A detailed preprocessing framework for the generation of polar performance diagrams.

Robert Schüler

February 24, 2023

Abstract

The performance of a given sailing vessel is traditionally depicted by polar performance diagrams estimating the relationship between wind and (top performance) boat speed. For some applications, it is desirable to create polar performance diagrams which reflect the real conditions of the sailing vessel in use. This can be done by analyzing data obtained during sailing trips of the particular vessel. The obtained data is likely polluted by several effects and contains data that does not reflect the top performance. We describe a modular preprocessing framework specialized for the use on sailing data. We then make informed choices for the different parts of the framework and test it on real sailing data.

1 Introduction

Polar performance diagrams and tables have been used by professional sailors for many years. They are motivated by the natural occurring task to predict the top boat speed (BSP) a given vessel can reach for given wind, usually given in terms of true wind, refering to the wind a stationary observer would measure. Note that wind is usually described as a pair of speed (TWS) and angle (TWA). An overview of different terms regarding wind is given in Figure 1.

In practice, this information is stored in tables with fixed wind speeds and angles, for example in steps of 2 knots (nautical miles per hour) and 5 degree respectively. A common way to visualize polar performance diagrams is by creating a polar plot of the TWA/BSP relation for several fixed values of TWS (see Figure 2).

Polar diagrams and tables given by manufacturers of sailing vessels or by official sailing organisations like the Offshore Sailing Congress [orc] are obtained from involved, but idealized numerical computer simulations – called Velocity Prediction Programs (VPPs). These simulations are used by designers even before a yacht is actually build. However, in reality it is not always reasonable to use a polar performance diagram obtained from a VPP. Even under idealized sea conditions, the status and capabilities of a used sailing vessel may differ notably from these numerical simulations, which are produced for new yachts.

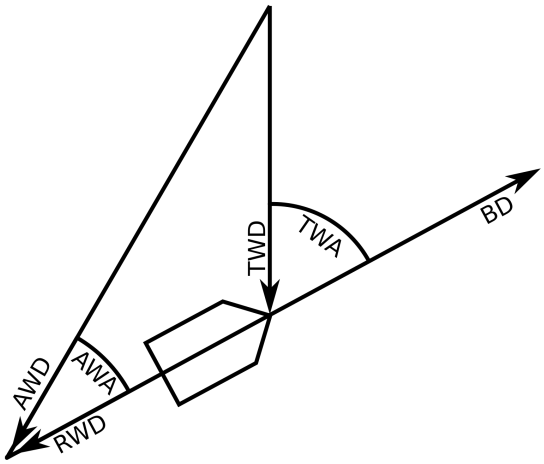

Figure 1: Relation of different wind terms (TWD - true wind direction, AWD - apparent wind direction, TWA - true wind angle, AWA - apparent wind angle, RWD - relative wind direction, BD - boat direction) (cf. [DSS22])

Also, some factors are not included in the polar diagram, as the competence of the sailor, the condition of the hull and the sail etc.

Therefore, many professional sailors try to obtain for their yachts actual polar performance data, from their own real-world measurements and experience. In this regard, automated real life measurements and sophisticated techniques from statistics and data science can help to obtain more realistic estimates for polar performance diagrams. It suits not only the vessel in use, but also can be adapted to specific skippers and their capabilities and habits. There are only a few publicly available studies to support such measurements (cf. [Sim17], [DSS22]).

In this paper, we refine the polar pipeline model described in [DSS22] loosely following concepts described in [GLH15] and use it on a real study case.

A general description of the framework is given in Section 2. In Section 3 we give detailed examples on how the different abstract modular parts of the framework might be realised and in Section 4 we use the framework and the described realisations to analyze real data as a study case.

2 Preprocessing Pipeline Framework for Sailing Data

Next to the usual problems to be solved by preprocessing methods (see [GLH15, Section 1.6.1]) there are some additional difficulties and properties in the context of analyzing sailing data for polar performance diagrams:

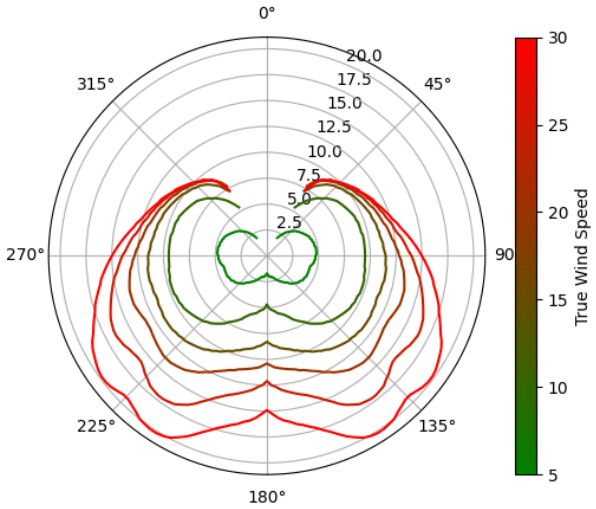

Figure 2: Plot of a polar diagram (cf. [DSS22]).

1. Sailing data is usually obtained over the course of a sailing trip from different measurement instruments and can be documented via logging files. Common file formats, as the NMEA 0183 log files (see [nme]), do not translate directly to organized data sets. But we can strongly assume, that these files describe the sailing trip in chronological order. The most important data (TWS, TWA, BSP and the timestamp) should be already present in these files.

2. The polar performance diagram should represent the top performance. However, the data also contains many phases of acceleration where the top performance is not reached.

3. Additional factors as current and waves can have a huge influence on the measured boat speed. So, in the best case, additional data can be used to minimize the negative effects on the resulting polar performance diagram. Weather data (which has not been recorded by on-board measurement equipment) is traditionally provided in gridded form, for example as GRIB or NetCDF files, and thus do not provide complete information about the conditions during the sailing trip.

4. Additional theoretical knowledge about sailing performance might be applied.

We extend the preprocessing framework discussed in [DSS22] by several parts, especially parts regarding additional data. We keep the general modular structure of the framework. The benefit of such a structure is, that single

parts of the framework can be identified and exchanged easily, making it easier to compare their effects.

A visual summary of the preprocessing framework is given in Figure 3. The framework includes these basic steps:

1. We assume we have multiple data sources containing sailing data from different trips that can be interpreted as (multi-dimensional) time series.
2. We split all data sources in training data and test data.
3. We use various preprocessing techniques on these data sets individually. Here we can use the assumption, that each data set represents the time series of a single sailing trip. Thus, it is reasonable to assume similar (or at least continuously changing) conditions for all records.
4. We concatenate all training data sets and test data sets respectively and convert them into data sets only containing information about TWA, TWS and BSP.
5. We use various preprocessing techniques on the simplified data.
6. We process the training data to obtain a polar performance diagram. A discussion of the processing is not part of this paper.
7. We use the preprocessed test data in order to compute several quality metrics.

Note, that good test data should be obtained independently from the training data and provide broad test cases, i.e. there should be test data available for a broad spectrum of TWA and TWS values. Of course it would be preferable to have the data, or at least the test data, be obtained in an undisturbed environment. However, this is often not practically possible. Therefore, in the presented framework, the test data gets almost the same preprocessing as the training data. This has the benefit, that polluted test data does not lead to bad quality metrics. Of course, this includes the risk to manipulate the training and test data in a way that the quality metrics are not meaningful. We have to take extra care in the choice of quality metrics such that this effect will not happen unnoticed.

In the following, we discuss the different parts of the framework in detail.

2.1 Data standardizing

In order to realize a modular framework, it is necessary to organize the data in a standardized manner.

Data reflecting multi-dimensional time series of various attributes will be treated as a table with named columns where each row represents a single record and the records are ordered chronologically. In a modular framework, it is important, that the names of these columns are consistent throughout the framework such that the different components can be exchanged individually.

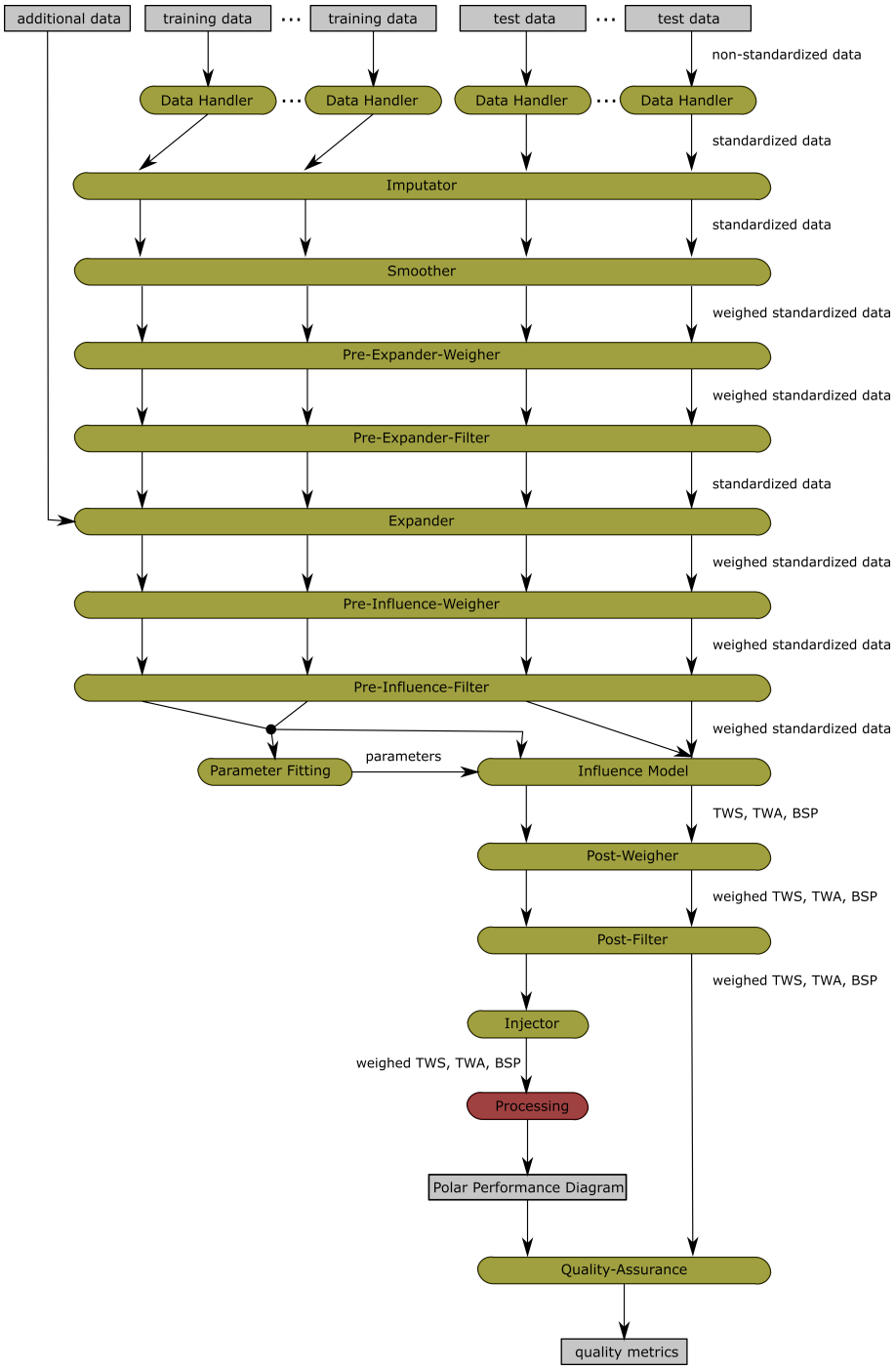

Figure 3: Overview of the extended preprocessing pipeline model.

Thus we require that the names of the columns are standardized. Which standard to use in an implementation is up to personal preferences. Furthermore, it is important that each column describes exactly one relevant value and each relevant measurement time is represented by exactly one column.

Organizing data containing only TWS, TWA and BSP values in contrast can easily be done using a three-column array.

2.2 Data Handling

In order to estimate a polar performance diagram, one might use various different forms of time series recorded during sailing trips under real conditions. These data sources might include files in NMEA Standard [nme], csv files and other sources. During the data handling, each of this data sources is supposed to be translated into the standardized data format of the framework. This step belongs to the preprocessing principle of *Data Integration*.

2.3 Data Imputation

This step is a well established preprocessing step (see for example [LGH12]). The different measurement equipment of the sailing vessel might send their information in different frequencies. Due to this and various other reasons, data obtained during the data handling might be incomplete. The imputation process is supposed to guarantee data without these gaps by removing records or interpolating missing data fields.

2.4 Smoothing

Often, measurements are only available in a certain precision. If this precision is very coarse, this might further pollute the data. Since, at this point, our data is given in the form of a time series and assume that the time series is continuous for most measurements. The precision error might be reduced by replacing the actual data with the graph of a suitable continuous function. A process of this type might be called *smoothing* and can be categorized as *Data Transformation* and *Data Cleaning* but can also be used for *Noise Identification* (see [SLH13]).

2.5 Expanding the data

The previous preprocessing tasks are suitable for data that has been obtained during a sailing trip and is thus given as a time series. However, for further processing it could be profitable to add data which is not specific for the observed vessel, such as weather data, geographic information, auxiliary attributes etc. The process of expanding the data is supposed to include a way to compute certain additional attributes from a given record. Most commonly, we use latitude, longitude and time information in order to estimate the respective conditions. These additional attributes are added to the current records for further processing. This process possibly requires further interpolation techniques.

2.6 Weighing and Filtering

The available sailing data does not necessarily reflect the data we want to process in order to obtain a polar performance diagram. This could be due to

1. Measurement errors,
2. relevant influences to the boat speed beyond wind,
3. the fact that real data contains many records which are not top-performance as needed for the polar performance diagram.

In order to cope with these problems, we evaluate the quality of the data points as a weight for each record and eliminate records with a bad quality (filtering). This process can be categorized as *Data cleaning* and *Data reduction*.

The presented framework contains three rounds of weighing and filtering in total.

The first round takes place even before the expansion of the data. The motivation of this step is to reduce the number of records to be expanded. This potentially leads to a relevant speed up. We call this round of weighing and filtering *Pre-Expander-Weighing* and *Pre-Expander-Filtering* respectively.

The second round of weighing and filtering takes place after the expansion of the data and before application of the influence model (see Section 2.7). The purpose of this round is to filter the data using the additional information, for example removing records with relatively bad weather. We call this round of weighing and filtering *Pre-Influence-Weighing* and *Pre-Influence-Filtering* respectively.

The third round of weighing and filtering, *Post-Weighing* and *Post-Filtering* is meant to weigh and filter points solely based on their TWA, TWS and BSP values after applying the influence model. This round should be used to filter points based on their consistency and/or density in the TWA, TWS, BSP plane, i.e. to remove outliers or to reduce the number of records in dense regions in the TWA, TWS plane.

The weighing and filtering can be categorized as *Data Reduction* and *Noise Identification*. Also, *Data normalization* should be regarded during these steps.

2.7 Influence Model

In most generality, we understand an influence model as a mapping of records containing various parameters to records only containing the values TWA, TWS and BSP in such a way, that the new records reflect the relationship of TWA, TWS and BSP under idealized conditions as good as possible. A sophisticated influence model might work with an analytical or numerical modelling of additional influences from current, waves etc.

We allow parameterized influence models. This allows, for example, for influence models which detect and correct structural measurement errors or the possibility to adjust some parameters of the model according to the observed data.

The parameters of the influence model are fitted before the application using only the training data (so the parameters are chosen independently from the test data). On the other hand, the influence model with fitted parameters is applied to both, training and test data.

2.8 Injecting Data

From the physics of sailing and polar diagrams which have been obtained via VPP, we have additional knowledge on the properties of polar performance diagrams. This knowledge can partly be modelled as additional data points which are added to the current data during this step.

These artificial data points are only added to the training data since they would unnecessarily pollute the test data.

It seems reasonable to use quality metrics which tests if the additional knowledge is represented in the resulting polar performance diagram.

3 Realisations

In this section we discuss the specific framework components which have been used in the study case in Section 4. This is meant to function as an illustrative example how the different parts of the framework might be realised, further explanation of the study case and as dicussion of possible issues and how they might be addressed. Note, that the modular form of this framework allows to compare the effects of exchanging different parts in the future.

The overall concept of our realisation is, to use the additional weather information exclusively for the pre-influence-weighing and not in the influence model. Furthermore, we formulated the parts in a very simplistic way, such that they still have the required effect. Note, that more sophisticated realisations should yield better results.

3.1 Data handling

We use several sailing time series given as files in NMEA 0185 standard. Such a file can be parsed line by line (sentence by sentence) in order to get different attributes for the standardized data. We interpret consecutive NMEA sentences containing distinct attributes as one record. Otherwise, the data would be too fragmented.

One difficulty is the way that the NMEA standard handles wind data (wind speed and wind angle). To be precise, the relevant NMEA attributes are “Wind angle”, “Wind speed” and “Reference” describing apparent wind or true wind depending on the value of “Reference”. If we would simply adopt the NMEA attributes, there would be columns referring to multiple attributes contradicting the principles discussed in Section 2.1. This problem is easily addressed by filling two columns TWA and TWS depending on the value of “Reference”.

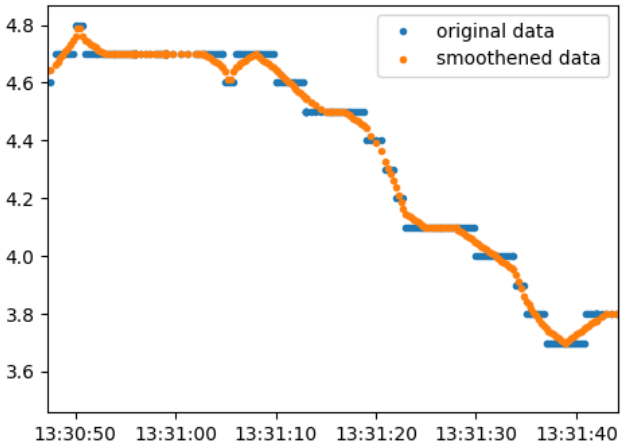

Figure 4: Caption

3.2 Imputator

In order to impute the missing data fields, we first delete all columns and rows that do not contain any information. Next, we find the first and last record containing a timestamp and delete all records before and after. Then we interpolate the timestamps since we need them for further processing. If two timestamps are too far apart (more than 2 minutes) we delete all records in between, since we assume that the interpolation error is too high in these cases. The remaining gaps are filled equidistantly.

The other fields are then imputed by affinely interpolating the data points time series.

3.3 Smoother

Time series measured with coarse precision tend to be composed of several time intervals with constant values. It seems plausible, that in between these intervals the data changes continuously in real life. In our tests we only kept the original data of the middle of the intervals and connected them via affine interpolation on the borders (30 seconds after first and before last record). An example plot for the smoothing procedure is given in Figure 4.

3.4 Pre-Expander-Weighing and Filtering

A polar performance diagram should only reflect the top performance and not the time intervals of acceleration and decelaration. We weigh the points according to the local fluctuation of the attributes TWA, TWS and BSP, roughly

following ideas from [Sim17].

For a fixed attribute and a given record, we consider the standard variation σ of all records that where measured at most 10 seconds before and at most 30 seconds after the examined data point. For a given upper bound γ on the standard variation, the weight is then computed by a mirrored and stretched ReLu function, i.e.

$$\sigma \mapsto \begin{cases} 0 \text{ if } \sigma > \gamma \\ (\gamma - \sigma)/\gamma \text{ otherwise} \end{cases} .$$

Thus, a standard variation of 0 will get a weight of 1 while standard variations exceeding γ will get a weight of 0.

We choose an upper bound of $\gamma = 20$ for TWA and upper bounds of $\gamma = 2$ knots for TWS and BSP since it seems reasonable that a variance exceeding these values is "too fluctuant". We then multiplied the respective weights and removed all the worst 60% of records.

3.5 Expander

We expand our data using weather data from meteostat [met] and the DWD [dwd] made available by the Leibniz Institute for Baltic Research Warnemünde. The weather data has been given as gridded data obtained via the meteostat Python package or given as a file in the NetCDF data format. In order to estimate the data at a given time and location, we use iterative affine interpolation.

3.6 Pre-Weighing and Filtering

This round of weighing and filtering is aimed on excluding records with unusual weather conditions. In order to do so, we use concepts from fuzzy logic (see for example [BB95]). In particular, we use the sigmoid function

$$x \mapsto \frac{1}{1 + e^{\sigma \cdot \alpha \cdot (x-c)}}$$

to define the truth value of $x \leq c$ ($\sigma = 1$) and the truth value of $x \geq c$ ($\sigma = -1$) respectively. We chose a slope coefficient of $\alpha = 10$ since then the "range of uncertainty" (length of the interval where the value is between 0.1 and 0.9) is less than 0.5 which seems reasonable. Furthermore, we model the boolean functions *and* and *or* using the minimum and the maximum respectively. Using this modelling, we check for the following conditions:

1. The Temperature is between $10C^\circ$ and $30C^\circ$,
2. The gusts are slower than $50km/h$ (≈ 21.6 knots),
3. The TWS is above 5 knots,
4. The meteostat weather condition code is smaller than 4.5 indicating clear, fair, cloudy or overcast weather,

5. The wave heights are lower than 1.5 meter.

We then remove all records with a truth value/weight of lower than 0.3.

3.7 Influence Model

We use a very simple influence model which is aimed to correct a structural measurement error in TWA. The underlying method assumes, that data in almost each direction relative to the wind has been recorded with the exception of the direction against the wind (0°). In the fitting phase of the influence model, we approximate a local density for a given TWA value wa and an interval length l (we use $l = 30$) by using Gauss-kernels as

$$\mathrm{dens}(wa) = \sum_{wa', ws' \in TWA, TWS} ws' \cdot e^{-\left(\frac{|wa - wa'| \pmod{360}}{l}\right)^2}.$$

We assumed that the wind angle with the lowest local density is the actual zero and corrected the data respectively.

That is to say, for a fitted *assumed real zero angle* of wa_z, the influence model is equivalent to the mapping

$$TWS, TWA, BSP, \text{ other data} \mapsto TWS, TWA - wa_z \pmod{360}, BSP$$

3.8 Post-Weighing and Filtering

We assign weights to a data point $d = (tws, twa, bsp)$ as follows. We define the boat speeds in a cylinder (for $r = 0.05$) over this data point by

$$C = \left\{ (bsp') \ : \ (tws', twa', bsp') \text{ is a data point and } \left(\frac{tws'}{40}\right)^2 + \left(\frac{twa'}{360}\right)^2 \leq r^2 \right\}.$$

Note that in this definition the true wind speed has been divided by 40 and the true wind angle is divided by 360 for normalization purposes.

We calculate the mean m over C and set the deviation error of the given data point as $e(d) = |m - bsp|$.

We define the weights by normalizing the deviation error with respect to the maximum and taking the difference to 1, i.e. the weight of d is defined to be

$$1 - \frac{e(d)}{\max(\{e(d') \ : \ d' \in D\}}.$$

We then filter the worst 10% according to these weights as outliers.

3.9 Injector

It is common knowledge amongst sailors that almost all sailing vessels can not sail directly against the wind. This should be reflected by the respective polar performance diagram. In order to get the processing method to produce a performance diagram accordingly, we extend the data by data points $(tws, 0, 0)$ and $(tws, 360, 0)$ where we choose for tws a total of 5‘000 equidistantly distributed points between the minimal and the maximal boat speed present in the data.

An alternative possibility is to choose a processing method that forces the polar diagram to be zero against the wind. However, the method used allows us to use a standard processing method and additionally lets the injected points influence the approximations on other boat speeds to obtain a “smoother” polar performance diagram.

3.10 Quality Assurance

One obvious method to measure the quality of the resulting polar performance diagram is to consider the average quadratic error when applied to the test data. We also want to weigh in the behaviour of the polar diagram around zero. To this extend, we also consider the average quadratic boat speed value for points at $TWA \in \{0, 360\}$ and TWS taken equidistantly between 0 and 20 knots. In order to carefully check the quality of the test data, we used 2 quality metrics. The first metric is the “test covering”, which we define as the number of tuples $(ws, wa) \in \mathbb{N} \times \{0, \dots, 359\}$ such that there exists a data point (ws', wa') with $|ws' - ws| \leq 0.5$ and $|wa - wa'| \pmod{360} \leq 0.5$. The second used metric for the pre-processed test data is the local result difference, i.e. the difference of the maximal and minimal boat speed at records with the same rounded TWS and TWA values. Lastly, we also measure the execution time.

4 Study case

For our study case we gathered wind and sailing data on a private sailing vessel of type *Hiddensee* during a journey in August 2020 from Rostock to Bornholm and back via Sweden. Yachts of these type are build very uniquely and no common polar performance diagram is known. Therefore it is an interesting object for our study. This data has already been used in [DSS22] for a proof of concept, but no quality metric has been measured there. Therefore, we can not compare our results.

Furthermore, we updated the `hrosailing`-Python package [Dan] developed at our institution in order to include the presented preprocessing framework. The following has been obtained using `hrosailing` version 0.10.1. The used source code can be found in the appendix and the used data files can be found at https://github.com/hrosailing/vela-trip-2021.

For the actual processing, we use a standard procedure which takes the arithmetic mean of nearby data points on a predefined grid. The resulting polar diagram is depicted in Figure 5. Note that due to the simplicity of the

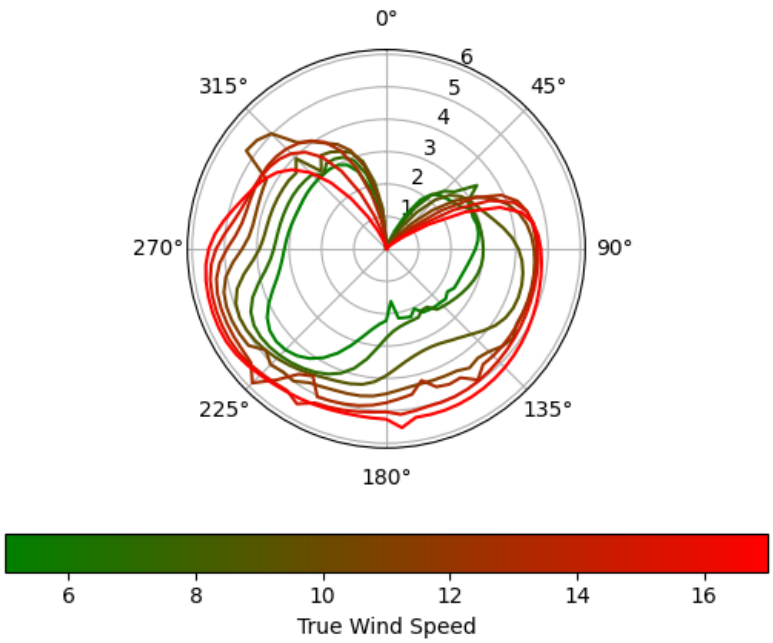

Figure 5: A plot of the resulting polar diagram.

components used, the presented results most probably can be improved largely by using more sophisticated parts of the framework.

Using the preprocessing techniques defined in 3, we obtain the following quality metrics:

Average quadratic error	4.844
Average quadratic value at zero	0.733
Test covering	474
Local test data difference	0.556
Execution time	02:43 hours

These quality metrics seem reasonable for a first benchmark example.

5 Conclusions

The presented preprocessing framework is a refinement of the preprocessing framework discussed in [DSS22] conserving the general modular structure and its benefits. Additional benefits of the refinement are

1. Additional data, as weather data, which has not been recorded during the trip can be taken into account,
2. It is possible to measure and compare the quality of the resulting polar performance diagram using test data,
3. Parametrized influence models can adapt their models according to the training data yielding an interface for machine learning approaches,
4. A distribution of the data into multiple data sources is supported.
5. Data recorded during a sailing trip can easily be handled as time series.

In Section 4 we discussed how the quality of the resulting polar performance diagram can benefit from suitable choices of components of the framework. We assume that better suited choices of the framework components one will lead to better results. Thus, a more detailed analysis of the different components and a broader study using several different data sets is desirable. On a technical side, the computations done during this study took quite some time. There is some obvious potential for significant speedups using parallel computing and relational data-bases.

References

[BB95] George Bojadziev and Maria Bojadziev. *Fuzzy sets, fuzzy logic, applications*, volume 5. World scientific, 1995.

[Dan] Dannenberg, Valentin and Schüler, Robert. hrosailing version 0.10.1. `https://github.com/hrosailing/hrosailing`, released: 2023-02-20, doi: 10.5281/zenodo.6322872.

[DSS22] Valentin Dannenberg, Robert Schüler, and Achill Schürmann. A data processing framework for polar performance diagrams. *Applied Sciences*, 12(6):3085, 2022.

[dwd] Deutscher wetterdienst - wetter und klima aus einer hand. `https://www.dwd.de/EN/Home/home_node.html`, Accessed: 14. November 2022.

[GLH15] Salvador García, Julián Luengo, and Francisco Herrera. *Data preprocessing in data mining*, volume 72. Springer, 2015.

[LGH12] Julián Luengo, Salvador García, and Francisco Herrera. On the choice of the best imputation methods for missing values considering three groups of classification methods. *Knowledge and information systems*, 32(1):77–108, 2012.

[met] Meteostat - the weather's record keeper. `https://meteostat.net/en/`, Accessed: 14. November 2022.

[nme] NMEA 0183 Interface Standard. https://www.nmea.org/content/STANDARDS/NMEA_0183_Standard, Accessed: 26. October 2022.

[orc] Orc world leader in rating technology. https://www.orc.org, Accessed: 26. October 2022.

[Sim17] Stefan Simon. Generating custom-fit polar diagrams from performance measurements on sailing yachts. Master's thesis, Graz University of Technology, 2017.

[SLH13] José A Sáez, Julián Luengo, and Francisco Herrera. Predicting noise filtering efficacy with data complexity measures for nearest neighbor classification. *Pattern Recognition*, 46(1):355–364, 2013.

Acknowledgements

The author likes to thank Ulf Gräwe from the Leibniz Institute for Baltic Research Warnemünde for providing valuable data.

Appendix

You need `hrosailing 0.10.1` and dependencies in order to run this code.

```
from hrosailing.pipelinecomponents import (
    NMEAFileHandler, WeatherExpander, AffineSmoother,
    FuzzyWeigher, FuzzyVariable, FluctuationWeigher,
    QuantileFilter, WindAngleCorrectingInfluenceModel,
    CylindricMeanWeigher, ComformingQualityAssurance,
    ZeroInjector, BoundFilter, FillLocalImputator
)
from hrosailing.cruising.weather_model import (
    NetCDFWeatherModel, MultiWeatherModel, GriddedWeatherModel
)
from hrosailing.pipeline import PolarPipeline
from hrosailing.pipeline.extensions import TableExtension

from datetime import timedelta
import os
import matplotlib.pyplot as plt

# prepare sailing files
dir_path="src/sailing_data"
sailing_files = [
    f"{dir_path}/{file_name}"
    for file_name in os.listdir(dir_path)
    if "FullSails" in file_name
```

```
]

test_files = sailing_files[0:2]
training_files = sailing_files[2:]

# Define all pipeline components
data_handler = NMEAFileHandler()

imputator = FillLocalImputator()

smoother = AffineSmoother()

pre_expander_weigher = FluctuationWeigher(
    timespan=(
        timedelta(seconds=30),
        timedelta(seconds=10)
    ),
    dimensions=["TWA", "TWS", "SOG"],
    upper_bounds=[20, 2, 2]
)

pre_expander_filter = QuantileFilter(60)

weather_model = MultiWeatherModel(
    NetCDFWeatherModel(
        "src/weather_data/WB600m.nc4",
        aliases={"lat": "latc", "lon": "lonc", "datetime":"time"},
        further_indices={"level": 0}
    ),
    GriddedWeatherModel.from_file(
        "src/weather_data/mwm.json"),
    exception_sensitive=True
)

expander = WeatherExpander(weather_model=weather_model)

x = FuzzyVariable(0.1)

pre_influence_weigher = FuzzyWeigher(
    (x["coco"] < 4) & (x["temp"] > 10) & (x["temp"] < 30)
    & (x["gust"] < 50) & (x["TWS"] > 5) & (x["waveH"] < 4)
)

pre_influence_filter = BoundFilter(0.5)

influence_model = WindAngleCorrectingInfluenceModel()
```

```
post_weigher = CylindricMeanWeigher()
post_filter = QuantileFilter(10)

injector = ZeroInjector(5000)

extension = TableExtension()

quality_assurance = ComformingQualityAssurance()

# setup pipeline
pipeline = PolarPipeline(
    data_handler=data_handler,
    imputator=imputator,
    smoother=smoother,
    pre_expander_weigher=pre_expander_weigher,
    pre_expander_filter=pre_expander_filter,
    expander=expander,
    pre_influence_weigher=pre_influence_weigher,
    pre_influence_filter=pre_influence_filter,
    influence_model=influence_model,
    post_weigher=post_weigher,
    post_filter=post_filter,
    injector=injector,
    extension=extension,
    quality_assurance=quality_assurance
)

# setup result files
log_path = "results/quality_metrics.log"
with open(log_path, "w", encoding="utf-8") as file:
    file.write("")

# process data
out = pipeline(
    training_data=training_files,
    test_data=test_files
)

stats = out.training_statistics.quality_assurance

#write logs
with open(log_path, "a", encoding="utf-8") as file:
    for key, value in stats.items():
        file.write(f"{key} : {value}\n")
    file.write("\n\n")
```

```
#save polar diagram

out.polardiagram.to_csv(f"results/vela.pd")

#plot polar diagram

ax = plt.subplot(projection="polar")

out.polardiagram.plot_polar(
    ax=ax,
    show_legend=True,
    legend_kw={"location" : "bottom"}
)

plt.show()
```