domain model

The following figure provides a simplified view on SAP Sailing Analytics domain model intersecting with this work:

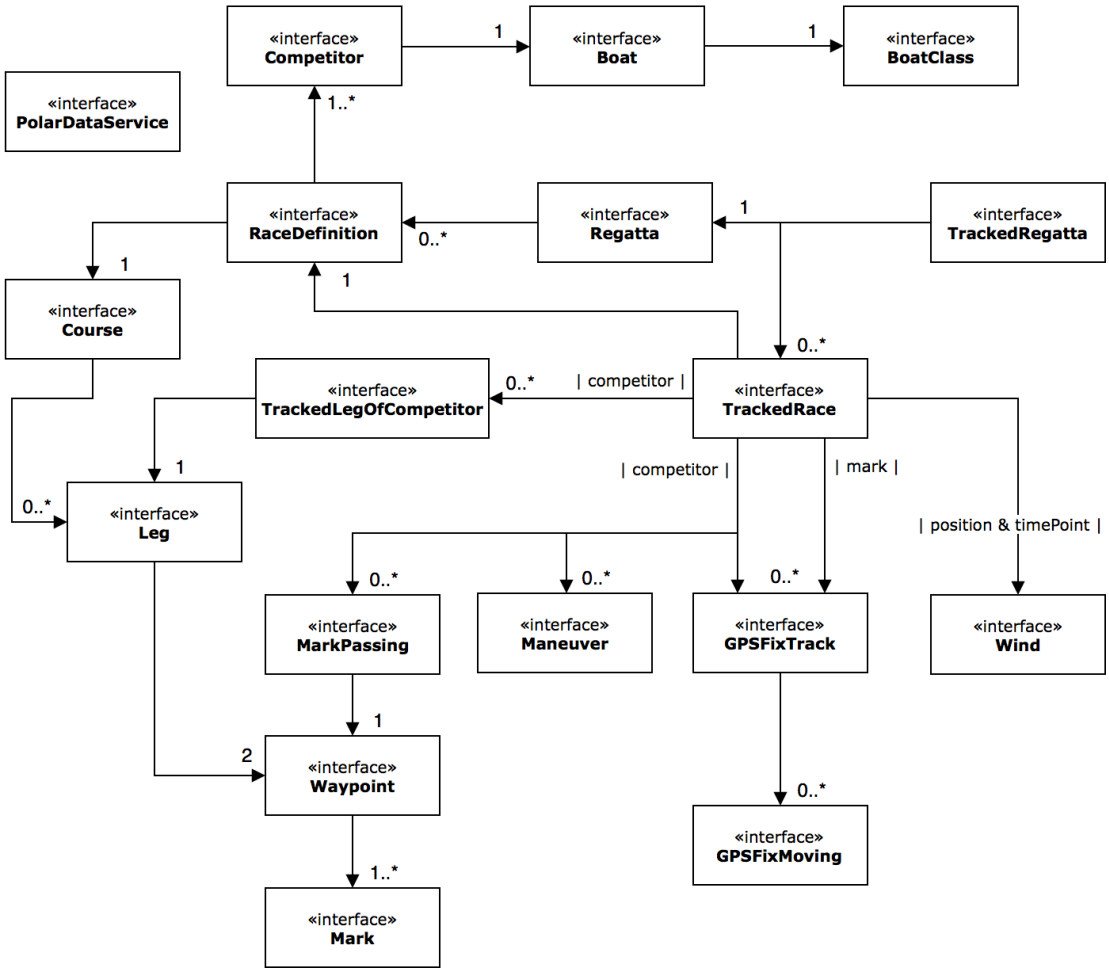

Figure 11: SAP Sailing Analytics domain model clipping

3.1.1 TrackedRace, TrackedRegatta

An instance of `TrackedRace` interface wraps all the information concerning a particular race. Thus, if the instance gets removed, the whole in-memory data regarding the corresponding race gets evicted as well. Therefore, it may be regarded as first point of access for all data concerning a particular race.

A regatta represents an event which is composed of a set of races. Therefore, analogously to `TrackedRace`, `TrackedRegatta` holds all data corresponding to an event.

3.1.2 Regatta, RaceDefinition, Competitor, Boat, BoatClass

Within a race definition, all metadata for race is defined. It includes race name, unique race id, the route to sail, and participating competitors. A boat is assigned to each competitor individually for each race. Each boat belongs to a particular boat class. A boat class defines the type of the boat with its characteristics regarding its sailing performance and shape. In order to guarantee a fair competition, most of race definitions are composed of competitors with boats of the same boat class. However, according to the domain model, this case must not be considered as a rule. And last but not least, `Regatta` contains its metadata including its name, and additionally the set of races, it is composed of.

3.1.3 Course, Leg, Waypoint, Mark, TrackedLegOfCompetitor

`Course` represents the target route which must be sailed as fast as possible by competitors in order to win the race. It is composed of a sequence of legs. The definition of leg, waypoint and mark provided in section *2.2.1 Sail racing and terminology* matches with its equally named interfaces `Leg`, `Waypoint` and `Mark`. Each leg represents a straight line of predefined race path. Hence, each leg is composed of two waypoints: a waypoint from which the leg starts, and a waypoint at which the leg ends. A waypoint can have different shapes with predefined mark passing strategies. Therefore, it can be composed of one or more marks. The mark passing strategy, as well as the set of marks is the information which `Waypoint`-interface provides.

The position of a mark is usually transmitted by its attached GPS sensor to SAP Sailing Analytics platform automatically. This means that the position of a mark is time dependent. It is handled as GPS track. The GPS tracks of marks are managed within `TrackedRace` which provides access to position of a mark for a certain time point.

The `TrackedLegOfCompetitor`-interface is huge. It contains information regarding time and additional statistics about the segment of GPS track sailed within its corresponding leg. The most important part of the interface for this thesis is the possibility to determine current sailed leg based on time. When appropriate `TrackedLegOfCompetitor`-instance gets determined, the information about previous and next waypoints can be accessed.

3.1.4 Wind

The `Wind`-interface is used within SAP Sailing Analytics to express the state of the wind which is dependent to time and position. It provides access to the following information:

5. TWD, e.g. in degrees
6. TWS, e.g. in knots
7. Position and time point of measurement

The wind information is captured from various wind sources, including wind measurement systems and manual user input. In most of races, there are few wind measurement systems spread over the racing area. The SAP Sailing Analytics platform continuously receives wind measurements and stores them within `TrackedRace`-instance for each wind source as a sequence of `Wind` which can be also denoted as wind track. Whenever wind information is queried, the appropriate wind fixes from each wind track are selected based on its time point and averaged considering the queried measurement position and time point. Finally, the averaged wind fix is constructed and returned.

3.1.5 GPSFixTrack, GPSFixMoving

A `TrackedRace`-instance contains GPS tracks for each competitor and for each mark. A GPS track is represented by `GPSFixTrack`-interface and is composed of a sequence of GPS fixes recorded for a sailboat or a mark. In SAP Sailing Analytics, each GPS fix is represented by `GPSFixMoving`-interface which provides access to the following information:

8. Longitude
9. Latitude
10. Time point
11. COG
12. SOG

The computation strategy of SOG and COG depends on the third-party services which supply SAP Sailing Analytics with GPS fixes. Such strategy could rely either on advanced measurements of tracking devices or track reconstruction by means of position points. Since this data is considered by project stakeholders as reliable, the computation strategy of COG and SOG will not be questioned.

3.1.6 Maneuver

The SAP Sailing Analytics platform is capable of detecting sailing maneuvers performed within a GPS track. A detected maneuver is characterized by the following properties:

- Maneuver **time point** and **position** which correspond to the fastest turning rate[30] recorded within the maneuver section
- **COG** and **SOG** before maneuver beginning and after maneuver finish
- **Maneuver angle** which gets negative in case of portside maneuvers and positive in case of starboard maneuvers.
- **Maneuver type**, which can be jibe, tack, bear-away, head-up, mark passing, penalty circle, or unknown when the wind data is not available
- **Maneuver loss**, which is defined as the distance that could be made good toward the wind, if the sailboat would continue sailing its course without maneuvering. The measure is relevant in context of tacks and jibes only.

This information is provided by `Maneuver`-interface which represents each maneuver individually. The list of maneuvers performed by a particular competitor can be retrieved from `TrackedRace`.

Since there are no standards for maneuver computation, the maneuver detection algorithm will be reviewed in order to understand the generation process of maneuver data.

3.1.7 MarkPassing

A tracked race contains a list of **mark passings** for each competitor. Within the context of SAP Sailing Analytics, a mark passing is regarded as a moment, when the boat is passing a start-line, a waypoint mark, or a finish-line. Thus, the `MarkPassing` interface includes time point and waypoint mark reference.

[30] The turning rate can be expressed in degrees per second

For wind estimation, the information about mark passings could be of relevance since it allows to draw conclusions about maneuver reasoning. However, the requirements for this work do not specify this information as mandatory input, and thus, it must not be required by wind estimation in order to produce reasonable output.

The calculation of mark passings is based on proprietary algorithm within SAP Sailing Analytics. Since it does not represent a crucial part for wind estimation, it will not be discussed further and taken as it is.

3.1.8 PolarDataService

The `PolarDataService`-interface provides access to polar data maintained by SAP Sailing Analytics platform. It includes following useful relations:

- (boat class, TWS, TWA) → (boat SOG)
- (boat class, TWS, point of sail) → (boat SOG, boat TWA)
- (boat class, boat SOG, point of sail) → (list of candidates (TWA, TWS))
- (boat class, boat SOG, maneuver angle, maneuver type) → (TWA, TWS)
- (boat class, maneuver type, TWS) → (maneuver angle)
- (boat class) → (boolean): Returns `true` when polars for the given boat class exist, otherwise `false`

For these relations the only two supported maneuver types are tacks and jibes, and the only six supported points of sail are maximum upwind, reaching and maximum downwind, each on either port or on starboard.

While the first two relations require true wind information for the input, the remainder does not, which might be useful for polars handling with unavailable wind information. However, for TWS inference, the point of sail parameter with its three available values might be too coarse-grained and less accurate than the boat SOG estimation methods which require TWA as parameter.

Overall, the `PolarDataService`-interface provides useful methods for TWS inference. However, to guarantee a correct usage of polar data for wind estimation, the polars aggregation strategy within `PolarDataService`-implementation will be reviewed and evaluated.

3.2 GPS-track smoothing

The GPS data is vulnerable to inaccuracy. Hence, outlier removal and smoothing strategies are applied when dealing with GPS tracks. Since SAP Sailing Analytics RaceBoard module is featuring GPS track visualization and calculation of various GPS track dependent measures such as current SOG, it incorporates appropriate algorithms for this issue. However, due to requirements imposed by sailing jury, GPS smoothing must not be omnipresent within all data visualizations provided for the sailing audience. Therefore, in order to meet these requirements and also produce reasonable data, two sets of GPS-fixes get maintained by SAP Sailing Analytics platform:

- Raw GPS-fixes set
- GPS fixes set with removed outliers

Additionally, a smoothing algorithm for COG and SOG is used for calculation of further analytical data.

3.2.1 GPS fixes set with removed outliers

This set is a subset of raw GPS fixes set which is free of significant outliers. The applied removal strategy consists of filtering out of fixes which require the sailboat to sail with an unrealistic SOG in order to reach that fixes from its preceding or following fixes. More precisely, the GPS-fixes set F with removed outliers is expressed by the following term:

$$\forall f \in F\ (F'_f = \emptyset \vee \exists f' \in F'_f(avgSpeed(f,f') \leq 40\ knots)$$

F represents raw GPS-fixes set
$F'_f \subseteq F$ with $\forall f \in F\ \forall f' \in F'_f\ (0 < duration(f,f') \leq 10\ seconds)$

In other words, a GPS-fix gets identified as outlier if there is at least one other GPS-fix contained within the raw GPS-fixes set lying 10 seconds apart from the analyzed fix and there is no fix within the range of 10 seconds from the analyzed fix which can be reached with average SOG ≤ 40 knots

In SAP Sailing Analytics, this GPS fixes set is used as the input for GPS track visualization within RaceBoard as well as the input for maneuver detection algorithm.

There are few issues which can be associated with this outlier removal strategy. The first issue is that boat speeds above 40 knots cannot be handled properly, because all fixes get regarded as outliers. Considering the world sailing speed record, which is 68 knots, such speeds cannot be considered as unrealistic. However, within the whole

history of sailing event accompanied with SAP Sailing Analytics, there were no issues with erroneous outlier removal encountered. Another issue comes into play when sailboats are sailing on low speed, e.g. due to poor wind conditions. In this case, the outlier filtering criterion with 40 knots might be too weak.

3.2.2 GPS-fixes smoothing algorithm

SAP Sailing Analytics incorporates a sophisticated algorithm for smoothing of COG-SOG tuples corresponding to a GPS fix. It is based on differential calculus with exponential averaging component and is regarded as computationally intensive. Therefore, a cache with already smoothed values is maintained within the implementation of `GPSFixTrack`-interface. The whole set of smoothed fixes is not maintained within SAP Sailing Analytics. The algorithm is accessible through `GPSFixTrack`-interface by the following relation:

(time point) → (boat COG, boat SOG)

Within `GPSFixTrack`-implementation, the smoothing algorithm is applied on GPS fixes set with removed outliers. A complete review of the algorithm has been omitted, due to its complexity which requires high effort and notable time for understanding. Nonetheless, the smoothing algorithm has been evaluated regarding its output characteristics in order make assessment about its suitability for this work. It turned out that the smoothing algorithm produces only plausible results when the queried time point is contained within GPS fixes set with removed outliers. When interpolation comes into play, erroneous inflection points get included in the interpolated section of the COG trend curve. The following figures illustrate this issue based on COG trend within a tack maneuver which was randomly picked from SAP Sailing Analytics data.

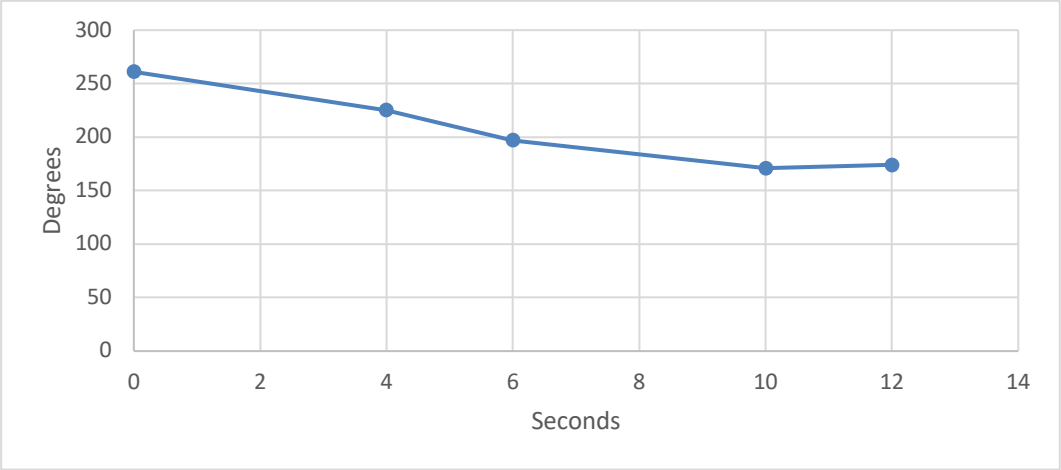

Figure 12: Smoothed COG trend diagram without interpolated fixes

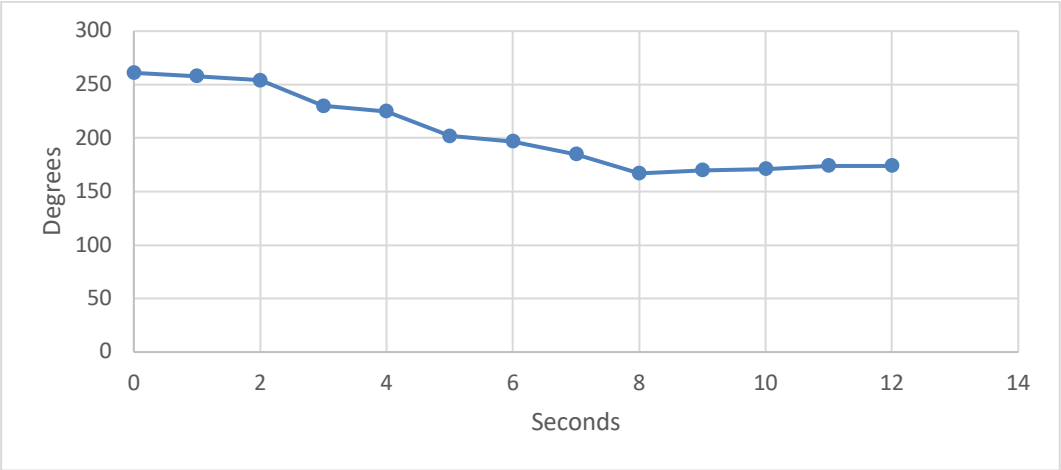

Figure 13: Smoothed COG trend diagram with interpolated fixes

While the values in the first diagram make sense for a tack maneuver, the values in the second diagram introduce falsely added inflection points within the COG trend. This statement gets underpinned by considering the derivatives of depicted curves which represent the turning rate trend.

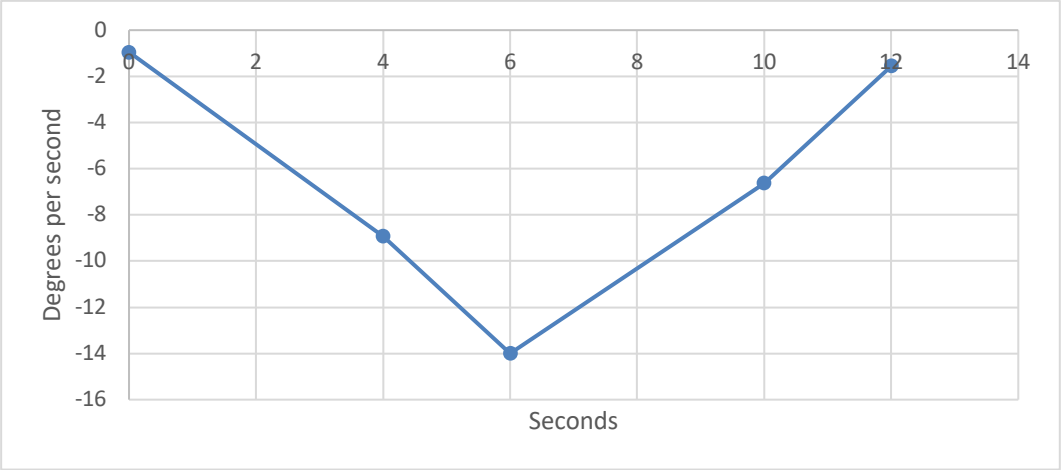

Figure 14: Turning rate trend diagram without interpolated fixes

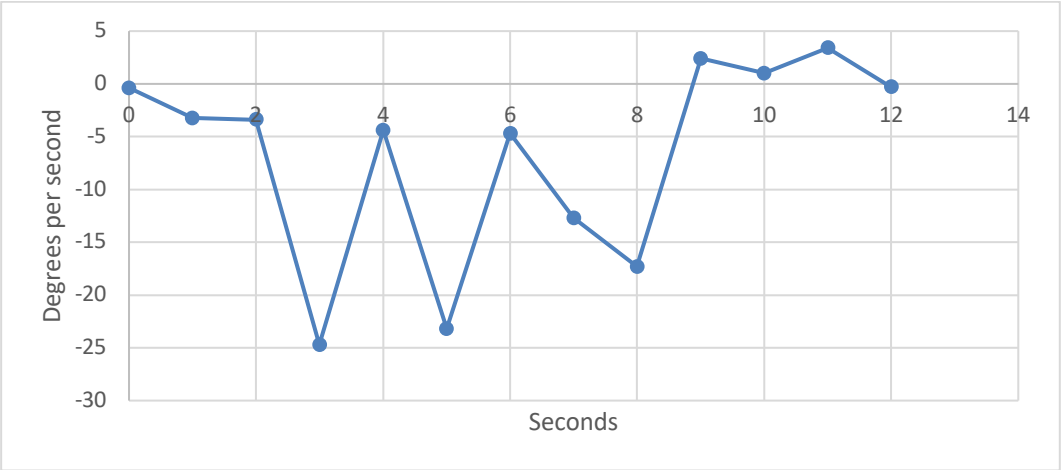

Figure 15: Turning rate trend diagram with interpolated fixes

When a sailboat is holding a stable course, the rudder is kept in the central position. To perform a port tack, a rudder needs to be moved to the portside. After the maneuver, the rudder must be moved back to the central position in order to stop the boat from turning and rearrange it to a new target course.

The first turning rate trend diagram reflects a usual performance of a portside tack maneuver where the rudder gets moved once to the portside at maneuver start until maneuver middle and then back to the center until maneuver end. To reproduce the values of the second diagram, the rudder must be moved back and forth multiple times, which is a nonsense when performing a tack in context of racing. Therefore, the interpolation strategy introduced by SAP Sailing Analytics smoothing algorithm is considered as incorrect. Hence, in order to retrieve plausible smoothed values by means of the

existing smoothing algorithm, the value set for its time point parameter must be limited to time points which are contained within GPS fixes set with removed outliers.

3.2.3 Position interpolation

The GPS fixes smoothing algorithm does not support smoothing of positional data. Hence, the COG and SOG are smoothed independently of altitude and longitude. However, `GPSFixTrack`-interface provides access to the following relation featuring linear interpolation between positions of GPS-fixes:

(time point) → (position)

Since a linear interpolation does not represent an ordinary maneuver curve which proceeds in logarithmic manner, its usage is not appropriate for maneuver analysis purposes.

3.2.4 Conclusion

The GPS fixes pre-processing provided by SAP Sailing Analytics is considered as not perfectly implemented, but still useful since it produces great results when used reasonably. Its reuse requires cautiousness; however, its reuse allows to focus on wind estimation rather than on value smoothing which is regarded as complex and time consuming.

3.3 Polars aggregation

Polars aggregation is another complex SAP Sailing Analytics component. Its development has filled a bachelor thesis and considerable number of development hours afterwards. Nevertheless, polars aggregation needs to be concisely reviewed, since it provides useful features for wind estimation.

3.3.1 Polars representation

SAP Sailing Analytics polars for boat types are maintained in a proprietary format. The polars consist of multiple regression functions for each boat class. Each regression function is represented by a polynomial of degree 3. The set of regression functions for each boat class is divided into three groups containing the following mapping:

- (TWS) → (TWA)

- (TWS) → (SOG)
- (SOG) → (TWS)

Each group of regressions consists of regression functions clustered in ranges of **absolute TWAs**. While the first two regression groups are separated only in three clusters representing upwind, reaching and downwind TWA ranges, the last group is split into $\frac{180}{5} = 36$ clusters each representing a TWA range of 5°. Based on this regression set, the `PolarDataService`-implementation is capable of providing the relations introduced in *3.1.8 PolarDataService.* Since absolute TWA ranges cannot get negative, a conclusion can be drawn that polars on port and starboard are equal which in turn guarantees SAP Sailing Analytics polars being axisymmetric.

3.3.2 Polars generation

The polars are generated using GPS fixes of sailboats and obtained wind information. By means of COG and TWD, the absolute TWA is determined. For each of the regression groups, the appropriate cluster for absolute TWA with contained regression function is determined. This means that from each regression group, one regression function gets picked and updated with regression fitting algorithms. For regression fitting, **Apache Commons Math**[31] library is used.

All raw GPS fixes of all races and competitors are considered for SAP Sailing Analytics polar data aggregation. This means, that outlier GPS fixes and maneuver sections are part of polar data as well, which is probably unintentional.

3.3.3 PolarDataService API implementation

All the methods of `PolarDataService` estimating the SOG of sailboats are implemented using the regression group with with fine-grained clusters. The coarse-grained regression groups are employed in methods which estimate absolute TWA and TWS by SOG and coarse-grained point of sail.

[31] "*Commons Math is a library of lightweight, self-contained mathematics and statistics components addressing the most common problems not available in the Java programming language or Commons Lang.*", see http://commons.apache.org/proper/commons-math/

3.3.4 Polar data import and access

Whenever a server instance of SAP Sailing Analytics is booting, polar data gets imported automatically irrespective whether the instance is executed within a cloud or local environment. Prerequisite for automatic polar data import is a working internet connection.

Since dealing with raw regression functions is considered as not trivial, the easiest way to access polars is by using `PolarDataService`-API.

3.3.5 Conclusion

`PolarDataService` is considered as an important component for this work because it provides relevant data for wind estimation, including target angle for tacks and jibes or upwind/downwind speed ratio. The implementation of `PolarDataService` is regarded as stable. Considering the number of events accompanied by SAP Sailing Analytics, and its average ratio of maneuvering time vs. the time of stable course sailing, it can be assumed that the unwanted fixes represent a small minority of all fixes. Therefore, it is expected that the effect on regressions of these fixes gets outweighed by regular fixes representing track segments at stable COG and SOG. However, the assumption is not proven and is based on intuition. Nevertheless, to maintain the focus on the topic of this thesis, further time-consuming investigations about the credibility of `PolarDataService` will be avoided.

The coarse-grained clusters with boat SOG to TWA regression are of special relevance, since they allow optimal TWA determination for upwind and downwind courses. However, there is a potential for tuning the TWS inference. According to sampled polars, the TWS can significantly vary between fine-grained TWA changes, e.g. between close hauled and close reaching, or running and broad reaching. Therefore, a conclusion can be drawn that for the TWS inference, the fine-grained TWA clusters are more meaningful than the coarse-grained clusters. To employ the fine-grained (TWS) → (SOG) regressions for TWS inference, one easy implementable strategy is to let the regression library determine the TWS value by solving the following equation:

$ax^3 + bx^2 + cx + d = s$	$a, b, c, d, x, s \in \mathbb{Q}$ with a, b, c, d representing regression coefficients, x representing regression unknown variable for TWS, s representing boat SOG

Since there are three values which can be determined from the equation, the value shall be used which is closest to the output of the coarse-grained (SOG) → (TWS)

regression. Thanks to the provided equation solving feature of Apache Common Math, this strategy has been easily implemented and will be used for TWS determination.

3.4 Maneuver detection

SAP Sailing Analytics features a maneuver detection algorithm which gets internally used by `TrackedRace`-implementation in order to infer the maneuver set for each GPS track. The main purpose of inferred maneuvers is its visualization within the race map of SAP Sailing Analytics RaceBoard. Each maneuver gets represented by a point within its visualized GPS track. Surprisingly, a maneuver within RaceBoard is not considered as a section. Hence, the maneuver position and its time point are related to a particular GPS fix contained within the corresponding GPS track. Since the main purpose of the SAP Sailing Analytics maneuver detection is not the calculation of maneuver sections which in turn is needed in order to separate the maneuver sections from sections with stable COG at target SOG, the algorithm must be examined in detail to assess its aptness for this work.

3.4.1 Maneuver spots location

The first step of maneuver detection is the application of **Douglas-Peucker algorithm**[32] on GPS track fixes with removed outliers. The aim is to simplify the shape of GPS track by reduction of GPS fixes set to a level, where remaining GPS fixes correspond to positions with significant course changes. The output fixes of Douglas-Peucker algorithm are called **Douglas-Peucker fixes** (**DP-fixes**). An example of DP fixes within a GPS track is provided in the following figure:

[32] Douglas Peucker algorithm is a line generalization algorithm which is widely used in cartographic and computer graphic applications [20, p. 13].

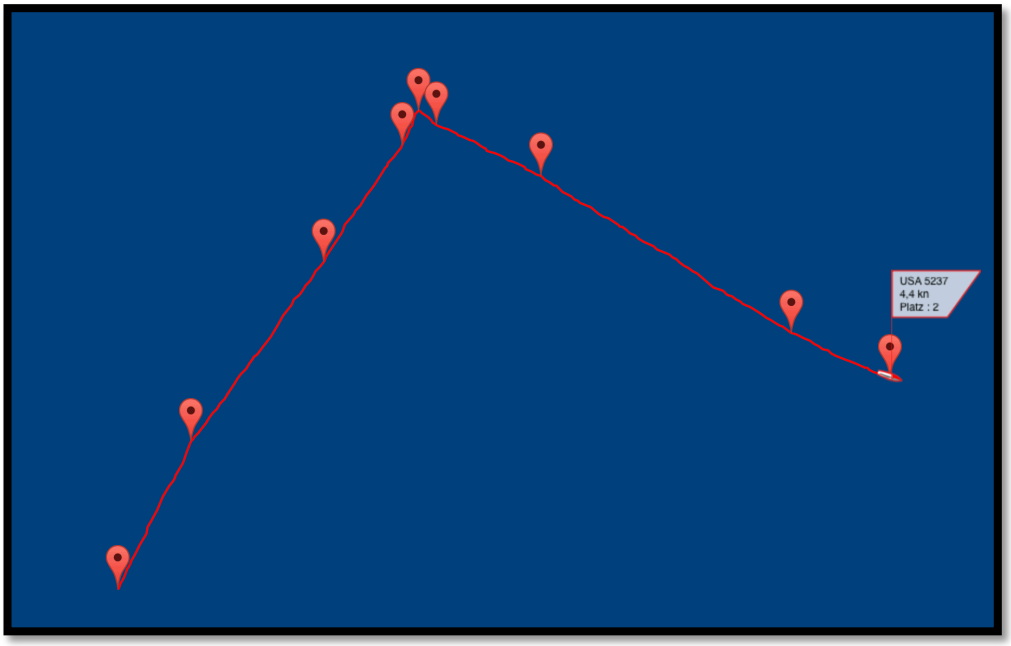

Figure 16: GPS-track with its DP-fixes visualized within SAP Sailing Analytics Raceboard map

Considering the figure with DP fixes, it becomes clear that all DP fixes except the first and the last fix are located within maneuver spots of a track. Depending on the maneuver length, the number of corresponding DP fixes to a maneuver can vary.

The second step of maneuver detection algorithm is grouping of inferred DP fixes to represent each maneuver spot by corresponding DP fixes group. A DP fix gets grouped with its previous DP fix, when the following three conditions are fulfilled:

- The distance to the previous DP fix is smaller than three times of the boat's hull lengths
- The temporal distance to the previous DP fix is not higher than 8 seconds
- The course change associated with the previous DP fix was performed in the same direction as the course change associated with the currently traversed DP fix.

3.4.2 Course change

The course change c_m within maneuver m is expressed in degrees, where $\{ c_m \in \mathbb{Q} \mid -\infty < c_m < +\infty \}$. It gets calculated by summation of course changes associated with all DP fixes corresponding to maneuver:

$$c_m = \sum_{k=1}^{N} courseChange(DPfix_{mk}) \quad \Big| \quad DPfix_{mk} \in \text{DP-fixes set of the manoeuvre } m$$

The course change associated with $DPfix_k$ is determined by following steps:

Step 1: Difference d_k between bearing of $DPfix_{k+1}$ from $DPfix_k$ and bearing of $DPfix_k$ from $DPfix_{k-1}$ is calculated.

$$d_k = bearing(DPfix_k, DPfix_{k+1}) - bearing(DPfix_{k-1}, DPfix_k) \quad \Big| \quad DPfix_k \in \text{DP-fixes set}$$

The bearing between two DP fixes is derived by the direction of the **great circle line**[33] that runs between those fixes.

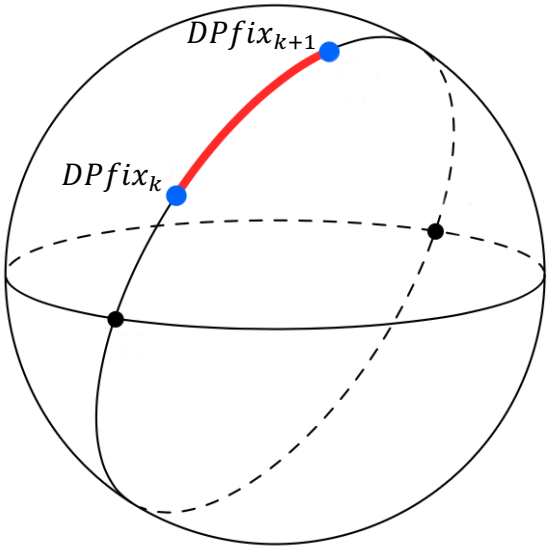

Figure 17: Example of a great circle line between two fixes[34]

Step 2: The real course change c'_k is determined by summation of course changes between each GPS fix lying within the time range between $DPfix_{k-1}$ and $DPfix_{k+1}$:

$$c'_k = \sum_{j=2}^{N} course(fix_j) - course(fix_{j-1}) \quad \Big| \quad fix_j \in \text{GPS-fixes set with removed outliers within the time range between } DPfix_{k-1} \text{ and } DPfix_{k+1}$$

[33] Great circle line defines the shortest path between two points on a sphere, where the sphere is representing the earth [18, p. 128]

[34] Source: *URL <https://en.wikipedia.org/wiki/Great-circle_distance>, Accessed 10.07.2018*

The course of fix_j is taken from the COG attribute provided by `GPSFixMoving`-interface.

Step 3: Course change c_k is inferred from d_k considering the sign of the real course change c'_k:

$$c_k = \begin{cases} d_k, & sign(d_k) = sign(c'_k) \\ d_k + 360 \cdot sign(c'_k), & sign(d_k) \neq sign(c'_k) \end{cases}$$

In other words, when the sign of d_k does not match the sign of c'_k, the degrees value of d_k gets flipped in order to approach the $DPfix_{k+1}$ by turning to the other side. For instance, when d_k is 90°, the flipped value will be -270 degrees. The case with unequal signs of d_k and c'_k occurs mostly within manoeuvres featuring a very high **turning rate**[35], e.g. penalty circles.

3.4.3 Maneuver beginning and end time points

While the `Maneuver`-interface does not provide access to beginning and finishing time of maneuver, the time points get computed internally within maneuver detection algorithm to deduce further maneuver data. The maneuver start time point is defined as the time point lying 4 seconds before the first DP fix of maneuver whereas the end time point as 4 seconds after the last DP fix.

3.4.4 Maneuver time point and position

The time point and position of maneuver are determined by location of its highest turning rate calculated between two successive GPS fixes contained within temporal maneuver boundaries. The turning rate between two fixes is calculated as follows:

$$s_j = \left| \frac{course(fix_{j-1}) - course(fix_j)}{duration(fix_{j-1}, fix_j)} \right|$$

$fix_j \in$ GPS fixes set with removed outliers lying between maneuver start and end time points

3.4.5 COG and SOG before and after maneuver

COG and SOG before maneuver are the COG and SOG of the first DP fix of maneuver. COG and SOG after maneuver are the COG and SOG of the last DP-fix. The associated

[35] Turning rate is regarded as course change speed which can be measured in degrees per second.

course c_k for a DP fix is derived from the bearing of the DP fix from to its preceding DP fix:

$$c_k = bearing(DPfix_{k-1}, DPfix_k)$$

The associated speed s_k for a DP fix is represented by the average SOG between DP fix and its preceding DP fix:

$$s_k = \frac{distance(DPfix_{k-1}, DPfix_k)}{duration(DPfix_{k-1}, DPfix_k)}$$

3.4.6 Maneuver type

There are several factors influencing maneuver type determination:

- Presence of waypoint marks
- Course change within maneuver
- COG before and after maneuver
- TWD

When a waypoint mark is passed, the maneuver is considered as a mark passing maneuver. In other case, the maneuver type gets determined by counting the number performed of tacks and jibes within maneuver based on the latter three influencing factors. When the number of tacks and jibes is zero, then the maneuver type gets regarded as head-up or bear-away, depending on turning direction with respect to TWD. When the number of tacks and jibes is higher than one, the maneuver type gets considered as a penalty circle. And if the number of tacks and jibes is one, the maneuver type is regarded as tack or jibe, depending on whether the number of tacks is one and the number of jibes is zero, or vice versa.

3.4.7 Maneuver loss

The maneuver loss is computed considering different maneuver section boundaries. The boundaries are determined by speed maxima search within GPS fixes set with removed outliers. The maneuver start is located by forward in time local speed maximum search starting from the maneuver start time point defined in *3.4.3 Maneuver beginning and end time points*. Analogously, maneuver end is determined by forward in time speed maximum search starting from maneuver end time point. However, to locate the maneuver end, the first 8 seconds are performed with a global maximum search followed by local maximum search. The yielded maneuver section is used for maneuver

loss calculation. This calculation is not discussed further because it introduces no relevance for the further course of this thesis.

3.4.8 Recursive maneuver detection algorithm calls

In some cases, the maneuver detection algorithm gets called recursively to detect maneuvers within a limited GPS fixes subset. Such recursive calls occur under following circumstances:

- A mark passing maneuver has been detected. In this case, two recursive executions are performed. The first execution gets performed with a GPS fixes subset contained within the time range of maneuver start time point and the time point of waypoint passing (excluding), whereas the second execution is performed with a GPS fixes subset contained between the time point of waypoint passing (excluding) and the maneuver end time point. The aim is to locate tacks and jibes within the section of mark passing maneuver for visualization purposes.
- A penalty circle has been detected and the number of tacks or jibes is higher than one. In such case, the maneuver gets split into multiple maneuvers where the course change of each maneuver gets limited to 360 degrees. The aim is to separate consecutive penalty circles for visualization purposes.

3.4.9 Conclusion

This review shows that maneuver detection is a non-trivial task. The idea of Douglas-Peucker algorithm employment for detection of base points for maneuver spots makes sense as well as the grouping strategy of DP fixes to infer maneuver spots. However, the inference strategy for maneuver section boundaries seems to be inappropriate for this work due to following reasons:

- DP fixes set is a subset of GPS fixes set with removed outliers. Hence, it can represent the maneuver curve in a more coarse-grained and imprecise manner compared to the curve based on smoothed GPS-fixes. Considering this fact, a conclusion can be drawn that the inference strategy for COG before and after maneuver might introduce a certain level of inaccuracy which can be avoided. Dealing with DP fixes can even cause a serious problem since the path based on DP fixes can contain abrupt and implausible turns like illustrated below:

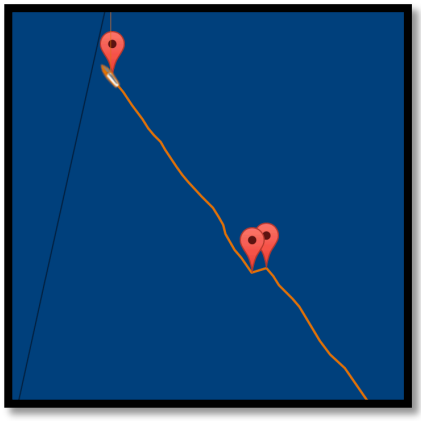

Figure 18: GPS-track with its DP fixes containing in the middle an abrupt implausible turn

Such GPS fixes constellations can produce maneuvers with a significant course change matching the usual course change range of tacks and jibes. Therefore, this type of maneuvers is regarded as incorrect. Moreover, it deteriorates the quality of maneuver data contained within SAP Sailing Analytics.

- The inferred maneuver beginning and end time points are based on naive assumption that all maneuvers start 4 seconds before the first DP fix and end 4 seconds after the last DP fix. These criteria are not sufficient to separate a GPS track into segments with stable course and speed from segments with course changes and speed instability influenced by maneuvers. The maneuvering is highly dependent to the boat class with its acceleration and turning characteristics. The boundaries determination introduced within maneuver loss calculation seems to be a more valid approach for maneuver detection.
- The calculation of SOG values before and after maneuver is based on DP fixes. SOG before maneuver gets averaged throughout the whole path starting from preceding maneuver. Analogously, SOG after maneuver incorporates the average of the whole track segment until the next following maneuver. Hence, these speeds do not represent the real beginning and exit speeds measured at the actual maneuver time frame. Instead, the speeds represent the average speed between maneuvers. Since the path between maneuvers can be of any duration and length, the wind and sea state can get arbitrarily changed. The averaging of such changes is unwanted for the wind estimation model because it hinders the creation of wind fixes related to the time point of a particular maneuver which in turn could be used for wind fix inference.
- The maneuver data of SAP Sailing Analytics includes the maneuvers determined by recursive execution of maneuver detection algorithm. Such

maneuvers include at least one boundary end which is defined either by the time point of waypoint mark passing or by reaching of course change with 360°. The courses and speeds associated with such boundary ends cannot represent stable courses at target speed. Therefore, maneuvers inferred by recursion are regarded as incomplete which leads to maneuvers which do not match the usual properties of its type.

The issues related with the maneuver detection are regarded as severe. Therefore, the adaption of maneuver detection algorithm and development of appropriate data cleansing strategy will be covered in detail in the data preparation phase.

3.5 Data storage

SAP Sailing Analytics backend persists its data in MongoDB[36]. However, most of its data, including GPS fixes, is hold in-memory without being persisted. When backend is booting, it loads raw GPS fixes of tracked races from third party services, such as TracTrac[37], and processes it accordingly to fit it in its own domain model.

The benefit of such data management includes fast data access, high throughput and minimal backup efforts. However, it also requires a server instance to provide at least 100 GB of main memory and multiple hours of booting time in order to load the data and fit it in its in-memory domain model.

3.6 Data access

Since the data required for this work is hold in-memory, there are no options of standardized data access as provided by ordinary database-management systems (DBMS). Therefore, other accessing options must be elaborated.

3.6.1 Data mining module

For data analysis purposes, SAP Sailing Analytics provides a UI-based data mining module where various predefined statistics can be queried. The statistics' values are calculated within a predefined logic which gets applied on targeted domain entities.

[36] MongoDB is a document-oriented NoSQL database, see https://www.mongodb.com

[37] TracTrac is a service which collects live GPS data for conducted events of any type, see http://www.tractrac.com/

Statistics queries support filtering and grouping by predefined criteria. Furthermore, aggregation strategies, such as minimal, maximal, mean and median value calculation are available. The set of predefined statistics in Data Mining module can be extended by adjustment of its Java code.

However, the Data Mining module does not provide any mechanism for utilization of any advanced data mining algorithms. The module provides only a limited set of proprietary data queries for statistics calculations which is selected within UI. The execution results are presented within UI either in charts or in raw numbers.

3.6.2 RESTful API

SAP Sailing Analytics makes its data publicly available through a RESTful API. The API allows retrievement of all data mentioned in chapter *2.2.3 SAP Sailing Analytics data* as JSON and is also extendable by Java code. For this work, this API is of special relevance because it allows the export of sailing data.

3.6.3 Data import into local server instance

The SAP Sailing Analytics platform is hosted by multiple server machines and supports data import from other SAP Sailing Analytics server instances. The data import comes into play when server instances are switched for a better load distribution, or for a fail-over. Therefore, it is possible to launch a server instance on a local machine and import sailing data from other active instances run within productive environment. By means of local code adjustments of SAP Sailing Analytics, the imported data can be written from memory to a persistent store. However, since a server instance of SAP Sailing Analytics requires at least 100 GB of main memory for operation with its whole dataset, and such resources were not made available for this work, this data access option gets complex. While it is still possible to import, persist and evict the in-memory data gradually by means of custom code, the effort for such setup is considered as high because it requires the understanding of the whole SAP Sailing Analytics import process including its configuration for all third-party data sources for each sailing event. However, this option is still regarded as useful for this work, especially when only a subset of SAP Sailing Analytics data is demanded. This is the case, when only polars without GPS-fixes are needed, or when integration tests are performed.

3.6.4 Third-party services

Since SAP Sailing Analytics platform does not persist GPS and wind data, it heavily relies on third-party services which supply the platform with that data. Each third-party service provides its proprietary interface for data queries which in most cases is a RESTful API. The SAP Sailing Analytics platform incorporates a client implementation for each of the used third-party services. Therefore, the client implementations can be used to retrieve GPS and wind data directly from third-party services.

This access option is regarded as error-prone and complex because it requires knowledge about each third-party service regarding its interface and data it provides. Furthermore, it requires analysis about how the data is transformed and fit into SAP Sailing Analytics domain model. Therefore, the RESTful API of SAP Sailing Analytics is considered as a more preferred access option.

3.7 Correlation concepts

There are various features which can be deduced from GPS-tracks of sailboats. The challenge for this section is to identify the data which might be relevant for this thesis. For this, first thoughts are made about possible correlation concepts with true wind in order to identify relevant data. Additionally, appropriate approaches for utilization of identified data are introduced.

3.7.1 Matching of actual polars with target polars

COG and SOG are important features of GPS track because it can be used to shape the **actual polars** which in turn can be matched with **target polars**. To match the actual polars A with the target polars T, the offset must be determined, such that for nearly all existing COG-SOG tuples contained within A, the following expression gets applicable:

$$A_c \approx T_{c-o}$$

$\{c, o \in \mathbb{Z}_{360} | \; c, \; o \geq 0\}$, c represents COG and o represents offset
$A_c \in \mathbb{Q}_{\geq 0}$ and represents the recorded SOG at COG c
$T_x \in \mathbb{Q}_{\geq 0}$ and represents the target speed from boat polars at $x = TWA + 360$, provided that $\{x \in \mathbb{Z}_{360} | \; x \geq 0\}$

When the offset is determined, no further calculations are required to infer TWD because the determined offset equals TWD.

However, when matching the actual SOG with the target SOG, it must be ensured that the boat is sailing a stable COG at a stable SOG and is not maneuvering. To achieve this, maneuver sections and its preceding and following acceleration sections must be located and filtered out for the polars matching process.

3.7.2 Maneuver analysis

As interviews with sailing experts revealed, the maneuver angle is another feature which could be matched with polar data to leverage inference of maneuver types.

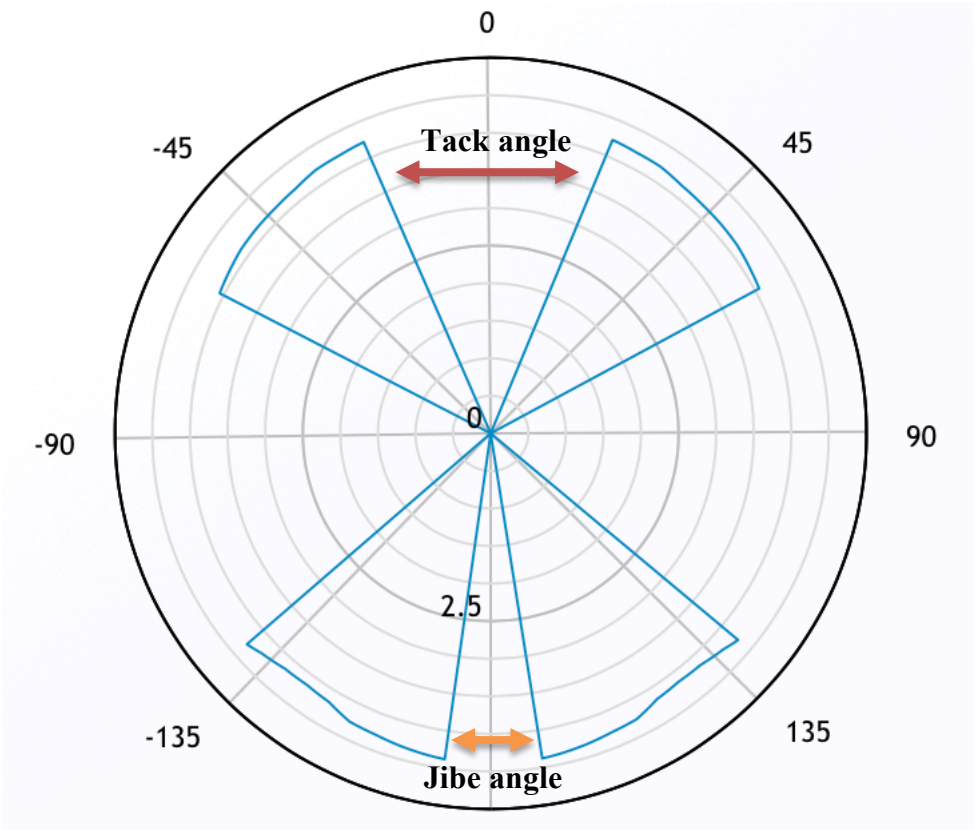

Figure 19: Maneuver type inference by maneuver angle

And it might be not the only feature contained within maneuver data which allows maneuver type inference. Turning rates and speed trends within maneuvers can contain additional clues pointing to a particular maneuver type. When a tack or jibe is known, TWD can be derived from the maneuver middle course. Moreover, the derived wind can be converted into a wind fix referencing the time point and the position of the corresponding maneuver.

3.7.3 Mark passing information

To win a race, a sailor must sail toward waypoints as fast as possible. Without consideration of wind, the most effective route from position A to position B is the line connecting position A and B directly. If the wind is blowing from or to the next waypoint, the sailor must sail in zigzag either by tacking or jibing. Sailing in zigzag with consecutive head-up and bear-away maneuvers toward next mark is counterproductive and is usually not performed during regatta racing. From this, a conclusion can be drawn that whenever a maneuver with a significant course change is recorded, and the maneuver is not a mark passing maneuver, the maneuver must be either tack or jibe. Head-up and bear-away with significant course changes are mostly performed during leg transitions. However, this assumption might turn out as wrong in cases, when TWD instantly changes by a significant amount. In this case, the sailors might need to adapt its COG to the new wind by bearing away or heading up with a significant course change. However, this is regarded a corner-case.

3.8 Confounders and mitigation

This section presents the elaborated factors that can distort the presented correlation concepts. The possible countermeasures for distortion mitigation are introduced as well.

3.8.1 Poor GPS quality

The recorded GPS track can get significantly distorted by poor quality of GPS tracking device. **Poor accuracy of GPS fixes** with significant outliers can lead to sections with wrong courses and unreasonable speeds as well as falsely located maneuvers. To mitigate the impact of poor GPS accuracy, **smoothing of GPS fixes** must be provided.

Small GPS sampling rate can negatively impact the quality of derived maneuver data, such that the analysis of maneuver course and speed trend becomes inappropriate. The only solution to handle this issue is to exclude the detailed analysis of maneuver data for the affected GPS track sections.

3.8.2 Improper usage of GPS-tracking devices

Some samples of SAP Sailing Analytics races revealed few cases, where a competing sailboat gets tracked within a car parking area during its participation within an ongoing race.

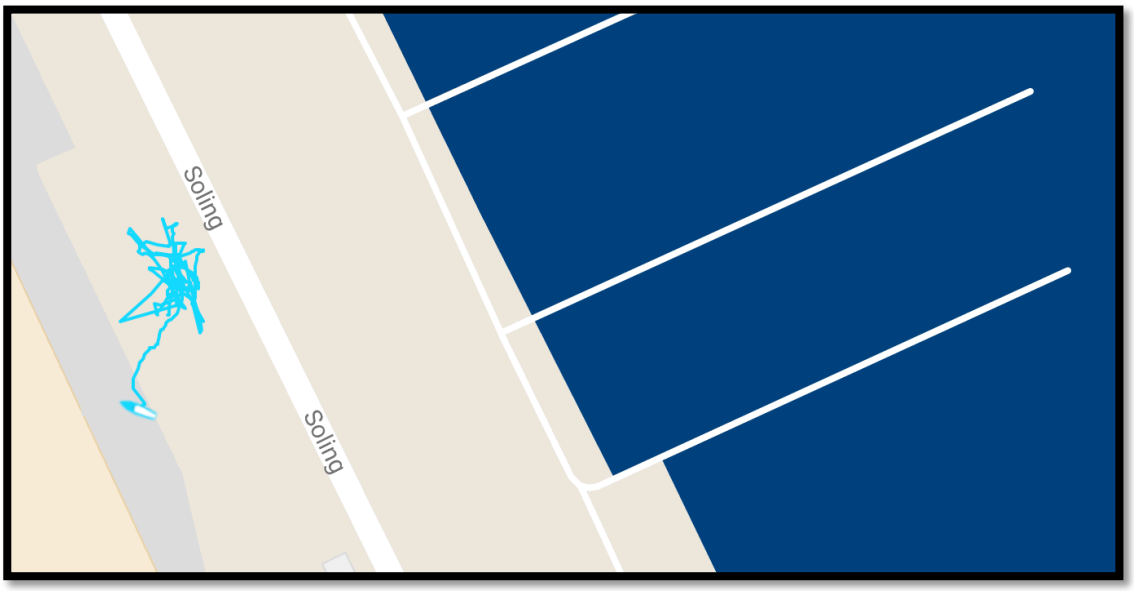

Figure 20: A sailboat is tracked during an ongoing sailing competition while it is standing still within a car parking area

This is an example of a human error. The affected tracking device could have been either forgotten in a car or the corresponding sailboat could have been excluded from the race and left on a trailer within car parking area. To prevent such an erroneous track, the tracking device had to be switched off, or the competitor had to be excluded from the competitors list. The consequence of this human mistake is that due to GPS inaccuracy, the non-moving tracking device is transmitting additional GPS fixes randomly positioned within an area of 20 meters. Within SAP Sailing Analytics platform, the falsely produced GPS track is processed normally and its data get regarded as part of the race, including incorrect SOG and falsely detected maneuvers. This issue affects the set of GPS tracks and maneuvers which will be analyzed and employed within this work. To mitigate this problem, implausible GPS tracks with corresponding maneuvers must be sorted out within data preparation phase. Eligible filtering criteria can be provided by definition of minimum and maximum for the average SOG of a GPS track.

3.8.3 Incorrect wind measurement

Interviews with sailing event organizers revealed that the wind measurement could also get incorrect. It can happen either due to technical issues with wind measurement systems, or due to a human mistake. An example for the latter case is an incorrect user input of manual wind records. Another already observed example is given when event staff decides to move a wind measurement system to another position without turning it off. If that case occurs during an ongoing race, the wind information recorded by the affected wind measurement system will be invalid because of produced apparent wind and obstacles surrounding the wind measurement device.

3.8.4 Sea state

The sea state is another crucial factor when it comes to matching of actual polars with target polars because the shape of actual polars can be significantly changed by a strong current. In worst case, the upwind versus downwind speed ratio can get inverted if a strong current against the wind is captured. The symmetry of the actual polars gets distorted if a strong current is flowing to the wind sideways. The countermeasures for the impact of currents on SOG can be a higher weighted consideration of estimation results coming from maneuver classification. Another countermeasure might provide the reshaping of actual polars with alignment to target polars.

However, currents flowing to the wind sideways cause another issue for this work by making all so far introduced features correlate with the current adjusted wind instead of the true wind, because all the features are inferred from GPS track which in turn provides only information about COG. The heading cannot be derived from GPS. Since the boat heading h is not influenced by the current and drag force on sail, it can be related to TWA a and true wind direction w as follows:

$$a' = h - w + 180$$

$$a = \begin{cases} a', & -180 < a' \leq 180 \\ a' - 360, & a' > 180 \\ a' + 360, & a' < 180 \end{cases} \qquad \{h, w \in \mathbb{Q} |\ 0 \leq h, w < 360\}$$

The COG c, however, gets influenced by the force of the current $cf()$ acting on the boat's hull and the drag force $df()$ acting on sail:

$$a' = c - w + 180 + cf(a) + df(a)$$

$$a = \cdots$$

$\{c, w \in \mathbb{Q} |\ 0 \leq c, w < 360\}$

$cf(a)$ and $df(a)$ are functions mapping TWA a to the drift between COG and

heading caused by force of current and drag force

Since $df(a)$ appears usually axisymmetric at 0° and 180° TWAs, it does not affect the issue with the determination of current adjusted wind instead of true wind. This also applies for $cf(a)$ if the current is flowing either directly against, or to the direction of true wind. But if the current is directed to the true wind sideways, the $cf(a)$ will become unknown. Since there will not be any information provided to the wind estimation component about the current and boat's heading, the only option for inference of true wind from determined current adjusted wind is the reshaping of actual polars to the alignment of the axisymmetric target polars. The reshaping process allows the inference of the motion vector of the current, which could be used to infer $cf(a)$.

3.8.5 Wind changes

True wind is a non-constant variable and can change over time. Wind gusts can lead to a time frame in which TWS gets significantly above the average. TWD is vulnerable to changes as well. However, according to interviews conducted with sailing experts, it is very unlikely that the TWD gets significantly changed within a short time frame. Therefore, the case where the wind flips its direction within few minutes will not be regarded as an issue of this work. Notwithstanding, the problem with wind changes underpins the necessity to relate estimated wind fixes to particular time point and position. To eliminate the confounders caused by wind gusts, the inferred data can be averaged.

3.8.6 Sailor skills

Beside boat characteristics, the performance of a sailboat on different TWAs is also influenced by the skills of sailors. A professional sailor knows the optimal TWAs which must be sailed in order to reach a destination as fast as possible. Depending on TWA, he also knows the best sail trim. Furthermore, professionals perform maneuvers more efficiently by loosing as less time and speed as possible. During sailing, professionals are also capable of predicting wind changes by observing the water in order to readapt the optimal TWA as soon as the sail gets affected by the changed wind. All these skills might not be perfectly developed by a novice or intermediate sailor. Therefore, depending on the sailor skills, the deviation level of actual polars with target polars can vary. To mitigate this confounder, deviations from actual polars to target polars must be tolerated. Best mitigation is performed by elaboration of general rules, which

apply irrespective of targets polars, e.g. “sailing upwind is slower than sailing downwind”.

3.8.7 SAP Sailing Analytics polars

The target polars provided by SAP Sailing Analytics are induced from all available GPS tracks. Hence, all the confounders of the sea state are contained within the target polars. However, because the target polars were generated considering a huge amount of GPS data, the polars represent the most likely confounders occurring within a sailing area. The benefit of such polars for this work is that the polars provided by SAP Sailing Analytics are related to COG instead of heading. Thanks to this fact, the matching of actual polars with target polars becomes easier because both polars are generated based on COG-SOG tuples derived from GPS.

3.8.8 Absence of target polars

SAP Sailing Analytics platform provides polars only for boat classes which were already racing within an event managed by the platform. Therefore, if an event with new boat classes gets conducted, there will not be any target polars provided to the wind estimation. To address this issue, general rules for polar data should be introduced.

4 Data preparation

This chapter covers all activities to produce the most appropriate data for wind estimation models. Firstly, the maneuver detection of SAP Sailing Analytics is adapted in order to support this work with demanded maneuver features and improve the data quality. Then, the relevant data contained in SAP Sailing Analytics is selected and imported from SAP Sailing Analytics into local environment for further pre-processing. The pre-processing includes feature engineering, data exploration by means of various data mining algorithms and visualization techniques, and final feature selection for maneuver type classification. Lastly, the remaining data pre-processing steps are conveyed.

4.1 Improvement of maneuver detection

The chapter *3.4 Maneuver detection* revealed severe issues related with SAP Sailing Analytics maneuver detection algorithm. These issues are summarized in section *3.4.9 Conclusion*. In a nutshell, it is error-prone to poor GPS quality and does not define a clear concept for maneuver section boundaries. Therefore, an improvement for the algorithm is initiated. The yielded algorithm improves the overall quality of maneuver data which is crucial for wind estimation accuracy. Additionally, it supports existing SAP Sailing Analytics components with a clean and reasonable maneuver concept producing more representative maneuver data.

4.1.1 Definition of maneuver

According to conducted interviews with experienced sailors, a maneuver is a section with unstable course and speed occurring due to rearrangement of sailboat to a new course. Thus, it starts already when the sailing crew is preparing to maneuvering because the preparation phase might already affect the speed. A maneuver ends when the boat reaches its new target course at target speed. From these considerations, a conclusion about maneuver parts is drawn:

1. Since the main purpose of a maneuver is turning, a maneuver must contain a turning section with a significant course change which is dominated either by port or starboard.
2. Before the turning section, the sailboat might lose speed due to crew preparations, e.g. due to sail trim change. Therefore, the start of maneuver is the point

when the boat's speed starts to drop continuously before the beginning of its turning section.

3. After the turning section, the sailboat is regaining its target speed and is rearranging to its target course. The course rearranging can even occur after the turning section, because in case of tacks, it is a proven strategy to oversteer for few seconds away from the wind in order to regain the boat's speed faster. Therefore, the end of maneuver can be regarded as the point when the boat's speed starts to drop after a continuous increase and boat's course gets stable.

The aforementioned considerations allow to draw another conclusion that a maneuver is composed of three ordered phases:

1. Preparation phase: sailboat prepares for maneuver
2. Turning phase: sailboat performs its main turn toward the new target course with possible oversteering
3. Recovery phase: sailboat regains its speed and rearranges to its target course

This definition of maneuver is of cause unofficial because it is underpinned only by opinions of few sailors and is not mentioned in any literature. However, for this work, it suits well since it allows the extraction of segments with stable course and speed for matching it with target polars. It also allows derivation of many measures about maneuvers concerning the turning section, speed recover section and final maneuver angle after rearrangement to target course.

4.1.2 Mark passing maneuvers

In previous maneuver detection the mark passing maneuvers have been treated as a separate maneuver type. This is considered as suboptimal for wind estimation and SAP Sailing Analytics race map because this maneuver type does not reveal any information regarding the wind. Depending on location of waypoints, a mark passing maneuver can be also considered as tack, jibe, bear-away and head-up. Therefore, it would be more informative to provide mark passing information within maneuver as an extra information besides the real wind dependent maneuver type. Following this consideration, mark passing has been removed from the set of possible maneuver types. Instead, the `Maneuver`-interface has been extended to provide mark passing information separately which is packed in `MarkPassing`-interface. If the maneuver is not a mark passing maneuver, `null` will be returned instead of a `MarkPassing`-instance.

4.1.3 SAP Sailing Analytics maneuver semantics

In context of races managed by SAP Sailing Analytics, some maneuvers must be split in order to provide a more suitable view for the audience and jury. For instance, the requirement for splitting applies when multiple consecutive penalty circles are sailed. In such a case, it is demanded by the audience to spot each penalty circle on the race map separately. Another case when splitting applies is when a penalty circle is sailed between two legs. In this case, the sailed penalty circle does not get recognized as such and the maneuver gets split into successive tack and jibe.

Since the maneuver boundaries affected by splitting do not represent adjacent segments with stable course and speed, and the splitting logic is not based on the circumstances involving true wind and propelling physics of sailboats, it gets considered as a confounder for wind estimation. Therefore, in order to eliminate this confounder, as well as to maintain the compliance with SAP Sailing Analytics race map, different maneuver data will be provided for both components. In contrast to maneuver data for the race map, the maneuver data for wind estimation will be free of any contra productive splitting logic.

4.1.4 Smoothed GPS-fixes data

The course and speed trend analysis are the most essential parts of maneuver detection. Therefore, it is crucial to processes sanitized and smoothed COG and SOG values which are inferred from GPS fixes. The section *3.4.9 Conclusion* has already pointed out that without application of a smoothing strategy, unreasonable maneuvers with high course changes might get falsely captured. Considering these facts, a decision has been made to employ SAP Sailing Analytics smoothing algorithm for deduction of smoothed COG and SOG values within maneuver detection code. Since the SAP Sailing Analytics smoothing algorithm is regarded as computation intensive, its usage must be limited. It is considered as a contra productive to map the whole GPS fixes set with removed outliers to the smoothed GPS fixes set. Instead, `GPSFixTrack`-interface with its implementations has been extended by the following method:

(*timePointFrom*, *timePointTo*) → (`SpeedWithBearingStepsIterable`)

The method is meant to provide a uniform method for querying course and speed trend information within provided time range throughout maneuver detection code. The time range queried by maneuver detection will be limited to the time range of possible maneuver sections.

The returned type `SpeedWithBearingStepsIterable` is internally composed as an ordered set of `SpeedWithBearingStep`-elements where each element contains the following information:

- **Time point** to which the measures of the instance are referring.
- **COG** and **SOG** sailed at associated time point. The calculation of COG and SOG is based on SAP Sailing Analytics smoothing algorithm which was introduced in *3.2.2 GPS-fixes smoothing algorithm*.
- **Course change** performed between the previous speed-with-bearing step and this step. The course change of the first step within a `SpeedWithBearingStepsIterable`-instance is always zero.
- **Turning rate** which is measured in degrees per second. It is calculated by dividing the course change of the current step by the duration between the time points of previous step and current step. The turning rate of the first step within a `SpeedWithBearingStepsIterable` -instance is always zero.

The sum of course changes of all steps contained within a `SpeedWithBearingStepsIterable`-instance represents the total course change performed between the time points of its first and its last step. Since the course change calculation for a step depends on the COG of the previous step, at least two steps are required to produce a nonzero total course change for the queried section.

The frequency of returned steps depends on the frequency of GPS fixes contained in GPS fixes set with removed outliers within the queried time range. Since the results of SAP Sailing Analytics smoothing algorithm get implausible when applying it on interpolated sections of a GPS track, the speed-with-bearing steps are sampled only at time points of GPS fixes which exist in GPS fixes set with removed outliers. Considering the total course change calculation and the frequency constraint of speed-with-bearing steps, a conclusion can be drawn that the aforementioned specification can introduce issues for maneuver detection when processing GPS tracks with poor GPS quality. Especially when dealing with low GPS sampling rates, the case might occur regularly where less than two GPS fixes might exist within the queried time range. This will hinder the maneuver detection to recognize maneuvers because no proper course change inference can be performed. In order to guarantee that at least two speed-with-bearing steps get returned for provided time range with *timePointFrom* < *timePointTo*, the following specification regarding the time points of the first and the last step for returned speed-with-bearing sets has been elaborated:

- The time point of the first step is the time point of the first GPS fix located at or before provided *timePointFrom*
- The time point of the last step is the time point of the first GPS fix located at or after the provided *timePointTo*

4.1.5 Average duration of turning sections

Some sailboats are more maneuverable than others. The name of corresponding boat class allows to draw conclusions about boat's maneuverability. Therefore, in order to query speed-with-bearing steps for maneuver detection purposes within a reasonable time range, `getAverageTurningDuration()` is introduced which specifies the assumed duration of turning sections with respect to the boat class of sailboat. It is used as reference value for duration of turning sections which are composed of exactly one DP fix.

Unfortunately, the existing code of SAP Sailing Analytics does not provide an option to derive a representative value for `getAverageTurningDuration()` for each boat class individually. Therefore, throughout this work the value returned by this method will be 8 seconds.

4.1.6 Main steps for new maneuver detection

With respect to specifications and considerations introduced in previous sections of this chapter, the new maneuver detection algorithm is introduced. The algorithm was designed by top-down approach which will be also used for its description. This section provides an overview of the main steps applied by the new maneuver detection algorithm. The calculation details of each step with its reasoning are presented in next sections.

Step 1: Detect maneuver spot locations using Douglas-Peucker algorithm and group the fixes by same criteria as introduced in *3.4.1 Maneuver spots location*. Each maneuver spot k gets represented by DP fixes group with $DPfix_1^k$ as first DP fix of the group and $DPfix_m^k$ as last DP fix of the group. Each DP fixes group has its associated course change direction. The following steps consider each detected maneuver spot individually.

Step 2: Determine the boundaries of the turning section for maneuver spot k. The turning section boundaries are expressed by two time points which represent a closed interval $[\ t_1^k;\ t_n^k\ ]$ covering the time range of the turning section.

Step 3: Determine the boundaries of maneuver section corresponding to maneuver spot k such that before and after the boundaries, segments with stable course and speed are expected. For this, $[\ t_1^k;\ t_n^k\ ]$ is taken as initial maneuver section and is extended in both chronological directions by speed maxima and course fluctuations analysis. The yielded boundaries are represented as $[\ t'^k_1;\ t'^k_n\ ]$.

Step 4: Create and initialize an instance of `CompleteManeuver` based on the turning section and maneuver section inferred in steps 2 and 3 for each maneuver spot k if the total absolute course change of the turning section is at least one degree.

Step 5: Map the `CompleteManeuver`-instance from maneuver spot k to one or more `Maneuver`-instances considering semantical rules for SAP Sailing Analytics race map mentioned in *4.1.3 SAP Sailing Analytics maneuver semantics*.

The step 5 is not required by wind estimation. It was only implemented due to imposed semantics of existing SAP Sailing Analytics race map. To eliminate the confounders introduced for wind estimation by maneuver splitting, the maneuver data represented by `CompleteManeuver`-interface will be used for wind estimation, whereas the maneuver data represented by `Maneuver`-interface will be used in already existing SAP Sailing Analytics components. Since the details of step 5 are irrelevant for wind estimation, they will not be discussed further.

4.1.7 Relationship of maneuver data

The relationship and cardinality of all aforementioned data representation classes is depicted by the following UML class diagram:

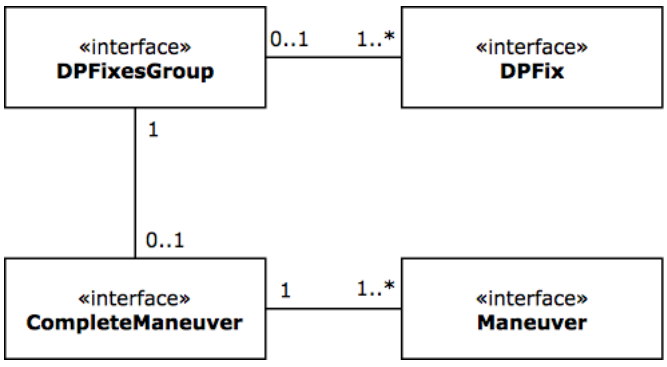

Figure 21: Cardinality of maneuver entities

4.1.8 Maneuver spot detection

The first step of the new maneuver detection leans heavily on the step of old maneuver detection algorithm described in *3.4.1 Maneuver spots location*. It utilizes Douglas-Peucker algorithm in order to locate the main points shaping the sailed path. The main points are DP fixes at which significant course change is performed. After DP fixes determination, the inferred DP fixes are grouped accordingly to represent a single maneuver spot by a DP fixes group. The difference between old and new maneuver spot detection algorithm is the calculation strategy of course changes associated with each DP fix which in turn is used for inference of turning direction associated with a DP fix and thus, influences the grouping of DP fixes. The main turning direction associated with $DPfix_i$ which is determined considering the sign of `totalCourseChange` which is defined as follows:

```
duration := getAverageTurningDuration() / 2
timePointFrom := max(timePoint(DPfix_i) - duration; timePoint(DPfix_i-1))
timePointTo := min(timePoint(DPfix_i) + duration; timePoint(DPfix_i+1))
steps := getSpeedWithBearingSteps(timePointFrom, timePointTo)
totalCourseChange := 0
foreach step in steps
    totalCourseChange = totalCourseChange + step.courseChange
```

Listing 1: Calculation of course change associated with a DP fix

Negative sign implies port, whereas positive sign implies starboard direction.

For instance, assuming that `getAverageTurningDuration()` is 8 seconds, the total course change associated with $DPfix_i$ will be calculated based on the speed-with-bearing steps queried within ±4 seconds time range around the time point of $DPfix_i$ if the interval to previous and next DP-fixes is ≥ 4 seconds.

4.1.9 Determination of turning section

The purpose of turning section is to represent the section where the main turn of maneuver is performed toward the direction associated by DP fixes group of a maneuver spot. It is assumed that a turning section is located close to the boundaries of DP fixes group. More precisely, each DP fixes group of a maneuver spot k contains exactly one turning section which is located between the following `timePointFrom` and `timePointTo` definitions:

```
timePointFrom := timePoint(DPfix_1^k)-getAverageTurningDuration() / 2
timePointTo := timePoint(DPfix_m^k)-getAverageTurningDuration() / 2
```

Listing 2: Assumed maximal interval for a turning section

To locate the boundaries of the turning section, the course change attribute of speed-with-bearing steps is analyzed. The aim of the course change analysis is to locate the smallest ordered subset of processed speed-with-bearing steps with the highest total course change matching the turning direction associated with its DP fixes group.

4.1.10 Determination of maneuver section

For this work, maneuver section is defined as the section of the whole maneuver encompassing preparation, turning and recovery phases described in *4.1.1 Definition of maneuver*. Thanks to determined turning section, the interval of the turning phase is already known. Hence, the goal of this step is to infer the intervals for the remaining preparation and recovery phases. To achieve this, the basic idea for existing SAP Sailing Analytics maneuver loss computation regarding speed maxima search is reused. Additionally, stable course search has been also employed in order to capture the rearrangment to target course after oversteering.

Speed maxima analysis

In order to detect the beginning of preparation phase, the first GPS fix needs to be located, from which the speed begins to drop continuously until the beginning of turning phase. More formally, local speed maxima search[38] is performed by traversing speed-with-bearing-steps backward in time starting from t_1^k. Conversely, after the turning phase, the sailboat begins to accelerate and regain its target speed after speed loss. Thus, the earliest end of recovery phase ends is when the sailboat stops continuously increasing its speed. Analogously to preparation phase beginning, one possible end of recovery phase can be determined by speed maxima search on speed-with-bearing steps traversed forward in time starting from t_n^k. However, the speed can get instable for a short time frame immediately after t_n^k, e.g. due to sailboat adjustments after turning. For previous considerations it means that a local speed maximum search could stop immediately after t_n^k and locate a local speed maximum which gets considerably lower than the speed followed after next few GPS fixes. To address this issue, it must be

[38] In contrast to global maximum search, a local search stops when the next traversed value is smaller than its previous value.

ensured that the speed search is firstly performed globally for an interval which is assumed to be the minimal interval for speed recovery for present boat class. Subsequently, local maximum search must be performed to locate the first fix with regained target speed.

Course fluctuation analysis

To mitigate the speed loss within maneuvers, oversteering tactic can be applied to accelerate the sailboat at more efficient TWA ranges right after suffered speed loss. Especially for tack-maneuvers, oversteering within recovery phase has been already proven as an effective strategy for time loss minimization within maneuvers. To capture oversteering within recovery phase, course fluctuation analysis has been added as a subsequent part of speed analysis. The aim of the course fluctuation analysis is to locate the first speed-with-bearing step with a turning rate not higher than a threshold value defining the maximal acceptable turning rate for a stable course between two GPS fixes.

Intervals for speed and course analysis

To avoid unreasonably long maneuver sections, interval limits for speed and course analysis are introduced. Since the average length of maneuver sections is highly dependent on boat class, the value of `getAverageTurningDuration()` is used as the boat class dependent scaling factor for applied interval limits. The applied limits are based on intuition of sailing experts only. A sophisticated derivation of limits has been avoided due to time reasons. It is assumed that the duration of preparation phase is much shorter than the duration of recovery phase because the speed loss is occurring primarily within the turning section, whereas speed recovery is performed throughout the whole recovery phase. Hence, a decision has been made for the following interval limits specification:

Table 3: Interval limits for speed and course analysis to locate maneuver section boundaries

Maximal duration of preparation phase	`getAverageTurningDuration()`
Maximal duration of recovery phase	`getAverageTurningDuration()` * 3
Duration for global speed maximum search within preparation phase before t_1^k:	`getAverageTurningDuration()` / 8

Duration for global speed maximum search within recovery phase after t_n^k	`getAverageTurningDuration()`

The aforementioned table implies that same time range is shared by speed analysis and course analysis. This also means that the farer the speed maximum gets located from t_n^k the shorter will be the remaining search interval for course analysis.

Colliding maneuver sections

In contrast to turning sections, the boundaries of maneuver sections are determined irrespective of the direction of performed course changes. The search interval limits for maneuver section inference are also longer than of turning section. This allows to conclude that maneuver sections can collide and overlap if the interval between two turning sections gets small. Moreover, considering the grouping strategy of DP fixes regarding associated turning direction criterion and the aim of maneuver sections to capture the oversteering part of maneuvers, another conclusion can be made that the maneuver section can even include another maneuver. This is the case when a sailboat performs a tack maneuver with oversteering. The part of rearranging the boat to its target course after oversteering can be recognized as a head-up maneuver. And because the head-up section is also considered as part of the tack after oversteering, it inevitably gets included inside the maneuver section of the tack. However, for SAP Sailing Analytics audience, this overlapping is considered more as a feature than as an issue because it allows to decompose a complex maneuver in its subcomponents. Thanks to this feature, the oversteering angle can be determined from the head-up maneuver located within oversteered tack.

However, while overlapping of oversteered tack with its head-up is considered as a positive effect of maneuver collisions, there are still other scenarios where overlapping gets regarded as an issue. For instance, two tacks performed in different directions within a short time frame could also overlap, especially due to course fluctuation analysis which will continue if it reaches the turning section beginning of the subsequent tack. Since overlapping of two tack maneuvers is considered as unwanted, additional limits concerning course change must be introduced for speed and course analysis. Similar to interval limits, the course change limits have been elaborated based on intuition which also means that these are the hooks for possible fining tuning of maneuver detection algorithm in the future. The basic idea is to allow maneuver section being extended with overlapping next maneuver only for maneuvers which are considered as more complex than its next maneuvers. The total course change within turning section

is used as maneuver complexity indicator. More precisely, the result of a maneuver extension step gets discarded if the absolute total course change produced within corresponding phase exceeds one third of absolute total course change within the turning section.

Pseudocode

To clarify the processing logic for speed and course analysis with its interval and course change limits, pseudocode for the whole maneuver section determination is shown. The snippet below shows the determination of preparation phase beginning:

```
globalSearchDurationLimit := getAverageTurningDuration() / 8
localSearchDurationLimit := getAverageTurningDuration()
globalSearchUntil := t_1^k – globalSearchDurationLimit
localSearchUntil := t_1^k – localSearchDurationLimit
steps := getSpeedWithBearingSteps(localSearchUntil, t_1^k)
sortStepsBackwardInTime(steps)
stepWithMaxSpeed1
    := findGlobalSpeedMaximum(steps, from: t_1^k, to: globalSearchUntil)
stepWithMaxSpeed2
    := findLocalSpeedMaximum(steps, from: globalSearchUntil, to: localSearchUntil)
stepWithHighestSpeed
    := getStepWithHighestSpeed(stepWithMaxSpeed1, stepWithMaxSpeed2)
stableCourseSearchTo := t_1^k
if (getAbsoluteTotalCourseChangeBetween(stepWithHighestSpeed.timePoint, t_1^k) ≤
          getAbsoluteTotalCourseChangeBetween(t_1^k, t_n^k) / 3)
    stableCourseSearchTo = stepWithHighestSpeed.timePoint
stepWithStableCourse
    := findStepWithStableCourse(steps, from: localSearchUntil,
       to: stableCourseSearchTo,
       turningRateThresholdForStableCourseInDegreesPerSecond: 1)
preparationPhaseBeginningTimePoint:= stableCourseSearchFrom
if (getAbsoluteTotalCourseChangeBetween(stepWithStableCourse.timePoint, t_1^k) ≤
          getAbsoluteTotalCourseChangeBetween(t_1^k, t_n^k) / 3)
    preparationPhaseBeginningTimePoint = stepWithStableCourse.timePoint
```

Listing 3: Determination of preparation phase beginning

The end of the recovery phase is determined as follows:

```
globalSearchDurationLimit := getAverageTurningDuration()
localSearchDurationLimit := getAverageTurningDuration() * 3
globalSearchUntil := t_n^k + globalSearchDurationLimit
localSearchUntil := t_n^k + localSearchDurationLimit
steps := getSpeedWithBearingSteps(t_n^k, localSearchUntil)
stepWithMaxSpeed1
    := findGlobalSpeedMaximum(steps, from: t_n^k, to: globalSearchUntil)
stepWithMaxSpeed2
    := findLocalSpeedMaximum(steps, from: globalSearchUntil, to: localSearchUntil)
stepWithHighestSpeed
    := getStepWithHighestSpeed(stepWithMaxSpeed1, stepWithMaxSpeed2)
stableCourseSearchFrom := t_n^k
if (getAbsoluteTotalCourseChangeBetween(t_n^k, stepWithHighestSpeed.timePoint) ≤
          getAbsoluteTotalCourseChangeBetween(t_1^k, t_n^k) / 3)
    stableCourseSearchFrom = stepWithHighestSpeed.timePoint
stepWithStableCourse
    := findStepWithStableCourse(steps, from: stableCourseSearchFrom,
        to: localSearchUntil,
        turningRateThresholdForStableCourseInDegreesPerSecond: 1)
recoveryPhaseEndTimePoint := stableCourseSearchFrom
if (getAbsoluteTotalCourseChangeBetween(t_n^k, stepWithStableCourse.timePoint) ≤
          getAbsoluteTotalCourseChangeBetween(t_1^k, t_n^k) / 3)
    recoveryPhaseEndTimePoint = stepWithStableCourse.timePoint
```

Listing 4: Determination of recovery phase end

Thus, considering the variables within the pseudocode above, the time range $[\ t'^{k}_{1};\ t'^{k}_{n}\ ]$ of manoeuvre section equals to:
$[\ preparationPhaseBeginningTimePoint;\ recoveryPhaseEndTimePoint\ ]$

4.1.11 CompleteManeuver-interface

The development of the maneuver detection has allowed a substantial amount of freedom for designing its output. However, since the whole maneuver data of all races is calculated during server booting and gets managed completely in-memory, caution was required to prevent extensive computation and memory footprint. Therefore, `CompleteManeuver`-interface has been designed to incorporate only the data which is highly demanded by SAP Sailing Analytics components, as well as by wind estimation component for calculation of further required features. Its structure is represented as follows:

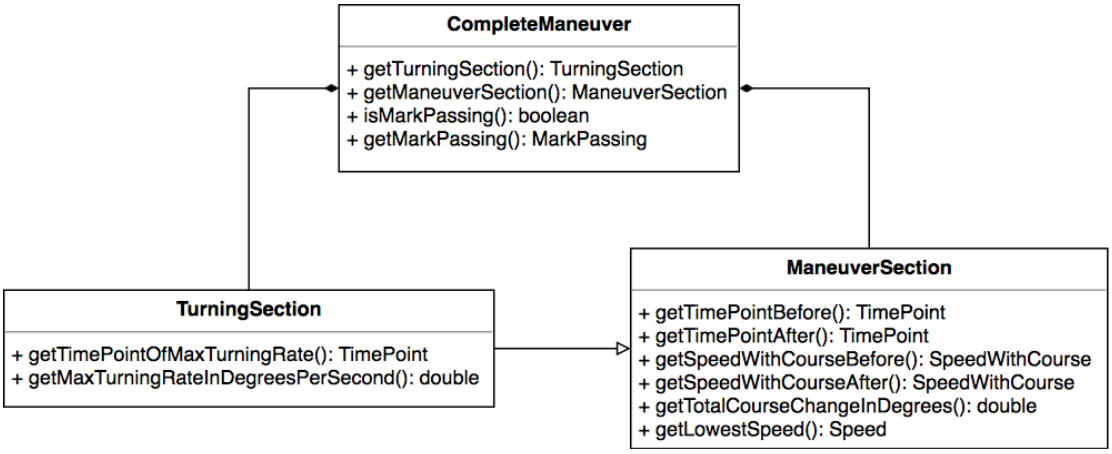

Figure 22: UML class diagram of `CompleteManeuver`-interface

4.1.12 Conclusion

The final maneuver detection algorithm provides various benefits for wind estimation, as well as existing SAP Sailing Analytics components. It defines a clean concept for turning and maneuver sections and incorporates GPS fixes smoothing to handle inaccurate GPS signals. Thanks to the development of maneuver detection, the complete understanding of maneuver data could be obtained. Furthermore, the `CompleteManeuver`-interface has been designed so that it optimally fits into the wind estimation component and provides a access to relevant data which is mentioned in *4.2.4 Maneuver data*.

4.2 Data selection

The chapter *3 Data understanding* has already introduced the data which is considered as relevant for this work. The aim of this section is to elaborate the identified data more precisely on entity and attribute level, as well as to consider the unavailable data which could additionally support this work.

4.2.1 GPS fixes

The GPS fixes are the most basic input which a wind estimation model can get. It can be used for derivation of further information such as polars and maneuvers. The aggregated information contains meaningful patterns which in turn can be used for wind inference. Thus, the question arises whether the whole set of GPS fixes contained within SAP Sailing Analytics is required for generation of wind estimation model.

There is also the possibility to design the wind estimation model so that it interacts directly with the aggregated polars and maneuvers. Since the size of GPS data is much bigger than of its derived polars and maneuvers, a considerable amount of computation and memory resources could be saved during model training, as well as during scoring which is beneficial for operation characteristics of the algorithm within production environment. Considering these facts, the decision has been made to refrain from employment of plain GPS fixes dataset and to use the maneuver dataset instead. Similar to speed and course information contained in GPS fixes, maneuver instances include COG and SOG before and after maneuver which reflect segments with stable course and speed.

4.2.2 Wind data

Nearly all races managed by SAP Sailing Analytics include wind measurements. It is vastly beneficial for this work since the wind data provides target values for wind estimation. It can be used for evaluation of estimation quality. It also allows employment of supervised learning algorithms. Therefore, the wind entity with following attributes is considered as relevant:

- Wind course in degrees
- Wind speed in knots
- Wind position which is composed of longitude and latitude
- Wind measurement time point represented in Unix-format

4.2.3 Polar data

In SAP Sailing Analytics, polar data is maintained within a complex proprietary service. Since its internals are complex, its relevant data cannot be mapped in entity attributes. Instead, the service methods are listed which seem to be relevant for this work:

1. (boat class, SOG, TWA) → (TWS)
2. (boat class, SOG, maneuver angle, maneuver type) → (TWA, TWS)
3. (boat class) → (boolean)

The first method has been added as part of improvement of `PolarDataService` mentioned in *3.3.5 Conclusion*. It is useful for TWS derivation for assumed TWA. The second method can be used to derive additional features for maneuver classification which indicate how close the maneuver angle to ordinary tacking angle is, as well as to

ordinary jibing angle. The third method shall indicate whether the polar data are available for a given boat class.

To make the polar data available for modelling, `PolarDataService` with its internal state of regressions will be completely imported and reused in order to generate the required data lazily on demand.

4.2.4 Maneuver data

Maneuver data is considered as highly promising data category for wind estimation. The most essential parts of maneuver data are maneuver sections and turning sections. While maneuver sections allow deduction of segments with stable course and speed for matching with target polars, turning sections might provide additional characteristics for maneuver classification. When the maneuver sections of tack and/or jibe maneuvers are known, wind can be derived from its middle courses.

The information whether the maneuver is a mark passing maneuver is useful because it provides additional insights about maneuver reasoning (see *3.7.3 Mark passing information*).

Overall, in terms of data selection the whole information contained within `CompleteManeuver`-interface is demanded. To preserve the order of maneuvers within each GPS track, it will be managed as sequences corresponding to a GPS track.

4.2.5 Boat class

The usage of polars requires the knowledge about the boat class. Therefore, it must be linked with each processed instance which is based on sailboat's GPS data. Considering the aforementioned data selection, these are maneuver sequences. Since the physical properties of each boat class are varying, the boat class information can also be used to train separate models for each boat class.

4.2.6 Race metadata

Within SAP Sailing Analytics, all wind measurements, GPS-fixes and maneuvers are related to a specific race. Each race is held at different time and place with its individual setup, including mark positions, wind measurement systems, competitors and boat class regulations. Therefore, a race can be seen as a wrapping element for a set of aforementioned data.

Considering the structure of `TrackedRace`-interface, each race belongs to a regatta. There are various regatta events which are organized by SAP Sailing Analytics. Some regattas limit its competitor list to professional sailors, while other regattas also include hobby sailors. Some regattas can be also associated with a specific race plan considering race duration and waypoints. Therefore, the information about regatta is considered to be as an additional extra which can be regarded as a criterion of data sampling for model training and evaluation. Furthermore, the information about regatta and race name, as well as competitor name for each GPS-track is regarded as important information for debugging. To sum it up, the following metadata from races is considered as important:

- Regatta name
- Race name
- Name of competitor corresponding to GPS track

4.2.7 Demanded data which is unavailable

This section elaborates further data, which could be collected with low efforts and costs in order to provide additional benefit for this thesis. Considering the capabilities of smartphones which are partially used as tracking devices within SAP Sailing Analytics, the mentioned data can be captured, provided that the smartphone is mounted horizontally parallel to the boat's center line.

Heading

The heading of sailboat is considered as an important variable since it can be used as a preferred alternative to COG for derivation of polars and maneuver angles. The polars and maneuver angles would be free of confounders introduced by current and wind drag. Furthermore, the drift between course and heading could be determined which in turn allows to draw conclusions about the current. Additionally, it might be a further feature candidate for wind estimation model.

Pitch, roll and yaw

A capturing strategy of pitch, roll and yaw by means of smartphone's gyroscope has been already introduced by RaceQs. These data could contain additional patterns which might be beneficial for maneuver classification.

4.3 Data composition

The previous section has introduced the data which seem to be relevant for this work. The next step is to define the datasets with its appropriate structures which wrap the selected data in a suitable and efficient manner to allow employment of supervised ML models.

In supervised learning algorithms, the model training is performed by providing a feature vector with its corresponding target values for prediction. In terms of wind estimation, the target values are TWD and TWS, while in terms of maneuver classification, the target value for each maneuver is its maneuver type. The aim of this section is to design an appropriate representation of datasets which contain all the information required for training and evaluation of a wind estimation models. For this, all the data presented in *4.2 Data selection* will be merged accordingly so that each single dataset provides all the information required for maneuver classification and/or wind regression.

4.3.1 Datasets

The selected data has been divided into the two following datasets:

- Sequences of maneuvers for each sailboat's GPS track
- Targets polars for each boat class

While the first dataset is meant to be used as model input, the target polars are meant to be used by wind estimation model internally on demand. In contrast to model input, the structure for polars dataset is already given by `PolarDataService`. The interface with its complex internals shall be completely reused. The structures for GPS fixes and maneuvers are defined in the following sections.

4.3.2 ManeuverWithWindEstimationData

`ManeuverWithEstimationData`-interface represents a single maneuver with true wind and waypoint information. The interface comprises the following information:

1. All the data provided by `CompleteManeuver`-interface, including time points, COGs and SOGs at the boundaries of maneuver section and turning section, lowest speed and total course change within each section, maximal turning rate and its time point within turning section, and mark passing information
2. TWD in degrees, TWS in knots

3. Highest speed within turning section
4. Longest interval between two GPS fixes within turning section and within maneuver section
5. Interval between the first fix of maneuver section and its previous fix
6. Interval between the last fix of maneuver section and its next fix
7. Interval between the end of the previous maneuver section and the beginning of the current maneuver section with its average COG and SOG
8. Interval between the end of the current maneuver section and the start of the next maneuver section with its average COG and SOG

While the first three list items are meant for feature engineering for maneuver classification, the remainder will be used for data cleansing. Since collisions of maneuver boundaries are tolerated by maneuver detection, the intervals between maneuvers can get negative. For those cases, the average COG and SOG values shall be set to `null`.

4.3.3 Wrapping of datasets

As discussed in *4.2.6 Race metadata*, it is regarded as beneficial to group maneuvers considering the race metadata. This section defines the appropriate data structures wrapping sequences of maneuvers. The structures are depicted as interfaces in UML diagram below:

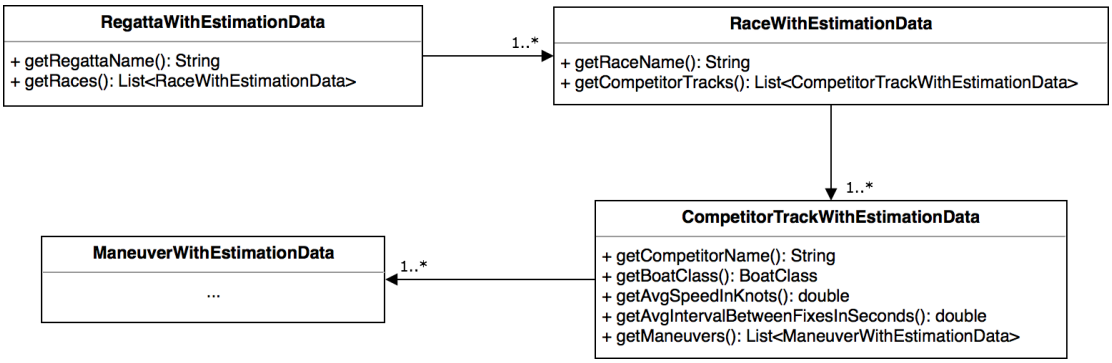

Figure 23: UML diagram of classes for imported data

4.4 Data acquisition

After the demanded data is known, appropriate data extraction strategy is elaborated considering the data access options of SAP Sailing Analytics which were discussed in *3.6 Data access*. The goal of this section is to set up a plan about how to retrieve the demanded data and import it in a local environment for further processing.

4.4.1 Selection of data access options

Within the section *3.6 Data access,* all possible access options for SAP Sailing Analytics data are listed. This section selects the most appropriate access options considering the effort and aptness for data extraction implementation.

SAP Sailing Analytics RESTful API

This is considered to be as the most preferable option due to following reasons:

- Domain classes with its JSON serializers and deserializers can be reused within local environment
- Since this work is part of SAP Sailing Analytics development, the RESTful API can be extended accordingly to provide maximal support for this work
- SAP Sailing Analytics is single point of access and does not require knowledge about third-party services

The RESTful API of SAP Sailing Analytics can be used for retrievement of all demanded data except `PolarDataService`. Currently, there are no JSON serializers and deserializers available for `PolarDataService`. Since its internals are complex, another access option shall be elaborated.

SAP Sailing Analytics polar data client

To find a strategy for the import of `PolarDataService` with its internal regressions, SAP Sailing Analytics bootstrap code has been investigated in order to locate the snippet for automatic polar data import. It turned out, that SAP Sailing Analytics platform houses a `PolarDataClient` class which initializes a local `PolarDataService`-instance with the state of the remote SAP Sailing Analytics instance. Internally, it queries a binary string from a server-instance and passes it to `PolarDataService`-instance through appropriate interface method. Therefore, to import `PolarDataService` locally, appropriate class was implemented which extends `PolarDataClient` with persistence of the retrieved string containing polars. Whenever a `PolarDataService` needs to be constructed locally, a new `PolarDataService`-instance is instantiated and initialized with the persisted string.

Excluded options

The remaining two access options described in *3.6 Data access* have been put aside. The data mining module does not provide any data export options except for aggregated data mining results. As already concluded in *3.6.4 Third-party services,* a direct data fetching over third-party services has been considered as too error-prone and complex when comparing to SAP Sailing Analytics RESTful API.

4.4.2 Data persistence

The data import process from remote SAP Sailing Analytics server-instance takes approximately 10 hours. Therefore, the data needs to be persisted in appropriate format locally in order to allow faster loading for further processing. Additionally, the persistence layer will be used as middleware for advanced data science tools and technologies which are not compatible with Java Virtual Machine (**JVM**). There have been following alternatives considered for data persistence implementation.

MongoDB

MongoDB is a highly scalable document-oriented NoSQL DBMS which is already used within SAP Sailing Analytics platform. It maintains its data in JSON-format and thus, is highly compatible with SAP Sailing Analytics RESTful API. Moreover, thanks to existing JSON serializers and deserializers, no extra mapping of objects to persistent format is required. MongoDB features its own query language which heavily leans and JavaScript and JSON. The Java driver for MongoDB is easy to use and is already included in SAP Sailing Analytics code. Therefore, the data import component just needs to satisfy appropriate dependencies to SAP Sailing Analytics modules in order to reuse its existing building blocks for JSON serialization and MongoDB management.

SQL oriented DBMS

Another alternative is the usage of a relational SQL oriented DBMS, such as MySQL or PostgreSQL. SQL is a well-known standard query language among computer scientists and thus, in contrast to MongoDB query language, it does not require an extra learning curve. However, its usage requires implementation of mapping between Java objects and relational structures. Therefore, this alternative for persistence is considered as time-consuming and not adequate.

CSV

CSV is simple data format where each CSV file is representing an entity. A row within CSV file represents an instance which is composed of attributes separated by comma. Overall, it is a simple format and at first glance it can be easy handled without an extra library. However, when commas and quotes are occurring within a row, they must be escaped, which introduces additional complexity level and hence, demands a CSV library for flawless file handling. Furthermore, compared to other DBMS it does not provide any efficient query mechanisms. However, when performing data analysis with data science tooling, there are many out of the box components which support data loading from CSV files.

Decision making

The decision making for persistence format of data has been influenced by the following criteria:

- Low implementation effort.
- High combability with JVM and Python
- Aptness for big data processing regarding robustness and scalability.

For maneuver data, the decision was made in favor of MongoDB. The top-level collection will contain `RaceWithEstimationData`-instances.

For `PolarDataService` persistence, custom code has been written which heavily leans on `PolarDataClient`. It downloads serialized string with polar data and persists in its original format in a file. Whenever a new `PolarDataService`-instance is required, utility class `PolarDataServiceAccessUtil` will be used to execute the following three steps:

- Instantiate a new instance of `PolarDataService`
- Load the persisted string with serialized polar data
- Let the created `PolarDataService`-instance import that string

4.4.3 Data loading from MongoDB

In MongoDB, the data is queried as JSON string. And because the SAP Sailing Analytics RESTful API is also based on JSON, same JSON serializers and deserializers can be used for mapping between Java objects and JSON objects. Since MongoDB is also supported by many other languages, it can be also accessed directly from non-Java

environments, such as Python. Since JSON is supported by Python natively due to its typeless nature, the mapping of data formats is performed automatically when appropriate libraries are used, e.g. PyMongo.

The size of imported data looks as follows:

- `PolarDataService` configuration string: 1 MB
- Maneuvers: 17,3 GB

Since the available technical resources for this work are limited to a notebook featuring 16 GB RAM, the available maneuver data will be processed lazily. Otherwise, the JVM operation of JVM gets terrible inconvenient and slow. Therefore, instead of loading the whole data in memory, Iterator design pattern has been applied to implement following iterators:

- `PersistedRegattasWithEstimationDataIterator`: returns `RegattaWithEstimationData` for each call of `next()` and guarantees that only the returned `RegattaWithEstimationData`-instance is hold in-memory.
- `PersistedRacesWithEstimationDataIterator`: returns `RaceWithEstimationData` for each call of `next()` and guarantees that only the returned `RaceWithEstimationData`-instance is hold in-memory.
- `PersistedCompetitorTracksWithEstimationDataIterator`: returns `CompetitorTrackWithEstimationData` for each call of `next()` and guarantees that only one `RaceWithEstimationData`-instance wrapping the returned `CompetitorTrackWithEstimationData`-instance is hold in-memory.
- `PersistedManeuversWithEstimationDataIterator`: returns `ManeuverWithEstimationData` for each call of `next()` and guarantees that only one `RaceWithEstimationData`-instance wrapping the returned `ManeuverWithEstimationData`-instance is hold in-memory.

4.5 Feature engineering

Thanks to previous sections, the demanded data has been identified and loaded into local environment and persisted for further processing. The datasets contain all the data which is relevant for maneuver classification and true wind inference. The current task

is to engineer feature sets which represent the information of acquired data in most efficient manner in order to facilitate pattern abstraction by ML algorithms.

This section introduces features which can be used as part of the input vector for maneuver classification. Even if some of the features appear redundantly, all of them are considered as valid candidates for the final feature set. The goal of this section is to derive features from `ManeuverWithEstimationData`-interface and transform it so that it easily correlates with maneuver type. The features will be included in the new interface called `ManeuverForDataAnalysis` which is used during data exploration and feature importance evaluation. The final feature selection will be performed after data exploration.

4.5.1 Maneuver type

For maneuver classification, the maneuver type is considered as the target variable for maneuver classification. Since it is not included in `CompleteManeuver` and `CompleteManeuverWithEstimationData`, it will be determined using the following logic:

```
numberOfTacks := Count the times when boat’s bow has crossed the TWD
numberOfJibes := Count the times when boat’s bow has crossed the inverse TWD
maneuverType := “Unknown”
if (TWD is given)
    if (numberOfJibes + numberOfTacks ≥ 2)
           maneuverType = “Penalty circle”
    else if ( numberOfJibes > 0 )
           maneuverType = “Jibe”
    else if ( numberOfTacks > 0)
           maneuverType = “Tack”
    else
           differenceOfAbsoluteTwas := |twaBeforeManeuver| - |twaAfterManeuver|
           if ( differenceOfAbsoluteTwas < 0 )
                  maneuverType = “Head-up”
           else
                  maneuverType = “Bear-away”
```

Listing 5: Maneuver type feature derivation

The logic introduced above is similar to SAP Sailing Analytics logic which is applied in context of `Maneuver`-instances construction. As a reminder, in contrast to `CompleteManeuver`-instances, the `Maneuver`-instances include an additional splitting logic for maneuvers which is considered as confounder for wind estimation.

4.5.2 Absolute maneuver angle

The maneuver angle is the total course change in degrees performed within maneuver section. It is an important attribute of maneuver because it allows to draw conclusions about maneuver type by matching the angle with optimal maneuver angles of tacks and jibes derived from target polars. For classification, the absolute value of maneuver angle will be used.

4.5.3 Absolute main curve angle

The main curve angle is the total course change in degrees performed within the main curve section of maneuver. Its correlation with maneuver type will be checked due to same reasons as for absolute maneuver angle.

4.5.4 Oversteering

Oversteering is regarded as the course difference between the course after the turning section and the course after the maneuver section. It is expected that within tacks, the oversteering is higher than of other maneuvers.

4.5.5 Speed loss ratio

High speed loss can be a hint for slow maneuvers such as a tack. The speed loss can be derived from the ratio between lowest speed SOG_{low} within manoeuvre section and manoeuvre entering speed $SOG_{entering}$:

$$\frac{SOG_{low}}{SOG_{entering}}$$

It is expected that the yielded value gets SOG neutral. However, according to [9], it is also expected that this value is highly dependent on sailor skills and might also correlate with the oversteering feature. The value range of this feature is [0; 1] which is beneficial in terms of normalization of feature scaling.

4.5.6 Speed gain ratio within turning section

If the maximal speed recorded within a turning section of maneuver gets significantly higher than the maneuver entering speed, it could be an indicator for a bear-away. This

feature shall be managed as ratio between the maximal recorded speed SOG_{max} within turning section and the turning section entering speed $SOG_{entering}$:

$$\frac{SOG_{entering}}{SOG_{max}}$$

The smaller the ratio value gets, the higher the likelihood for maneuver being bear-away is expected. Similar to speed loss ratio, the value range of this feature gets also [0; 1].

4.5.7 Lowest speed vs. maneuver exiting speed ratio

This feature is another variant of speed loss measure. However, instead of comparison of lowest speed with maneuver entering speed, this feature relates the lowest speed SOG_{low} with manoeuvre exiting speed $SOG_{exiting}$:

$$\frac{SOG_{low}}{SOG_{exiting}}$$

It is expected that the value gets 1 if a head-up maneuver with a significant course change is performed. The assumption comes from the knowledge, that most boat classes perform slower when sailing upwind than downwind. Therefore, the lowest speed is expected to be recorded at maneuver finish since the boat starts continuously losing its speed after its upwind rearrangement.

4.5.8 Speed-in speed-out ratio

If the maneuver entering SOG and exiting SOG are nearly equal, it is highly likely that the maneuver is either tack or jibe because both maneuvers start and end at same absolute TWAs. Considering the symmetry property of polars, the absolute TWA on both tacks must be sailed with equal speed when the wind is stable. In terms of maneuver classification, the ratio between maneuver entering speed $SOG_{entering}$ and manoeuvre exiting speed $SOG_{exiting}$ will be used as additional feature:

$$\frac{SOG_{entering}}{SOG_{exiting}}$$

It is expected that when the ratio gets close to 1, the likelihood for manoeuvre being a tack or a jibe gets increased. If the value gets above 1, it might be a signal for a bear-away, below 1 for a head-up.

4.5.9 Scaled speed before and after

The stable speeds recorded at maneuver boundaries constitute important information which can be mapped to target polars of the corresponding boat class. However, the polars fitting requires knowledge about all available speeds sailed within a race which contradicts the idea of classification of individual maneuvers. The classifiers will be trained on dataset coming from all races conducted with various boat classes under different TWS conditions. It means that upwind courses with higher TWS can be sailed at same speeds as downwind courses with lower TWS. Therefore, there is no possibility for maneuver classifiers to derive a reasonable polars concept from absolute SOG values without any reference to a specific TWS range from a particular race. To address this issue, instead of absolute SOG values, relative SOG values before and after maneuver are introduced. The idea is to relate the SOGs before and after maneuver to the maximal SOG recorded within the same competitor track. The relation is done by dividing all SOG values by the 99% highest SOG record. The 99-percentile constraint is added to avoid that the maximum value is taken from an outlier SOG record. The new features will be referred as following:

- Scaled speed before (maneuver)
- Scaled speed after (maneuver)

4.5.10 Maximal turning rate

The turning rate is another promising feature candidate for maneuver classification. It is assumed that tack maneuvers have a higher turning rate than jibes because within tacks the boat's bow is steered through the dead-wind zone which in turn causes the sail to be untrimmed.

4.5.11 Deviation of maneuver angle from optimal angle of tack and jibe

The following method of `PolarDataService` allows derivation of optimal maneuver angles for tacks and jibes:

(boat class, boat SOG, point of sail) → (list of candidates (TWA, TWS))

The optimal tack maneuver angle α_{tack} is calculated as follows:

$$\alpha_{tack} = TWA * 2$$

Whereas the optimal jibe maneuver angle α_{jibe} is determined as follows:

$$\alpha_{jibe} = (180 - TWA) * 2$$

Since the polar service operates internally only with positive TWA values, the returned TWA of the aforementioned service method cannot get negative. The actual maneuver angle of maneuver is the absolute value of its total course change within maneuver section. Thus, considering the total course change c of maneuver, the deviations d_{tack} and d_{jibe} of actual maneuver angle from expected maneuver angle for tacks and jibes are calculated by:

$$d_{tack} = |c| - \alpha_{tack}$$

$$d_{jibe} = |c| - \alpha_{jibe}$$

According to definition of the aforementioned method of `PolarDataService`, multiple optimal TWAs can be returned for a point of sail. In that cases, the TWA with the closest derived optimal angle to actual maneuver angle will be selected.

4.5.12 Mark passing

As already preluded in *4.2.4 Maneuver data*, the mark passing feature is considered as important since it allows to draw conclusions about maneuver reasoning which in turn can be useful for discrimination between tack and jibe vs. head-up and bear-away. Since the mark passing maneuvers are performed primarily due to rearrangement of COG toward next waypoint mark, and not because of performance based alignment to the true wind, the physical properties of mark passing maneuvers such as maneuver angle might be not representative and thus, not match the usual maneuver angle of regular tacks and jibes derived from polars. This knowledge might be also learned by classification models. To make the feature compatible with classification models, the Boolean type has been transformed to Integer by mapping `true` to `1` and `false` to `0`.

4.5.13 Duration of maneuver, main turn and recovery phase

The mentioned durations of maneuver parts can provide additional correlations with maneuver type. However, considering the concept of maneuver detection, the durations get highly dependent to GPS sampling rate, wind stability and skipper's speed recovery strategy. Notwithstanding, its correlation will be examined more in detail as part of data exploration task.

4.5.14 Time loss

Maneuver loss measure is already used in SAP Sailing Analytics for maneuver performance measurement. It was introduced in section *3.4.7 Maneuver loss*. However, since it highly depends on speed loss, it might get redundant. Nevertheless, it will be checked whether this measure can provide an added value for maneuver classification.

The SAP Sailing Analytics maneuver loss measure is treated as a distance which allows to draw conclusion that it highly depends on boat's speed. To make the maneuver loss SOG neutral, the measure was converted in duration-based unit which will be referred as time loss. It is calculated by dividing maneuver loss distance by SOG at maneuver beginning.

4.6 Data cleansing

The aim of this section is to filter out noisy data which is affected by confounders described in *3.8 Confounders and mitigation*. Some of the elaborated filters are meant to be integrated into the input pipeline of wind classifiers. All of the filters will be applied when the dataset for model training are generated. In the following course of this thesis, all of the introduced filters will be used for different purposes and referred by its name which corresponds to the title of its dedicated subsections.

4.6.1 Start-line and finish-line

Most of the races contained within SAP Sailing Analytics include a start-line and a finish-line. It is assumed that after a sailboat has crossed a start-line, it is in race mode which means that it sails at optimal TWAs and SOG until it reaches the finish-line. In terms of wind estimation, it is regarded as the most valuable section for the wind estimation model. In contrast to racing section, the sections before start-line or after finish-line can be sailed arbitrarily. To understand the sailboats behavior within non-racing part of tracks, random races have been sampled and visualized using SAP Sailing Analytics RaceBoard UI. It turned out that in many cases the sailboats sail back and forth close to the start-line at similar speeds as during the race. In some other cases, the sailboats sail at much lower speeds than during the race, probably due to keeping the sail intentionally untrimmed in order to burn the time. An example is illustrated below.

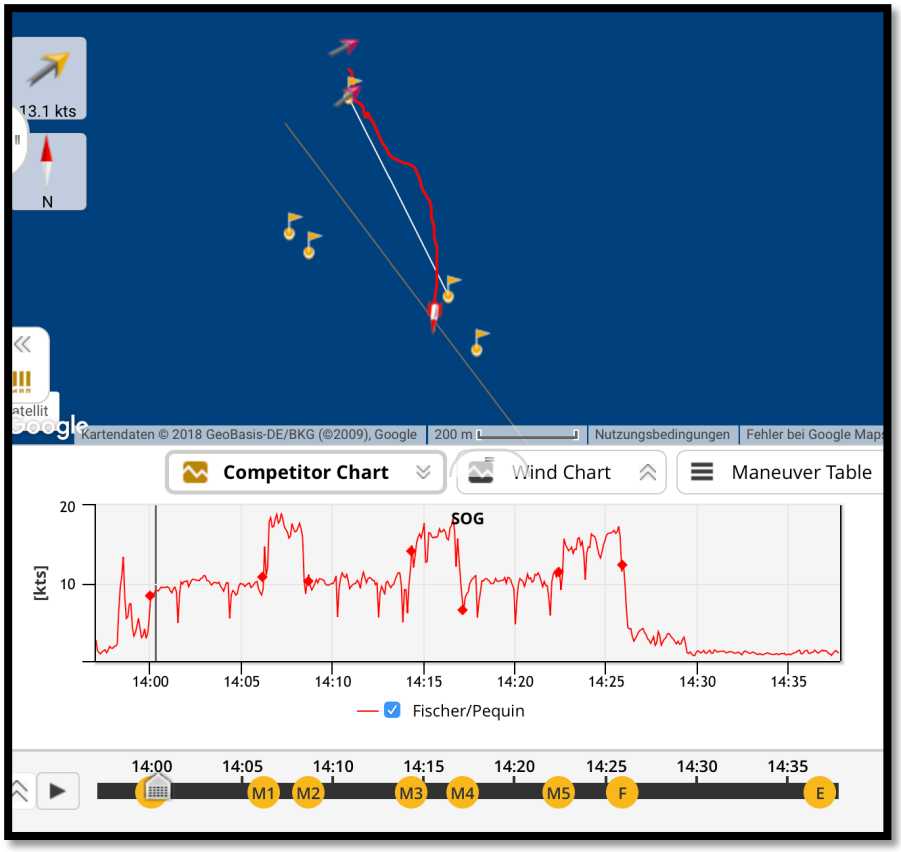

Figure 24: SOG comparison before and after start-line crossing[39]

According to SOG chart displayed below the race map, the sailboat is sailing nearly at same TWA, but with a significant SOG difference before and after the start-line crossing. This behavior is a confounder for wind estimation because it highly deteriorates the matching processes between actual polars and target polars. After the finish-line, nearly all race samples revealed a continuous speed drop of sailboats. Therefore, it is desirable to ignore the sections after finish-line crossing.

Considering these facts, following filters are introduced:

- A filter for `RaceWithEstimationData`-entities which imposes the races to be composed of at least two waypoints, one for start-line and one for finish-line.
- A time range filter for maneuver instances which requires the value of the time point attribute of each of the instances to be contained between the time points of start-line and finish-line crossing.

[39] Race: Kieler Woche 2018 – 49er Q2

4.6.2 Invalid competitor tracks

This filter targets all competitor tracks with implausible speeds and poor GPS sampling rates and shall be applied to all `CompetitorTrackWithEstimationData`-instances. In order to pass the filter, a `CompetitorTrackWithEstimationData`-instance must meet the following conditions:

- Average speed of track > 1 knot
- Average interval between raw fixes < 8 seconds
- Number of mark passings equals the number of waypoints contained within the race

The last filtering criterion is meant to ensure that the sailboat was indeed racing through the all defined waypoints and was not been sunk on its way.

4.6.3 Maneuvers within GPS holes

The quality of maneuver data is highly sensitive to GPS inaccuracy. Even with a tolerable GPS sampling rate there is still a chance that the tracking device either enters a GPS hole, or samples multiple consecutive GPS fixes which get identified as outliers. If these scenarios occur within maneuver section, most of the features of the affected maneuver such as lowest speed, total course change and even the locations of maneuver boundaries will get unreasonable and untypical for the given maneuver type. The result would be a deteriorated maneuver classification due to introduced noise within maneuver features.

The following filters are meant to address this issue:

- Longest interval between two fixes within maneuver section < 4 seconds
- Average interval between two GPS fixes from the end of previous maneuver until the beginning of current maneuver ≤ 8 seconds
- Average interval between two GPS fixes from the end of current maneuver until the beginning of next maneuver ≤ 8 seconds
- Interval between the first fix of maneuver section and its previous fix ≤ 4 seconds
- Interval between the last fix of maneuver section and its next fix ≤ 4 seconds

While the first filtering criterion preserves the quality of maneuver attributes derived from its speed and course trends, the remaining criteria target the accuracy of determined maneuver boundaries.

4.6.4 Close maneuvers

Depending on the sailing circumstances and approaching waypoints, multiple maneuvers can be performed successively. For instance, it has been already observed multiple times when two consecutive tack maneuvers were performed one by another within a short time frame in order correct wrongly estimated route. In terms of revised maneuver detection, those maneuvers can even collide with its boundaries. For maneuver classification, close maneuver boundaries introduce a confounder which affects the SOG at maneuver boundaries. If a maneuver is followed closely by another maneuver with similar maneuver angle, it is expected that SOG after of the preceding maneuver and SOG before the following maneuver will not represent sections with stable speed and course. The features of affected maneuvers such as speed loss and oversteering might get for the maneuver type untypical as well. Therefore, the following filtering criteria are introduced to filter out colliding maneuvers:

- Interval between the end of the previous maneuver and the beginning of the current maneuver > 4 seconds, OR the absolute total course change of the previous maneuver < 30% of the absolute total course change of the current maneuver
- Interval between the end of the current maneuver and the beginning of the next maneuver > 4 seconds, OR the absolute total course change of the next maneuver < 30% of the absolute total course change of the current maneuver

4.6.5 SOG before and after maneuver

When a sailboat starts moving after a halt, or the skipper decides to slow down the boat intentionally, a maneuver might get captured which does not represent usual SOG and COG at its boundaries. This introduces an additional confounder for maneuver classification because the SOG values at maneuver beginning and end do not represent stable segments at target SOG and target TWA. Since the maneuver features such as speed loss and gain highly depend on SOG before and after maneuver, it gets also skewed. To solve this issue, following filters have been elaborated:

- SOG at maneuver beginning > 2 knot and SOG at maneuver end > 2 knot
- The absolute difference between SOGs at maneuver beginning and maneuver end < triple of the lower SOG value from both SOG samples

4.6.6 Regular maneuvers

In order to improve the capabilities of classification to classify maneuvers by its type, only maneuvers will be considered which provide features with enough diversity for discrimination. It is expected that the features of maneuvers with small angle get either nearly equal or randomly shifted due to noise coming GPS-sensors or wind instability. The features of maneuvers with wide angle introduce no benefit, since it is not possible to use their middle course to predict the wind. It gets even worse if the type of such maneuvers is tack or jibe because the maneuver irregularities will influence the classification model and thus, produce an extra noise of irregularity which in turn might deteriorate the overall classification accuracy. Therefore, such unusual maneuvers are considered as confounder. The maneuvers which are expected to be useful for classification and wind estimation purposes will be filtered by the following criterion:

- $120 \geq$ absolute main curve angle in degrees ≥ 30

4.7 Data exploration

The aim of this section is to glance through the data to gather understanding about its distribution and characteristics. This section is the groundwork for *4.8 Feature selection*. All charts and evaluations were derived from dataset with cleaned maneuvers filtered by all criteria introduced in *4.6 Data cleansing*.

4.7.1 Data exploration strategy

To explore maneuver data, various statistical methods and data visualization techniques were employed. There are various tools available for data visualization and statistics computation. One of the tools is provided by SAP Sailing Analytics DataMining module which offers visualization of predefined statistics for a subset of SAP Sailing Analytics domain entities including `GPSFixMoving`, `Maneuver` and `PolarDataService`. It features generation of polar charts and histograms based either on already aggregated polars or on GPS-fixes. For `Maneuver`-entities, it provides bar charts which reflect various statistics such as min, max, mean, or median of an entity attribute, e.g. maneuver angle. However, at the time of writing, SAP Sailing Analytics DataMining does not support advanced visualization techniques such as scatter plot and histograms for data other than polars. Furthermore, each of the statistical feature which is not available in the set of predefined statistics needs to be coded manually into DataMining module which in turn requires understanding of its architecture, text

internationalization and redeployment of DataMining module on the productive server instance to access the added statistics. Therefore, SAP Sailing Analytics DataMining UI has been used to collect initial insights. More advanced exploration was leveraged by means of Python by processing the data in local environment.

4.7.2 Distribution of maneuver types

The following diagram depicts the distribution of maneuver types contained in regular maneuvers:

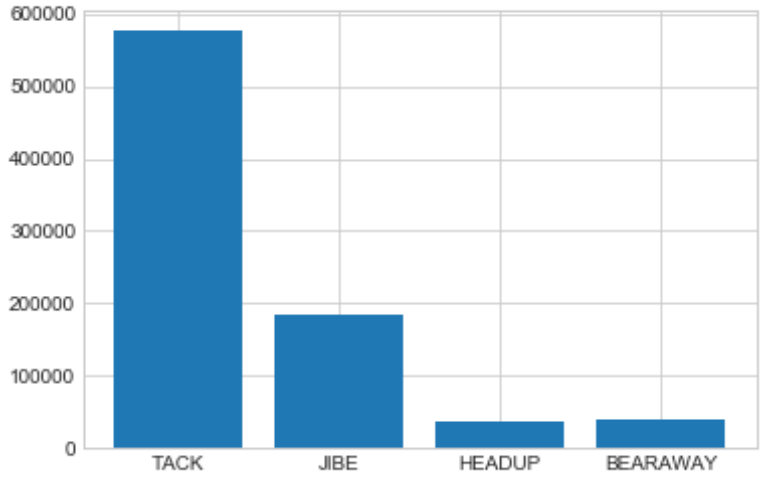

Figure 25: Maneuver type distribution in cleaned maneuvers dataset

The diagram points out two important facts:

- From 8,572,582 imported maneuver instances there are only 834,731 clean instances which have passed through data cleansing criteria. It is about 10%.
- The distribution of maneuver type variable is vastly shifted toward tack.

The last fact requires special attention, since the data distribution also affects the evaluation of maneuver type classifiers. To mitigate the influence of the skewed maneuver type distribution, appropriate metrics for scoring of classifiers must be chosen. For instance, if the classifier ignores the head-up and bear-away maneuver completely, it will still get an accuracy of 90% if the accuracy metric is as follows:

$$\frac{correct\ predictions}{all\ predictions}$$

The distribution for train and test datasets will not be modified because it is representative for the real sailing world with applied filtering criteria and hence, shall be considered by classifiers.

4.7.3 Correlation of maneuver features

Correlated features imply redundancy and might introduce the risk for falsely weighted feature importance during training of ML algorithms. The task of this section is to discover correlations between features to consider them within feature selection task. The following figure presents a heat map for feature correlations.

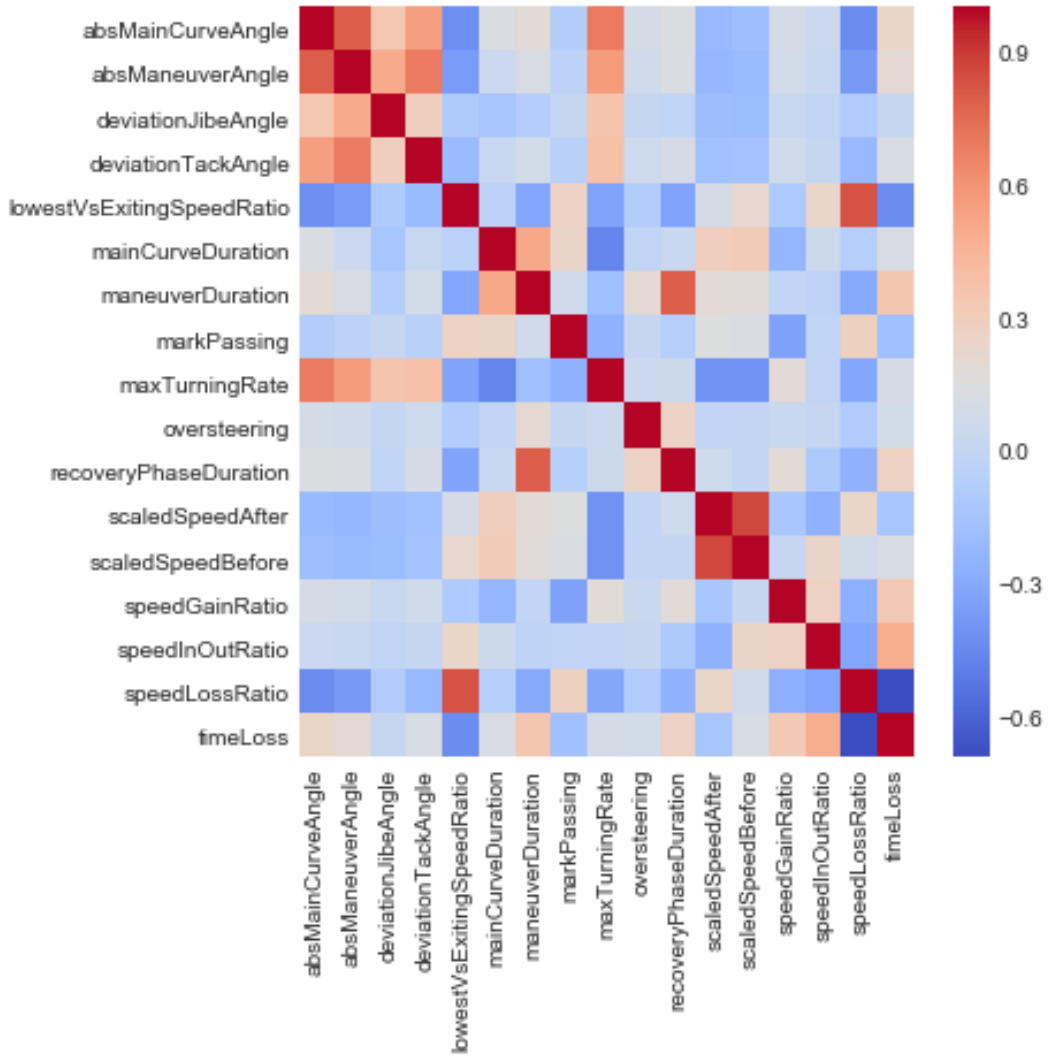

Figure 26: Correlation heatmap for maneuver features

Red color signals a strong positive correlation between two features, whereas blue color a strong negative correlation. The whiter the color, the less is the correlation between two features. In the following, the most important correlations are listed:

- Strongly positive correlated features:
  - *Speed loss ratio* and *lowest speed vs. exiting speed ratio*: The correlation is obvious since tack and jibe maneuvers contain usually equal

speeds at maneuver beginning and maneuver end. However, the main purpose of the feature is to identify head-up maneuvers. Therefore, further examination of this feature is required.

  - *maxTurningRate* and *absMainCurveAngle*: The higher the main curve angle, the higher the turning rate. The reason might be that the highest turning rate is recorded during tack maneuvers which in turn are performed usually with a higher maneuver angle than other maneuvers. One noticeable fact is that the main curve angle correlates stronger with the main curve angle than with maneuver angle.
  - *AbsMainCurveAngle* and *absManeuverAngle*: The correlation is self-explanatory because both angles are nearly equal. The main difference between maneuver angle and main curve angle is that in contrast to maneuver angle, the main curve angle includes the oversteering section. Considering the correlation of the main curve with max turning rate and with maneuver angle, a conclusion can be drawn that the main curve angle is redundant.

- Strongly negative correlated features:
  - *speedLoss* and *timeLoss*: The smaller the speed loss value gets, the more SOG is lost during maneuver performance. Consequently, the higher the difference between lowest speed and maneuver entering speed, the higher the time loss.

4.7.4 Feature importance test with ANOVA

The following figure presents the feature importance ranking of maneuver features with ANOVA F-test. The values have been scaled by the maximal occurring value.

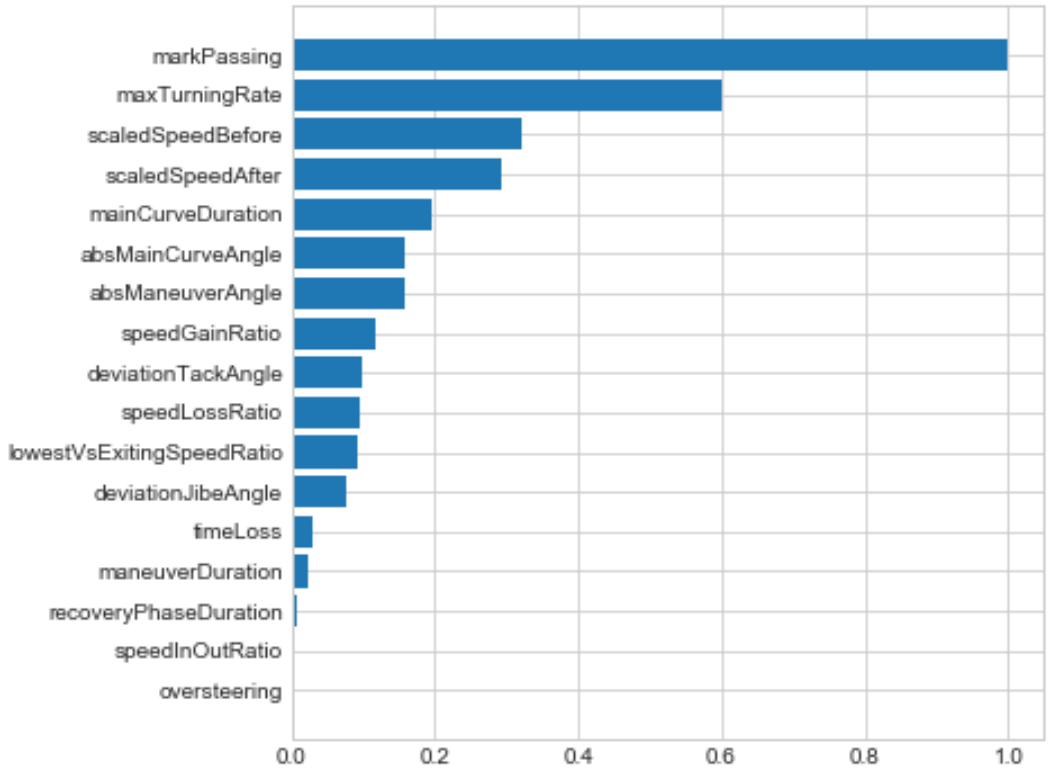

Figure 27: Importance ranking of maneuver features with ANOVA

According to the chart, the features with the lowest information gain are:

- Maneuver duration
- Recovery phase duration
- Speed-in speed-out ratio
- Oversteering

While the first two features might indeed introduce no correlation with the maneuver type due to its randomness affected by GPS sampling rate, boat class performance and skipper skills, the ranking of the last two features is disappointing since it vastly contradicts the ideas about the purpose of those features mentioned in *4.5 Feature engineering*. Therefore, further investigations need to be made to verify this result.

4.7.5 Feature importance test with Random Forest

The following figure represents the feature importance ranking derived from a trained scikit-learn-based Random Forest model composed of 50 trees:

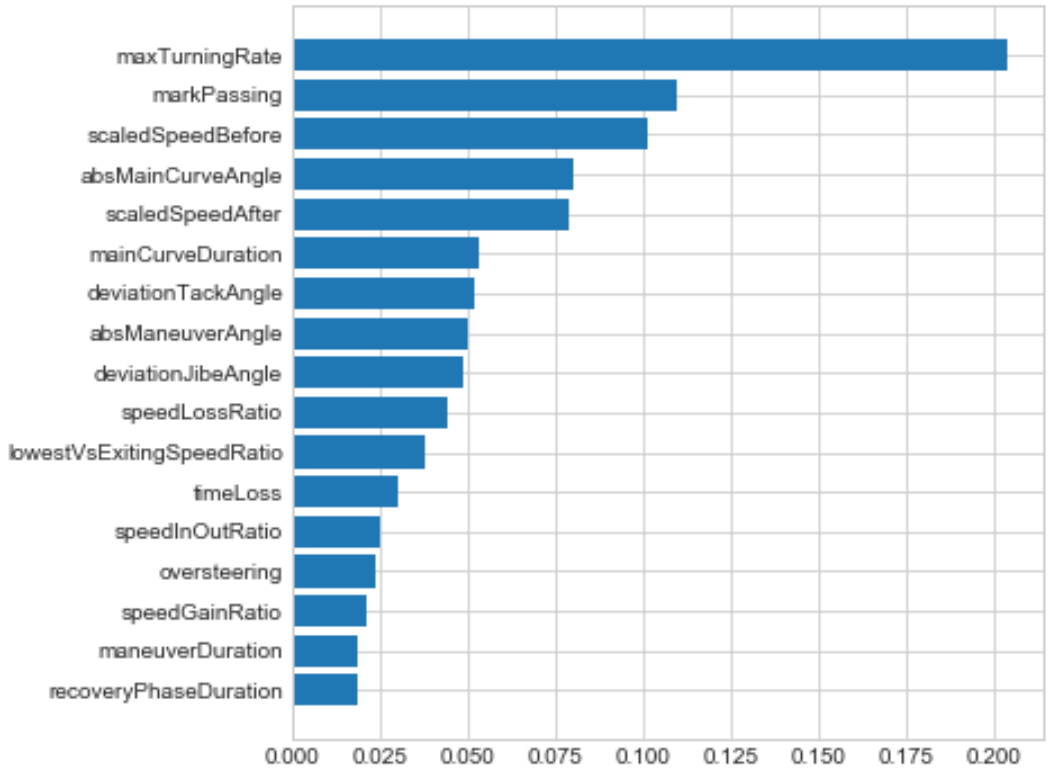

Figure 28: Importance ranking of maneuver features with Random Forest

While the importance of top features remains similar to ANOVA feature ranking, the set of lowest ranked features has changed. Main curve duration, speed gain and time loss have been ranked nearly on par with the features which were identified by ANOVA as irrelevant. While time loss is not considered as a serious feature candidate due to its strong correlation with speed loss, the speed gain might get irrelevant due to skewed distribution of maneuver types to the detriment of bear-away maneuvers. Therefore, the distribution of speed gain must be investigated.

4.7.6 Maneuver angle and main curve angle

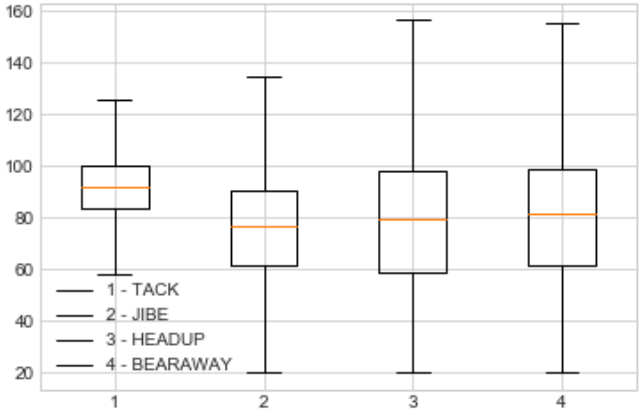

Figure 29: Box plot of absolute maneuver angles

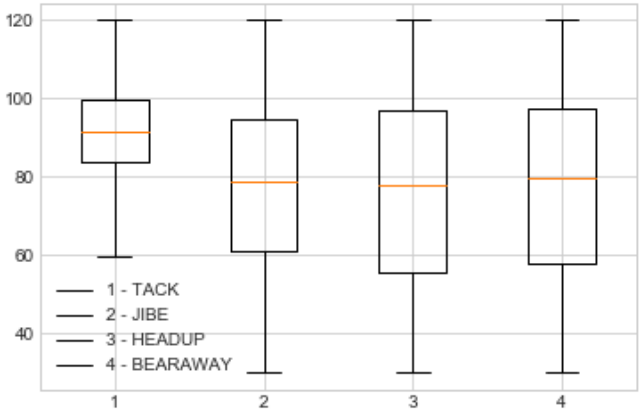

Figure 30: Box plot of main curve angles

In overall, both features introduce a strong correlation. However, in contrast to maneuver angles, the variance of main curve angles of jibe, bear-away and head-up maneuvers is considerably higher. A possible explanation might be given by the fact that there are various oversteering strategies applied which depend on experience level of individual skippers. From that, a conclusion can be drawn that maneuver types are characterized stronger by maneuver angles than by main curve angles.

4.7.7 Oversteering

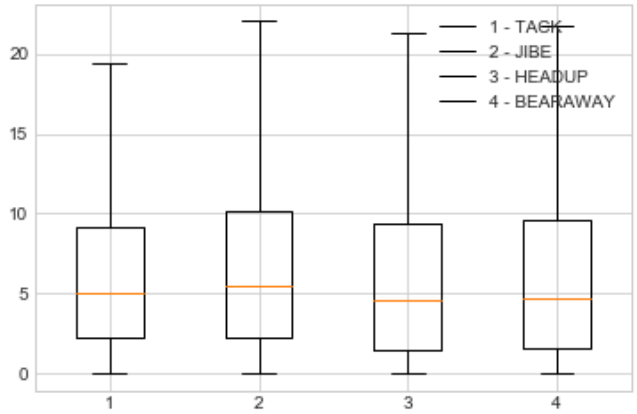

Figure 31: Box plot of oversteering

The mean and variance of oversteering looks nearly same for all maneuver types which proves that the feature is irrelevant, as proposed by feature importance ranking tests.

4.7.8 Speed loss and lowest speed vs. exiting speed ratio

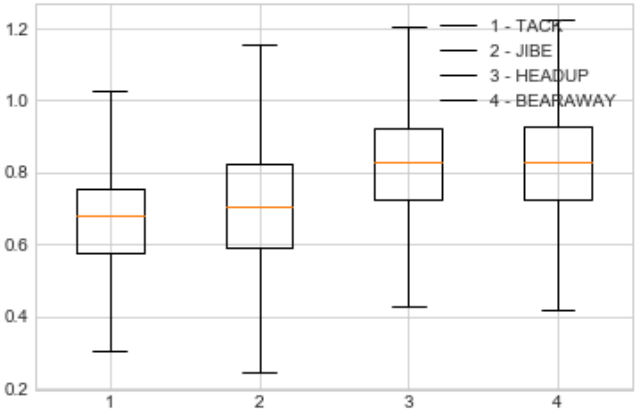

Figure 32: Box plot of speed loss ratio

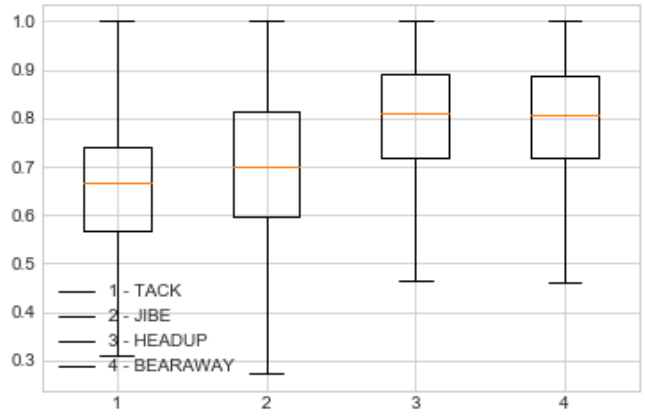

Figure 33: Box plot of lowest speed vs. exiting speed ratio

The speed loss and lowest speed vs. exiting speed ratio seem to have equal means and variances for all maneuver types. The assumption about the lowest SOG of head-up maneuvers being nearly equal to the SOG after maneuver turned out to be wrong. Therefore, there is no reason keep the lowest speed vs. exiting speed ratio feature in the input feature set of classifiers.

4.7.9 Speed gain ratio

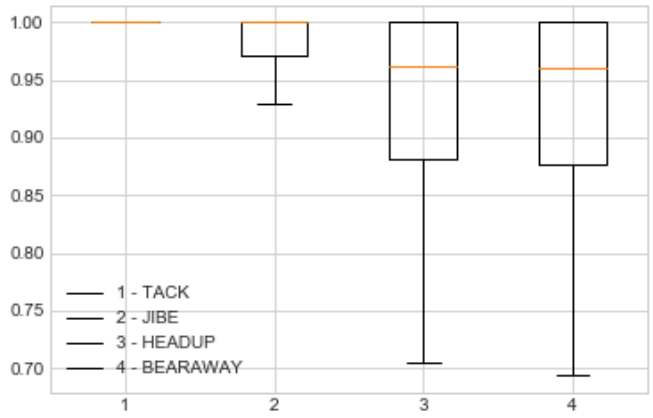

Figure 34: Box plot of speed gain ratio

The mean and variance of speed gain ratio significantly differs among maneuver types except for head-up and bear-way maneuvers. The equal mean and variance in head-up and bear-away contradicts the initial idea of the feature to characterize bear-away maneuver by consideration of its transition from upwind to downwind with a continuous speed increase during its main curve. Nevertheless, the distribution of values over all maneuver types significantly differs which allows to conclude that the feature is relevant.

4.7.10 Speed-in speed-out ratio

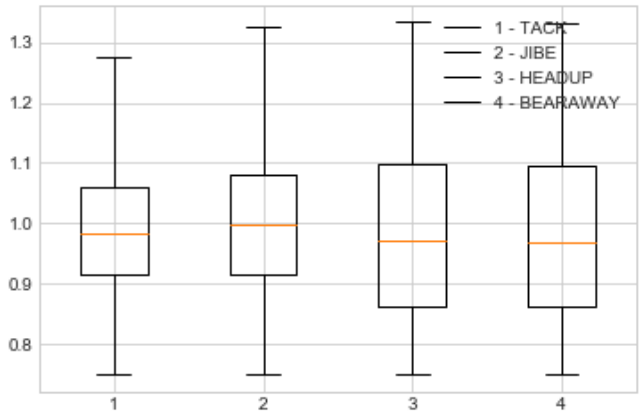

Figure 35: Box plot of Speed-in speed-out ratio

The ratio between speed-in and speed-out has been ranked by both feature ranking methods as irrelevant. According to its box plot, there are no significant differences between maneuver types. This highly contradicts the sailing theory, since it is assumed that tacks start and end at upwind, jibes at downwind, while head-up starts at upwind and end at downwind and bear-away vice versa. The contradiction is strange, but the sampling of individual races by means of SAP Sailing Analytics RaceBoard confirmed that the assumed theory is false in terms of given data. Therefore, since the feature does not introduce any information gain, it will be excluded from the final input set for classifiers.

4.7.11 Deviation of maneuver angle from optimal angle of tack and jibe

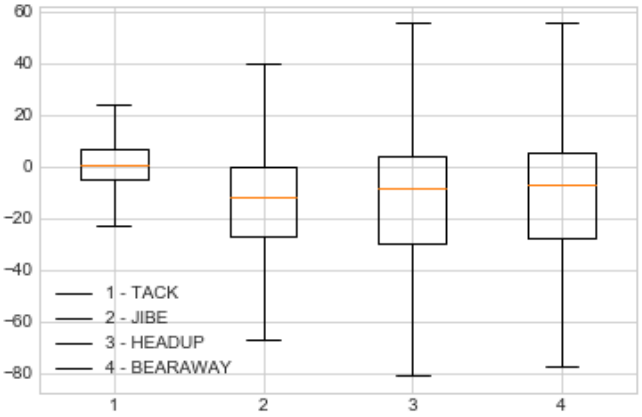

Figure 36: Box plot of deviation of maneuver angle from optimal tack angle

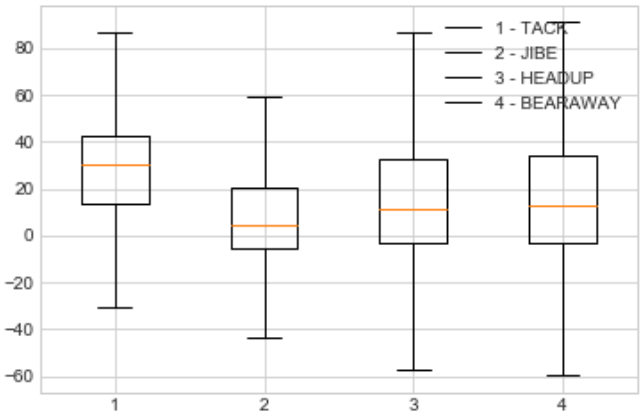

Figure 37: Box plot of deviation of maneuver angle from optimal jibe angle

The deviation of tacks from optimal tack angle is vastly smaller than of other maneuvers which underpins the importance of this feature. When comparing it with the deviation of jibes from optimal jibe angle, the conclusion can be drawn that maneuver angle of tacks is more characteristic for its maneuver type than of jibes because the variance of tack angle deviations is considerably smaller than of jibe angle deviations. This might be an important insight for the derivation of TWD considering the maneuver middle course.

4.7.12 Maximal turning rate

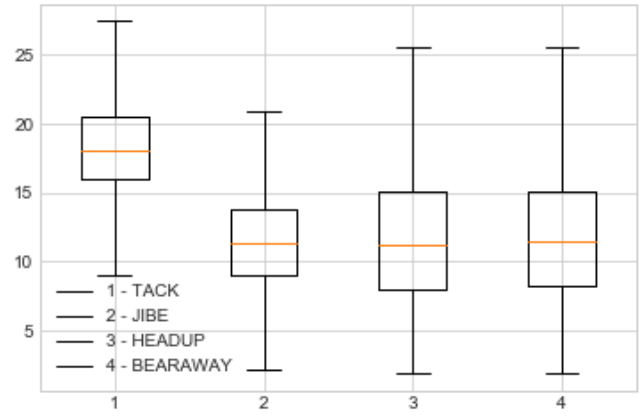

Figure 38: Box plot of maximal turning rate

The turning rate of tack maneuvers considerably differs from other maneuver types which confirms its high importance rank.

4.7.13 Mark passing

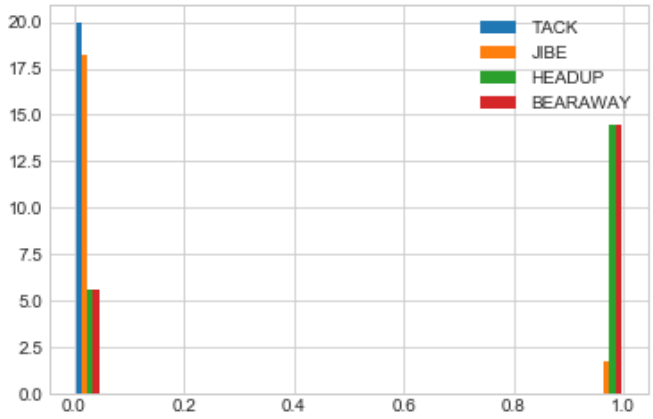

Figure 39: Histogram of mark passing feature

The histogram reveals that the most head-up and bear-away maneuvers with absolute main curve angle ≤ 120° and ≥ 30° are performed as mark passing maneuvers. Therefore, it can be seen as important discrimination criterion between tacks and jibes vs. head-up and bear-away maneuvers.

4.7.14 Remaining features

The detailed presentation about investigations results of the features time loss, maneuver duration, main turn duration and recovery phase duration are omitted because they do not provide any special insights. While the duration-based features do not provide any useful maneuver type characteristics, the time loss feature shows similar characteristic properties as speed loss, but its distribution is affected by more noise which makes the feature being obsolete.

4.8 Feature selection

In previous section many insights about the importance of each feature candidates were collected. In this section, these insights are considered to select the final features for maneuver classifiers. Additionally, the value set of maneuver type output variable will be reconsidered.

4.8.1 Included features for maneuver classification

The following list presents the features which have been included in the input feature set for maneuver classification:

- Absolute maneuver angle
- Speed loss
- Speed gain
- Maximal turning rate
- Scaled speed before
- Scaled speed after
- Mark passing
- Deviation of maneuver angle from optimal tack angle
- Deviation of maneuver angle from optimal jibe angle

4.8.2 Excluded features

The following list presents the features which have been excluded from the input feature set for maneuver classification:

- Absolute main curve angle
- Oversteering
- Lowest vs. exiting speed ratio
- Speed-in speed-out ratio
- Maneuver section duration
- Main curve duration
- Recovery phase duration
- Time loss

The reasoning for exclusion is stated in *4.7 Data exploration*

4.8.3 Feature categories

Some of selected features depend on data which might be unavailable for some races. For instance, if the mark information is not known, the mark passing maneuvers will not be known either. When a sailboat of a new boat class is tracked, its polars are unknown. Thus, the minimal requirement for the wind estimation component is the availability of GPS data which allows derivation of the following basic features:

- Absolute maneuver angle
- Speed loss ratio
- Speed gain ratio
- Maximal turning rate

The scaled speed before and after maneuver can be only determined, if there was at least one downwind course sailed. It is required in order to determine the real speed maximum to be used for scaling of all speed values. In case if only speeds at upwind courses are given, the scaled speed feature will introduce a confounder, as all the upwind speeds will be scaled to much higher values than expected by classification models. Therefore, the scaled speed before and after must be only used, if at least one downwind course was sailed at stable speed. For the already tracked races of SAP Sailing Analytics platform, this is nearly always the case. However, when the application of wind estimation is demanded during a live race, the presence of downwind course must be verified. If the speed at downwind course is not known, the scaled speed features must be excluded from the input feature list.

When the information about marks with its positions is unavailable, the mark passing feature must be excluded, otherwise it will be always zero which is a confounder for classifiers.

When the polars of a sailboat are known, the following features must be excluded:

- Deviation of maneuver angle from optimal tack angle
- Deviation of maneuver angle from optimal jibe angle

Overall, there were four feature categories identified:

- Basic features
- Scaled speed features
- Mark features
- Polar features

While the first category contains mandatory features for wind estimation, the remainder is optional and must not be required.

4.8.4 Head-up and bear-away merge

Besides feature importance, data exploration revealed another important insight about head-up and bear-away maneuvers. In all evaluated feature plots, there was no significant difference between means and variances of head-up and bear-away maneuvers

recorded. This allows to conclude that the two maneuver types cannot be discriminated considering the available feature set. Therefore, the decision has been made to merge the two maneuver types and represent it as a single discrete value for maneuver classification. The three possible discrete values of maneuver type are as follows:

- Tack
- Jibe
- Other

4.9 Further pre-processing

This section summarizes further pre-processing steps applied to the loaded maneuver data. The steps include splitting strategy for train and test datasets, as well as feature scaling and PCA.

4.9.1 Split of train and test datasets

„*The only way to know how well a model will generalize to new cases is to actually try it out on new cases*" [10, p. 29]. To follow best practices of ML, the maneuver data will be split in train and test datasets in order to evaluate its performance on new data. The splitting criterion will be the year of the sailing event which is contained within regatta name. If a maneuver instance is related to a regatta which contains the string "*2018*" in its name, it will be assigned to the test dataset, otherwise to the train dataset. The train dataset will be used for cross-validation-based evaluation and hyperparameter tuning of each of the classification algorithm, while the intention of the test dataset is to determine the final scoring performance of classification algorithms on future data by simulating a new sailing year.

4.9.2 Feature scaling

Some of the classification algorithms are highly affected by different feature scales, while others are not. To provide each of the algorithm its preferred data pre-processing variant, feature scaling has been added to the data pre-processing pipeline which can be activated on demand. The decision for the scaling method was made in favor of standardization due to its robustness against outliers and simple employment provided by evaluated ML libraries.

4.9.3 PCA

Considering the small number of selected maneuver features and the huge amount of available training data, the curse of dimensionality is not expected to be an issue for this work. Nevertheless, it shall be tested whether it can provide an added value to ML algorithm candidates during its evaluation. For this, PCA with 95% variance preservation has been added as an optional feature to the data pre-processing pipeline which can be turned on and off similarly to feature scaling.

5 Modelling

The data for supervised learning is ready. Next step in ML is to devise an appropriate model which learns the mapping between input and target output. All of the introduced models within this chapter will be evaluated in chapter *6 Evaluation*. Within this work, four different wind estimation models have been designed and evaluated. Three of the models are ensemble models built on top of probabilistic maneuver classifiers. Thus, maneuver classifiers are introduced at first. Afterwards, the design of each elaborated wind estimation model is presented.

5.1 Probabilistic maneuver classification

The purpose of probabilistic maneuver classification is to estimate posteriori probabilities for each possible type of a maneuver instance considering its maneuver features elaborated in the previous chapter. The task for this section is to select suitable classification algorithms and elaborate its optimal setup for Java environment with SMILE-library.

5.1.1 Eligible algorithms

Following list of eligible algorithms has been elaborated:

- Support Vector Machines (SVM)
- Naive Bayes
- Logistic Regression
- Linear Discriminant Analysis (LDA)
- Quadratic Discriminant Analysis (QDA)
- Random Forest
- Gradient Boosting
- Neural Network
- K-Nearest Neighbor

All of the listed algorithms support probability output for each possible value of its output variable. Furthermore, the algorithms are available both in SMILE and in scikit-learn which facilitates performance evaluation and provides compatibility with SAP Sailing Analytics backend. These were the main requirements during algorithm selection. The training and scoring interfaces of the algorithms are same. Therefore, it is not

necessary to know the internal concepts of each of the algorithm in order to make use of it.

5.1.2 Handling optional features

As already stated in *4.8.3 Feature categories*, the availability of some feature categories is not guaranteed. With the exception of basic features derived from GPS track, all other features are considered as optional features which can provide additional information for classification models to improve its classification performance. Since all of the introduced classification algorithm require the input features to be a vector of a fixed length, appropriate concept is demanded to handle variable feature sets.

In total, there are four feature set categories, including one mandatory and three optional. Thus, there are $2^3 \rightarrow 8$ possible variants of feature sets. To provide support for all these variants, the decision has been made to train for an own classification model for each possible feature set. The appropriate model is determined for each maneuver instance individually considering its available information.

5.1.3 Boat class specialization and generalization

The boat class provides important information about performance characteristics of sailboats. Some of the maneuver features including deviation of maneuver angle to optimal angle of tacks or jibes refer to polars of corresponding boat class, while other features such as speed loss or turning rate do not refer to any boat class information, although it could be beneficial since its values might introduce individual patterns for each boat class. To capture these individual patterns, considerations were made to train classification models for each boat class individually. However, since the datasets for each boat class will contain much less training instances, its performance could get worse. Therefore, the decision has been made to train all classifiers with all feature variations for all boat classes individually, as well as with all boat classes together. For classification of a maneuver instance, the classification model with the highest achieved test score shall be selected dynamically.

5.1.4 Training and persistence

As mentioned in previous section, classification models get trained for each possible maneuver feature set, with maneuvers for each boat class individually as well as with maneuvers of all boat classes together. The states of each of the models will be persisted

on filesystem employing Java serialization[40] which is supported by SMILE-library for all implementations of selected algorithms. With each trained classification model, the following additional information will be persisted:

- Maneuver features on which the model was trained on. It is the minimal feature set which must be provided to the model for proper classification
- The boat class which the model supports. `null` means that the model was trained with all boat classes
- F1- score on test data

5.1.5 Classifier model management

As previous sections preluded, there will be hundreds of classification models trained and managed for wind estimation task. To provide an easy and flawless relation of the most suitable classification model to a maneuver instance, the `ManeuverClassifiersCache` component is introduced. Its architecture looks as follows:

[40] By means of serialization objects can be encoded as a sequence of bytes. By means of deserialization, objects can be reconstructed from its serialized byte sequence.

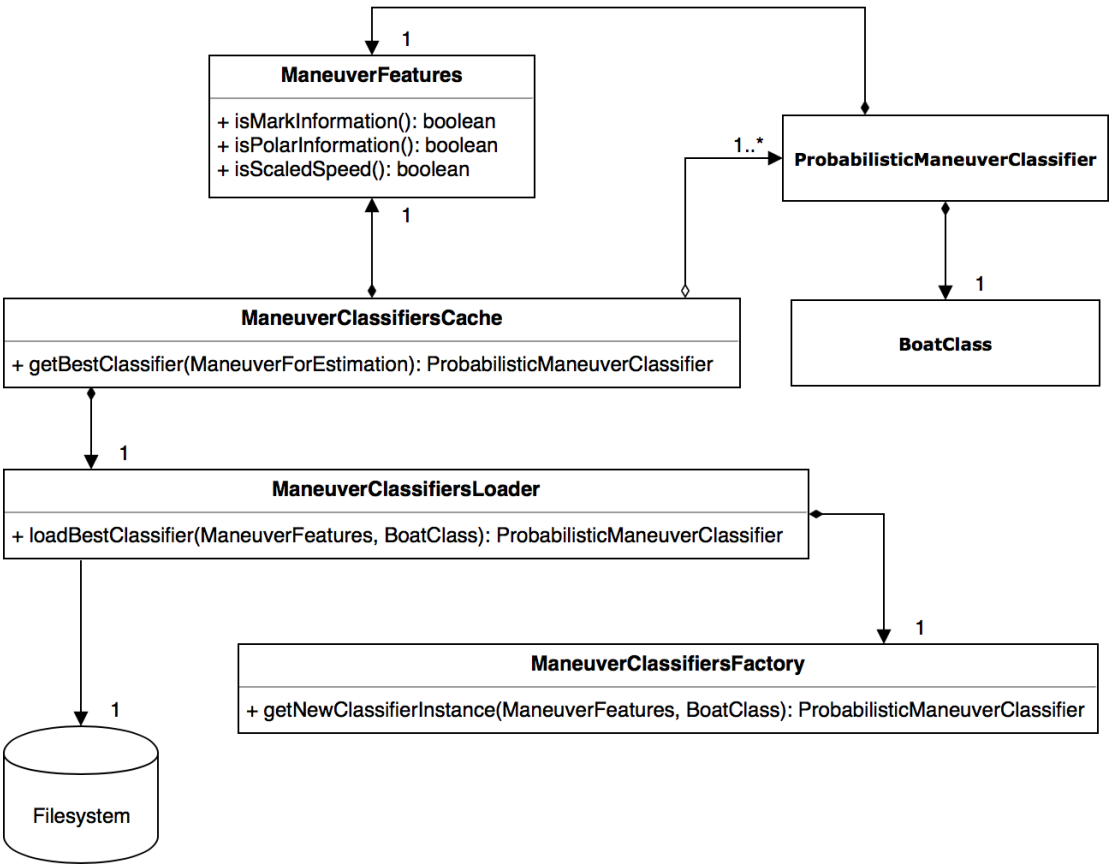

Figure 40: Maneuver cache UML class diagram

The purpose of `ManeuverClassifiersCache` is to retrieve the most suitable classification model for a given maneuver instance. Internally, it maintains a cache with already loaded classification models of type `ProbabilisticManeuverClassifier`, relating each loaded model to a combination of `ManeuverFeatures`-instance and `BoatClass`-instance which are derived from the provided `ManeuverForEstimation`-instance within `getBestClassifier()`. Besides the derived maneuver features from a maneuver instance, `ManeuverClassifiersCache` itself contains its own `ManeuverFeatures`-instance as well. It is used + for programmatical deactivation of particular feature categories. It takes precedence over availability of feature categories in maneuver data. For instance, it can be used to deactivate the scaled speed features due to unavailable downwind segments within GPS-track. Furthermore, the feature deactivation allows evaluation of classifier performances on all possible features sets.

Whenever a new combination of maneuver features and boat class gets demanded which is not contained within the internal cache of `ManeuverClassifiersCache`,

ManeuverClassifiersLoader will be instructed to determine and load the most suitable classifier which implements the following logic:

```
clfsToLoad := empty list
Iterate over all possible maneuvre feature subsets from given manoeuvre features
    clf1 := ManeuverClassifiersFactory.getNewClassifierInstance(
                current manoeuvre features, null)
    clf2 := ManeuverClassifiersFactory.getNewClassifierInstance(
                current manoeuvre features, given boat class)
    add clf1 and clf2 to clfsToLoad
Load all persisted classification models of instances contained in clfsToLoad
Determine and return the model with the highest test score
```

Listing 6: Pseudocode for determination of the most suitable classifier model for given maneuver features and boat class

As provided by pseudocode, ManeuverClassifiersFactory is used to instantiate the instance of the classifier without loading its persistent state. The purpose of ManeuverClassifiersFactory is to centralize the definition of ProbabilisticManeuverClassifier implementations which is meant to be used for particular maneuver features and boat class. For instance, it can be programmatically specified within ManeuverClassifiersFactory that for all maneuver features and all boat classes, neural network-based classifier implementation must be used.

When a new classification model gets loaded, it will be put in the internal cache of ManeuverClassifiersCache. For further calls of getBestClassifier() with same parameter values, the already loaded model will be retrieved from the cache directly.

5.2 HMM-based wind estimator

The most accurate wind estimation model has been achieved by means of a customized HMM built on top of classification results for each maneuver instance. The customization employs concepts of graph theory in order to determine the most probable TWD track considering the COG values contained within maneuvers. It can be regarded as a regularization component which is meant to eliminate the impact of wrong classifications by introducing penalties for abrupt TWD changes. In comparison to other implemented wind estimator, the merit of HMM-based estimator is its capability for adaption to continuous and smooth wind changes.

5.2.1 Context mapping

The contextual mapping between HMM components and wind estimation is the following:

- Subjects to observe: maneuver instances of all competitors within a single race
- Observation $o_1..o_n$: features of a maneuver instance
- Hidden states $s_1..s_n$: sequence of manoeuvre types.
- Set of hidden state values $S_1..S_m$: Tack, Jibe, Bear-away, Head-up
- Initial probabilities: $P(S) = \frac{1}{m}$ for all S
- Transition probabilities: function which maps $s_t = S_x$ in TWD range inferred from S_x and o_t, and $s_{t-1} = S_y$ in TWD range inferred from S_y and o_{t-1}. If both TWD ranges are intersecting, the transition probability gets high. In other case, the transition probability gets scaled down considering the smallest difference between the two TWD ranges.
- Emission probabilities: classification probabilities for each maneuver type

5.2.2 Consequences of customization

Overall, there are two customizations introduced in HMM for wind estimation:

- Transition probabilities are calculated dynamically depending on observations o_{t-1} and o_t. Thus, the HMM constraint $P(s_t = S_x \,|s_{t-1} = S_y)$ is readapted to $P(s_t = S_x \,|s_{t-1} = S_y, o_t, o_{t-1})$
- Emission probabilities are derived from probabilities output of maneuver classifiers

Especially the former customization point is considered as severe. Consequently, it is unsupported by HMM implementations of all known ML libraries for Java. Therefore, the whole HMM with its customization has been implemented in Java by its own, including representation of states and observations, as well as compatible variants of Viterbi and Forward-Backward algorithms which are used for wind track inference.

5.2.3 Mapping between classifications and emission probabilities

As decided in *4.8.4 Head-up and bear-away merge*, the probability of head-up and bear-away maneuver types is represented by a single probability in classification

results. To provide compatibility to the defined HMM, the classification probabilities shall be mapped to emission probabilities as follows:

```
classificationProbabilities := get from classifier for a maneuver
emissionProbabilitiesForManeuver := empty map
emissionProbabilitiesForManeuver[Tack] = classificationProbabilities[Tack]
emissionProbabilitiesForManeuver[Jibe] = classificationProbabilities[Jibe]
emissionProbabilitiesForManeuver[BearAway] = classificationProbabilities[Other]
emissionProbabilitiesForManeuver[HeadUp] = classificationProbabilities[Other]
Normalize emissionProbabilities by dividing each its value by sum of all values
```

Listing 7: Mapping between classification probabilities and emission probabilities of HMM

The last expression within the listing normalizes the emission probabilities so that its values sum up to 1 as required by HMM.

5.2.4 TWD range inference from maneuver types

Each maneuver includes three important features from which the range of possible TWDs can be restricted:

- COG before maneuver
- COG after maneuver
- Maneuver type

Although the real maneuver type of a maneuver is unknown, the algorithms of HMM is iterating over all possible maneuver type values in order to infer the most probable one. Therefore, the task for TWD range inference is to deduct a range of possible TWDs for a maneuver considering a particular maneuver type assumed by HMM.

Depending on maneuver type, the inference strategy of valid ranges for TWD is different. For tack and jibe maneuvers, the range of valid TWDs is determined as follows:

$$TWD\ range = maneuver\ middle\ course \pm buffer$$

For tacks, the $buffer$ will be the absolute deviation of maneuver angle from its optimal tack angle. If polar information is unavailable, then it gets 10 % of maneuver angle. The reasoning is that tack maneuvers mostly start and end at same absolute TWA if it gets perfectly performed. The lesser the absolute deviation from optimal angle gets, the more perfect the maneuver is assumed to be, and the more definite the TWD inference gets which yields in narrower ranges of possible TWDs.

Analogously, for jibes, the $buffer$ will be the absolute deviation of manoeuvre angle from its optimal jibe angle. If polar information is unavailable, then it gets 20 % of maneuver angle. The reasoning is that the TWA variance of jibes at its middle courses

is significantly higher than of tacks. Therefore, a conclusion can be drawn that the middle course of a tack deviates less from TWD than of jibes. Considering this fact, the decision has been made to make TWD ranges of jibes being wider and thus, less definite than of tacks. And finally, since the middle course of jibes is shifted from TWD by 180°, the so far calculated TWD range must be inverted by 180°.

The inference of valid TWD ranges for head-up and bear-away maneuvers is more sophisticated. For bear-away maneuvers, the valid TWD range is derived as follows:

- Starts from: the inverted course after maneuver
- Ends at: course before maneuver shifted by `MIN_ABSOLUTE_UPWIND_TWA` toward the inverted course after maneuver

`MIN_ABS_UPWIND_TWA` is a constant representing the minimal absolute TWA which can be sailed upwind. It is considered to restrict TWD range by the knowledge about the dead-wind zone. For downwind, there was no analogous concept defined because it is possible to sail straight way from the wind, even if it is considered as dangerous and inefficient. The derivation logic is exemplified by the following figure:

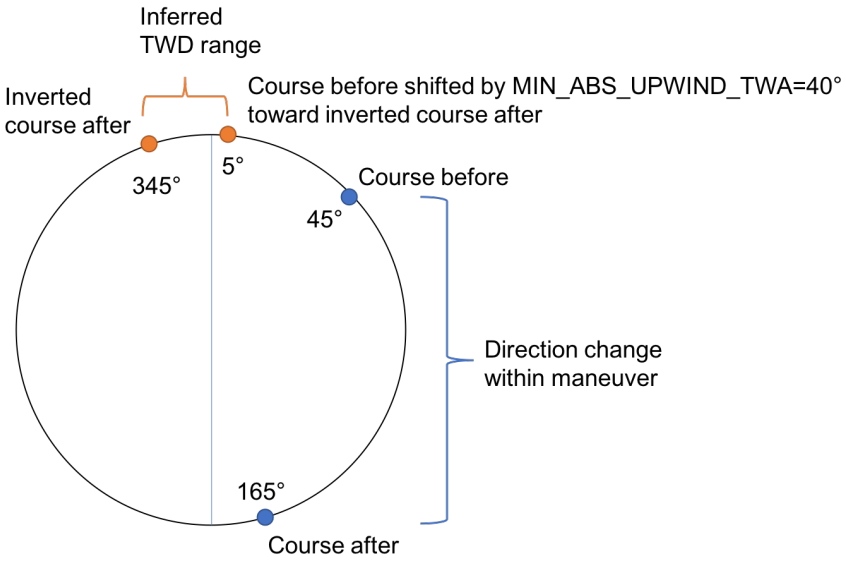

Figure 41: TWD range inference for bear-away maneuvers

For head-up maneuvers, the TWD range is inferred similarly:

- Starts from: the inverted course **before** maneuver
- Ends at: course **after** maneuver shifted by `MIN_ABSOLUTE_UPWIND_TWA` toward the inverted course **before** maneuver

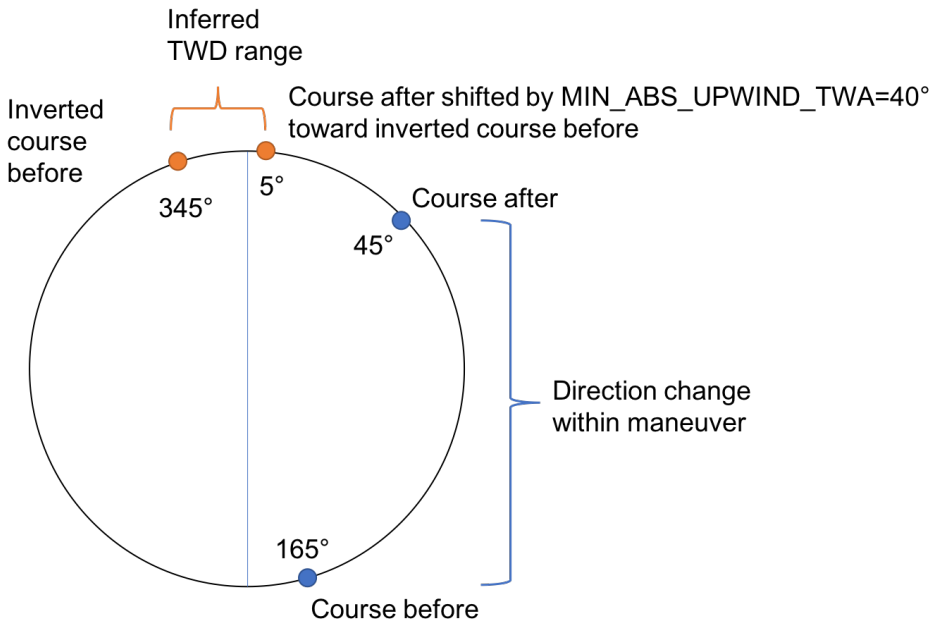

Figure 42: TWD range inference for head-up

Considering the aforementioned inference strategies, a conclusion can be drawn that the higher the maneuver angle of a head-up/bear-away maneuver is, the narrower and more definite its valid TWD gets.

5.2.5 Transition probability calculation

In conventional HMM the transition probabilities can be managed in a single $n \times n$ matrix with n representing the number of possible hidden state values. It is valid because in conventional the HMM transition probabilities are independent of observations. In the customized HMM version of this work this is no longer the case because the derivation of TWD depends on observed features of a maneuver. Therefore, a transition probability matrix containing $P(s_t = S_x \,|s_{t-1} = S_y, o_t, o_{t-1})$ must be calculated for each observation time point t individually.

The calculation strategy for transition probabilities is based on punishment of deviations between inferred TWD ranges from transitioning states. The implementation for transition probability matrix calculation is provided by the following pseudocode:

```
transitionProbabilitiesMatrix := empty map
for each possible S_x:
  twdRangeAfter := infer TWD range for s_t = S_x considering o_t
  for each possible S_y:
    twdRangeBefore := infer TWD range for s_{t-1} = S_y considering o_{t-1}
    if twdRangeAfter intersects with twdRangeAfter:
      transitionProbabilitiesMatrix[S_x, S_y] = 1
    else:
      diff := calculate absolute difference in degrees of between
              closest boundaries of twdRangeBefore and twdRangeAfter
      if diff ≤ MAX_TOLERATED_DIFF:
        transitionProbabilitiesMatrix[S_x, S_y] = 1 / (1 + (diff / MAX_TOLERATED_DIFF)^2)
      else:
        transitionProbabilitiesMatrix[S_x, S_y] = 1 / (1 + diff^2)
    Normalize transitionProbabilitiesMatrix[S_x,*] so that it sums up to 1
```

Listing 8: Calculation of transition probability matrix for HMM

The constant `MAX_TOLERATED_DIFF` is considered as a hyperparameter which can be tuned during model evaluation. Its purpose is to define the maximal tolerated deviation between TWD transitions in degrees. The deviations within tolerance will be penalized significantly less than deviations outside of tolerance. The reason is to tolerate small TWD changes and exclude abrupt and rapid ones which are considered as unusual. It is expected that the transition penalties will act as a regularization mechanism to produce wind fix sequences with soft TWD transitions.

5.2.6 Wind track inference

The inference of wind track can be divided in three main tasks:

- Determination of the most probable sequence of maneuver types with Viterbi algorithm
- TWD and TWS mapping from maneuvers with determined maneuver types
- Calculation of confidence with Forward-Backward algorithm

Most probable maneuver type sequence

Firstly, the most probable sequence of maneuver types is determined by means of Viterbi algorithm. For this work, an own implementation of Viterbi has been developed to make it compatible with the customized HMM. The main difference between conventional and own implemented Viterbi algorithm is the derivation strategy of transition probabilities. Furthermore, the Viterbi algorithm has been extended to compute forward probabilities for each possible hidden state value within its iteration loops in

order to reuse the iteration overhead for partial calculation of confidence values for each produced wind fix. The yielded algorithm is defined by the following pseudocode:

```
n := number of observations
bestPrev := empty list
forwardProbs := empty list
viterbiProbs := empty list

bestPrev[1] = empty map
viterbiProbs[1] = empty map
for each possible maneuver type value S_x:
  bestPrev[1][ S_x] = S_x
  viterbiProbs[1][ S_x] = P(s_1 = S_x) * P(o_1 | s_1 = S_x)
forwardProbs[t] = viterbiProbs[t]

for each t starting from 2 until n:
  bestPrev[t] = empty map
  viterbiProbs[t] = empty map
  forwardProbs[t] = empty map
  calculate transition probability matrix P(s_t = S_x |s_t-1 = S_y, o_t, o_t-1) for all S_x × S_y
  for each possible maneuver type value S_x:
    bestViterbiProb := -1
    bestViterbiS_y := null
    forwardProbs[t][S_x] = 0
    for each possible maneuver type value S_y:
      transitionEmissionProb := P(s_t = S_x |s_t-1 = S_y, o_t, o_t-1) * P(o_t | s_t = S_x)
      viterbiProb := viterbiProbs[t-1][ S_y] * transitionEmissionProb
      if bestViterbiProb < viterbiProb:
        bestViterbiProb = viterbiProb
        bestViterbiS_y = S_y
      forwardProb := forwardProbs[t-1][ S_y] * transitionEmissionProb
      forwardProbs[t][S_x] = forwardProbs[t][S_x] + forwardProb
    viterbiProbs[t][S_x] = bestViterbiProb
    bestPrev[t][S_x] = bestViterbiS_y

bestLastViterbiProb := -1
bestLastS_x := null
forwardProbSum := 0
for each possible maneuver type value S_x:
  lastViterbiProb := viterbiProbs[n][S_x]
  if bestLastViterbiProb < lastViterbiProb:
    bestLastViterbiProb = lastViterbiProb
    bestLastS_x = S_x
  forwardProbSum = forwardProbSum + forwardProbs[n][S_x]
path := empty list
path[n] = bestLastS_x
For each t starting from n-1 until 1:
  lastS_x := path[t+1]
  path[t] = bestPrev[t][lastS_x]
```

Listing 9: Pseudocode for combined implementation of customized Viterbi and Forward algorithm

The algorithm outputs the most probable sequence of maneuver types within `path`. Additionally, it computes the forward probabilities for each possible S_x of s_t and stores them within `forwardProbs`. The sum of forward probabilities of all $s_n = S_x$ is also

computed as `forwardProbSum` because it is required for confidence calculation with Forward-Backward algorithm.

The implemented Viterbi algorithm introduces similarities with Dijkstra algorithm which is used in graph theory for computation of the shortest path with minimal distance sum from a start node toward a destination node considering the distances between each node. Within Viterbi algorithm, instead of distances transition and emission probabilities are used with the aim to find the best $s_{1..n}$ with maximal likelihood which can be also considered as best path of a graph with $n \times m$ nodes where n is the number of observations and m the number of possible hidden state values S_x. The idea is exemplified by the illustration below:

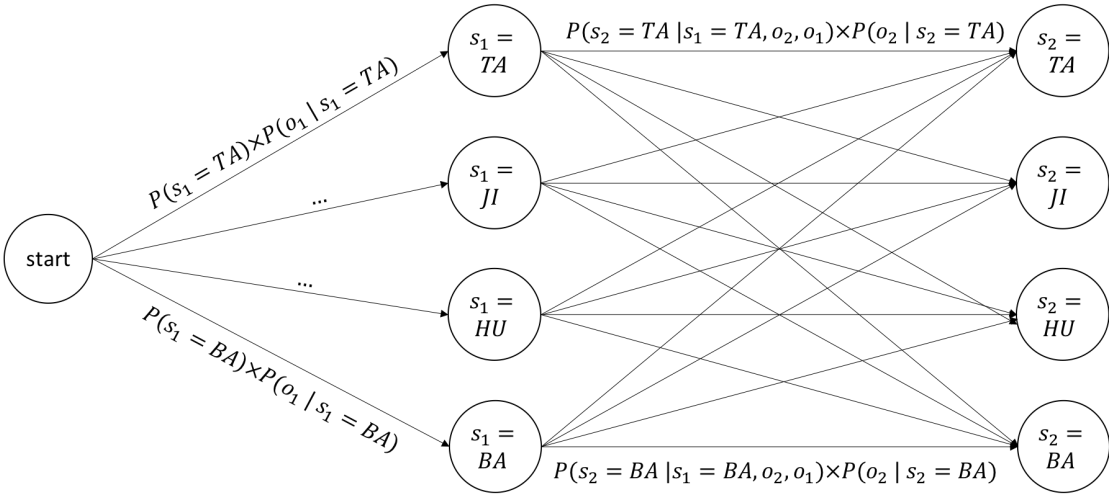

Figure 43: Viterbi as algorithm for directed graph

The hidden state values in the illustration above are tack (TA), jibe (JI), head-up (HU) and bear-away (BA).

TWD and TWS mapping

After the maneuver types of maneuvers were determined, wind fixes with TWD and TWS can be constructed. To provide the most accurate fixes, only tack and jibe maneuvers are considered, while head-up and bear-away maneuvers get ignored. The reasoning is that tacks and jibes are mostly initiated and finished at same absolute TWAs and thus, allow TWD derivation from its middle courses. Head-up and bear-away maneuver are performed arbitrarily which does not allow a meaningful TWD derivation. Furthermore, additional filtering criteria were added to improve the certainty of produced wind fixes. The following list summarizes the criteria for determination of maneuvers which are mapped to wind fixes:

- Maneuver type must be either tack or jibe
- Derived TWD range for maneuver must be not wider than 30°
- The TWD range of previous maneuver must intersect with TWD range of currently analyzed maneuver

If the criteria are satisfied by a maneuver, then it will be mapped to a wind fix as follows:

- TWD is the middle course of maneuver if the maneuver is tack. In case of jibe, TWD will be the inverted middle course of maneuver
- If polar information is available, TWS is determined from `PolarDataService` as follows:
    - TWS for TWA and SOG before maneuver are queried from `PolarDataService`
    - TWS for TWA and SOG after maneuver are queried from `PolarDataService`
    - The final TWS will be the mean of both queried TWS values
- Position and time point of wind fix will be the position and time point of corresponding maneuver

For TWS query, the new method of `PolarDataService` is used which was introduced in *3.3.5 Conclusion*.

When the wind track is determined, the TWS within all wind fixes gets replaced by the median TWS value. It is justified by the fact the TWS is affected by wind gusts and further instabilities. Furthermore, the SOG is affected by sailing mistakes and airflow stagnation due to obstacles and other boats. The median of all TWS fixes is meant to provide the most common and outlier free TWS.

Confidence calculation

The confidence of each derived wind fix is derived by means of Forward-Backward algorithm. Forward-Backward algorithm relies on Forward and Backward probabilities, as well as the sum of Forward-Probabilities of s_n for all possible values S_x. The formula for confidence calculation for each wind fix at time point t is the following:

$$confidence_t = \frac{forwardProbs[t][s_t] \times backwardProbs[t][s_t]}{forwardProbSum}$$

While `forwardProbs` and `forwardProbSum` are already given from the custom Viterbi and Forward algorithm implementation introduced in *Most probable maneuver*

type sequence section of *5.2.6 Wind track inference*, `backwardProbs` must be calculated by means of Backward algorithm. Its implementation is based on the following pseudocode:

```
n := number of observations
backwardProbs := empty list
backwardProbs[n] = empty map

for each possible maneuver type value S_x:
  backwardProbs[n][ S_x] = 1

for each t starting from n-1 until 1:
  forwardProbs[t] = empty map
  recycle transition probability matrix P(s_t+1 = S_y | s_t = S_x, o_t, o_t+1) for all S_x × S_y
    from Viterbi and Forward algorithm implementation
  for each possible maneuver type value S_x:
    backwardProbs[t][S_x] = 0
    for each possible maneuver type value S_y:
      transitionEmissionProb := P(s_t+1 = S_y | s_t = S_x, o_t, o_t+1) * P(o_t+1 | s_t+1 = S_y)
      backwardProb := backwardProbs[t+1][ S_y] * transitionEmissionProb
      backwardProbs[t][S_x] = backwardProbs[t][S_x] + backwardProb
```

Listing 10: Backward algorithm

5.3 Outlier removal-based wind estimators

The concept of outlier removal-based wind estimators is to determine the wind considering all maneuvers which have been classified as tack. From tack maneuvers, wind fixes are constructed. Lastly, an outlier removal method is applied to produce a smooth wind track. Within this work, two outlier removal-based wind estimators were implemented. The difference between both is the applied outlier removal strategy.

5.3.1 Inference of temporary wind track

The first step of the algorithm is to infer a temporary wind track on which an outlier removal strategy will be applied. For this, the classification probabilities of each maneuver are mapped to maneuver type. Afterwards, a wind fix is created for each tack maneuver and initialized as follows:

- Position and time point of maneuver
- TWD is maneuver middle course
  TWS is derived from `PolarDataService` considering the average of TWA, SOG before and after maneuver
- Confidence is the classification probability of maneuver being a tack

Analogously to clustering-based approach, there is at least one tack maneuver required by outlier removal-based wind estimators to produce a non-empty wind track. A decision was made to exclude wind fixes which could be derived from jibes due to expected classification inaccuracy of jibes. As data exploration and classifier evaluation revealed, the maneuver features of jibe are less characteristic than of tacks. For the applied outlier removal strategies, it is crucial that the number of outliers gets considerably below the number of non-outliers. Otherwise, outliers get almost indistinguishable from non-outliers. This issue shall be mitigated by considering of maneuvers with maximally accurate classification which is provided only by tacks.

5.3.2 Mean-based outlier removal

The first outlier removal strategy is to determine the mean TWD and remove all wind fixes differing with the mean more than 30°. It is considered to be simple and efficient for environment where TWD is changing no more than ±30° throughout the sailing session.

5.3.3 Neighbor-based outlier removal

The second strategy is more sophisticated and supports any TWD changes, as long as they happen gradually. It was inspired by the present outlier removal strategy of SAP Sailing Analytics for GPS-fixes which is introduced in *3.2.1 GPS fixes set with removed* outliers. The strategy is composed of the following three steps:

- Detection of the first non-outlier TWD
- Elimination of all wind fixes with TWD which differs from TWD of its preceding non-outlier fix more than 30°

The outlier removal strategy is exemplified by the following pseudocode:

```
windFixes := derived wind fixes from tack maneuvers
firstRealNonOutlier := null
firstPossibleNonOutlier := null
previousFix := null
currentFix := null
for each nextFix in windFixes:
  if currentFix is not null:
    absoluteDiffInDegrees := |TWD difference between currentFix and nextFix|
    if absoluteDiffInDegrees ≤ 30:
      if firstPossibleNonOutlier is null:
        firstPossibleNonOutlier = currentFix
      if previousFix is not null:
        absoluteDiffInDegrees = |TWD difference between …
          … currentFix and previousFix|
        if absoluteDiffInDegrees ≤ 30:
          firstRealNonOutlier = currentFix
          abort for each loop
includedWindFixes := empty list
excludedWindFixes := empty list
if firstPossibleNonOutlier is null:
  includedWindFixes = windFixes
else:
  lastNonOutlier := firstRealNonOutlier
  if lastNonOutlier is null:
    lastNonOutlier = firstPossibleNonOutlier
  for each windFix in windFixes:
    absoluteDiffInDegrees := |TWD difference between windFix and lastNonOutlier|
    if absoluteDiffInDegrees ≤ 30:
      includedWindFixes.add(windFix)
      lastNonOutlier = windFix
    else:
      excludedWindFixes.add(windFix)
```

Listing 11: Pseudocode for neighbor-based outlier removal

5.3.4 Confidence calculation

There is only one confidence value determined which gets set to all produced wind fixes. Its calculation is based on relation between the confidences of included and excluded wind fixes which is defined by the following pseudocode:

```
confidenceSumOfIncluded := sum of all confidence values …
  … contained in includedWindFixes
confidenceSumOfExcludedWindFixes := sum of all confidence values …
  … contained in excludedWindFixes
finalConfidence := confidenceSumOfIncluded / (confidenceSumOfIncluded+confidenceSumOfExcluded)
```

Listing 12: Pseudocode for confidence calculation of outlier removal-based wind estimators

5.3.5 Final wind track

The final wind track is derived from `includedWindFixes`. All wind fixes from `includedWindFixes` are added to the final track. Additionally, the wind fixes are modified as follows:

- Confidence value is set to `finalConfidence`
- TWS:
  - For mean-based outlier removal: median TWS of all included wind fixes
  - For neighbor-based: original TWS of all included wind fixes

The purpose of different TWS calculation for each outlier removal-based strategy is to evaluate both TWS inference strategies within evaluation phase.

5.4 Clustering-based wind estimator

Clustering-based wind estimator is another implementation which is built on top of maneuver type classifiers. Its idea is to employ SAP Sailing Analytics k-Means algorithm implementation to cluster all maneuver instances of a race. Afterwards, the maneuver type is determined per cluster considering the classification results of each maneuver within the cluster. When the maneuver type of a cluster is known, it gets assigned to all maneuvers within the cluster. Finally, a wind track is inferred from tack and jibe maneuvers. This estimator concept is expected to perform well in environment without significant TWD changes.

5.4.1 K-Means setup

To cluster maneuvers, k-Means clustering implementation of SAP Sailing Analytics is used with the following configuration:

- k is 16
- Euclidian distance function considering the following maneuver features:
  - Maneuver middle course
  - Signed maneuver angle
- Custom initialization of centroids as provided by the following pseudocode:

```
k := 16
centroids := empty list
iterate with i from 0 until (k / 2) in (360 / k) steps:
  middleCourseInDegrees := i * 360 / k / 2
  starboardManeuverAngle := 45
  portManeuverAngle := -45
  centroids.add(middleCourseInDegrees, starboardManeuverAngle)
  centroids.add(middleCourseInDegrees, portManeuverAngle)
```

Listing 13: Equidistant centroid initialization of k-Means

The clustering configuration is meant to group maneuvers performed within a similar course range and maneuver angle and determine the maneuver group with tacks considering maneuver classification results. A k-Means implementation with such configuration was already provided within SAP Sailing Analytics platform. Its setup was not further questioned and taken as it was.

5.4.2 Maneuver type inference

The maneuver types are inferred by determination of a cluster which is dominated by tack maneuvers. For this, the classification result of each maneuver within a cluster is considered. The higher the probabilities of cluster maneuvers for being a tack, the higher the likelihood for the cluster being dominated by tack maneuvers. Furthermore, the opposite cluster is considered regarding average score for its maneuvers being jibes. The derivation logic for cluster scoring is introduced by the following pseudocode:

```
tackCluster := cluster with maneuvers to evaluate
tackProbabilities := get tack probabilities for each maneuver within tackCluster
avgTackScore := mean of tackProbabilities
weightedMiddleCourse := calculate weighted average of middle course of all …
  … maneuvers within tackCluster considering its tack probability as weight
jibeCluster := determine cluster with closest weighted middle course to …
  … inverted weightedMiddleCourse
jibeProbabilities := get jibe probabilities for each maneuver within jibeCluster
avgJibeScore := mean of jibeProbabilities
jibeBoostFactor := 0.2 * avgJibeScore
finalTackScore := (avgTackScore + 0.2 * avgJibeScore) / (1 + 0.2)
```

Listing 14: Pseudocode for probability calculation of a cluster being a tack cluster

At the end, the cluster with the highest `finalTackScore` gets labelled as tack cluster and its opposite cluster as jibe cluster.

5.4.3 Confidence calculation

To compute the confidence, the probability of the tack cluster gets related to the highest probability of cluster which is 45° apart from the tack cluster.

```
clusters := all clusters
tackClusterProb := probability of tack cluster being a tack cluster
tackMiddleCourse := weighted middle course of tack cluster
bestChallengingProb := 0
for each cluster in clusters:
  middleCourse := determine the weighted middle course of the cluster
  prob := determine the probability for cluster being a tack cluster
  absoluteDiffInDegrees := absolute difference between …
    … middleCourse and tackMiddleCourse
  if absoluteDiffInDegrees > 45 and prob > bestChallengingProb
    bestChallengingProb = prob
finalConfidence := tackClusterProb / (tackClusterProb+prob)
```

5.4.4 Wind track inference

The wind is inferred from maneuvers of the tack cluster. For each maneuver, a wind fix is created and initialized as following:

- Position and time point of maneuver
- TWD is the middle course of maneuver
- TWS is the median of TWS derived with `PolarDataService` for each maneuver. To test the added value of `PolarDataService` modification introduced in *3.3.5 Conclusion*, the original TWS derivation method of `PolarDataService` was used which is the following:
  (boat class, boat SOG, point of sail) → (list of candidates (TWA, TWS))
  For this method, *upwind*-value is provided as point of sail instead of TWA.
- Confidence is `finalConfidence`

The inference strategy allows to conclude that at least one tack maneuver is required by clustering-based wind estimation in order to produce a non-empty wind track.

5.5 Polars fitting wind estimator

The last implemented wind estimator does not rely on maneuver classification. Its concept is based merely on fitting of actual polars into target polars without employment of any ML. It is not expected that the performance of this estimator gets on par with ML-based model. Notwithstanding, the approach will be evaluated in order to capture the added value provided by maneuver features and ML. Furthermore, in contrast to ML-based estimators, the polars fitting wind estimator is expected to be robust against low sampling rates of GPS sensors.

5.5.1 Algorithm overview

The algorithm can be divided in three parts:

- Accumulation of actual polars
- Fitting of actual polars into target polars

Each of the parts is introduced by the following sections.

5.5.2 Accumulation of actual polars

The actual polars are accumulated as follows:

- There are $\frac{360}{10} = 36$ clusters created each representing average SOG of a COG range which is 10° wide. The width of the COG range of cluster is a hyperparameter of the model and will be referred as `CLUSTER_SIZE`
- For each maneuver, SOG and COG tuples before and after maneuver are extracted. Each SOG-COG tuple is mapped to the cluster with COG range which includes the extracted COG. The SOG of the tuple is included in the average SOG of the cluster.

5.5.3 Fitting of actual polars into target polars

The implemented polars fitting strategy resembles the method of least squares by minimizing the sum of the squares of TWS residuals of all clusters. Its implementation is provided by the following pseudocode:

```
boatClass := ...
totalNumberOfFixes := number of (COG, SOG) tuples averaged by all clusters
windCandidates := empty list of triples
iterate with twd starting from 0 until 350 in 10 steps:
  twsWithFixesCountList := empty list
  iterate over each non-empty cluster:
    middleCog := middle COG of COG range of the cluster
    numberOfClusterFixes := number of (COG, SOG) tuples averaged by the cluster
    avgSog := average SOG of the cluster
    absTwa := |twd - middleCog|
    if absTwa > 180:
      absTwa = 360 - absTwa
    twsInKnots := PolarDataService.getTws(boatClass, avgSog, boatClass)
    if 2 < twsInKnots ≤ 20:
      twsWithFixesCount := (twsInKnots, numberOfClusterFixes)
      twsWihtFixesCountList.add(twsWithFixesCount)
  weightedTwsInKnots := 0
  for each (twsInKnots, fixesCount) tuple in twsWithFixesCountList:
    weight := fixesCount / totalNumberOfFixes
    weightedTwsInKnots = weightedTwsInKnots + weight * twsInKnots
  variance := 0
  for each (twsInKnots, fixesCount) tuple in twsWithFixesCountList:
    twsDeviation := weightedTwsInKnots – twsInKnots
    squaredTwsDeviation := twsDeviation * twsDeviation
    partialVariance := fixesCount / totalNumberOfFixes * squaredTwsDeviation
    variance = variance + partialVariance
  triple := (twd, weightedTwsInKnots, variance)
  windCandidates.add(triple)
finalWind := TWD and TWS with lowest variance in windCandidates
```

Listing 15: Fitting of actual polars into target polars

The pseudocode logic can be summarized as follows:

- Possible TWD values are iterated in 10th steps clockwise starting from 0°
- TWS of each non-empty cluster is determined. For this, TWA is derived from currently iterated TWD and average COG of the cluster in order to query TWS from `PolarDataService`.
- Weighted TWS is derived for the currently iterated TWD considering the yielded TWS of each cluster and its number of fixes. The TWS variance introduced by all clusters is calculated as well. Currently iterated TWD with its TWS and variance are added as a triple to the list of candidates.
- From the list of candidates, TWD and TWS with the lowest variance are determined.

5.5.4 Confidence calculation

The confidence of the final TWD and TWS are calculated considering the computed TWS variances of each evaluated TWD. More precisely, the smallest variance of a

TWD candidate is determined which is at least 45° apart from the final TWD, and then it gets related to the variance of the final TWD. The logic is as follows:

```
finalWind := Final wind determined from previous listing
windCandidates := list of wind candidates from previous listing
finalTwd := TWD from finalWind
finalVariance := variance from finalWind
bestChallengingVariance := maximal positive value
for each windCandidate in windCandidates:
  twdCandidate := TWD of windCandidate
  varianceCandidate := variance of windCandidate
  absoluteDiffInDegrees := |difference between finalTwd and twdCandidate|
  if absoluteDiffInDegrees ≥ 45 and bestChallengingVariance > varianceCandidate:
    bestChallengingVariance = varianceCandidate
finalConfidence := (bestChallengingVariance − bestVariance) / (bestChallengingVariance + bestVariance)
```

Listing 16: Confidence calculation of polars fitting-based wind estimator

5.5.5 Wind track inference

In contrast to other wind estimators, the concept of polars fitting wind estimator is not meant to support the inference of wind track. Instead, the wind is estimated for the whole track and provided as a single TWD-TWS tuple which is represented by `finalWind` variable in the listing above. However, to make the estimator compatible to the evaluation process designed for all wind estimators, the single wind fix will be mapped to a wind track as follows:

```
maneuvers := all maneuvers provided as input for the wind estimator
finalWind := Final wind determined from previous listing
twd := TWD from finalWind
tws := TWS from finalWind
finalWindTrack := empty list
confidence := Final confidence determined from previous listing
for each maneuver in maneuvers:
  twaBefore := calculate TWA before maneuver considering twd and …
    … COG before maneuver
  twaAfter := calculate TWA after maneuver considering twd and …
    … COG after maneuver
  if signs of twaBefore and twaAfter do not match:
    if ||twaBefore|-|twaAfter|| ≤ 20:
      windFix := new wind fix with twd, tws, confidence, and with time point …
        … and position of maneuver
      finalWindTrack.add(windFix)
```

Listing 17: Pseudocode for wind track inference of polars fitting-based wind estimator

6 Evaluation

This chapters covers the whole evaluation process of maneuver classification algorithms and wind estimation models. Firstly, benchmark of all selected algorithms for maneuver classification is performed to select the most appropriate algorithm for ensemble models. Then, all engineered wind estimation models are evaluated and discussed regarding its estimation accuracy and credibility of its confidence.

6.1 Probabilistic maneuver classifiers

This section is dealing with evaluation of maneuver classifiers which are used as part of all engineered wind estimation models with the exception of polars fitting-based model.

6.1.1 Evaluation plan

The aim of maneuver classifiers evaluation is to select the most accurate classification algorithm and reuse it with its optimal configuration in Java by means of SMILE-library. To guarantee maximal accuracy for each evaluated ML algorithm, the preferable pre-processing pipeline and optimal hyperparameters will be determined for each of the algorithms individually. For all the tasks, the following action plan was elaborated:

1) Determination of optimal pre-processing pipeline: each classifier gets evaluated with its default hyperparameters on three available pre-processing pipelines employing the following pre-processing:
    a. No feature scaling, no PCA
    b. Feature scaling, no PCA
    c. Feature scaling, PCA

    For each ML algorithm the pre-processing pipeline with the highest achieved evaluation score is selected.
2) Hyperparameter search: various hyperparameters are tested for each classifier. The optimal hyperparameters are selected.
3) Selection of best algorithm: Benchmark with all classifiers is performed considering its individual hyperparameters and pre-processing preferences. The algorithm with the highest test score is selected.
4) Evaluation on Test-dataset: The best algorithm is trained on train dataset and tested on test dataset to report its final performance on new data.

6.1.2 Training and scoring

Within all evaluation steps of classifiers with the exception of step 4), 3-fold cross-validation is applied on train dataset. The evaluation metric will be the averaged F1-score of each maneuver type which is also known as **macro-averaged F1-Score**. In context of maneuver classification it is computed as follows:

$$f1_{macro} = \frac{f1_{tack} + f1_{jibe} + f1_{other}}{3}$$

An alternative to macro-averaged F1-Score is weighted/micro-averaged F1-Score which is derived as follows:

$$f1_{weighted} = {}^{t}/_{a} \times f1_{tack} + {}^{j}/_{a} \times f1_{jibe} + {}^{o}/_{a} \times f1_{other}$$

With t: number of tacks, j: number of jibes, o: number of other maneuvers, a = number of all maneuvers

However, the selection of the evaluation metric was carried out taking into account the skewed distribution of maneuver types. The aim of F1-score is to maintain the balance between optimistic and pessimistic classifications. However, considering the number of tacks vs. remainder and the definition of precision which is used in F1-score for penalization of optimistic classifiers, the penalty for false positive tacks in $f1_{tack}$ is expected to be small because the precision formula includes the number of true positives in the numerator. To mitigate this effect, macro-averaging is used. Whenever a tack gets falsely classified, it will significantly affect $f1_{jibe}$ and $f1_{other}$ and result in a lower average value.

6.1.3 Maneuver data statistics

In the following, the numbers of all datasets used for this evaluation are listed:

- Total clean maneuvers: 838,187
  - Tacks: 576,108 (69 %)
  - Jibes: 185,355 (22 %)
  - Others: 76,724 (9 %)
- Training maneuvers (year 2018 excluded): 667,015
  - Tacks: 452,096 (68 %)
  - Jibes: 150,891 (23 %)
  - Others: 64,028 (9 %)
- Test maneuvers (year 2018 only): 171,172
  - Tacks: 124,012 (72 %)

- Jibes: 34,464 (20 %)
- Others: 12,696 (8 %)

The clean maneuvers are maneuvers satisfying all filtering criteria introduced in *4.6 Data cleansing*.

6.1.4 Optimal hyperparameters and pre-processing pipeline

The following table presents the optimal hyperparameters and pre-processing pipeline determined for each selected ML classification algorithm:

Table 4: Determined optimal hyperparameters and pre-processing pipelines for selected maneuver classification algorithms

Algorithm	**Pre-processing**	**Hyperparameters**
SVM	No scaling no PCA	Kernel: Linear C: 1
	Polynomial and RBF kernels significantly improve the prediction score, but due to its $O(n^2)$ training complexity it does not scale from > 40,000 training samples	
Naive Bayes	Scaling PCA	None
Logistic Regression	No scaling no PCA	Penalty: L2 C: 1
LDA	No scaling no PCA	None
QDA	No scaling no PCA	None
Random Forest	No scaling no PCA	Trees: 30 Max depth: unlimited
	The prediction score minimally increases with each additional tree. However, training and scoring time increases proportionally with each added tree as well, while the overall prediction score does not improve more than 1 %. 30 Trees is considered as compromise between prediction score and training/scoring time.	

Gradient Boosting	No scaling no PCA	Boosting stages: 100 Learning rate: 0.1 Max depth: 3
Neural Network	Scaling no PCA	Hidden layers: 2 neurons in hidden layer 1: 200 neurons in hidden layer 2: 200 activation for hidden layers: logistic sigmoid
	Analogously to Random Forest tree number, the accuracy of Neural Network increases minimally with each added neuron. The selected hyperparameters constitute a compromise between accuracy and training/scoring time. Furthermore, <1% higher accuracy was achieved with ReLU activation function. Since ReLU is not available in SMILE, it is not used within evaluation process.	
K-Nearest Neighbor	Scaling no PCA	Neighbors: 25

6.1.5 Benchmark

The benchmark of classification algorithms was performed on datasets with all possible feature category combinations introduced in *4.8.3 Feature categories*. The most meaningful feature category combinations are included in the benchmark result presentation. Within benchmark charts, each classifier name corresponds to the name of its implementation class within scikit-learn. The less obvious scikit-learn class names are explained in the following:

- **MLPClassifier**: Neural Network (Multi-Layer Perceptron classifier)
- **LinearSVC**: SVM classifier with linear kernel
- **GaussianNB**: Gaussian Naive Bayes

The following charts present the test score benchmark of classifiers:

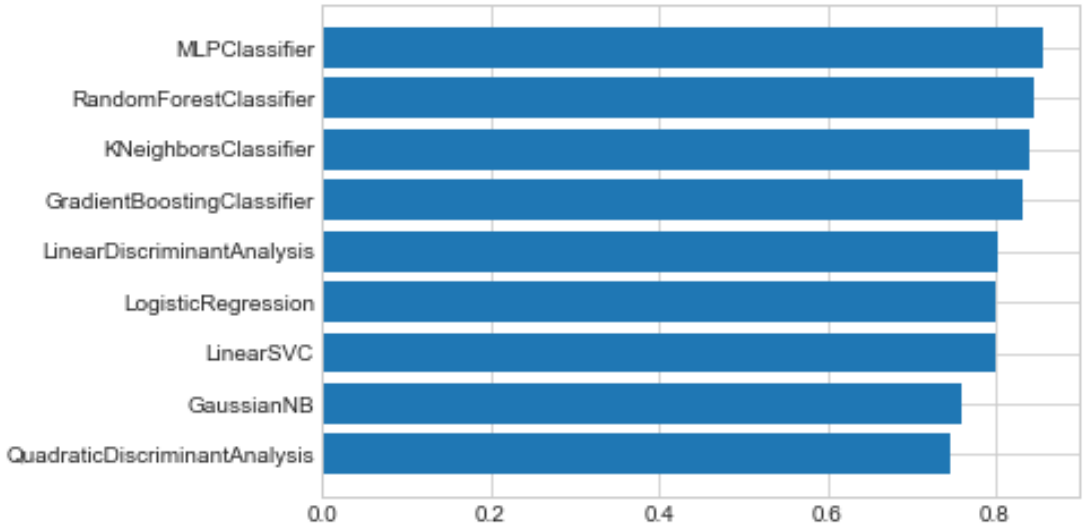

Figure 44: Test score benchmark of selected algorithm with enabled polar, mark and scaled speed features

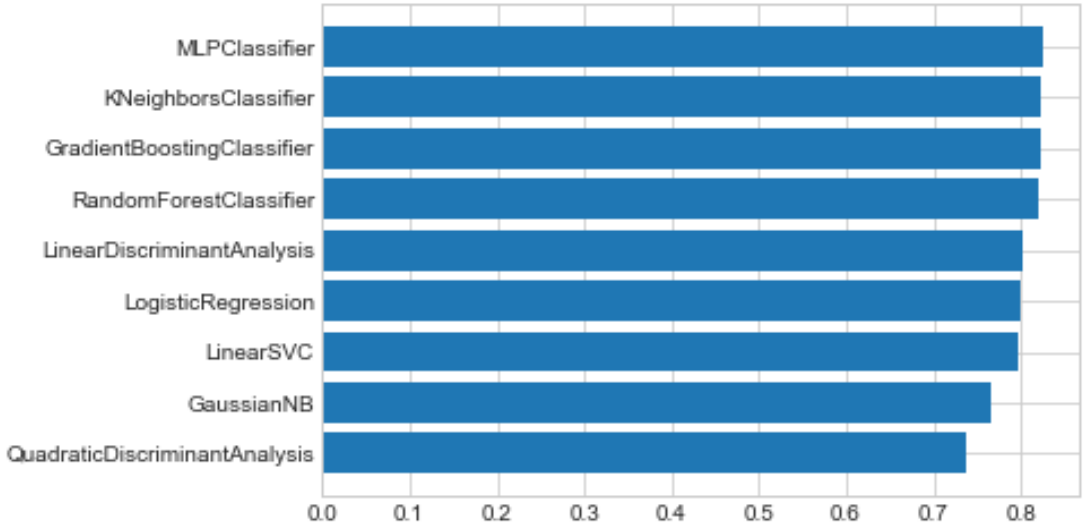

Figure 45: Test score benchmark of selected algorithm with enabled mark and scaled speed features

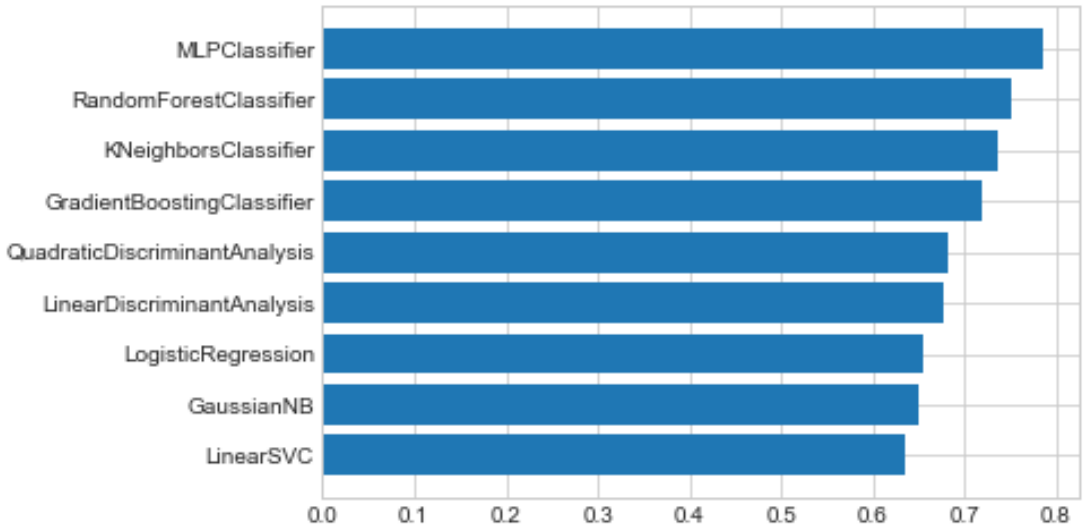

Figure 46: Test score benchmark of selected algorithm with enabled polar and scaled speed features

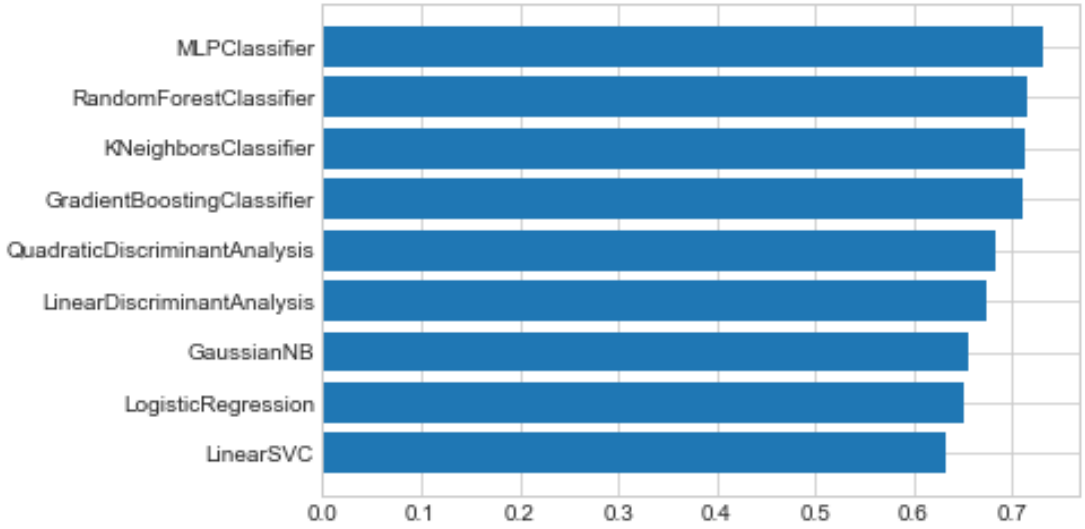

Figure 47: Test score benchmark of selected algorithm with enabled scaled speed features

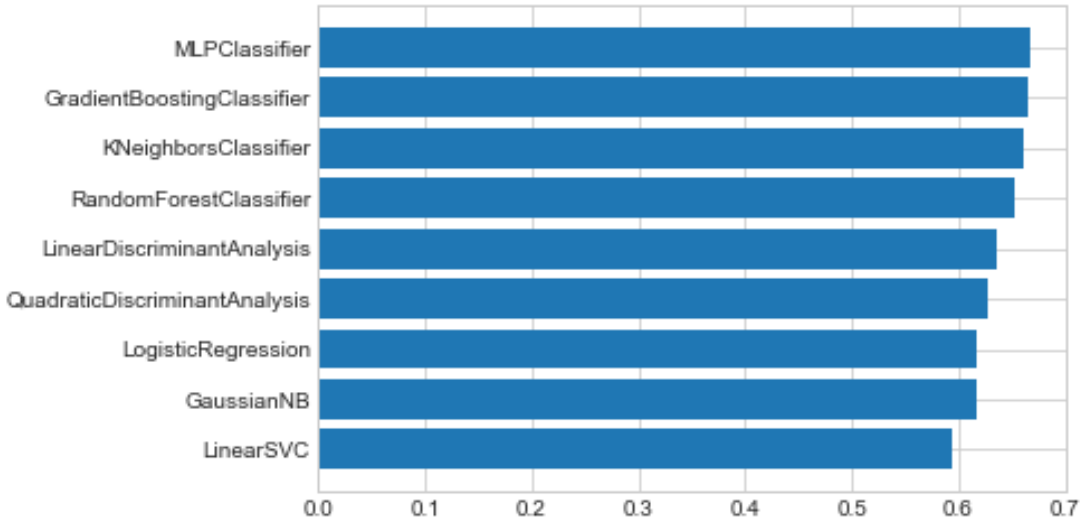

Figure 48: Test score benchmark of selected algorithm with basic features only

Considering the results of all benchmarks, there is a clear winner. Throughout all benchmarks the highest classification score was recorded with Neural Networks. Furthermore, the benchmark underpins the added value for each individual feature category. With all optional features provided, the score of Neural Networks classifier is $\frac{0.857-0.668}{0.668} = 28\%$ higher than with basic features only.

6.1.6 Final classification performance report

After the decision for ML classification algorithm and its optimal setup was made, its final performance is tested on new data which was not used during algorithm evaluation and hyperparameter tuning. The aim is to report the expected classification accuracy on new data without the influence of hyperparameter tuning. For this, the classification model is trained using the whole training dataset. Afterwards, the trained model is used to classify the whole test dataset. The following tables present the classification results for each maneuver type individually using precision, recall and F1-score metrics.

Table 5: Final classification report with enabled polar, mark and scaled speed features

	precision	recall	f1-score	support
JIBE	0.83	0.64	0.72	34464
OTHER	0.59	0.73	0.65	12696
TACK	0.94	0.98	0.96	124012
avg / total	0.89	0.89	0.89	171172

Table 6: Final classification report with enabled mark and scaled speed features

	precision	recall	f1-score	support
JIBE	0.80	0.81	0.80	34464
OTHER	0.69	0.73	0.71	12696
TACK	0.97	0.96	0.97	124012
avg / total	0.92	0.91	0.92	171172

Table 7: Final classification report with enabled polar and scaled speed features

	precision	recall	f1-score	support
JIBE	0.75	0.61	0.68	34464
OTHER	0.47	0.55	0.50	12696
TACK	0.93	0.96	0.94	124012
avg / total	0.86	0.86	0.86	171172

Table 8: Final classification report with enabled scaled speed features

	precision	recall	f1-score	support
JIBE	0.78	0.78	0.78	34464
OTHER	0.57	0.46	0.51	12696
TACK	0.95	0.97	0.96	124012
avg / total	0.89	0.89	0.89	171172

Table 9: Final classification report with based features only

	precision	recall	f1-score	support
JIBE	0.70	0.61	0.65	34464
OTHER	0.52	0.41	0.46	12696
TACK	0.91	0.96	0.93	124012
avg / total	0.84	0.85	0.84	171172

Table 10: Final classification report with enabled polar features

	precision	recall	f1-score	support
JIBE	0.53	0.57	0.55	34464
OTHER	0.43	0.45	0.44	12696
TACK	0.91	0.89	0.90	124012
avg / total	0.80	0.79	0.79	171172

6.1.7 Conclusion

The final classification report revealed the following interesting facts about the engineered maneuver classifier:

- Tack classifications achieve the highest score. However, considering the formula of scoring metrics, the classification error for tacks can get widened by factor 2 or 3 if the skewed distribution of maneuver types gets normalized. Notwithstanding, the distribution is based on real world data. Additionally, the distribution is affected by data cleansing which can be considered as classification helper because it considers maneuver angle which is a maneuver feature. Therefore, despite the skewed data distribution, the high classification score has been accepted as it is.
- Polar features seem to disrupt the classification of test dataset which contradicts the benchmark results. Particularly surprising is the deterioration of tack classification score despite the strong correlation of deviation of maneuver angle from optimal tack angle feature. There are various possible reasons for this issue. One reason can be that within year 2018 new boat classes were introduced which affects the quality of polar data. Another reason might be new category of sailors with different sailing skills. However, these considerations are based on speculation and will not be investigated within this work due to time reasons.
- The most uncertainty in classification results is present within jibe and other maneuvers, even with enabled scaled speed features. The classification of these maneuvers significantly improves if mark features are enabled. This insight is underpinned by data exploration results which revealed that most of head-up and bear-away maneuvers with a maneuver angle > 30° are mark passing maneuvers.

6.2 Wind estimators

This section covers the evaluation process of all wind estimation models.

6.2.1 Evaluation plan

The purpose of the wind estimation model is to predict TWD, TWS and provide appropriate confidence value underpinning the credibility of estimation results. Thus, all these model outputs must be systematically evaluated. For this, the following action plan was elaborated:

1) Specification of measurements: All measurements including its measures for errors and correctness are defined.
2) Evaluation setup: The maneuver classification model is prepared for wind estimators. Training and test datasets are defined. Environment for evaluation is created
3) TWD evaluation: TWD estimation results of all wind estimators are analyzed regarding its accuracy
4) TWS evaluation: TWS estimation results of all wind estimators are analyzed regarding its accuracy
5) Confidence evaluation: The credibility of confidence outputs of all wind estimators is analyzed
6) Conclusion: Final assessment of evaluation results

6.2.2 Specification of measurements

The task for this section is to specify what to measure and how it should be measured. In contrast to maneuver classification, the appropriate measurements for wind estimation evaluation are less obvious due to the nature of its input and output. The input of all wind estimation models is a sequence of maneuvers corresponding to a single race, while the output is a wind track of the race. The correctness of the inferred wind track depends on the target wind track. Within this work, the target wind track is composed of wind fixes sampled from SAP Sailing Analytics platform at positions and time points of maneuvers. Since all wind fixes within implemented wind estimators are also created at positions and time points of maneuvers, they can be easily mapped to target wind fixes for comparison.

Wind fix measures

There are following measures which will be collected from comparison of inferred vs. target wind fixes:

- TWD deviation (non-negative)
- TWS deviation (non-negative)

TWD of a wind fix will be considered as correct if the following criterion is satisfied:

- TWD deviation is $\leq 20°$

For TWS correctness the following fulfillment criteria apply:

- TWD is correct
- TWS deviation is ≤ 2 kn

Since the highest attention was paid to TWD inference which does not depend on TWS inference, a decision was made to evaluate TWD inference individually without consideration of TWS derivation fulfillment criteria. However, TWS determination depends on TWD due to TWA derivation which is required for `PolarDataService`. Thus, the premise for correct TWS estimation is that TWD estimation is also correct. Therefore, a TWS measure will be considered only as correct if both, TWD and TWS are correct.

Wind track measures

A wind track will be considered as correct if 75% of its wind fixes are considered as correct. For polars fitting-based wind deviation the fulfilment criteria were intentionally weakened due to its coarse grained wind fix inference. The TWD deviation tolerance was increased to 45°. Its adapted wind track will be considered as valid if more than half of its wind fixes (51 %) get considered as correct.

Empty wind tracks

Since there is at least one positive tack or jibe classification required by all wind estimators for the inference of non-empty wind tracks, a strategy for treatment of empty wind tracks must be defined. There are following reasons which may lead to absence of tack/jibe classifications:

- Invalid TWD inference
- No tacks and jibes with acceptable data quality are contained within the race

Empty wind track indicates that there is not enough data available for wind estimator to produce a reasonable wind track. This result is considered as more valuable than an incorrect result. Therefore, to distinguish between correct estimations, wrong predictions and definite uncertainty, the following measures are used for wind track evaluation:

- Number of correct predictions
- Number of correct and incorrect predictions
- Percentage of correct predictions vs. correct and incorrect
- Number of ignored races due to empty wind tracks

Confidence measures

The purpose of the confidence is to provide a level of certainty which is given by wind estimator for its estimation result. Low confidence indicates high uncertainty of the model which in turn allows to conclude that there is a high probability for the estimation being wrong. This information can be used by a high-level estimation component for decision making whether the estimation result shall be presented to a user or not due to expected inaccuracy. A confidence calculation strategy will be considered as successful, if the average confidence of correct estimations gets considerably higher than the average confidence of incorrect estimations. It is useful, if the difference between both measures allows definition of a threshold value to determine whether the estimation is certain or not.

6.2.3 Evaluation setup

To supply wind estimators with maneuver classification results, a Neural Network-based maneuver classifier has been implemented in Java by means of SMILE-library. The implementation is compatible with the whole infrastructure for classification model management which was introduced in *5.1 Probabilistic maneuver classification*. It was trained on the whole training dataset which contains maneuvers of all races except of the year 2018. There were multiple models trained and persisted considering the concepts defined in *5.1.2 Handling optional features* and *5.1.3 Boat class specialization and generalization*. The races of year 2018 are used to test the performance of wind estimators within this evaluation.

For this work, a custom Java evaluation framework was created in order to collect the specified measures in a compatible manner to SAP Sailing Analytics and to already implemented Java code including wind estimators and data loading. The framework makes use of lazy loading iterators which are listed in *4.4.3 Data loading from MongoDB*.

6.2.4 TWD evaluation

The whole evaluation process was performed only with two combinations of optional feature categories:

- Enabled mark and scaled speed features, disabled polar features
- All optional features disabled

Its purpose is to compare concisely best results with the worst results. The following results were recorded for wind estimators with enabled mark and scaled speed features:

Table 11: TWD evaluation results of wind estimators with enabled mark and scaled speed features

Measurement description	HMM	Cluster	Outlier Mean	Outlier Neighbor	Polars Fitting
Correct races	772/883 (**88.4** %)	767/882 (87 %)	711/832 (85.5 %)	705/839 (84.0 %)	371/630 (58.9 %)
Ignored races	14 (**1.6** %)	15 (**1.7** %)	65 (7.2 %)	58 (6.5 %)	267 (29.8 %)
Avg. TWD deviation within correct races	5.1°	5.2°	5.1°	5.1°	9.3°
Avg. TWD deviation within incorrect races	**70.1°**	84.5°	96.8°	102.6°	146.9°
Avg. TWD deviation within all races	**12.6°**	15.5°	18.4°	20.7°	65.9°

The following table presents the results of wind estimators with all optional maneuver features disabled. Since the polars fitting-based wind estimator does not depend on the optional maneuver features, it was excluded from the table as its result are same as in the table above.

Table 12: TWD evaluation results of wind estimators with enabled mark and scaled speed features

Measurement description	HMM	Cluster	Outlier Mean	Outlier Neighbor
Correct races	693/881 (78.7 %)	755/880 (**85.8** %)	690/831 (83.0 %)	684/840 (81.4 %)
Ignored races	16 (**1.8** %)	17 (**1.9** %)	64 (7.1 %)	57 (6.4 %)

Avg. TWD deviation within correct races	5.0°	5.2°	5.1°	5.1°
Avg. TWD deviation within incorrect races	111.5°	**88.5°**	105.2°	111.2°
Avg. TWD deviation within all races	27.8°	**17.0°**	22.1°	24.8°

The aforementioned results allow to conclude following facts:

- The polars fitting approach is significantly worse than ML-based approaches
- With enabled mark and scaled speed features, HMM-based wind estimator achieves the highest estimation accuracy. However, with basic features it gets the worst accuracy in comparison to all other ML-based models. The reason is probably its weak restriction of wind changes in comparison to other estimators. Therefore, a conclusion can be drawn, that HMM-based model is highly affected by invalid maneuver classifications due to its weak restriction of wind change because it gets outperformed by models with strong wind change restrictions. However, it is considered as too early to nominate maneuver clustering wind estimator as the benchmark winner, because there are many hyperparameters and implementation details of HMM which can be tuned, including transition penalty calculation.
- Outlier-based wind estimators are outperformed by maneuver clustering-based estimator in both cases. Considering this, a conclusion can be made that ignoring classifications of jibe and other maneuvers is contra productive.

6.2.5 TWS evaluation

The TWS estimation was evaluated with enabled mark and scaled speed features. Its evaluation results are presented in the following table.

Table 13: TWS evaluation results of wind estimators with enabled mark and scaled speed features

Measurement description	HMM	Cluster	Outlier Mean	Outlier Neighbor	Polars Fitting
Correct races	417/818 (**51.0** %)	40/825 (4.8 %)	366/786 (46.6 %)	235/792 (29.7 %)	300/592 (50.7 %)
Avg. TWS deviation within correct races	0.8 kn	1 kn	0.8 kn	0.8 kn	0.8 kn
Avg. TWS deviation within incorrect races	4.6 kn	6.6 kn	4.9 kn	6.2 kn	4.4 kn
Avg. TWS deviation within all races	2.7 kn	6.4 kn	3.0 kn	4.6 kn	2.6 kn

The aforementioned results allow to conclude following facts:

- TWS derivation of `PolarDataService` is highly sensitive to TWA. The maneuver clustering-based estimator is the only estimator which employs the original method of `PolarDataService` for TWS derivation based on coarse-grained point of sail, instead of TWA. Thus, the small improvement introduced in *3.3.5 Conclusion* is considered as highly beneficial.
- The median TWS is more representative for a wind track, than different TWS determined for each wind fix individually.
- TWS derivation is significantly less accurate than TWD derivation. The reason is probably that TWS is less stable than TWD. Instability of TWS implies various confounders for this work. First, since the wind is considered as a position dependent variable, and most of the employed wind measurement systems within SAP Sailing Analytics events are stationary, a conclusion can be drawn that the target wind introduces inaccuracy as well, because it is not measured at positions of sailboats. Furthermore, the TWS derivation strongly depends on the skills of sailors and the results of maneuver detection algorithm. If the derived speed before or after maneuver does not represent the target speed, the TWS derivation will become incorrect. Overall, a conclusion can be drawn, that

TWS inference from GPS tracks is error-prone and cannot be considered as reliable.

6.2.6 Confidence evaluation

Analogously to TWS evaluation, confidence evaluation was performed also with two combinations of optional feature categories to compare the confidences of best results with confidences of the worst results. The recorded results are as following:

Table 14: Confidence evaluation results of wind estimators with enabled mark and scaled speed features

Measurement description	HMM	Cluster	Outlier Mean	Outlier Neighbor	Polars Fitting
Avg. confidence within correct races	96.9 %	80 %	97.3 %	96.0 %	37.4 %
Avg. confidence within incorrect races	87.3 %	56.4 %	90.8 %	76.2 %	24.6 %
Avg. confidence of all races	95.8 %	76.9 %	96.3 %	92.7 %	32.2 %

Table 15: Confidence evaluation results of wind estimators with enabled mark and scaled speed features

Measurement description	HMM	Cluster	Outlier Mean	Outlier Neighbor
Avg. confidence within correct races	94.7 %	67.1 %	93.2 %	91.8 %
Avg. confidence within incorrect races	88.3 %	52.0 %	87.4 %	77.0 %

Avg. confidence of all races	93.4 %	65.0 %	92.2 %	89.1 %

The aforementioned results allow to conclude following facts:

- The difference between average confidence of correct estimations and average confidence of incorrect estimations is small. It is hard to find a threshold. However, when considering the plain average values, there is still a possible range for threshold values. In case of HMM, it can be at 91 %, for clustering at 60 %, for neighbor-based outlier removal at 85 %. For the remainder, the confidence is less informative. However, without consideration of confidence distributions for incorrect and correct estimations, no meaningful threshold can be determined.
- The average confidences of clustering-based wind estimation appear to be the most meaningful because it provides the highest difference between confidences of correct and incorrect estimations. However, without its distribution analysis, no further conclusions regarding threshold inference can be made.

6.2.7 Conclusion

The evaluation showed that most of the implemented estimators are capable of proper TWD inference. The polars fitting-based wind estimator is considered as significantly less successive and reliable than the ML-based wind estimators. The most successful wind estimator implementations are:

- HMM-based estimator
- Maneuver clustering-based estimator

In comparison to maneuver clustering-based wind estimator, there are various fine-tuning options which can be applied to HMM-based estimator. Therefore, it is considered as the most favorable wind estimator implementation. Due to the given time frame for this thesis, the fine-tuning of HMM, as well as the further analysis of confidence credibility will not be covered within this work.

7 Ending

This chapter concludes the results of this of this work considering its initial goals. It also presents further tasks for possible future works. Finally, an outlook is given about how the results of this work might be used in the future by SAP.

7.1 Achieved results

In general, the work showed that GPS-based wind inference is feasible in context of sailing. The most important data for this task is present in maneuvers. The work uncovered all important features within maneuver data for maneuver classification. The importance of each feature was underpinned by means of known statistical tests such as ANOVA. It was also confirmed by final classification report of the elaborated maneuver classifier. One crucial and underestimated task for achievement of high maneuver classification accuracy was the reengineering of maneuver detection. Thanks to new maneuver detection, all elaborated maneuver features were made available. Furthermore, the whole spectrum of available maneuver data could be understood and evaluated in terms of feature engineering for maneuver classification. The maneuver classification is a crucial component for all successfully tested wind estimation models. The evaluation of wind estimation models on different feature sets showed that the accuracy of wind estimation highly depends on the accuracy of maneuver classification. That is why the detailed feature engineering and elaboration of classification model are considered as important. The final evaluation of wind estimation models showed that with the exception of polars fitting-based model, all ML-based models provide an acceptable accuracy for TWD inference. The most outstanding models are HMM- and maneuver clustering-based. In terms of TWS, the estimation results are less reliable. Thus, its aptness as default value is questionable. The work also introduced various strategies for confidence determination. However, its actual suitability still needs to be determined within further investigations.

Overall, from the perspective of science, the results of this work may be considered as valuable. However, there is always room for improvement which also applies to this thesis.

7.2 Further ideas

This section summarizes further ideas of this thesis which were not tackled due to time reasons.

7.2.1 Maneuver number dependent benchmark

The question which remains unanswered by this thesis is the following: How does the number of maneuvers affect the performance of each wind estimation model? To answer this question, the evaluation framework must be adapted accordingly in order to sample the wind estimation results from wind estimation models after each additional maneuver provided. The number of correct and incorrect estimations including TWD/TWS deviation from target must be collected and averaged per number of maneuver types. For benchmark presentation, a 2D line chart can be used where the x-axis represents the number of maneuvers, and y-axis the averaged measure.

7.2.2 Detailed confidence evaluation

The previous evaluation of confidence showed that the difference between average confidence values of correct and incorrect estimations is not as great as demanded. Notwithstanding, it might be still possible to determine a threshold which separates correct estimations from the most incorrect. To achieve this, confidence values for all individual races and/or wind fixes must be collected and labelled as corresponds or incorrect. Afterwards, the distributions of both correct and incorrect labels must be plotted as box plot or histogram in order to see if both distributions intersect.

7.2.3 Tuning of HMM-based wind estimator

HMM-based wind estimator introduces various hooks which can be used for fine-tuning. The most basic hyperparameter is `MAX_TOLERATED_DIFF` which is introduced in *5.2.5 Transition probability calculation*. Furthermore, the penalty calculation formula for TWD deviations can be tweaked in order to regulate between strong and weak restrictions of wind changes. Another idea is to capture TWD intersections between best hidden states throughout the whole best path of hidden states which is calculated by Viterbi-algorithm. However, it violates the first Markov property $P(s_t = S_x \,|s_{t-1} = S_y)$. Notwithstanding, in terms of regulation of TWD changes, it might reveal as beneficial.

7.2.4 Employment of complex neural networks

Within this work, only basic neural networks architectures could be tested. The type of tested neural networks is also known as multi-layer feedforward neural networks. Today, there are more advanced architectures available. However, in SMILE and scikit-learn such networks are not implemented. The ML technology research showed, that such network implementations can be leveraged in Java by means of Deeplearning4j library. It is considered as a time-consuming task because the implementation and evaluation process of such networks is more complex. Furthermore, the training of such models is considered as tedious because it requires more complex computations which can lead to training times of multiple days. For this task, the use of a CUDA[41]-compatible computer is highly recommended because it allows utilization of GPU to speed up the training process. In the following, the most promising network architectures are listed:

- Generative Adversarial Networks (GAN): GAN is considered as one of the most advanced ML methods for classification and regression tasks. It can be used for maneuver classification
- Long-Short-Term-Memory (LSTM): LSTM is a type of recurrent neural networks which is used in context of sequential data input. One approach which can be tested is whether an LSTM can predict the maneuver type sequences better than HMM-based wind estimator. Another approach can be to use LSTM as a sequence to number regression component which derives TWD and TWS from maneuver data by itself. And finally, LSTM can be also used to predict TWD and TWS by means of plain GPS fixes of a GPS track. However, this approach is considered as problematic due to the amount of training data which must be provided to LSTM model for training. Its training time and scoring time will be probably too high for productive usage.

7.3 Outlook

The results of this work are considered as groundwork for SAP Sailing Analytics platform to allow introduction of default wind data for sailing races where no wind was recorded. The work has explored the solution space for wind inference and presents

[41] "*CUDA® is a parallel computing platform and programming model developed by NVIDIA for general computing on graphical processing units (GPUs). With CUDA, developers are able to dramatically speed up computing applications by harnessing the power of GPUs.*", see https://developer.nvidia.com/cuda-zone

multiple wind estimation prototypes. The next task is to select the appropriate prototype and develop it further so that it fits in the code infrastructure of SAP Sailing Analytics. Since all estimation models are Java compatible and implemented within the code base of SAP Sailing Analytics, no new technologies and time-consuming adaptions are required. The only crucial part is the management of persistent state of maneuver classification models. Furthermore, it is highly likely that further tuning of HMM-based wind estimator will be performed so that it finally outperforms the maneuver clustering-based approach with all feature sets.

List of Abbreviations and Acronyms

ANOVA	Analysis of Variance
COG	Course Over Ground
CPU	Central Processing Unit
CRISP-DM	Cross-Industry Standard Process for Data Mining
DBMS	Database-Management System
DL	Deep Learning
DP	Douglas-Peucker
GPS	Global Positioning System
GPU	Graphics Processing Unit
JNI	Java Native Interface
JVM	Java Virtual Machine
kn	knot(s)
LDA	Linear Discriminant Analysis
ML	Machine Learning
QDA	Quadratic Discriminant Analysis
SAP	Systems, Applications & Products in Data Processing
SMILE	Statistical Machine Intelligence and Learning Engine
SOG	Speed Over Ground
SVM	Support Vector Machines
TWA	True Wind Angle
TWD	True Wind Direction
TWS	True Wind Speed
UI	User Interface
VMG	Velocity Made Good

List of Tables

List of Figures

References

[1] T. M. Blog, "Understanding Analysis of Variance (ANOVA) and the F-test," 18 05 2016. [Online]. Available: http://blog.minitab.com/blog/adventures-in-statistics-2/understanding-analysis-of-variance-anova-and-the-f-test. [Accessed 2018 09 15].

[2] R. MacGregor, "Macgregor 26," Bluewater Yachts, [Online]. Available: http://www.macgregor26.com/how_to_sail/how_to_sail.htm. [Accessed 09 09 2017].

[3] G. Jobson, Sailing Fundamentals, New York: Touchstone, 2005.

[4] "Wayzata Yacht Club," [Online]. Available: http://www.wyc.org/portals/0/forms-docs/seminars-tsang.pdf. [Accessed 09 09 2017].

[5] "NAD Pathfinders Wiki," [Online]. Available: http://www.investitureachievement.org/wiki/index.php?title=Adventist_Youth_Honors_Answer_Book/Recreation/Navigation. [Accessed 09 09 2017].

[6] R. G. Golledge, Wayfinding Behavior: Cognitive Mapping and Other Spatial Processes, USA: Johns Hopkins University Press, 1999.

[7] J. Evans, P. Manley and B. Smith, The sailing bible: The complete guide for all sailors from novice to experienced skipper, London: Adlard Coles Nautical, 2009.

[8] C. Mason, The Best of Sail Trim, Sheridan House, 2000.

[9] RaceQs, "YouTube - raceQs 3D Replay: Analyzing Tacks," 18 03 2015. [Online]. Available: https://www.youtube.com/watch?v=36RSSN7IVXI. [Accessed 14 09 2017].

[10] A. Géron, Hands-On Machine Learning with Scikit-Learn and TensorFlow: Concepts, Tools, and Techniques for Building Intelligent Systems, Sebastopol, USA: O'Reilly UK Ltd., 2017.

[11] Scikit-Learn, "1.13. Feature selection," [Online]. Available: http://scikit-learn.org/stable/modules/feature_selection.html. [Accessed 15 09 2018].

[12] H. M. Huan Liu, Feature Selection for Knowledge Discovery and Data Mining, New York, USA: Springer, 1998.

[13] Z. Ghahramani, "An Introduction to Hidden Markov Models and Bayesian Networks," International Journal of Pattern Recognition and Artificial Intelligence, London, England, 2001.

[14] G. Bonaccorso, Machine Learning Algorithms: Popular algorithms for data science and machine learning, Second Edition ed., Birmingham, UK: Packt Publishing, 2018.

[15] B. Lantz, Machine Learning with R, Birmingham, UK: Packt Publishing, 2013.

[16] M. Brown, Data Mining For Dummies, Hoboken, USA: John Wiley & Sons, 2014.

[17] B. D. Vijay Kotu, Predictive Analytics and Data Mining: Concepts and Practice with RapidMiner, Morgan Kaufmann, 2014.

[18] E. Maor, To Infinity and Beyond: A Cultural History of the Infinite, Princeton, USA: Princeton University Press, 1991.

[19] Wikipedia, "Wikipedia - Python (programming language)," 07 09 2018. [Online]. Available: https://en.wikipedia.org/wiki/Python_(programming_language). [Accessed 09 14 2018].

[20] Y. Zheng and X. Zhou, Computing with Spatial Trajectories, New York, USA: Springer, 2011.

[21] K. Sharan, Beginning Java 8 APIs, Extensions and Libraries: Swing, JavaFX, JavaScript, JDBC and Network Programming APIs, New York, USA: Apress, 2014.