UNIVERSIDAD AUTÓNOMA DE MADRID

ESCUELA POLITÉCNICA SUPERIOR

MASTER THESIS

PROPOSAL AND EVALUATION OF THE MACHINE LEARNING MODELS FOR CORRECTING ERA5 STRESS EQUIVALENT WIND FORECASTS AS A FUNCTION OF ATMOSPHERIC AND OCEANIC CONDITIONS

Master in Data Science

Author: MAKAROVA, Evgeniia
Advisor: PORTABELLA ARNÚS, Marcos
Lecturer: DORRONSORO IBERO, José Ramón
Department: Ingeniería Informática

November 2022

PROPOSAL AND EVALUATION OF THE MACHINE LEARNING MODELS FOR CORRECTING ERA5 STRESS EQUIVALENT WIND FORECASTS AS A FUNCTION OF ATMOSPHERIC AND OCEANIC CONDITIONS

Author: MAKAROVA, Evgeniia
Advisor: PORTABELLA ARNÚS, Marcos
Lecturer: DORRONSORO IBERO, José Ramón

Ingeniería Informática
Escuela Politécnica Superior
Universidad Autónoma de Madrid

November 2022

Abstract

Abstract — This work aims at creating a preliminary machine learning (ML) model for correcting the European Centre for Medium-Range Weather Forecasts (ECMWF) ERA5 reanalysis stress-equivalent local wind biases, based on atmospheric and oceanic parameters. Several errors in the ECMWF global output for near surface ocean winds have been reported when validated against scatterometer observations. An existing approach for correcting these biases (the so-called ERA* method) consists of scatterometer-based corrections accumulated over a certain time window at each grid point, which allows to reduce local persistent biases. This approach is sensitive to scatterometer sampling and, to collect a statistically significant number of samples, assumes that such biases are static. This is not the case for errors due to moist convection or the diurnal cycle. For operational purposes, the temporal window is lagged with respect to the reanalysis forecast time and the time difference between scatterometer-based correction (SC) and sample data collections can be ten days.

We propose a preliminary ML setup that looks for the functional relationship between several oceanic and atmospheric variables that describe the persistent NWP errors as observed in the NWP-scatterometer differences. This would allow to predict the biases of the stress-equivalent wind forecasts and using the bias corrections in coupled weather or seasonal forecasts, or to account for these in climate runs. Such variables are first identified as ECMWF model parameters, such as stress-equivalent winds, their derivatives (curl and divergence), atmospheric stability related parameters, i.e., sea-surface temperature (SST), air temperature (Ta), relative humidity (rh), surface pressure (sp), as well as SST gradients and ocean currents. This work evaluates the feasibility of such approach and provides an overview of possible implementations of this regression.

Several ML algorithms are trained on a dataset that covers a period of 65 days and further evaluated. These algorithms include two libraries based on Gradient Boosting Decision Trees (GBDT), such as XGBoost and LightGBM, and feed-forward neural networks, implemented with the sklearn library (MLP Regressor) and with the Tensorflow and Keras API.

The models are trained to reproduce the differences between collocated scatterometer (ASCAT-A) and ERA5 U10S. The resulting models are further evaluated against a test dataset that covers a period of 23 days posterior to the training period. The best performing models are then further selected to generate the corrections for the entire ERA5 forecasts.

The corrected forecasts are then collocated with an independent scatterometer HSCAT-B that has a local pass time that differs 3.5 hours from that of ASCAT-A. Globally, the best performing model is a Tensorflow-based neural network with 4 hidden layers with 256, 128, 64, 32 neurons per layer, with dropout used for regularization. It shows a 5.54% of square error reduction globally, and in particular up to 7.66% in the extra-tropics, compared to ERA5 (test period). In the tropics and high latitudes, the error variance reduction is of 3.67% and 5.47%, respectively. This neural network setup outperforms the ERA* product in the extra-tropics and

high latitudes, although not in the tropics.

This work demonstrates that it is possible to reduce ERA5 local biases by using only NWP variables as model inputs, which makes this approach promising for operational setup purposes.

Key words — scatterometer-based corrections, ERA5 biases, Machine Learning, Ocean forcing

Acronyms

ADAM adaptive moment estimation.

API application programming interface.

ASCAT Advanced Scatterometer.

AWDP The ASCAT Wind Data Processor.

BUFR Binary Universal Form for the Representation of meteorological data.

CDO Climate Data Operator.

CMEMS Copernicus Marine Service.

CNN Convolutional Neural Network.

CPU central processing unit.

DL Deep Learning.

DNN Deep Neural Network.

ECMWF European Centre for Medium-Range Weather Forecasts.

EO Earth Observation.

ERA5 ECMWF Reanalysis v5.

EUMETSAT European Organization for the Exploitation of Meteorological Satellites.

FNN Feed-forward Neural Network.

GBDT Gradient Boosting Decision Trees.

GMF Geophysical Model Function.

GPU graphics processing unit.

GRIB General Regularly distributed Information in Binary form.

HSCAT-B Haiyang-2B scatterometer.

HY-2B Haiyang-2B.

ICM Institute of Marine Sciences.

IFS Integrated Forecasting System.

KNMI Royal Netherlands Meteorological Institute.

LightGBM Light Gradient Boosting Machine.

LST local solar overpass time.

MABL Marine Atmospheric Boundary Layer.

MARS Meteorological Archival and Retrieval System.

MetOp Meteorological Operational satellite.

ML Machine Learning.

MLP Multi-layer Perceptron.

MSE Mean Squared Error.

NetCDF Network Common Data Form.

NRT near real time.

NSOAS National Satellite Ocean Application Service.

NWP Numerical Weather Prediction.

OSCAT2 OceanSat Scatterometer.

OSI-SAF Ocean and Sea Ice Satellite Application Facility.

PenWP Pencil beam Wind Processor.

ReLU rectified linear unit.

rh relative humidity.

RMSE Root Mean Square Error.

RMSProp root mean squared propagation.

SC scatterometer-based corrections.

ScatSat-1 Scatterometer Satellite-1.

SDG stochastic gradient descent.

SGD stochastic gradient descent.

sp surface pressure.

SST sea surface temperature.

SVR Support Vector Regression.

Ta air temperature.

U10S stress-equivalent winds.

UAM Universidad Autónoma de Madrid.

VRMS Vector Root Mean Square difference.

WOC World Ocean Circulation.

XGBoost Extreme Gradient Boosting.

Contents

List of Tables

List of Figures

1 Introduction

Scatterometer-derived sea surface winds have been assimilated into the European Centre for Medium-Range Weather Forecasts (ECMWF) Integrated Forecasting System (IFS) for more than two decades. Scatterometers are radar (active microwave) instruments that measure the sea surface roughness, i.e., the microwave energy scattered from the ocean surface gravity-capillary waves, which are induced by the local wind. When the same ocean area is measured from three or more different look angles, it is possible to determine the wind speed and direction at the ocean surface through the numerical inversion of the Geophysical Model Function (GMF), i.e. the empirical relationship between the backscatter and the sea-surface wind vector [1].

Scatterometer winds are successfully assimilated in NWP after thinning them. Nevertheless, the assimilation of the Advanced Scatterometer (ASCAT onboard MetOp series) wind data into the IFS can be further improved, such as by using scatterometer wind products after spatial aggregation (so-called superobbing), applying flow-dependent errors, or refining the quality control [2]. Unfortunately, as reported by Belmonte and Stoffelen [3] and Trindade et al. [4], the Numerical Weather Prediction (NWP) wind output shows relatively large and persistent local biases over the ocean, which ideally should be corrected before data assimilation. Furthermore, biases need to be corrected since they mostly represent unresolved geophysical processes by NWP models.

A particularly relevant problem in NWP is the stability-dependent errors in the boundary layer formulation [5]. For example, as pointed out in [6, 7], in the ECMWF model formulation, momentum mixing seems too strong in the stable and neutral boundary layer and too weak under unstable conditions. Stress-equivalent ECMWF winds under stable conditions are thought to be too strong, under unstable conditions too weak, and this shows up as a residual wind speed bias when compared to scatterometer winds. Moreover, a lack of NWP cross-isobaric flow has been reported [6, 8], i.e., NWP wind directions are biased with respect to the observed winds with opposite sign in the Southern and the Northern Hemispheres, particularly in neutral and stable stratification.

Scatterometer-based corrections (SC) of the mentioned systematic and persistent effects present in the ECMWF ERA-Interim [4] and ERA5 [9] reanalyses output have been developed.

The method consists of accumulating the mean differences between scatterometer and model wind data over long enough periods of time to reduce sampling errors and well correct persistent NWP biases.

With this approach, scatterometer-based corrections (SC), using accurate, unbiased, high spatial resolution ocean vector winds from several scatterometers, i.e., the C-band Advanced Scatterometer (ASCAT) on board MetOp satellites and several Ku-band scatterometers such as the OSCAT2 scatterometer on board SCATSat-1, are developed. The NWP-corrected wind output by SC, the so-called ERA*, outperforms NWP (ERAi, ERA5) when compared against independent scatterometer data ([4], [10], [9]). In particular, a SC based on a 3-day time window of a combination of the 3 ASCATs (A, B, and C) and OSCAT2 leads to an overall 9% reduction of error variance in ERA*, with respect to that of ERA5.

A main extension of the above method, would be to attribute the biases to local processes or conditions, such that corrections may be computed for forecasts too. This is relevant to weather forecasting, seasonal forecasting and climate predictions, particularly when made with coupled atmosphere and ocean models. Furthermore, such attribution would aid research into improved parameterization of processes in the models. Finally, in case of only one scatterometer, a longer sampling period is needed to compute accurate local biases, while these will be somewhat less representative of the verification time where the corrections are applied. If the model marine boundary layer process errors were attributed by deep learning methods, then more representative and hence accurate local model errors may be computed and applied in data assimilation or coupling.

Deep learning methods [11] are currently being used to generate sea level forecasts, driven partly with direct sea level observations [12] or to provide better estimates of relevant variable fields [13, 14] which can be used to force numerical models. These methods are fast and numerically cheaper than data assimilation and have already been demonstrated to allow reliable reconstructions of sparse satellite measurements of sea surface temperature and sea level anomaly [13, 14].

We aim at a preliminary machine learning (ML) model set up, which uses scatterometer and ECMWF forecast fields of stress-equivalent winds (U10S) and derivatives (divergence and curl), atmospheric stability related parameters, i.e., sea-surface temperature (SST), air temperature (Ta), relative humidity (rh), surface pressure (sp), as well as SST gradients and ocean currents, to predict the SC, and correct for the mentioned NWP errors. This is particularly relevant for both atmospheric (e.g., NWP data assimilation) and oceanic (improved wind forcing) applications.

Several non-linear regression models are evaluated to find the relationship between these atmospheric (stability) and oceanic (SST gradients and currents) parameters and the SC, in order to optimize the correction of the NWP output errors. The atmospheric stability parameters are retrieved from the ERA5 reanalysis dataset [15], while the dataset for the currents is obtained through the Copernicus Marine Service (CMEMS). The regression does model the SC derived from geolocated differences between ASCAT-A scatterometer and ERA5 stress-equivalent winds. The evaluated models include ensemble algorithms based on Gradient Boosting Decision Trees (GBDT), such as XGBoost [16, 17] and LightGBM [18]. Another type of model implemented is based on Multi-layer Perceptron (MLP) neural network [19].

The NWP output is corrected using the range of selected models and the performance is then first assessed against the test dataset, i.e., against the same ASCAT-A scatterometer, and finally against an independent scatterometer, i.e., HSCAT-B, whose sun-synchronous orbit samples the

ocean surface at a different local solar time of the day than that of ASCAT-A.

In this work, we compare the performance of preliminary models trained on a small dataset (less than 3 months of data) to choose the optimal approach, which will be further sophisticated and trained over several years of data during the second part of this study.

2

State of Art

This chapter gives an overview of the applications of machine learning algorithms in Earth observation and remote sensing, as well as introduces the problem of ERA5 biases and the previous attempts to mitigate them. The section about scatterometer observations gives a brief description of the instrument that is used as ground truth for this work, to estimate the ERA5 biases.

2.1 Machine learning applied to Earth Observation

In the last decades, the amount of meteorological and Earth Observation (EO) data has been growing at an exponential rate, e.g., the growth rate of the ECMWF Meteorological Archival and Retrieval System (MARS) is about 45% per year [20]. In particular, satellite acquisitions, which provide imagery and measurements from various types of sensors, contribute the most to the mentioned data growth.

Due to the rising volumes of data, data-driven solutions have become more and more important in Earth Sciences, including weather forecasting and climate change studies. Machine Learning (ML) algorithms are becoming more extensively used for satellite image processing (segmentation [21] and object detection [22]), bias correction [23] and cross-calibration, time series prediction [12] and filling of the observational gaps [14].

2.2 Persistent biases in ECMWF ocean wind forcing output

Global Numerical Weather Prediction (NWP) model sea-surface wind output is commonly used to force ocean models for their time and space continuity. One of the most commonly used NWP models is the ECMWF reanalysis ERA5.

Several issues with global NWP output, and ERA5 in particular, have been reported [9].

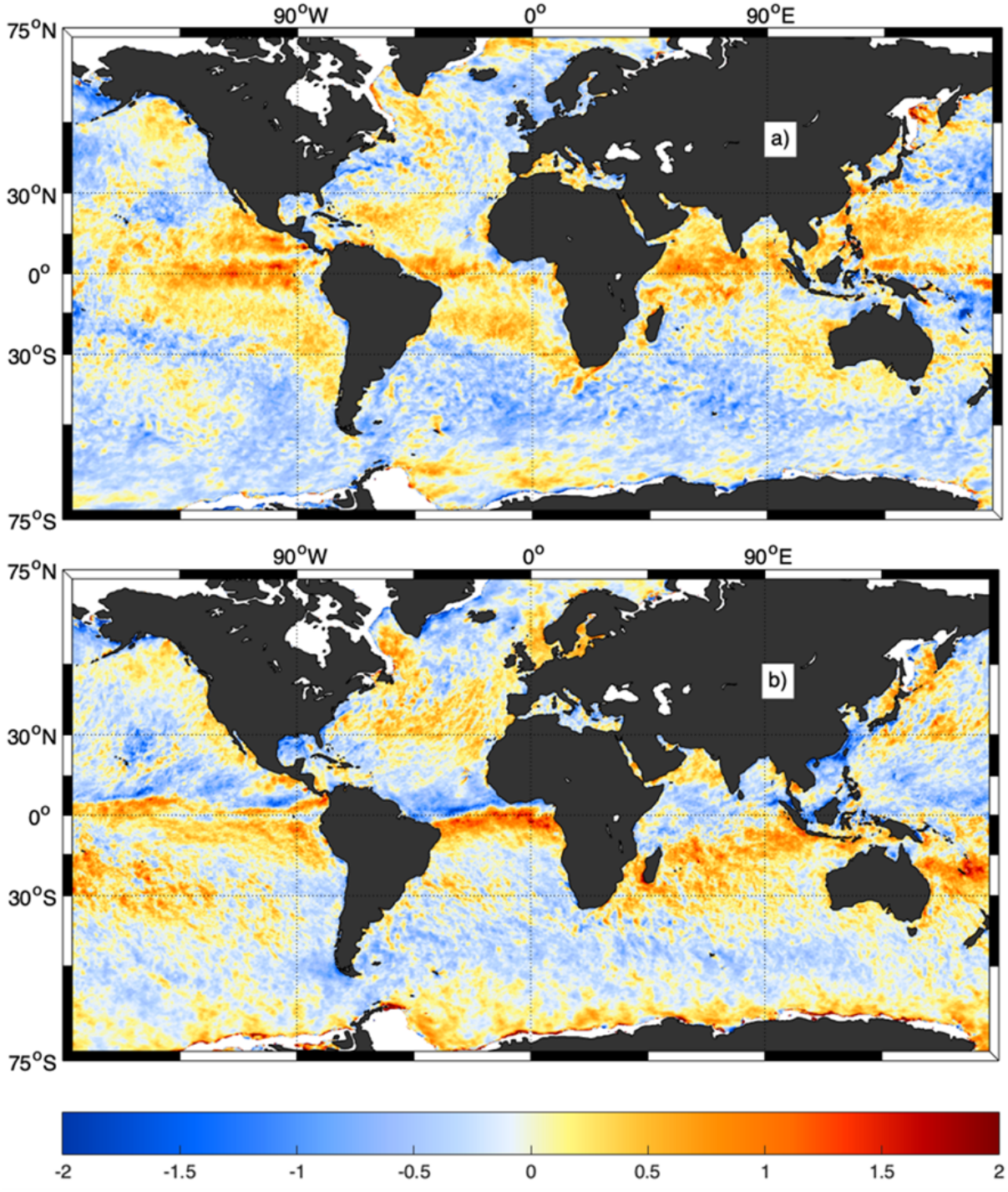


Figure 2.1: Differences between ASCAT-A/B/C and collocated ERA5 U10S for the zonal (a) and the meridional (b) wind components, accumulated over a 30-day temporal window centred on February 15th 2019 around 09 UTC. The colours represent the differences in m/s (see colour scale). Adopted from [9].

Belmonte et al. [3] reports large scale circulation errors for ERA5, such as excessive mean model westerlies in the middle latitudes that are connected with insufficient mean model meridional (poleward) flow between 30 and 60° in both hemispheres.

Sandu et al. [24] analyse the causes of wind direction biases of the ECMWF Integrated Forecasting System, which is considered one of the most prominent systematic errors in the near-surface weather forecasts. They bring up the already known issue of the forecasted surface wind direction being generally veered (rotated clockwise) with respect to observations in the Northern Hemisphere (NH) both over land and over the oceans, while in the Southern Hemisphere (SH) it is backed (rotated anticlockwise) with respect to observations.

Belmonte et al. [3] show that the local wind errors in ERA5 reanalysis are reduced by 20% as compared to the previous ERA-Interim version, while still present. The causes of the most common biases were summarized in [25]:

1. Air-sea interaction parameterization errors, e.g., the link between SST and wind gradients;
2. Marine Atmospheric Boundary Layer (MABL) closure, particularly the neutral and stable MABL;
3. Representation of mesoscale phenomena, such as moist convection [26];
4. Mesoscale dissipation of 3D turbulence; NWP model dynamical closure;
5. Ocean currents, particularly for the zonal (equatorial) component.

As a result numerical weather prediction can miss relevant ocean-atmosphere interactions at both the large scales and the oceanic mesoscale. These persistent errors result in systematic differences between scatterometer and NWP sea surface winds. These differences are referred to as persisting local biases, and can be seen when scatterometer and model winds are collocated [9]. Figure 2.1 represents an example of persistent local wind biases globally for the zonal (a) and the meridional (b) components, accumulated over 30 days (February 2019).

2.3 Scatterometer observations

Historically, the main part of the weather observations was collected over land and in highly populated areas. Wind data over the oceans mostly came from ships that normally follow established routes and avoid extreme weather. Another available source of continuous in-situ ocean observations are buoys that carry meteorological stations but their geographical distribution is highly sparse and uneven.

Over the last decades, remote sensing is being used to provide meteorological information over the global ocean, in particular with the use of various instruments on board satellites.

There are two types of sea-surface-wind remote-sensing instruments: passive and active microwave sensors [27]. The passive sensors measure the electromagnetic radiation coming from the Earth surface and the atmosphere and several meteorological parameters can be obtained from such measurements, including winds [1]. Passive microwave sensors (radiometers) measure the wind speed over the ocean by analysing the power spectrum of the electromagnetic radiation that is emitted by the wind roughened surface [28]. The surface roughness depends on the near surface stress-equivalent wind speed [29].

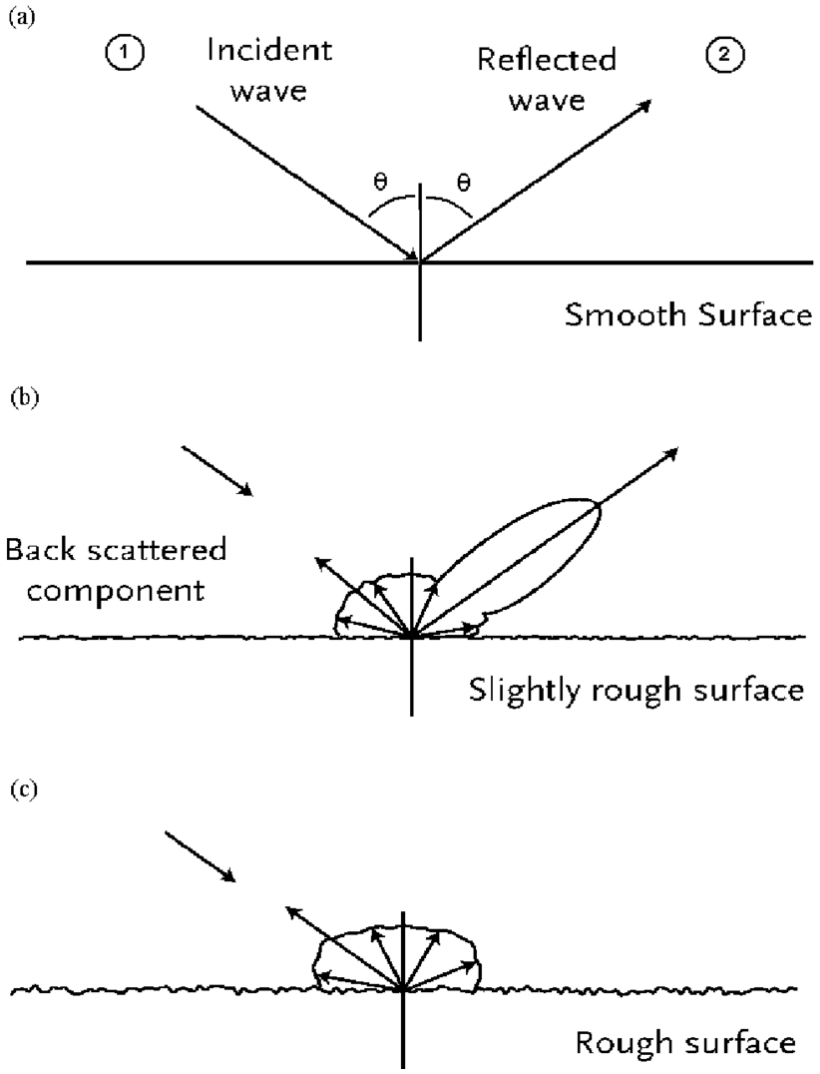


Figure 2.2: Schematic representation of the scatterometer acquisitions over the ocean surface. Adopted from [31].

The active microwave sensors (radars) emit electromagnetic radiation towards the Earth and measure the properties of the received back-scattered signal. Scatterometers measure the power returned from ocean surface roughness elements by Bragg resonance (figure 2.2). At the e.m. wavelength of the radar (5 cm for ASCAT) and incidence angle (ranging 25-65° for ASCAT), the backscatter radiation mostly comes from the gravity-capillary waves that are in equilibrium with the local wind stress. The measured backscatter depends on the magnitude of the stress magnitude and the direction relative to the radar beam [1, 30].

The stress-equivalent wind dataset that is used in this work is obtained from the European Organization for the Exploitation of Meteorological Satellites (EUMETSAT) Ocean and Sea Ice Satellite Application Facility (OSI-SAF) ASCAT-A coastal wind product. Advanced Scatterometer (ASCAT) is one of the instruments carried on-board the EUMETSAT Meteorological Operational satellite (MetOp) polar satellites. Metop-A, the first in a series of three satellites, was launched on 19 October 2006 [34] and decommissioned in November 2021.

Figure 2.3 represents the swath geometry of the ASCAT scatterometer and the position of two

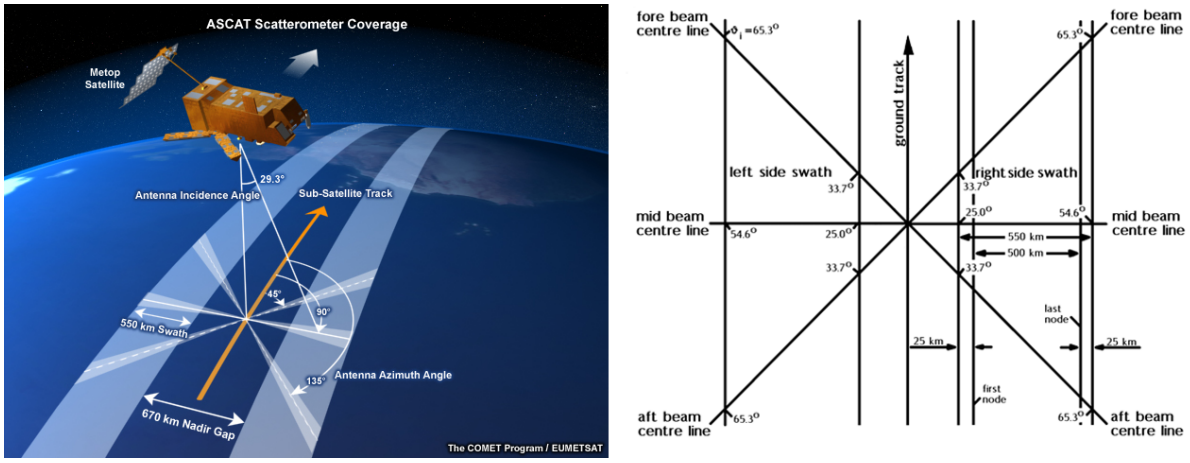


Figure 2.3: ASCAT-A swath geometry. Adopted from [32, 33]

sets of three antennas (45 degrees forward, sideways, and 45 degrees backwards with respect to flight direction) that are used to generate radar beams. The beams illuminate two approximately 550 km-wide swaths on both sides of the satellite ground track, separated by about 700 km. The measurements are operationally available with two grid sizes, i.e., 12.5 km (the so-called coastal) and 25 km [34]. The instrument operates at C-band frequency (5.255 GHz), which makes it rather insensitive to rain as compared to Ku-band instruments [23, 26].

2.4 ERA*

A data series correcting for local, persistent NWP stress-equivalent wind biases was produced in the framework of the World Ocean Circulation (WOC) project, which led to the generation of the so-called ERA* dataset [35], for the period 2010-2020. The ERA* product aims to correct persistent errors of ERA5 reanalysis with the use of the varying scatterometer constellation over time [4, 9].

The rationale of the method is that when the scatterometer wind data are accumulated over certain periods of time, it is possible to overcome sampling errors and maintain some of the scatterometers most beneficial features, i.e., those related to relatively small-scale ocean processes, such as wind-SST interaction and ocean-current relative winds, and furthermore, correct for the other small- and large-scale NWP parameterization and dynamical errors.

The correction is based on the temporally-averaged difference between scatterometer (u_{10s}^{SCAT}) and ERA5 stress-equivalent wind components (u_{10s}^{ERA5}), at grid point (i,j) and time sample (t), as described in equation 2.1 (same for the meridional component v) [9].

$$SC(i,j,t_f) = \frac{1}{M}\sum_{t=1}^{M}(u_{10s}^{SKAT_k}(i,j,t) - u_{10s}^{ERA5}(i,j,t)) \tag{2.1}$$

As such, for each t_f and fixed temporal window configuration (N), the collocated scatterometer/ERA5 fields within $t_f \pm N/2$ days are collected and the corresponding scatterometer/ERA5 differences averaged at each grid point. This leads to the generation of hourly (at each t_f) SC U10S fields for a predefined time window (N) and combination of

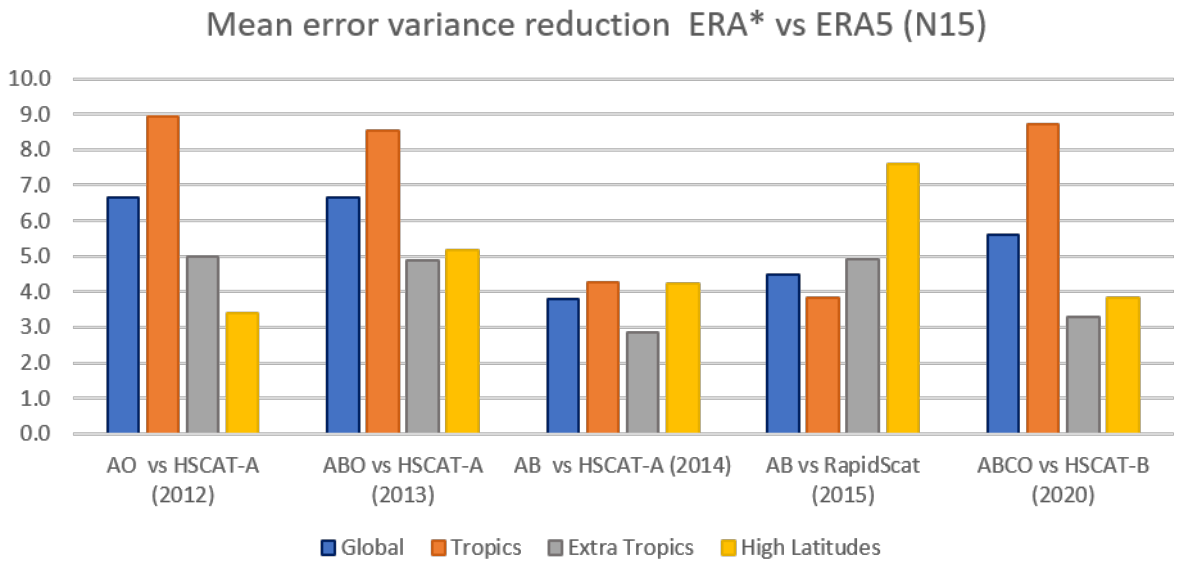


Figure 2.4: Mean error variance reduction (in percentage, with respect to that of ERA5) of the selected ERA* 15-day configurations against independent scatterometer data for global, tropical, extra-tropical, and high-latitude regions. Adopted from [9]

scatterometers (k).

Finally, the scatterometer corrections for the zonal and the meridional components, $SC(i, j, t_f)$, are added to the ERA5 U10S forecasts, as shown in equation 2.2.

$$u_{10s}^{ERA*}(i, j, t_f) = u_{10s}^{ERA5}(i, j, t_f) + SC(i, j, t_f) \tag{2.2}$$

ERA* outperforms NWP, and ERA5 in particular, when compared against independent scatterometer data [4, 9, 10, 36]. For example, the scatterometer-based corrections (SC) accumulated over a 3-day time window and a combination of 4 scatterometers, i.e., the Advanced Scatterometers ASCAT-A, -B, and -C (onboard Metop satellite series) and the OSCAT2 (onboard ScatSat-1), leads to an overall 9% reduction of the error variance in ERA*, with respect to that of ERA5.

The default configuration of the dataset, however, is based on a 15-day time window (N15), as it shows more stable behaviour for the 11-year dataset, accounting for the varying scatterometer constellation and the many, relevant data gaps of some of the operating systems. For the periods of reduced scatterometer sampling, the SC accumulated over shorter (e.g., 3-day) temporal windows results in a degradation of performance compared to that of ERA5. For larger (e.g., 15-day) temporal windows, when the constellation consists of 3 and more scatterometers and data gaps are absent, the ERA* performance is similar to that of shorter temporal window configurations. Figure 2.4 shows the reduction of ERA* error variance compared to that of ERA5, using several independent scatterometers as reference. For example, in 2020 the error variance reduction is 5.6% globally, 8.7% in the tropics, 3.3% in extra tropics and 3.8% in high latitudes.

While ERA* aims at removing local NWP biases, it is not always effective in doing so. In

particular, the effectiveness of the correction will strongly depend on scatterometer sampling at each particular grid point, while weather-dependent biases are not grid point dependent. A more functional relationship between atmospheric state variables, sea surface temperature (SST), ocean currents, and scatterometer versus NWP differences might lead to more effective SC and therefore improved NWP bias mitigation. More importantly, such correction may also be exploited in weather, seasonal and climate forecasting applications to correct for model errors.

2.5 ML applied to NWP bias corrections

Over the last years, machine learning and in particular Deep Learning (DL) techniques are often used to enhance NWP forecast accuracy. The most common models for NWP corrections are being developed for temperature [37] and precipitation, including nowcasting [38].

Models for wind bias corrections are mostly done for land observations, where the ground truth data comes from the meteorological observational stations and the correction depends on the local terrain roughness.

In [39], Tang et al. propose an adaptation of the LightGBM model to correct the prediction of temperature and wind from the ECMWF model. They use data from 18 observation stations in the Hainan Province (China) to calculate the corrections. They introduce the spatial LightGBM model that includes data from several neighbouring stations. In this work, the spatial LightGBM outperforms the standard LightGBM, as well as Random Forest and Gradient Boosting Decision Trees (GBDT).

Han et al. [40] propose a deep learning method called CU-net to correct the gridded forecasts of four weather variables from the ECMWF IFS: 2-m temperature, 2-m relative humidity, 10-m wind speed, and 10-m wind direction, with a forecast lead time of 24 h to 240 h in North China. CU-Net stands for correction U-net, a CNN-based network architecture normally used for image segmentation. In training, they use operational ECMWF IFS model as input and ERA5 reanalysis as ground truth.

Chan et al. [41] use XGBoost to correct the biases in the intensity of tropical cyclones, which is usually underestimated by the ECMWF forecasts. They demonstrate that their ML model is able to reduce the errors in typhoon intensities, despite being highly dependent on the quality of NWP predictors used for model input. Another consideration that they discuss concerns whether XGBoost model predictions are suitable for extremely violent typhoons.

Park et al. [42] apply a DNN to calibrate ASCAT-based wind speed in the Korean Sea using 10 buoy stations. In this case, they used the ML approach not for NWP bias correction, but for tuning the scatterometer 25-km footprint winds to the local buoys. They used wind speed, direction, location, date and time of the day as input for the neural network and fitted the model to the buoy-based wind speed. The DNN has 5 hidden layers with 256-128-64-32-16 nodes per layer. The DNN-based calibration showes lower RMSE compared to linear and SVR regressions.

Another recent work [43], published by the NVIDIA research group, shows that it is possible to substitute the ECMWF IFS with its underlying physics by FourCastNet (Fourier Forecasting Neural Network), a DNN based on Adaptive Fourier Neural Operator (AFNO). They claim that the trained model produces forecasts 45,000 times faster than the traditional NWP approach with the forecast accuracy comparable to that of the current IFS.

2.6 SC generation for NWP using ML models

In this work, we aim to create a preliminary regression model that links ERA5 model winds and atmospheric stability and oceanic related variables with ERA5 model biases for the zonal and the meridional stress-equivalent wind components.

ERA5 U10S biases can be detected comparing the reanalysis data with the scatterometer observations. In this work, we focus on the ASCAT-A dataset described in section 2.3. The model data is collocated with the ASCAT-A data through temporal and spatial interpolation and the corresponding differences are calculated. The resulting regression should be able to model such differences between ERA5 and scatterometers in the absence of the observational data, only by using other NWP model variables as input. The resulting ML model should represent the functional relationship between the NWP atmospheric stability and oceanic related variables and ERA5 local biases to improve ocean wind forcing predictions.

As compared to the first version of ERA*, which required accumulated scatterometer observations over a period of several days, or even weeks, as well as centered time windows, which makes it difficult to implement in the operational forecasting, the resulting model should be able to predict the NWP biases to improve operational NWP ocean forcing forecasts.

This works focuses on testing of the feasibility of such approach. The models under evaluation are trained only on a small set of data. We try different machine learning algorithms to check which are the most suitable for the task to finally choose the one with the best performance and most adequate for this specific goal (i.e., ocean forcing prediction improvement). The resulting model will be further sophisticated and trained over a larger dataset of several years in the future.

3 Datasets

This chapter describes the datasets that are being used in this work. The model data, such as ERA5, currents and variables derived from them are used as the ML model input features. The scatterometer data (we focus on ASCAT-A only in this work) is used to estimate the ML model target (ERA5 biases). The last section introduces the independent scatterometer dataset that is used for validation purposes.

3.1 ASCAT-A scatterometer data

The ASCAT-A scatterometer stress-equivalent wind data are used to determine the ERA5 biases, and are therefore used as the ML model target. The modelled output is the difference between the scatterometer measurements and ERA5 stress-equivalent winds.

The OSI-SAF 12.5-km ASCAT-A scatterometer U10S data are available for the period between January 2007 and November 2021. We use the period 02 January - 01 April 2020 data for training and initial evaluation of the model, while in the continuation of this project the best performing model will be trained and evaluated over a much longer period of 5 years, i.e., 2017-2021.

The scatterometer raw data are processed at Royal Netherlands Meteorological Institute (KNMI) and provided on a Level-2 swath grid and then reprocessed at Institute of Marine Sciences (ICM).

Figure 3.1 shows an example of ASCAT-A descending swaths for a given day. Stress equivalent scatterometer winds are available both in cartesian (zonal and meridional or u and v components, in m/s) and polar (wind speed in m/s and wind direction in degrees) coordinates, as shown in Table 3.1.

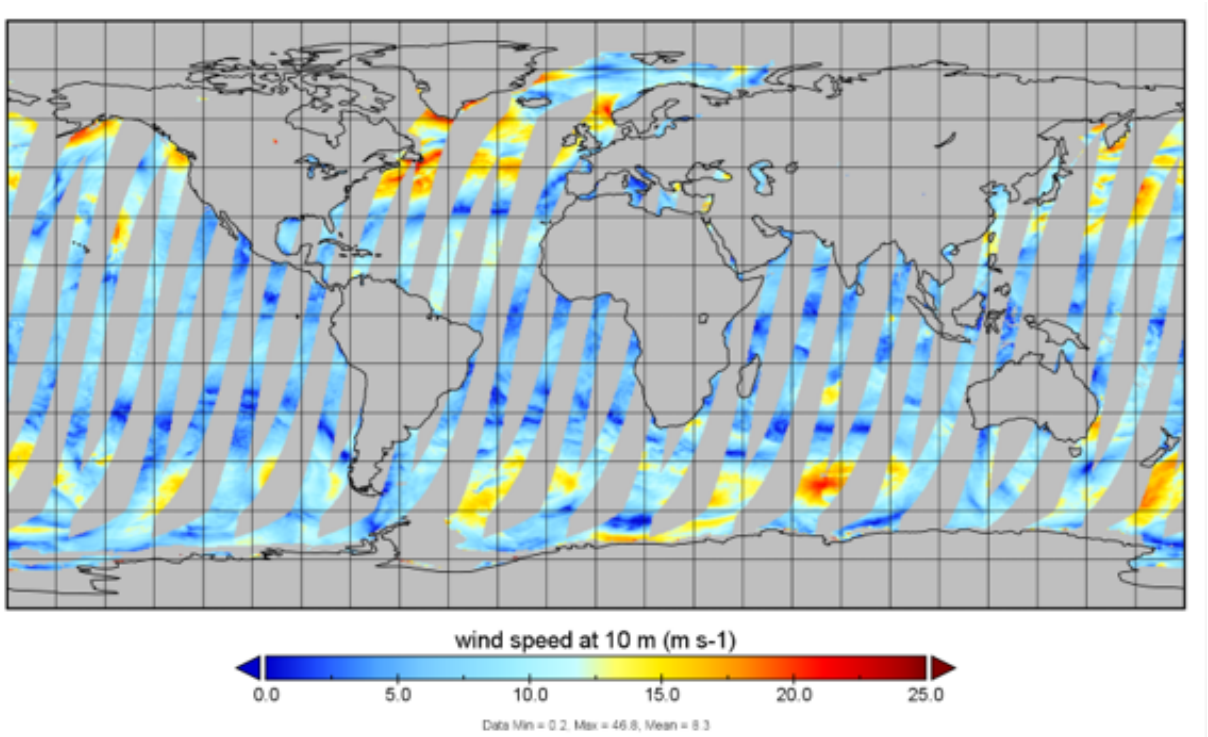


Figure 3.1: ASCAT-A descending swaths for February 8th, 2020

3.2 ERA5 model data

The Numerical Weather Prediction (NWP) model data that is used as the input for the regressions comes from ERA5 reanalysis dataset [44]. We choose ERA5 reanalysis instead of the operational historical forecasts as it uses the same version of the atmospheric model for all the years of the dataset, while the characteristics of the operational forecasts change from year to year.

ERA5 reanalysis data has 2 analysis cycles, at 06 and 18 hours UTC. At the analysis time, the model assimilates the observational data into the current state of the system and runs the numerical model to generate the forecasts of the future state of the system. ERA5 generates forecasts with hourly time resolution and only those forecasts ranging from +3h up to +18h from the analysis cycle time are used. In case of overlapping forecasts from two different analysis cycles, we use the closest ones for training. sea surface temperature (SST) data is available only once per analysis cycle and is therefore provided twice per day.

ERA5 datasets have a 30-km grid and 1 hour temporal resolution. The datasets are available for download through the ECMWF Meteorological Archival and Retrieval System (MARS).

The ASCAT-A and ERA5 variables used in the model are listed and briefly described in table 3.1.

Note that the ASCAT-A dataset includes collocated ERA5 U10S as background model wind, which is used as wind reference for the processing and quality assurance. The collocated stress-equivalent background wind is calculated from the ERA5 neutral 10-meter winds using an equation 3.1, where ρ_{air} is the local air density and $< \rho_{air} >$ is the average global air density

taken as 1.225 kg/m^3 [29].

$$U10S = U10N\sqrt{\frac{\rho_{air}}{<\rho_{air}>}} \tag{3.1}$$

3.3 Currents

The ocean surface currents dataset is obtained through Copernicus Marine Service (CMEMS) [45], [46]. We use the daily means for model input to avoid additional noise in the data. Currents are more persistent over time than winds or other atmospheric variables, and thus hourly changes are less significant (besides tidal currents in the coastal zones that are not included in this study).

The original dataset includes surface currents at 0 and 5 m depths, but we only include currents at 0 meter depth as they have more influence on the atmospheric conditions.

The dataset includes the zonal and the meridional velocities of the current (uo, vo in m/s) at 1/4° regular grid.

Source	Variable short name	Variable long name	Units
Common	lat	Latitude	degree
Common	lon	Longitude	degree
KNMI	wind_speed	ASCAT-A stress equivalent wind speed at 10 m	m s-1
KNMI	wind_to_dir	ASCAT-A wind direction at 10 m	degree
KNMI	eastward_wind	ASCAT-A stress equivalent wind u component at 10 m	m s-1
KNMI	northward_wind	ASCAT-A stress equivalent wind v component at 10 m	m s-1
ERA5	se_model_speed	Stress equivalent model wind speed at 10 m	m s-1
ERA5	model_wind_to_dir	Model wind direction at 10 m	degree
ERA5	se_eastward_model_wind	Stress equivalent model wind u component at 10 m	m s-1
ERA5	se_northward_model_wind	Stress equivalent model wind v component at 10 m	m s-1
ERA5	msl	Mean sea level pressure	Pa
ERA5	air_temperature	Aire temperature	K
ERA5	q	Specific humidity	kg kg-1
ERA5	sst	Sea surface temperature	K
CMEMS	uo	Eastward sea water velocity at 0 m	m s-1
CMEMS	vo	Northward sea water velocity at 0 m	m s-1

Table 3.1: Variables used in models and their origin

Variable short name	Variable long name	Units
se_model_wind_divergence	Model divergence of stress equivalent wind at 10m	s-1
se_model_wind_curl	Model rotation of stress equivalent wind at 10m	s-1
sst_dx	SST zonal gradient	K m-1
sst_dy	SST meridional gradient	K m-1
current_speed	Sea water velocity at 0 m	m s-1
cos_sst_grad	Cosine between SST gradient vector and stress-equivalent model wind	NA
cos_currents	Cosine between sea current vector and stress-equivalent model wind	NA

Table 3.2: Additional variables selected as possible ML model inputs

3.4 Wind derivatives and additional input variables

Several input features that may potentially improve ERA5 U10S output corrections are calculated from some of the variables in the original datasets. The full list is represented in the table 3.2.

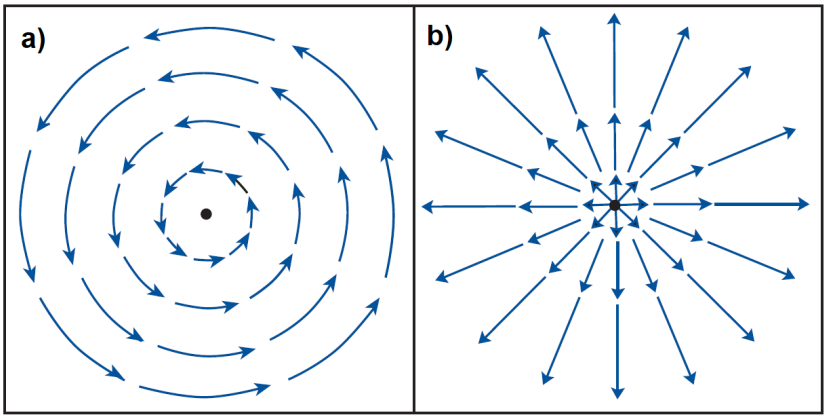


Figure 3.2: Example of a flow with positive curl (a) and divergence (b). Image adopted from [47]

Based on the ERA5 stress-equivalent wind components, we calculate U10S vorticity (curl) 3.2 and divergence 3.3. The equations are adapted to the Cartesian coordinate system, where the wind derivatives are calculated with respect to the x (zonal) and y (meridional) axes.

$$\zeta = \nabla \times \vec{v} = \frac{\partial v}{\partial x} - \frac{\partial u}{\partial y} \tag{3.2}$$

$$\delta = \nabla \cdot \vec{v} = \frac{\partial u}{\partial x} + \frac{\partial v}{\partial y} \tag{3.3}$$

The vorticity characterizes the rotational movements or shear in the air, and in case of a two-dimensional flow, the vorticity vector is a scalar rotational quantity perpendicular to the ground. The vorticity is positive for counter-clockwise rotation (cyclonic in northern hemisphere and anticyclonic in the southern hemisphere) [48]. The divergence of a vector field is positive when it behaves as a source. When the flux is directed inward the divergence is negative and called

convergence. Figure 3.2 illustrates idealized flows with positive wind vorticity (a) and divergence (b).

Besides derivatives from the ERA5 U10S fields, we also include as input to the model the sea surface temperature (SST) gradients. Xin Jin et al. in [49] show the existence of a statistical relationship between surface wind stress and SST gradients. We thus calculate the corresponding zonal and meridional gradients (i.e., sst_dx, sst_dy) and the cosine of the angle between the SST gradient direction and the ERA5 stress-equivalent wind direction (cos_sst_gradient).

Finally, we derive two additional input features from the ocean currents dataset, i.e., the surface sea velocity norm (current speed) and the cosine between the current vector and the ERA5 stress-equivalent wind vector (cos_currents).

3.5 HSCAT-B dataset (validation)

We use the HSCAT-B scatterometer data for independent verification purposes (see verification methodology description in section 5.4).

The HSCAT-B on board the HY-2B satellite has a local pass time that differs about 3.5 hours from that of ASCAT-A. It passes over about 3.5 hours earlier or 8.5 hours later. The HSCAT-B is a Ku-band (13.256 GHz) conically scanning scatterometer with two beams sweeping at incidence angles of 41° (horizontally polarized) and 48° (vertically polarized) in swaths of 1350 and 1750 km, respectively [50].

The Chinese National Satellite Ocean Application Service (NSOAS) HSCAT-B data is available at 25 km grid resolution and has been reprocessed with the EUMETSAT Numerical Weather Prediction (NWP) SAF Pencil beam Wind Processor (PenWP) developed at KNMI, exploiting improved QC in the newer PenWP version [51].

3.6 Preliminary data analysis

This section provides a detailed description of the dataset used for training. The training dataset consists of 65 days of data, out of which only 5% randomly subsampled data points are used to reduce its size. The period of the training data is 2 January - 6 March 2020.

Table 3.3 shows the statistics of the scatterometer corrections represented in components (u_diff for zonal and v_diff for meridional) and model variables (i.e., atmospheric variables and SST from ERA5 and ocean currents from CMEMS) that are used as input features. The statistics were calculated on a global scale to understand the range of variables (for normalization purposes, as required for regression with neural networks).

Globally, we can see that the ERA5 mean zonal U10S (u_diff) are slightly biased with respect to scatterometer U10S, while the ERA5 meridional U10S does not show a significant bias. Note also that the latter presents a somewhat larger standard deviation of the differences compared to the former.

Figure 3.3 represents the distributions of scatterometer corrections for the training data divided by regions, such as the tropics (between 30°S - 30°N), the extra-tropics (30°N - 55°N and 30°S - 55°S) and the high latitudes (beyond 55°N & 55°S). Zonal U10S differences (u_diff)

Training dataset, total count 6,262,033						
Variable name	**mean**	**std**	**min**	**1%**	**99%**	**max**
u_diff	-0.17	1.36	-23.95	-3.99	3.37	25.16
v_diff	-0.02	1.46	-25.94	-3.89	3.92	21.92
se_eastward_model_wind	0.38	6.89	-24.38	-12.50	15.70	28.21
se_northward_model_wind	-0.37	5.59	-27.60	-13.26	13.01	24.02
se_model_speed	8.10	3.65	0.03	1.32	18.07	33.16
model_wind_to_dir	182.5	97.6	0.0	8.3	351.5	359.9
se_model_wind_divergence	-3.4E-07	1.8E-05	-5.0E-04	-6.2E-05	3.8E-05	2.9E-04
se_model_wind_curl	1.6E-07	3.0E-05	-5.4E-04	-9.1E-05	8.1E-05	7.4E-04
msl	1.0E+05	1.5E+03	9.3E+04	9.7E+04	1.0E+05	1.0E+05
air_temperature	287.2	10.5	242.7	265.9	301.7	306.2
q	0.009	0.006	0.000	0.001	0.019	0.024
sst	288.6	10.3	271.5	272.5	303.2	305.0
sst_dx	1.4E-07	1.0E-05	-4.0E-04	-2.8E-05	3.1E-05	4.2E-04
sst_dy	1.1E-06	1.3E-05	-6.5E-04	-3.5E-05	3.7E-05	5.0E-04
cos_sst_grad	0.51	0.58	-1.00	-0.99	1.00	1.00
uo	0.01	0.21	-4.16	-0.54	0.57	7.00
vo	0.00	0.16	-6.22	-0.43	0.42	5.01
current_speed	0.21	0.16	0.00	0.02	0.78	7.02
cos_currents	0.11	0.72	-1.00	-1.00	1.00	1.00

Table 3.3: Statistics for scatterometer corrections of zonal and meridional components (u_diff, v_diff) and model inputs

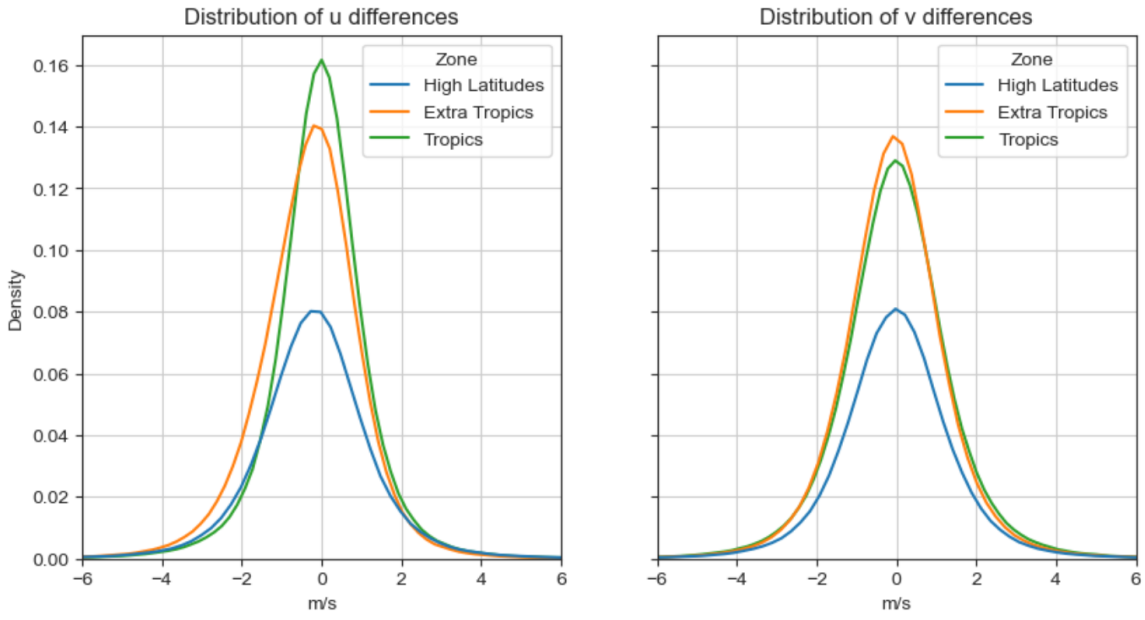


Figure 3.3: Distributions of scatterometer corrections for training data for zonal (u) and meridional (v) components in Tropics, Extra Tropics and High Latitudes.

have the smallest standard deviation in the tropics, i.e., 1.25 m/s, while 1.34 m/s in the extra tropics (1.56 m/s in Northern hemisphere and 1.23 in Southern) and 1.53 m/s at high latitudes (1.82 m/s Northern and 1.41 Southern). In the extra-tropics and high latitudes, the errors are generally larger in northern hemisphere, while in the tropics they are of similar magnitude in both hemispheres. In the extra-tropics, the zonal differences are biased in both hemispheres, while the mean differences are of -0.31 m/s.

The meridional U10S differences (v_diff) are biased in the tropics, -0.23 m/s in the northern hemisphere and 0.21 m/s in the southern. The standard deviation of the differences is 1.45 m/s in the tropics, 1.41 m/s in the extra-tropics (1.60 m/s and 1.31 m/s in the northern and southern hemispheres, respectively), and 1.52 m/s in the high latitudes (1.85 m/s and 1.38 m/s in the northern and southern hemispheres, respectively).

Figure 3.4 shows the spatial distribution of the mean scatterometer differences for zonal (top) and meridional (bottom) components. To calculate the statistic, the binning was performed on 0.5 ° x 0.5 ° grid. The means are calculated over 65 day period at the beginning of 2020. We can see the biases mentioned previously for zonal component (u) in extra-tropics (predominance of blue color, best seen in southern hemisphere). The biases for the meridional component (v) are also well seen in tropics.

If we compare this figure with 2.1 that represents mean accumulated scatterometer corrections for February 2019, the spatial distribution of the biases is quite similar. If the NWP model errors are seasonally dependent and repeat over years, it should be feasible to simulate this model errors based on location and other NWP variables that can be used as input features.

Figure 3.5 represents Pearson correlation between NWP variables and scatterometer corrections (globally). Some of the variables that are used as input features are highly correlated, such as sea surface temperature (SST), surface air temperature and relative humidity (q). There is a correlation between various variables and latitude, for example mean sea level pressure (msl), surface air temperature, SST, humidity and zonal wind component (se_eastward_model_wind). Current components (uo and vo) and model wind components (se_eastward_model_wind, se_northward_model_wind) are also correlated as model winds are used in ocean forcing models.

The presence of highly correlated variables makes the input feature importance analysis more difficult (section 6.3). As the weights get distributed between highly correlated variables, it is hard to tell which of them have more influence on the model output.

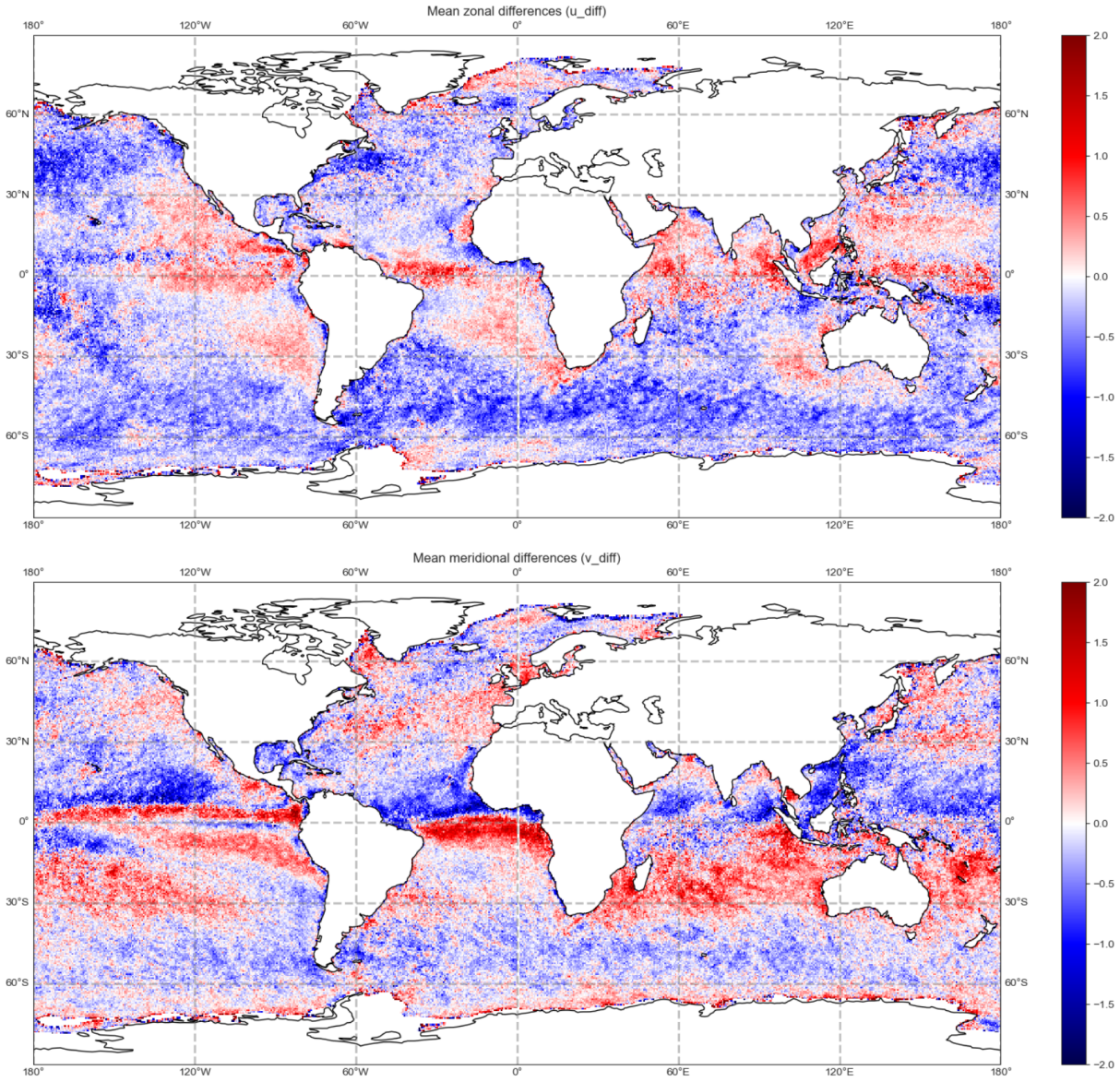


Figure 3.4: Mean scatterometer corrections for training data for zonal (u) and meridional (v) components binned in 0.5 ° x 0.5 ° grid.

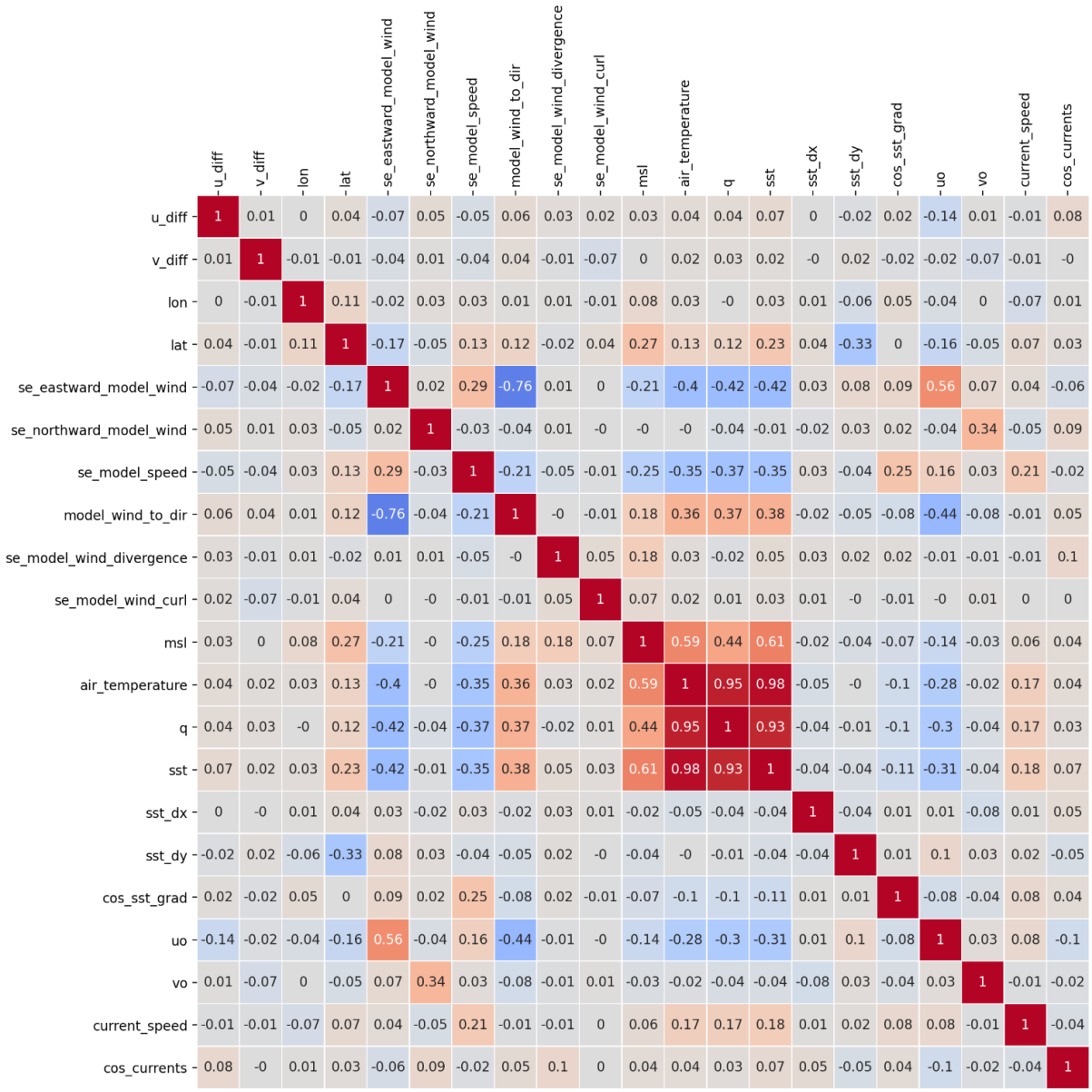


Figure 3.5: Correlations between scatterometer corrections (u_diff, v_diff) and model inputs

4

ML models

This chapter provides an overview of the of the different ML algorithms that are trained and evaluated to create preliminary regression models for generating ERA5 bias corrections. Section 4.1 gives a short introduction to supervised learning, regression models and terminology used further in the chapter. Next section 4.2 describes general requirements for the algorithms to fit the project purposes and sections 4.3 - 4.6 introduce ML algorithms that are further evaluated.

4.1 Common definitions

The regression model (predictor) is a function that describes the relationship between several independent variables, or input features, and a dependent variable, further referred as target.

The models described below are part of the supervised machine learning algorithms that map feature vectors (inputs) to labels (output), based on example input-output pairs [52]. The goal of learning is to find a model and its corresponding parameters such that the resulting predictor will perform well on unseen data [53].

There are three distinct algorithmic phases of machine learning algorithms [53]:

1. Training or parameter estimation
2. Hyperparameter tuning or model selection
3. Prediction or inference

The training or parameter estimation phase is when we adjust our predictive model based on our objectives and the training data. In this case, we train non-probabilistic models and follow the principal of maximum likelihood estimation to adjust the model parameters. The estimation of model parameters (θ) is also referred to as fitting the model to the train data.

The loss function ($\ell(y_n, f(x_n;\theta))$) measures how well the model fits the training data by computing a normalized difference between the model estimation and the target training data. When training the model, the goal is to find a parameter vector that minimizes the average loss (difference) on the set of the training samples (equation 4.1) [19, 53].

$$\hat{\theta} = \underset{\theta}{\operatorname{argmin}} \frac{1}{N} \sum_{n=1}^{N} \ell(y_n, f(x_n;\theta)) \tag{4.1}$$

The most commonly defined loss to measure the performance of the regression models is the square loss function (4.2) that heavily penalizes in particular large residuals $y - \hat{y}$.

$$\ell(y_n, f(x_n;\theta) = (y_n - f(x_n;\theta))^2 \tag{4.2}$$

The difference when using the square loss function is equal to Mean Squared Error (MSE) (4.3).

$$\text{MSE} = \frac{1}{N} \sum_{n=1}^{N} (y_n - f(x_n;\theta))^2 \tag{4.3}$$

Different loss functions will typically lead to different outcomes, reflecting the expected error distribution between model and reference data.

The model behaviour on the unseen data is simulated either with cross-validation or with a separate evaluation dataset. To perform well on unseen data, there should be a balance between fitting well on training data and still making generalized predictions that do not reduce model performance on the new data. Different regularization techniques are used to avoid overfitting, which allow the model to generalize well. Performing well also implies that the sampling distribution of the training data should well represent the general distribution of unseen data.

Commonly used regularization techniques include L1 (lasso) and L2 (ridge). In ridge regularization, the objective loss function is a sum $J(w)$ of mean squared error and a criterion that expresses a preference for the weights that lead to smaller squared L2 norm (second term in the equation 4.4), where λ controls the preference for smaller weights [54].

$$J(w) = \text{MSE} + \lambda w^T w \tag{4.4}$$

When minimizing $J(w)$, there is a trade-off between fitting the data and the weights being small. This also puts weights on fewer features.

The L1 (lasso) regularization adds a penalizing term that is a sum of the absolute values of the parameters w instead of the squared norm. If the regularization coefficient λ is sufficiently large, some of the weights w_j are driven to zero, leading to a sparse model [55].

During the training, the parameters are being adjusted to minimize the loss function. The training process and the model structure are controlled by model hyperparameters, that determine the values of the model parameters after the learning algorithm converges [56]. Hyperparameter tuning, or hyperparameter optimization, is the process of finding the

configuration of hyperparameters that results in the best model performance.

The hyperparameter tuning can be automatized using Grid Search, where the set of possible hyperparameter values is selected and the algorithm trains a model on each possible combination of these values. The algorithm returns the combination of hyperparameters that shows the best performance for a defined metric on the evaluation subset. If the evaluation subset is not defined, the cross-validation can be used to evaluate the models.

Random search is an alternative to grid search that is less computationally expensive when, instead of the predefined grid, the hyperparameter values are being randomly sampled from predefined distributions. It is not used in this work, however, as one of the hyperparameters is the neural network structure, which cannot be sampled randomly.

Finally, prediction or inference phase is when we use a trained predictor on previously unseen data, after the parameters of the model are fixed (trained) and the resulting predictor is applied to new input data [53].

4.2 General requirements

To be able to model the corrections of a 2-D vector field, we should generate the U10S corrections to both the zonal and the meridional components or, alternatively, the wind speed and direction components. The regression should be able to produce 2 outputs that are not independent. However, the difference characteristics of individual vector components are generally close to normal, while the difference characteristics of speed and direction are skew and not normal. The latter is not statistically compatible with using a squared loss difference (MSE) [57, 58].

Anyway, this 2-D vector field requirement leads to the selection of ML models with the support of multi-output regression. In this work we focus on modelling ERA5 biases for both the zonal (u) and the meridional (v) components, following the ERA* approach.

Another constraint for selecting the appropriate regression algorithms is the possibility to build a model based on a large quantity of satellite and NWP data. One day of ASCAT-A data in the original swath grid has more that 2 million points and slightly less when interpolated onto a 0.125 degree regular grid. In this work, a reduced dataset is used, such that it can fit into available RAM during the training. However, for the next training stage over a larger dataset, the algorithm should support either custom batch generator or parallel computing libraries.

Incremental learning is also useful for the operational implementation of the model, as the ECMWF model errors drift seasonally and the model can be updated with the latest data to improve predictions. Note that ECMWF is changing model version twice per year, following parallel testing, where the yearly biases remain rather constant up to now. So, some care is needed in a near real time (NRT) implementation to follow ECMWF model changes, while ERA5 changes should be rather gradual instead, given current experiences.

Most of the experiments are made for the algorithms based on Gradient Boosting Decision Trees (GBDT), such as Extreme Gradient Boosting (XGBoost) [16], [17] and Light Gradient Boosting Machine (LightGBM) [18], as well as for neural networks with 3-4 hidden layers implemented with Multi-layer Perceptron (MLP) [59] from scikit-learn library and with a similar Feed-forward Neural Network (FNN) with Keras and Tensorflow libraries [60].

4.3 XGBoost

Extreme Gradient Boosting (XGBoost) [16, 17] is a scalable and distributed Gradient Boosting Decision Trees (GBDT) library that belongs to the group of ensemble machine learning algorithms based on decision trees.

Gradient boosting is based on improving a weak model by adding a number of other weak models to generate a strong model [61]. The weak additive models are being targeted to minimize the errors between the previous model output and the target function. The outcomes of each added model represent the gradient of the error residuals with respect to the prediction. Figure 4.1 represents schematic diagram of gradient boosting. The resulting function $F(x)$ is the sum of the function $f_1(x)$ that is estimated using targets Y, while the consecutive functions $f_i(x)$ are trained with the residuals as the model targets $(Y - f_{i-1}(x))$.

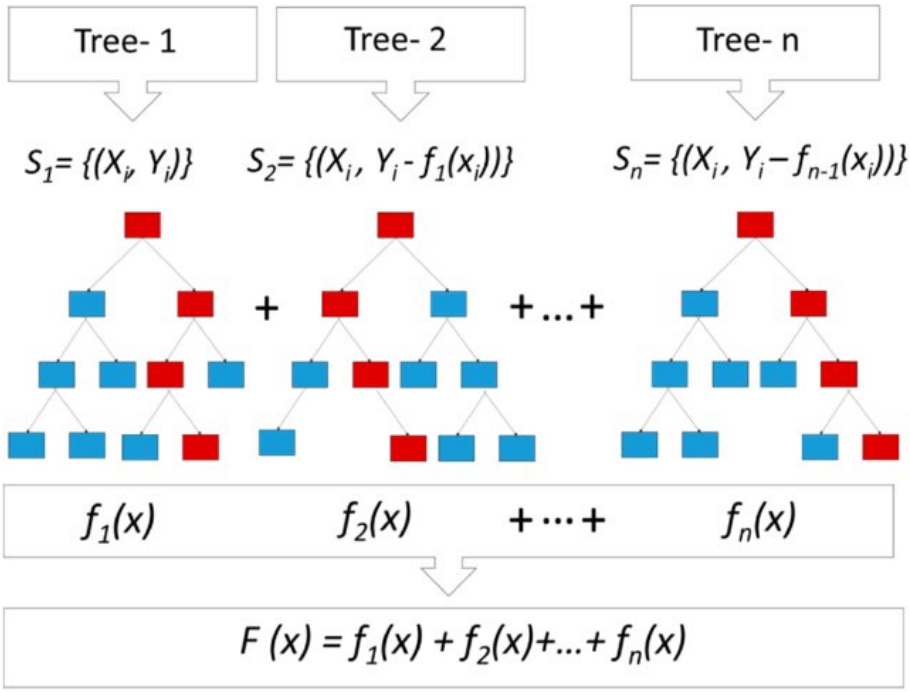


Figure 4.1: Gradient boosting schematic diagram. Adopted from [62].

GBDTs iteratively train an ensemble of shallow decision trees, with each iteration using the error residuals of the previous model to fit the next model. The final prediction is a weighted sum of all the tree predictions.

The Extreme Gradient Boosting implementation of the gradient boosting offers parallel building of trees instead of sequential which reduces the time required for training over large sets of data. It also provides various options for regularization that are used to avoid overfitting. XGBoost can be used both for classification and regression problems. Starting with version 1.6, XGBoost has a native support of multi-output that is required for our model.

The library has many hyperparameters that can be tuned for better model performance. We mainly tune the number of estimators and regularization parameters, such as alpha (L1 regularization term, see the description above) and gamma (minimum loss reduction required to

make a further partition on a leaf node of the tree). We also add randomization of samples and features used in the construction of the estimators by modifying the subsample (by randomly sample the training data prior to growing trees) and the colsample_bytree (ratio of features used in constructing each tree). The tuned hyperparameters are discussed in detail in section 5.2.

4.4 LightGBM

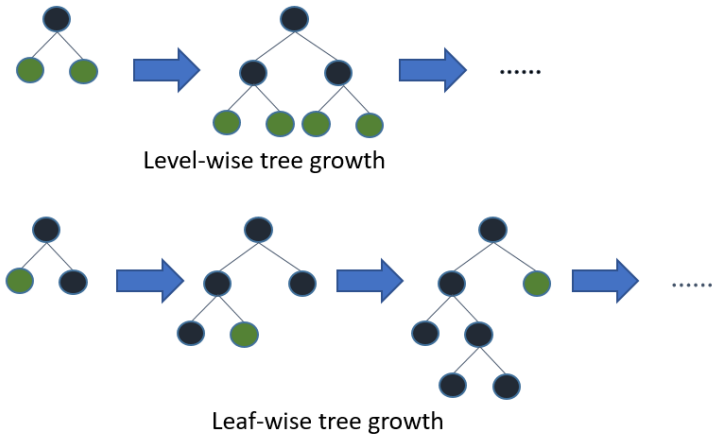


Figure 4.2: Level-wise (above) vs leaf-wise tree growth. Image adopted from [63].

The Light Gradient Boosting Machine (LightGBM) [18] is another popular gradient boosting algorithm. Compared to XGBoost, LightGBM grows trees leaf-wise instead of level-wise (depth of the tree) [63] (see figure 4.2).

The LightGBM regressor doesn't have native support of multi-output, so we have to wrap it into the MultiOutputRegressor or the RegressorChain models [64] provided by scikit-learn. The resulting metrics are similar for both of the multi-output wrappers.

4.5 MLP Regressor

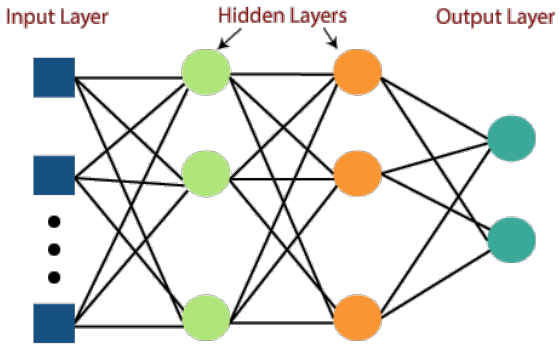


Figure 4.3: Multi-layer Perceptron with 2 outputs. Image adopted from [65].

The Multi-layer Perceptron (MLP) regressor provided by the scikit-learn library is an implementation of a fully-connected feed forward neural network (see image 4.3) that can learn and approximate a non-linear function.

Multi-layer Perceptron consists of the input layer that contains neurons representing input features, several hidden layers and an output layer. Each neuron of the hidden layer transforms the values from the previous layer with a weighted linear summation, followed by a non-linear activation function. There are several types of activation functions, we used the rectified linear unit (ReLU): $f(x) = max(0, x)$. The final output layer receives the values from the last hidden layer and returns the resulting sum without applying any activation function (as compared to classification tasks where a softmax or sigmoid activation function is applied) [66].

The training of the model is being done through a back-propagation algorithm by updating the weights of the sums of the layer outputs. Back-propagation is used to compute the gradient of a loss-function, which then is passed to a gradient-based optimization algorithm [19].

There are several types of solvers that optimize the gradient descent available in this library. In this work, we use the stochastic gradient descent (SDG) with adaptive learning rate. Adaptive learning keeps the learning rate constant to the initial learning rate, as long as the training loss keeps decreasing; otherwise, it is divided by 5 [67].

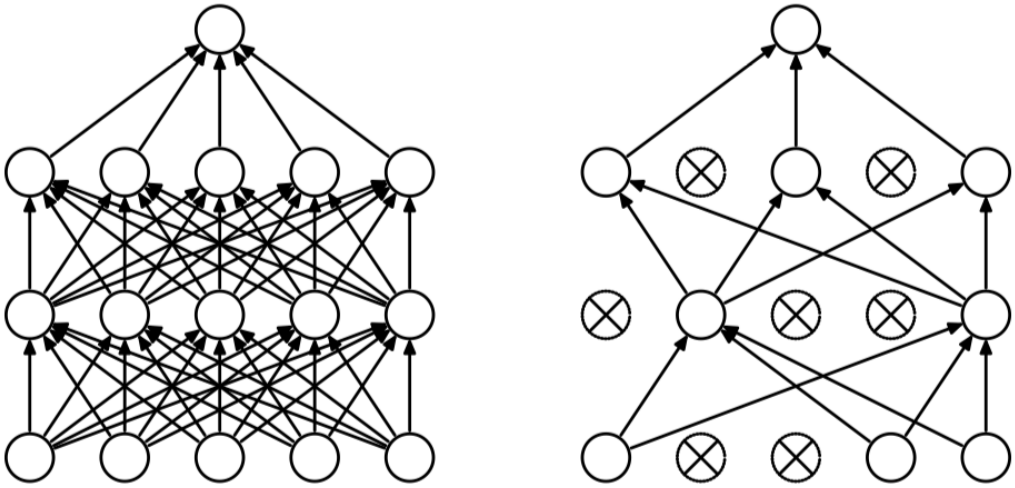
The stochastic gradient descent minimizes empirical risk by following the gradient of the risk evaluated on a random sample or a subset of data [68]. The MLP regressor implementation uses mini batches that include 200 samples by default. It also implements the Nesterov momentum that allows to move faster along the directions where the loss is changing slowly, and slowing down along the directions where the gradient has suddenly changed [19].

The MLP regressor supports L2 regularization (ridge) that is used to avoid model overfitting. The L2 regularization is penalizing weights that become too large in magnitude. In this implementation the L2 regularization term is divided by the sample size when added to the loss [67]. The library also supports online-learning using partial_fit that allows to update model weights with new data. The MLP is sensitive to feature scaling [66] and requires normalization of the datasets for training and for generation of the model output.

4.6 Feed-forward Neural Network with Tensorflow and Keras

The MLP regressor from scikit-learn library is easy to use. However, it doesn't have GPU support, fails during the training of larger models, and doesn't have the flexibility of model design as offered by Tensorflow and Keras. Keras is an application programming interface (API) for Tensorflow infrastructure that provides a user interface for deep learning [60].

We build similar feed-forward neural networks with 3-4 hidden layers with Tensorflow and Keras. However, instead of the L2 regularization term, this time we use Dropout layers between the hidden layers. The Dropout layer randomly sets input units to 0 with certain probability (rate) at each training step (weight update), which helps to prevent overfitting [60]. Figure 4.4 represents the differences between the fully-connected neural network (a) and the network after the dropout has been applied (b). The weights are being updated only using the inputs that stay active. In this work, the dropout rate is being estimated for each neural network configuration through cross-validation.



(a) Standard Neural Net (b) After applying dropout.

Figure 4.4: Fully-connected neural network (a) and the network after applying dropout (b). Image adopted from [69].

We use the adaptive moment estimation (ADAM) optimizer and a learning rate scheduler with exponential decay to reduce the learning rate at the end of the training epochs, and thus to refine the gradient descent. The ADAM optimization method computes individual adaptive learning rates for different parameters from the estimates of the first and the second moments of the gradients [70].

5 Methodology

This chapter explains the methodology and workflow that was used in this work. Section 5.1 explains the data preparation steps. Further sections 5.2 and 5.3 explain the search of optimal hyperparameters and the training stage, while last section 5.4 describes the validation approach.

5.1 Data preparation

Due to the high volume of the scatterometer and model data, the pre-processing is an important part of this project. The input files used in the resulting dataset come from different sources and in various file formats, temporal and spatial resolutions, which requires collocation of the datasets as well as spatial and temporal interpolation.

As the data preparation involves multiple steps and has a certain computational cost, it is being done in a separate chain from the training and validation of the models. The pre-processing for each day of data is being run in parallel on a HPC cluster and the outputs are stored, so that the pre-processing step is done only once. The processed data is then accessed during training and validation of the models.

5.1.1 Spatial interpolation

The collocation of the geo-referenced datasets involves aligning the data in space and time. As all datasets are in different formats and/or on different spatial grids, the first step of the data pre-processing is to interpolate them to the same grid.

As the target grid, we use the same as that of the first version of the ERA* product, i.e., a 1/8° regular grid. Table A.1 includes the regular grid definition for the NWP variables and the half grid for calculation of the derivatives such as SST gradients, wind curl and divergence. The half grid is defined so that the middle point of the gradient value has the same coordinates as the target grid.

The spatial interpolation is done with several software tools depending on the file format and its source. The scatterometer datasets are provided in Binary Universal Form for the Representation of meteorological data (BUFR) format in orbit files, which already include spatially and temporally interpolated ERA5 U10S fields. During the processing, the orbit files are concatenated in daily files and further processed with GenScat tools provided by KNMI (most of which are part of the NWP-SAF AWDP software [71, 72]). Input scatterometer files are first converted into NetCDF format at level 2 (L2) on a regular swath grid, and curl and divergence are calculated for model and scatterometer winds. The full command line options used in GenScat can be consulted in Appendix B.1.

Then, with the nc_L2_to_L3 (B.2) tool from the GenScat toolbox, we perform the spatial interpolation of the L2 NetCDF files to the fixed, regular level3 (L3) grid described in A.1.

Finally, the ERA5 model parameters (other than U10S) and ocean currents model fields are spatially interpolated to regular grid using the Climate Data Operator (CDO) tool [73].

5.1.2 Temporal interpolation

When all the datasets are on the same regular grid, we temporally interpolate them to the scatterometer ground truth data. Scatterometer data are continuous in time, while NWP model data are provided hourly (atmospheric data), and oceanic data twice-daily (SST and SST gradients) or daily (currents).

As atmospheric data has hourly resolution, and can have large temporal variance (e.g., the sea surface winds), we use three different forecast times to make the temporal interpolation, one right before the scatterometer acquisition and two after. In case there are two forecast files available for the same hour, we choose the one with the shortest forecast range, but making sure that all three forecasts used in the interpolation originate from the same analysis cycle. To get the interpolated value between three data points we use quadratic interpolation, as shown in the equations 5.1 - 5.5:

$$t' = \frac{time - t_2}{t_3 - t_1} \tag{5.1}$$

$$a = (v_3 + v_1 - 2v_2)/2 \tag{5.2}$$

$$b = (v_3 - v_1)/2 \tag{5.3}$$

$$c = v_2 \tag{5.4}$$

$$v_{interpolated} = at'^2 + bt' + c \tag{5.5}$$

where time corresponds to the scatterometer acquisition time, $t_1 - t_3$ correspond to the three mentioned NWP forecast times, $v_1 - v_3$ to the atmospheric variable to be interpolated at the three different times, and $v_{interpolated}$ to the final temporally interpolated value.

In case of SST data, we use the nearest available forecasts, as only two forecasts are available at ERA5 analysis times (6am and 6pm). As a result, all data that is obtained before noon are collocated with the 6am forecast, while afternoon data are collocated with the 6pm forecast.

The currents are collocated directly by reading the corresponding daily file, as the daily currents average is centered at 12 UTC.

The collocated files are first saved on a regular grid in NetCDF format (mainly to be able to visualize the collocated data) and then converted into two-dimensional arrays (with python numpy library) with features as columns. Longitude and latitude coordinates are added as features as well. The arrays pass additional filtering of unrealistic values and are saved in pickle format along with the list of feature names (used further for feature selection in model training).

5.1.3 Collocation of NWP data for output generation

To be able to validate the machine learning model performance against independent scatterometer data, the model output should be generated in a similar format as the ERA5 forecasts. To do this, we need to collocate ERA5 atmospheric hourly forecasts with twice-daily SST files and daily ocean velocity files, such that the NWP model biases can be computed with the trained predictor model.

When the resulting corrections SC are calculated, the input NWP fields are available over the whole ocean area, and not only over the scatterometer swaths, and the ML model output is generated globally for each forecast time.

For this type of collocations no temporal interpolations are required, we just merge the corresponding files of SST and currents with those of ERA5 atmospheric data as input to the SC computation by the ML model.

5.1.4 Normalization

This step is omitted for decision tree ensembles such as XGBoost or LightGBM but necessary for neural networks.

As some of the variables are angles in degree units, we've chosen the normalization in the range [-1, 1] to be in line with sine and cosine used for these features. For example, we use sine for the latitude normalization as the values are in the range [-90, 90] and both sine and cosine for the longitude, as well as the wind direction, as they range from 0 up to 360 degrees.

Other non-circular variables are normalized using the minmax normalization with the manually designed minimum and maximum values shown in table A.2. We analyse the distributions of a sample dataset and define a common range of values for each variable (based on analysis of maximum and minimum, as well as of 1 and 99 percentiles) and make them centered at 0 for the variables with zonal and meridional components.

The targets were scaled linearly as well as other input features. In future work we will try transformed target regressor [74] to see if non-linear transformations of the targets give any added value to the model.

5.1.5 Sample size reduction

The 3-month dataset used in training occupies around 30GB of disk space and is split into 180 files (i.e., 2 files per day). This amount of data makes it difficult to train the ML models without the use of parallel computing and implies the use of incremental training techniques. However, LightGBM (with sckit-learn multi-output wrapper) incremental training over multiple files instead of loading the entire dataset into the memory does not show good performance as

compared to XGBoost (not shown), and as such we try a different approach by reducing the dataset size.

To make the dataset more manageable for the model training, we perform downsampling by randomly choosing 5% of the data, which results in about a 2GB dataset. This approach leads to a more computationally efficient search for the optimal parameters of the first ML model prototypes and a better performance. For the follow-up activities, we will seek ways of training the tuned best performing model on a larger dataset.

5.2 Search for optimal hyperparameters

The dataset is split into train (02/01 - 06/03/2020), validation (07/03 - 09/03/2020) and test (10/03 - 01/04/2020) periods. Test dataset represents 25% of total data by the end of the period, while the validation files correspond to 3 days in the middle and the train dataset to the beginning of the period.

The first approximation for the hyperparameters of the different ML models is done on a small dataset that includes several days of data. For this task, we use GridSearch with cross validation [75] from the scikit-learn library splitting data with TimeSeriesSplit [76].

5.2.1 XGBoost

The example of the hyperparameter search for XGBoost over a small subset of data can be seen in the code listing 5.1.

Code 5.1: XGBoost grid search example

```
param = {
        'early_stopping_rounds': [True, False],
        'booster':['gbtree'],
        'tree_method':['hist'],
        'n_estimators':[100, 500, 1000],
        'min_child_weight': [.5, .8, 1],
        'gamma': [0.1, 0.2, 0.5, 1],
        'subsample': [.5,1.0],
        'colsample_bytree': [0.5, 1.0],
        'max_depth': [5, 10, 20],
        'alpha': [0, 0.1, 0.5]
        }
```

We use the default **gbtree** booster with the histogram tree method. It is an approximated solution that builds a gradient histogram for each node and iterates through the histogram instead of through the entire dataset [77].

In this grid search, we determine the optimal number of estimators (**n_estimators**). This is a first approximation on a smaller dataset. When training over the whole period, the models with a larger number of estimators are being tested as well.

The **min_child_weight** parameter is the minimum sum of instance weights (hessian) needed in a child. If the sum of instance weights of a resulting leaf node is lower than

min_child_weight, then the partitioning stops. The larger the min_child_weight is, the more conservative is the algorithm [17]. The models that are evaluated in the results section have this parameter set to a default value 1.

The **Gamma** parameter defines the minimum loss reduction required to make a further partition on a leaf node of the tree. The resulting model has this parameter set to 0.2.

Subsample is the ratio of the data that is randomly sampled during the training step. We use the default uniform distribution sampling method. A subsample value of 1 means that no random subsampling is done during the training and all the available data is used. The resulting model has a subsampling value of 0.5.

Colsample_bytree is another randomization technique that is used to avoid overfitting. It defines the subsample ratio of columns when constructing each tree. Subsampling occurs once for every constructed tree. A ratio of 0.5 is chosen for the resulting model.

The **max_depth** parameter defines the maximum depth of the tree and its complexity. A large max_depth value can lead to model overfitting. In our model, it is set to 10, while the default value is 6.

Alpha is the L1 regularization term on model weights. The L1 and L2 regularization was explained in the previous chapter 4. The bigger the regularization term is, the more conservative is the model. This value is set to 0.2.

The **learning rate** is also adjusted, but mostly manually through training runs over the whole training dataset. Learning rate is set to 0.01.

5.2.2 LightGBM

For LightGBM models, the following hyperparameters are tuned [78]:

- **n_estimators** defines the number of estimators of the model, similar to XGBoost.
- **num_leaves** is the maximum number of leaves in one tree, and controls the complexity of the model.
- **max_depth** limits the max depth of the tree model.
- **min_data_in_leaf** defines the minimum number of data in one leaf, and depends on the number of samples and the number of leaves.
- **max_bin** is the maximum number of bins that feature values will be bucketed in; reducing this number can help to avoid overfitting.
- **learning_rate** is the learning rate of the model.
- **lambda_l1** is the L1 regularization term.
- **feature_fraction** defines the fraction of the randomly selected subset of features on each iteration (tree).
- **num_iterations** is the number of boosting iterations.
- **bagging_freq** is the frequency of bagging, used to perform bagging at every k iteration.

- **bagging_fraction** randomly selects part of the data without resampling.

5.2.3 MLP Regressor

The MLP regressor hyperparameters are tuned in various grid searches, as well as through many training attempts by observing the metrics on the validation dataset throughout the training.

The code listing 5.2 shows one of the first grid searches that is done on a small subset of data.

Code 5.2: MLP grid search example

```
regr = MLPRegressor()

param_grid = {'hidden_layer_sizes': [(50, 25, 10), (50, 25, 25, 10),
    ↪ (100, 50, 25), (100, 50, 25, 5)],
              'activation': ['relu'],
              'alpha': [1, 0.1, 0.01, 0.001],
              'solver': ['adam', 'sgd'],
              'verbose': [True]
             }
```

In this grid search, we choose the optimal layers configuration, as well as the **alpha** regularization term (L2). We also try two different optimizers, stochastic gradient descent (SGD) with Nesterov's momentum and adaptive moment estimation (ADAM).

At first, the attempt was made to train the MLP Regressor on a non-reduced dataset using partial fit and training it over a number of separate files. During this type of training, SGD performed better than ADAM, and when training the final model on the reduced dataset, we use SGD with the adaptive learning rate as well.

For the MLP Regressor, the grid search over the entire training dataset is performed (see code listing 5.3). In this grid search, we compare the performance of different layer configurations, as well as the regularization term alpha.

Code 5.3: MLP grid search example

```
params_fixed = {'activation': 'relu',
           'solver': 'sgd',
           'learning_rate_init': 0.03,
           'learning_rate': 'adaptive',
           'shuffle': True,
           'max_iter':200,
           'verbose': True,
           'tol': 1e-5,
           'early_stopping': True
           }

param_grid = {'hidden_layer_sizes': [(256, 128, 64),
                                     (256, 128, 64, 32),
                                     (256, 128, 64, 32, 16),
```

```
                                    (512, 256, 128),
                                    (512, 256, 128, 64),
                                    (512, 256, 128, 64, 32),
                                    (512, 256, 128, 64, 32, 16),
                                    (1024, 512, 256),
                                    (1024, 512, 256, 32)],
            'alpha': [0.005, 0.001],
            'verbose': [True]
        }
```

5.2.4 KerasRegressor

A similar parameter search for a Tensorflow-based neural network with the use of the SciKeras [79] is made, a library that allows the use of Tensorflow with SciKit-learn and GridSearch.

The grid search listed in 5.4 is an example of the hyperparameter search for a fixed layer configuration.

Code 5.4: Grid search with KerasRegressor

```
param = {
param_grid = {
    'optimizer__learning_rate': [1e-4, 1e-5, 1e-6],
    'optimizer': ['adam', 'sgd', 'rmsprop'],
    'model__hidden_layer_sizes': [(256, 128, 64, 32)],
    'model__dropout': [0, 0.1, 0.25, 0.5],
    'batch_size': [64, 128, 256, 512, 1024],
    'epochs': [25]
}

tscv = TimeSeriesSplit(n_splits=2)
grid_search = GridSearchCV(
    reg, param_grid=param_grid, scoring = 'neg_mean_squared_error',
        ↪ cv = tscv, verbose = True
)
```

This search compares the performance of three optimisers, i.e., ADAM, SGD, and RMSProp. Different learning rates are tested, including the rates for the Dropout layers. We also analyze if the batch size is influencing the model performance. The Tensorflow model is trained using the GPU cluster node and the batch size significantly change the time required for the training. For the trained model with 4 hidden layers configuration and 256, 128, 64, 32 perceptrons per layer, respectively, we use a batch size of 512, a dropout rate of 0.25 and a learning rate scheduler with initial learning rate of 1e-4 and final learning rate of 1e-5.

5.3 Training

In this work, preliminary models are trained and evaluated, for which the training is only done over 65 days of data.

Initially, the model is trained on the full training data set. However, for the first prototype, it was decided to train it only on 5% of the data, since the scatterometer data at this resolution (12.5 km) is quite repetitive and neighbouring points normally have similar values. This way, the time for the initial model tuning is substantially reduced.

Most of the models are trained on CPU, however the Tensorflow models have been trained on a GPU cluster node, since the latter is computationally more intensive.

5.4 Validation approach

There are several validation steps for selecting the best performing ML models. As mentioned previously, the validation is first performed over smaller datasets of 3 days, and the worst performing models are then discarded at this step.

The main metric that is used for model selection is the Vector Root Mean Square Difference (VRMSD or VRMS), where the differences are calculated for both the zonal and the meridional U10S components (5.6). This metric is equivalent to Root Mean Square Error (RMSE), but applied to zonal and meridional vector components.

$$VRMS = \sqrt{\frac{1}{N}\sum_{i}(u_i^{scat} - u_i^{model})^2 + (v_i^{scat} - v_i^{model})^2} \tag{5.6}$$

As we compare the ML model performance to that of the baseline ERA5 that we're trying to correct, another metric that is used is the relative error variance reduction (5.7).

$$Relative\ error\ variance\ reduction(\%) = \frac{VRMS_{ERA5}^2 - VRMS_{model}^2}{VRMS_{ERA5}^2} * 100 \tag{5.7}$$

Validation on ASCAT-A test dataset

As mentioned in section 5.2, before training the models on the collocated ASCAT-A dataset, we leave 25% of data at the end of the period (23 days) for test purposes. This set is not used for tuning the hyperparameters, and is used to compare the model performances against each other and against ERA5. At this validation step, we generate corrected U10S fields only for data points that are collocated with ASCAT-A swaths without generating full forecasts, which is done in the next section. The metrics are calculated for all available ASCAT-A swath points, without performing any sample size reduction as we did previously to reduce the training time of the models (described in 5.1.5).

At the first validation step, the model errors are calculated against ASCAT-A scatterometer U10S, i.e., the same scatterometer instrument that is used as reference for the ML model development. This is just a first step to check the rough performance of the models and the validation is limited to the time of the ASCAT-A passes, i.e., the same time for which the models are developed.

Validation against an independent scatterometer

The HSCAT-B scatterometer on board the sun-synchronous Haiyang-2B (HY-2B) satellite is used as an independent U10S source for model validation purposes. The local solar overpass time (LST) of this satellite is separated from that by ASCAT-A by about 3.5 hours, which enables an assessment of the ML model applicability at different times of day and at times that were not used in the training. We note that HSCAT-B has slightly different accuracy characteristics than ASCAT-A, due to instrument and retrieval noise, reduced spatial resolution and rain contamination [80].

This step includes the application of ML-based SC to all used ERA5 global forecasts for forecast time steps from +3h to +18h twice a day. For this, all the necessary ML model input data is collocated with ERA5 forecasts as mentioned in 5.1.3 and, depending on the ML model, the input data are normalized as described in 5.1.4 before inference.

Then, the collocated feature layers of 2-D data are being transformed into 2-D arrays with features as columns and the model output is being generated. The model output is being transformed again into regular 2-D grid and no output is performed over land or sea ice [Note that the original ERA5 data is kept over land and sea ice, while the focus is on ocean forcing]. This way we have corrections only over open water, where the model actually has been trained. When dealing with the ML models that require normalization, the output values are also rescaled into their original range.

The model output is saved into NetCDF files that include the model zonal and meridional U10S components, as well as the calculated corrections that are applied to ERA5. The zonal and meridional U10S components are also saved in GRIB format as this is used as the NWP input (background) by Pencil beam Wind Processor (PenWP) [81].

The PenWP tool is used for processing the Ku-band scatterometer data, such as HSCAT-B, and is therefore used to perform the spatial and temporal interpolations of the background (i.e., either ERA5 or the different ML model output) to the scatterometer data. The spatial and temporal interpolations to the L2 scatterometer acquisitions are described in sections 5.1.1 and 5.1.2. An example of the PenWP run can be found in B.3.

Finally, the VRMS (equation 5.6) of collocated ERA5 / ML model output with respect to HSCAT-B U10S data is computed per day. The ERA5 and ML model output performances are compared for a period of 3 months, i.e., February and April 2020, to assess the performance for both training, validation and test periods.

6

Results and discussion

This chapter shows the inter-comparison of the preliminary regression models that are trained and evaluated over a small subset of 3 months of data. The model that shows the best performance in terms of error variance reduction, computational effort for training and inference, as well as the possibility to adapt it to a larger training dataset is then selected for future developments.

6.1 Validation on ASCAT-A test dataset

As mentioned in section 5.4, the first validation step is done over the 25% of the ASCAT-A 3-months dataset, i.e., the test subset. The test subset includes ASCAT-A collocations between 10/03/2020 and 01/04/2020. The validation is done for the original data collocated with ASCAT-A, and not the randomly-sampled 5% reduced data set that was used in the training. Each day of the period has two files corresponding to ascending and descending ASCAT-A orbits for that day. The metrics provided for the training period also refer to the complete orbit files, and not to the reduced training subset. The main metric that is used for comparison is the VRMS defined in the equation 5.6.

Table 6.1 shows the VRMS scores of ERA5 and the different ML models evaluated, for the training, validation and test periods. The first row shows the ERA5 VRMS scores (benchmark) that we are trying to reduce with the different ML models. Two LightGBM models with 500 and 1000 estimators (LGB500 and LGB1000) were evaluated. The models were wrapped into Multioutput Regressor to predict at the same time zonal and meridional differences. For XGBoost regressor we compare 3 configurations XGB500, XGB1500 and XGB2000 with 500, 1500 and 2000 estimators, respectively. Two configurations of MLP Regressor models are evaluated, one has 3 hidden layers with 256 - 128 - 32 perceptrons per layer, the other 4 hidden layers with 256 - 128 - 64 - 32 perceptrons. The latter configuration is repeated with Tensorflow library (TF 256 128 64 32 Dr 0.25), but instead of L2 regularization we implement Dropout layers between the hidden layers with the Dropout rate of 0.25. We also test a larger model with 512 256 128 64 hidden layer sizes and a higher dropout rate of 0.4. The last row N represents the number of samples

VRMS	Training	Validation	Test
ERA5	2.0061	2.0158	1.9916
LGB500	1.9065	1.9216	1.9021
LGB1000	1.9638	1.9678	1.9485
XGB500	1.8627	1.9068	1.8890
XGB1500	1.8219	1.8994	**1.8833**
XGB2000	1.8096	1.8989	1.8830
MLP 256 128 32	1.8953	1.9095	1.8924
MLP 256 128 64 32	1.8929	1.9089	**1.8912**
TF 256 128 64 32 Dr 0.25	1.8630	1.9032	**1.8838**
TF 512 256 128 64 Dr 0.4	1.8531	1.9019	1.8833
N	125,241,930	5,744,099	43,524,815

Table 6.1: VRMS difference between collocated ERA5 / ML models and ASCAT-A U10S, for training, validation and test datasets for different model configurations. The first row shows the baseline ERA5 metrics. N corresponds to the number of samples in each subset.

used in each subset.

The models that are highlighted in bold are selected because of its initial performance and then tested in the second validation stage by generating the full forecast outputs and collocating them with independent scatterometer HSCAT-B U10S (see sections 5.4 for methodology and 6.2 for results). The detailed configurations of the selected models are shown in Appendix C.1 for XGBoost, C.2 for MLP regressor and C.3 for Tensorflow feed-forward network.

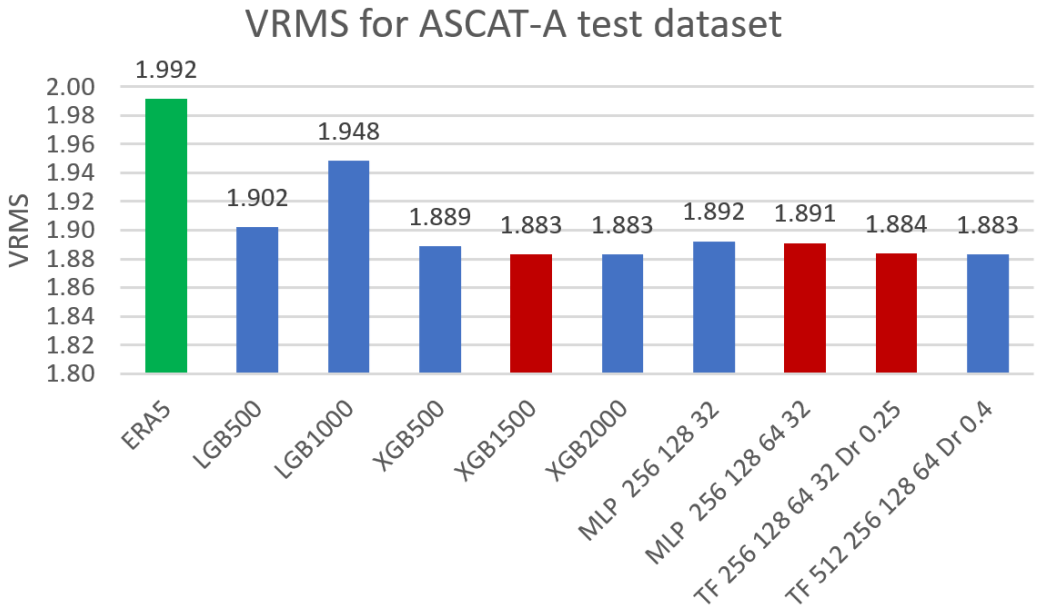


Figure 6.1: VRMS difference between collocated ERA5 / ML models and ASCAT-A U10S on the test dataset. The ERA5 VRMS values (benchmark) are plotted in green. The ML models plotted in red are the ones selected for the second verification step with full forecast generation and validation against HSCAT-B.

Square error reduction, %	Training	Validation	Test
LGB500	9.7	9.1	8.8
LGB1000	4.2	4.7	4.3
XGB500	13.8	10.5	10.0
XGB1500	17.5	11.2	**10.6**
XGB2000	18.6	11.3	10.6
MLP 256 128 32	10.7	10.3	9.7
MLP 256 128 64 32	11.0	10.3	**9.8**
TF 256 128 64 32 Dr 0.25	13.8	10.9	**10.5**
TF 512 256 128 64 Dr 0.4	14.7	11.0	10.6

Table 6.2: Mean error variance reduction of the different ML models with respect to ERA5, for the training, validation and test subsets.

Figure 6.1 is a graphical representation of the metrics shown in table 6.1 but for the test subset only. The ERA5 VRMS scores (benchmark) are highlighted in green, while the scores for the ML models selected for the second validation step against independent HSCAT-B are marked in red. Generally, the ML models show similar metrics on the test dataset, except for the LightGBM model, whose performance is probably affected by the lack of native multi-output support and the use of an additional wrapper. For the second stage of the verification, we select, on the basis of performance and computational efficiency, one configuration per library, except for the LightGBM.

The XGBoost with 1500 estimators shows better performance than smaller models with 500 estimators, however it doesn't improve much on the reduced training dataset when more estimators are added (see table 6.1). The computational time for output generation, however, is significantly longer than for the other models. When run on a CPU cluster for final forecast generation, the XGBoost is the slowest model, compared to the MLP regressor and Tensorflow implementations.

The MLP regressor with 4 hidden layers shows slightly better performance than the one with 3 layers and smaller number of perceptrons, while the Tensorflow implementation of 4 hidden layers with resp. 256, 128, 64 and 32 neurons and dropout layers shows better performance, similar to that of XGBoost with 1500 estimators. A larger model with subsequently 512, 256, 128 and 64 neurons per layer does not show much improvement while more computationally expensive during both training and inference. When run on a CPU cluster, the MLP regressor generates output faster than the Tensorflow models, however the latter provide much more flexibility in possible model architectures and training approaches.

Table 6.2 and figure 6.2 show the relative error variance reduction of the different ML models as compared to ERA5 (see Equation 5.7). It is clear that the XGBoost models significantly better adjust to the training data set than the neural networks. At the same time, the Tensorflow implementations with dropout better fit the training data set than the MLP regressor with L2 normalization.

Figure 6.3 shows the quantiles of the distribution of relative error variance reduction calculated for separate files of the test subset. As mentioned previously, each day has 2 files corresponding to ascending and descending ASCAT-A passes. The XGBoost with 1,500 estimators shows similar mean performance than the Tensorflow models but with more variation.

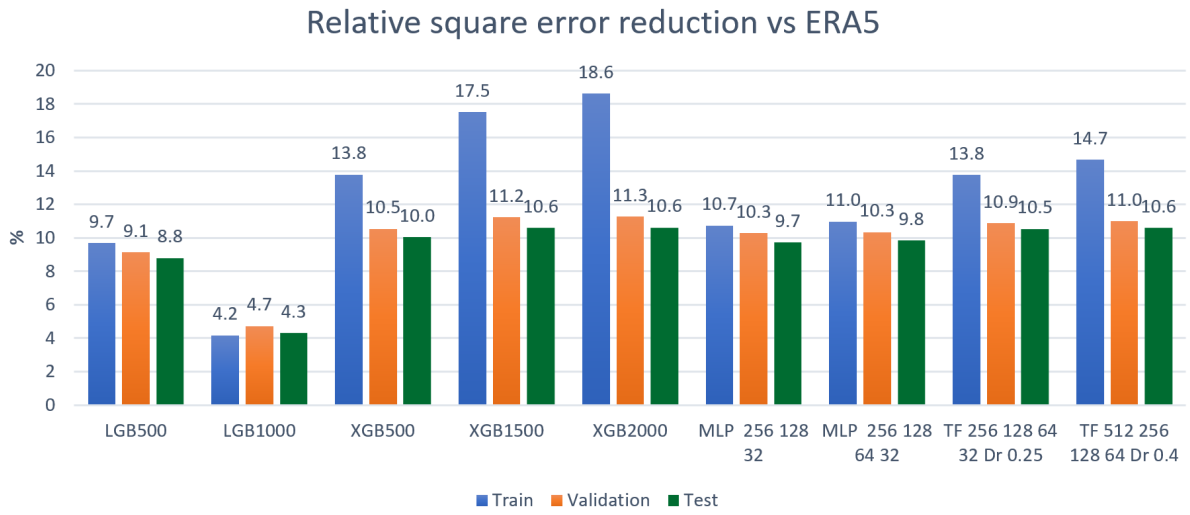


Figure 6.2: Mean error variance reduction of the different ML models with respect to ERA5, for the training, validation and test data sets.

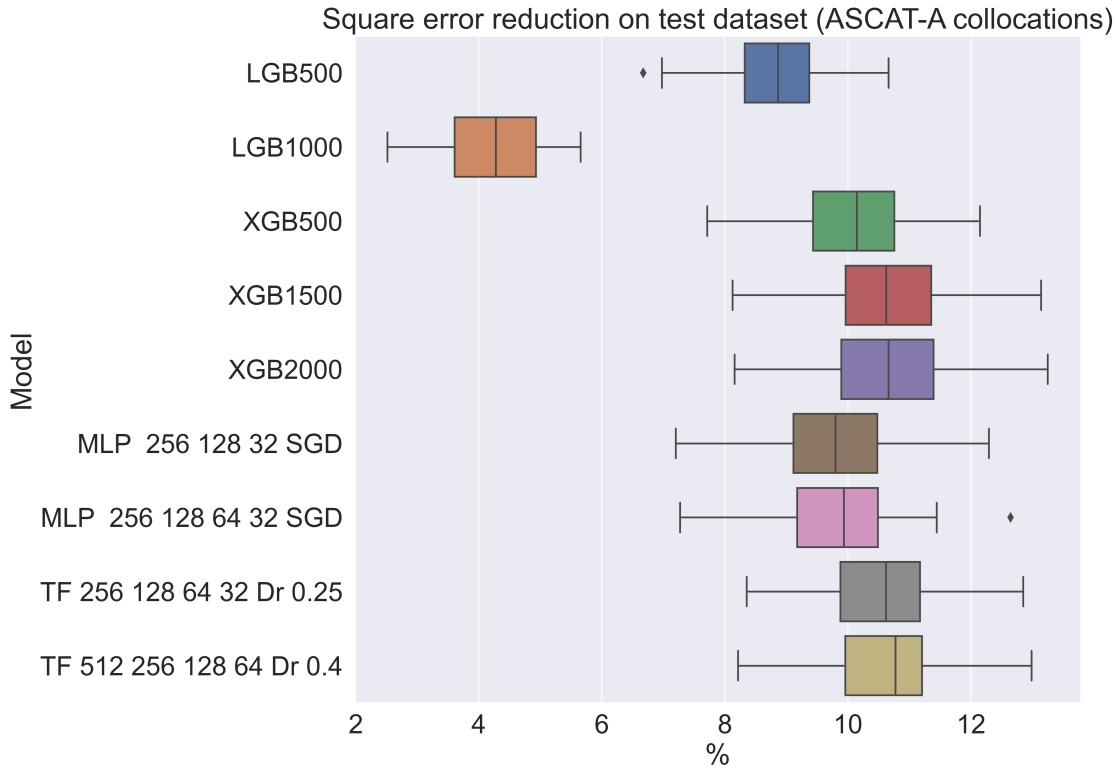


Figure 6.3: Distribution of daily error variance reduction of the different ML models with respect to ERA5, for the ASCAT-A test dataset.

6.2 Generation of complete forecasts and validation against HSCAT-B

The second validation step includes the generation of full NWP corrected forecasts and their collocation with independent HSCAT-B scatterometer data (see section 5.4). The forecasts are generated for the period 01/02/2020 - 29/04/2020, including the period 01/02/2020 - 06/03/2020, which is part of the period that is used for training, and the period 06/03/2020 - 09/03/2020 used for hyperparameter tuning. The period 10/03/2020 - 29/04/2020 is used as test dataset. At this step, a larger period is being validated compared to the test period verified against ASCAT-A, which only includes March data. This is done to verify how the model performance degrades with time, as it is trained for only a small contiguous subset of data.

Table 6.3 shows the overall VRMS scores for the three selected models (see section 5.4) against HSCAT-B U10S, for the training, validation and test periods. Besides the global statistics, we also assess the performance of the models in the tropics (between 30°S - 30°N), extra-tropics (30°N - 55°N and 30°S - 55°S) and high latitudes ($>$ 55°N & 55°S). The ERA5 metrics represent the benchmark for the developed regression models.

Table 6.4 shows the mean error variance reduction for the different ML models with respect to ERA5 (in %), for the training, validation, and test periods, while figure 6.4 shows the same, but only for the test period. Generally, on the test dataset, the Tensorflow model shows the best performance, with an error variance reduction of up to 5.54% globally, which is comparable to the performance of the first version of the ERA* N15 metrics for the same period (4.8%). In this case, however, we are dealing with a functional relationship between atmospheric and oceanic variables and NWP U10S corrections, trained on scatterometer data. It should further be taken into account that the preliminary ML models are trained on only a very small data set of 65 days, that was furthermore sub-sampled to reduce the training time, where training on a larger data set should significantly improve the model performance.

In the tropics, all 3 ML models show similar performance. However, the largest differences are observed in the extra tropics, where the best performing model reaches 7.66% of error variance reduction compared to ERA5, while for the ERA* 15-day baseline configuration (ERA* N15) the reduction is only about 3.66%. At high latitudes, the Tensorflow model also shows better performance than the XGBoost and the MLP regressor models, with 5.47% of error variance reduction, that again outperforms the 3.36% reduction of the ERA* N15 configuration).

Figure 6.5 shows the mean error variance reduction of the different ML models for the training period. As expected, the reduction is significantly higher than that for the testing period.

The application of the ML generated corrections for the training period can be useful for correcting biases in reanalysis data sets, that provide forecasts in the past, or for weather, seasonal and climate forecasts in the future. They are useful for climate studies as there are many gaps in the observational data over the ocean. The corrections generated by the models can also be used for past forecasts, including periods when scatterometer observations were ingested in the ML model for training. For the training period, we are able to reduce the variance up to 9.22% globally, 9.61% in the Tropics, 10.23% in the extra Tropics and 7.08% (Tensorlfow) or 8.37% (XGBoost) at high latitudes.

Figure 6.6 represents the error variance reduction of the different ML models with respect to ERA5, for every day of the testing period and the global ocean, while figure 6.7 shows the same, but for the extra-Tropics only, i.e., where the ML models show the best performance. The goal

VRMS	Train	Validation	Test	Train	Validation	Test
	Global			Extra Tropics		
ERA5	1.624	1.644	1.631	1.588	1.554	1.601
XGB1500	1.550	1.601	1.594	1.520	1.497	1.556
MLP 256_128_64_32	1.570	1.607	1.595	1.528	1.500	1.557
TF 256_128_64_32	1.547	1.590	1.585	1.505	1.477	1.538
N	26,137,168	2,322,781	36,921,180	9,622,053	869,584	13,948,660
	Tropics			High Latitudes		
ERA5	1.566	1.613	1.546	1.808	1.877	1.891
XGB1500	1.489	1.569	1.518	1.731	1.858	1.849
MLP 256_128_64_32	1.514	1.576	1.518	1.762	1.865	1.851
TF 256_128_64_32	1.489	1.564	1.518	1.743	1.849	1.838
N	11,458,334	1,020,372	16,586,660	5,056,781	432,825	6,385,860

Table 6.3: VRMS difference of ERA5 / ML models and HSCAT-B U10S, for the training, validation and test periods, for the global ocean, the tropics, the extra tropics, and the high latitudes.

Variance reduction, %	Train	Validation	Test	Train	Validation	Test
	Global			Extra Tropics		
XGB1500	8.91	5.18	4.50	8.48	7.23	5.53
MLP 256_128_64_32	6.48	4.56	4.42	7.45	6.92	5.43
TF 256_128_64_32	9.22	6.50	5.54	10.23	9.75	7.66
	Tropics			High Latitudes		
XGB1500	9.59	5.38	3.65	8.37	2.01	4.33
MLP 256_128_64_32	6.50	4.56	3.66	5.03	1.31	4.16
TF 256_128_64_32	9.61	5.96	3.67	7.08	2.99	5.47

Table 6.4: Same as 6.3, but showing the mean error variance reduction of the different ML models with respect to ERA5 (in %).

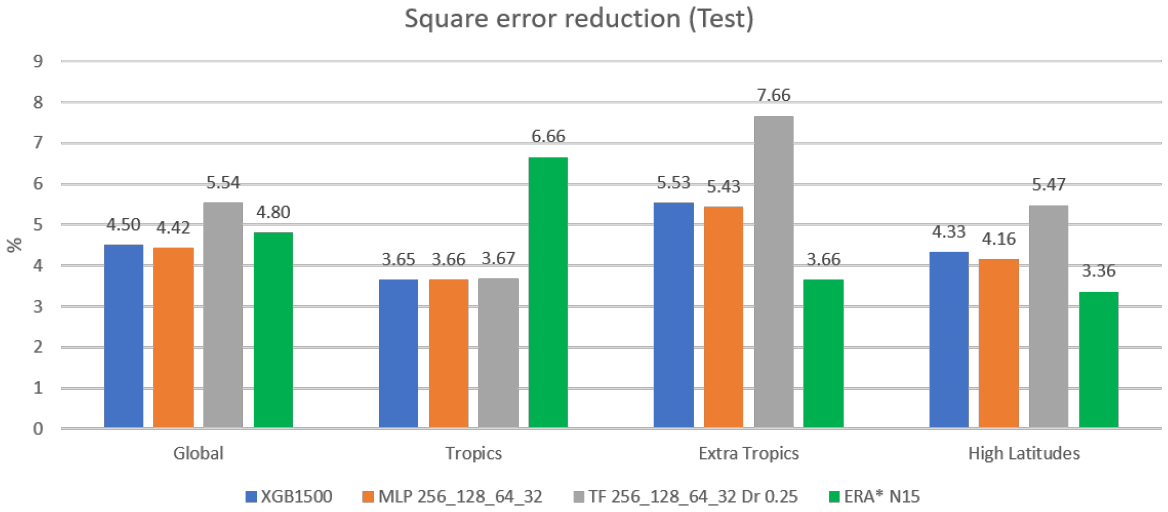


Figure 6.4: Mean error variance reduction of the different ML models with respect to ERA5 for the test period and the different ocean regions.

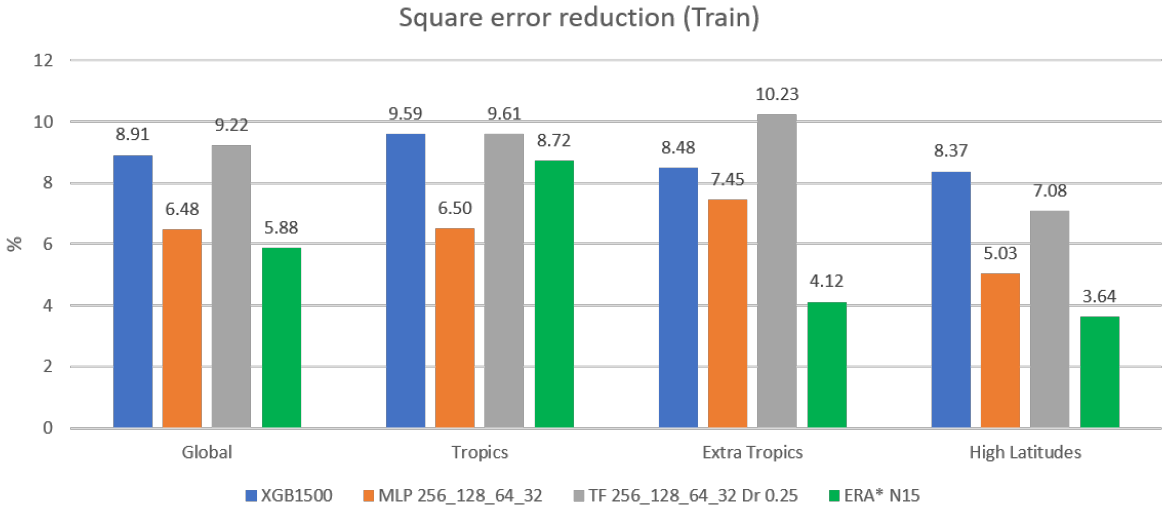


Figure 6.5: Same as Figure 6.4 but for the train period.

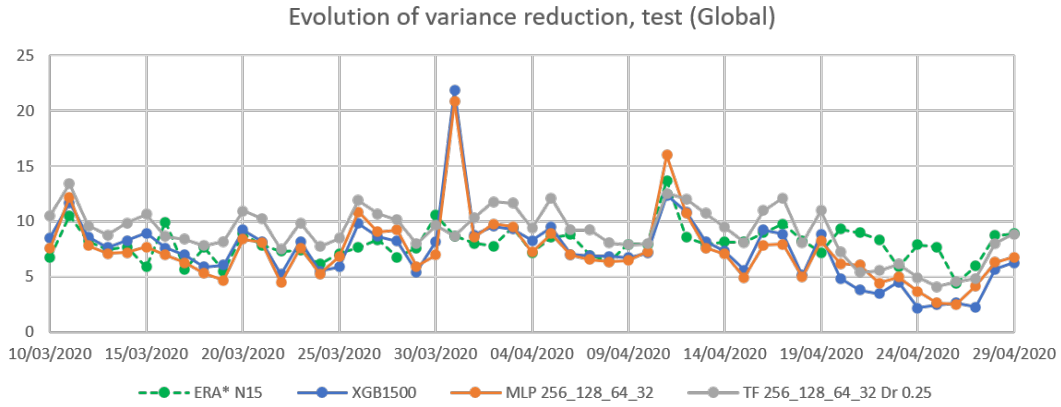


Figure 6.6: Daily evolution of the error variance reduction of the different ML models with respect to ERA5, for the testing period over the global ocean.

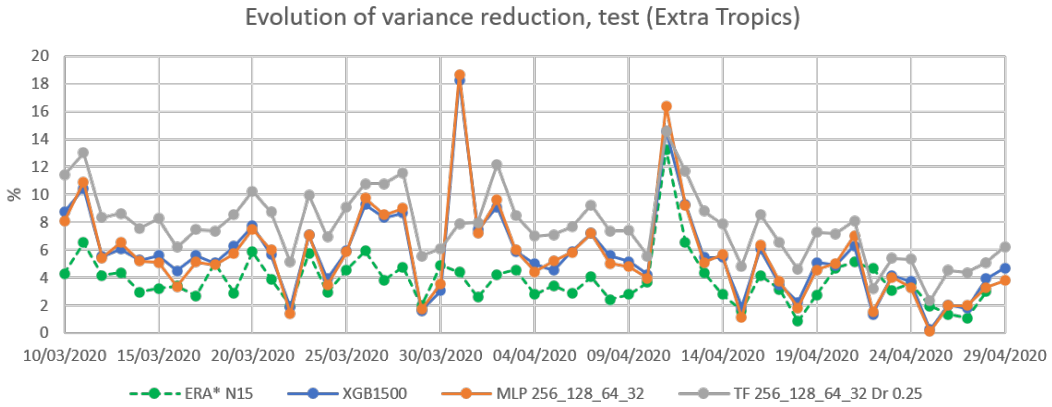


Figure 6.7: Same as figure 6.6 but for the extra tropics

is to analyze whether there is a model performance degradation with time, beyond the training period. Indeed, a certain degradation is seen, especially for the XGBoost and MLP regressor models, while the Tensorflow model still shows noticeable improvement at the end of the test period. Note also that the Tensorflow model outperforms the ERA* N15 up to more than one month beyond our reduced training data set, and for the extra-Tropics almost until the end of the testing period. If the model is trained on a larger data set, this degradation effect may] be much smaller. Nevertheless, when used in operational mode, the ML model should be regularly updated to minimize potential drifts in performance.

6.3 Feature importance

While the Tensorflow model generally shows better performance than the MLP regressor and XGBoost models, decision tree ensembles can provide information about the input feature importance. Figure 6.8 shows the feature importance after training the XGBoost regressor with 2000 estimators. This information can be used to interpret the model and to reduce its dimensionality, by removing the features with less importance. In this work, the ML models have been trained with all the available input features. However, in a subsequent study the ML model performance will be assessed after the elimination of the less important features.

Figure 6.8 shows that the feature with the most weight is the latitude, followed by the ERA5 wind direction and the meridional wind component. The longitude is also an important feature, which tells us that the model takes into account the position of the generated correction and the orientation of the wind field. From the atmospheric stability related variables, the XGBoost shows slightly higher importance for the relative humidity q, followed by the sea surface temperature (SST) and the zonal component of surface currents (uo). The following two features are wind related, such as the the wind curl and the zonal wind component. Similar importance is assigned to the surface air temperature, followed by air pressure (msl).

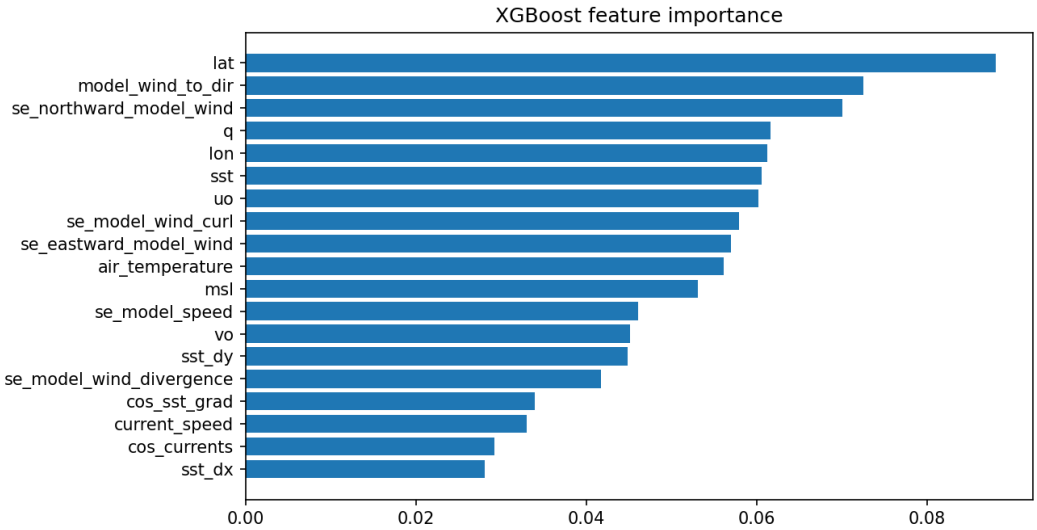


Figure 6.8: Feature importance in the XGBoost model with 2000 estimators

The wind speed has less importance than the wind direction or the wind components which means that the correction depends more on the wind direction than its intensity. This is in line with the physical closure problem in the Marine Atmospheric Boundary Layer (MABL) quoted earlier [8]. On the other hand, it is more difficult to understand how deficient physical processes depend on latitude or longitude, other than due to coincidental occurrence, i.e., without direct physical cause. It is interesting to relate the ERA5 biases to their physical causes, which might also be further elaborated in future ML models.

7

Conclusions and Future Work

In this work, we demonstrate the feasibility of reducing ERA5 stress-equivalent wind biases with machine learning (ML) techniques that use both atmospheric and oceanic data as the regression input and observations from the scatterometer constellation as learning data set.

The obtained preliminary ML models, which are trained on only a small dataset of 2 months and 10 days, show a reduction of error variance with respect to ERA5 U10S which is globally comparable to that of the ERA* N15 baseline product (figure 6.6) and significantly better in the extra-Tropics (figure 6.7) for the 1.5 month period following the training period. When validating the models against the independent HSCAT-B scatterometer, a 5.5% error variance reduction is achieved globally for the test data set, and up to 7.66% in the extra-Tropics.

When we generate full global forecasts for the training period, the error variance reduction is up to 9.61% globally and up to 10.23% in the extra-Tropics. The generation of hourly global predictions for the periods used in the training (where the ML model "sees" only about 14 1100-km wide swaths per day) can be used for correcting the ERA5 reanalysis data-set, similar to what has been done for the ERA* N15 baseline product, but with significantly better performance. The metrics for the test data set also show the ML model potential for operational use.

During the second validation step, we compare three ML models with similar performance, but implemented with three different ML libraries. The Tensorflow model shows a slightly better performance on the test dataset than the other two. As such, this model is selected for future developments in the OSI SAF follow-on project since the Tensorflow library is much more flexible than the MLP regressor implementation. To train the model over a 100 - 200 times larger data set than the one used in this preliminary study, we need the ability to design a custom training process and the possibility to train the model on a GPU cluster node. The XGBoost also has GPU support and can be trained over large datasets using Dask parallel computing. However, the models with a large number of estimators are much slower at generating the forecasts after the model has been successfully trained, compared to the neural networks.

In the OSI SAF follow-on project, we plan to train the neural network over several years of data which will require the creation of a custom training procedure, while the computational

effort for training will be considerable. As part of future work, to reduce the training time and model complexity, we will perform a study of input feature elimination, as some of the features have less weight than others (see section 6.3).

While the small 2 GB training data set did not allow ML model performance improvement, when adding more hidden layers or perceptrons per layer, and required implementation of regularization techniques to avoid overfitting, one might need to augment the complexity of the model, when trained on a considerably larger data set.

In section 6.3 the feature importance of the preliminary ML model is introduced, which was possible to estimate only with the XGBoost model. If we use a feed-forward neural network with an elevated number of parameters (weights), while training the final model over the entire data set, a sensitivity study will be required for interpreting the model, as generally the fully-connected neural networks are treated as black-box solutions and additional effort is required to understand how certain input features influence the final output.

In this work, we try the simplest feed-forward neural networks and calculate the derivatives such as gradients, wind curl and divergence which are also used as inputs. An alternative is to further develop the Convolutional Neural Network (CNN) model, in which the derivatives are calculated by the neural network itself and over larger areas. This will also allow to create spatial dependencies with surrounding points, which may give a better representation of the underlying physical dynamics of the system, as well as possible reduction of noise in the ML model output.

Acknowledgements

The work has been funded under the EUMETSAT Ocean and Sea Ice (OSI) Satellite Application Facility (SAF) Visiting Scientist activity (reference OSI_VSA22_01).

I would like to thank Marcos Portabella (ICM) for supervising this work and giving me the opportunity to develop this project. I want to give special thanks to Ad Stoffelen (KNMI), he is a person who came up with the idea of this project and provided his guidance on the data to be used and the way it should be processed. Both Marcos and Ad were a great help when creating this report, as well as the proposal. I also appreciate the help of José Dorronsoro (UAM) who was very kind to accept the role of lecturer and gave his helpful insights on this work.

I want to thank Anton Verhoef (KNMI) for providing the scatterometer datasets, ERA5 stress-equivalent wind data, as well as other atmospheric variables. Anton also gave his support on the use of the scatterometer data processing software and on the interpolation algorithms.

I appreciate the help of Manuel Arias (ICM-CSIC), who gave useful advises on training approaches of the ML models.

Acknowledgments to Copernicus Marine Service (CMEMS) for publishing the global currents dataset that was used in this work.

Bibliography

[1] M. Portabella, "Wind field retrieval from satellite radar systems," Ph.D. dissertation, University of Barcelona, Barcelona, 2002, oCLC: 1123837672.

[2] W. Lin, M. Portabella, A. Stoffelen, J. Vogelzang, and G. de Chiara, "Optimization of ASCAT data assimilation in global NWP," accepted: 2018-03-20 Publisher: EUMETSAT. [Online]. Available: https://digital.csic.es/handle/10261/162529

[3] M. Belmonte Rivas and A. Stoffelen, "Characterizing ERA-Interim and ERA5 surface wind biases using ASCAT," *Ocean Science*, vol. 15, no. 3, pp. 831–852, Jun. 2019, publisher: Copernicus GmbH. [Online]. Available: https://os.copernicus.org/articles/15/831/2019/

[4] A. Trindade, M. Portabella, A. Stoffelen, W. Lin, and A. Verhoef, "ERAstar: A High-Resolution Ocean Forcing Product," *IEEE Transactions on Geoscience and Remote Sensing*, vol. 58, no. 2, pp. 1337–1347, Feb. 2020, conference Name: IEEE Transactions on Geoscience and Remote Sensing.

[5] H. Hersbach, "Comparison of C-Band Scatterometer CMOD5.N Equivalent Neutral Winds with ECMWF," *Journal of Atmospheric and Oceanic Technology*, vol. 27, no. 4, pp. 721–736, Apr. 2010, publisher: American Meteorological Society Section: Journal of Atmospheric and Oceanic Technology.

[6] A. R. Brown, A. C. M. Beljaars, and H. Hersbach, "Errors in parametrizations of convective boundary-layer turbulent momentum mixing," *Quarterly Journal of the Royal Meteorological Society*, vol. 132, no. 619, pp. 1859–1876, 2006. [Online]. Available: https://onlinelibrary.wiley.com/doi/abs/10.1256/qj.05.182

[7] Q. Song, D. B. Chelton, S. K. Esbensen, N. Thum, and L. W. O'Neill, "Coupling between Sea Surface Temperature and Low-Level Winds in Mesoscale Numerical Models," *Journal of Climate*, vol. 22, no. 1, pp. 146–164, Jan. 2009, publisher: American Meteorological Society Section: Journal of Climate.

[8] I. Sandu, A. Beljaars, P. Bechtold, T. Mauritsen, and G. Balsamo, "Why is it so difficult to represent stably stratified conditions in nwp models?" no. 684, p. 23, 08 2012. [Online]. Available: https://www.ecmwf.int/node/12086

[9] M. Portabella, A. Trindade, G. Grieco, and E. Makarova, "Algorithm Theoretical Basis Document for ERAStar v. 2.0," Tech. Rep., Jun. 2022, ESA report WOC-ESA-ODL-NR-009_T1_ERAstar_V2.0 (Contract No. 4000130730/20/I-NB). [Online]. Available: https://data-cersat.ifremer.fr/projects/woc/products/theme1/ocean_winds/woc-l4-se-erastar-h/v2.0/atbd.pdf

[10] E. Makarova, "Evaluation of the ERA* ocean forcing product under storm surge conditions in the adriatic sea," Trabajo final de grado, Universidad de Barcelona, publisher: Digital CSIC. [Online]. Available: http://hdl.handle.net/10261/236102

[11] M. Sonnewald, R. Lguensat, D. C. Jones, P. D. Dueben, J. Brajard, and V. Balaji, "Bridging observations, theory and numerical simulation of the ocean using machine learning," *Environmental Research Letters*, vol. 16, no. 7, p. 073008, Jul. 2021, publisher: IOP Publishing. [Online]. Available: https://doi.org/10.1088/1748-9326/ac0eb0

[12] L. Žust, A. Fettich, M. Kristan, and M. Ličer, "HIDRA 1.0: deep-learning-based ensemble sea level forecasting in the northern Adriatic," *Geoscientific Model Development*, vol. 14, no. 4, pp. 2057–2074, Apr. 2021, publisher: Copernicus GmbH. [Online]. Available: https://gmd.copernicus.org/articles/14/2057/2021/

[13] A. Barth, A. Alvera-Azcárate, M. Licer, and J.-M. Beckers, "DINCAE 1.0: a convolutional neural network with error estimates to reconstruct sea surface temperature satellite observations," *Geoscientific Model Development*, vol. 13, no. 3, pp. 1609–1622, Mar. 2020, publisher: Copernicus GmbH. [Online]. Available: https://gmd.copernicus.org/articles/13/1609/2020/

[14] A. Barth, A. Alvera-Azcárate, C. Troupin, and J.-M. Beckers, "DINCAE 2: multivariate convolutional neural network with error estimates to reconstruct sea surface temperature satellite and altimetry observations," Oceanography, preprint, Nov. 2021. [Online]. Available: https://gmd.copernicus.org/preprints/gmd-2021-353/gmd-2021-353.pdf

[15] H. Hersbach, B. Bell, P. Berrisford, S. Hirahara, A. Horányi, J. Muñoz-Sabater, J. Nicolas, C. Peubey, R. Radu, D. Schepers, A. Simmons, C. Soci, S. Abdalla, X. Abellan, G. Balsamo, P. Bechtold, G. Biavati, J. Bidlot, M. Bonavita, G. De Chiara, P. Dahlgren, D. Dee, M. Diamantakis, R. Dragani, J. Flemming, R. Forbes, M. Fuentes, A. Geer, L. Haimberger, S. Healy, R. J. Hogan, E. Hólm, M. Janisková, S. Keeley, P. Laloyaux, P. Lopez, C. Lupu, G. Radnoti, P. de Rosnay, I. Rozum, F. Vamborg, S. Villaume, and J.-N. Thépaut, "The ERA5 global reanalysis," *Quarterly Journal of the Royal Meteorological Society*, vol. 146, no. 730, pp. 1999–2049, 2020. [Online]. Available: https://onlinelibrary.wiley.com/doi/abs/10.1002/qj.3803

[16] T. Chen and C. Guestrin, "XGBoost: A Scalable Tree Boosting System," in *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, Aug. 2016, pp. 785–794, arXiv:1603.02754 [cs]. [Online]. Available: http://arxiv.org/abs/1603.02754

[17] "XGBoost Documentation — xgboost 1.6.1 documentation." [Online]. Available: https://xgboost.readthedocs.io/en/stable/

[18] G. Ke, Q. Meng, T. Finley, T. Wang, W. Chen, W. Ma, Q. Ye, and T.-Y. Liu, "LightGBM: A Highly Efficient Gradient Boosting Decision Tree," in *Advances in Neural Information Processing Systems*, vol. 30. Curran Associates, Inc., 2017.

[19] K. P. Murphy, *Probabilistic Machine Learning: An Introduction*, ser. Adaptive Computation and Machine Learning series, F. Bach, Ed. Cambridge, MA, USA: MIT Press, Mar. 2022.

[20] J. A. P. León, "Data archive growth: Escaping from the black hole," Jan. 2020. [Online]. Available: https://www.ecmwf.int/en/newsletter/162/news/data-archive-growth-escaping-black-hole

[21] R. Pal, S. Mukhopadhyay, D. Chakraborty, and P. N. Suganthan, "Very high-resolution satellite image segmentation using variable-length multi-objective genetic clustering for multi-class change detection," *Journal of King Saud University - Computer and Information Sciences*, vol. 34, pp. 1275–1544, Jan. 2022.

[22] G. Matasci, J. Plante, K. Kasa, P. Mousavi, A. Stewart, A. Macdonald, A. Webster, and J. Busler, "Deep learning for vessel detection and identification from spaceborne optical imagery," in *ISPRS Annals of the Photogrammetry, Remote Sensing and Spatial Information Sciences*, vol. V-3-2021. Copernicus GmbH, Jun. 2021, pp. 303–310, ISSN: 2194-9042. [Online]. Available: https://www.isprs-ann-photogramm-remote-sens-spatial-inf-sci.net/V-3-2021/303/2021/

[23] X. Xu and A. Stoffelen, "Support vector machine tropical wind speed retrieval in the presence of rain for Ku-band wind scatterometry," *Atmospheric Measurement Techniques*, vol. 14, pp. 7435–7451, Nov. 2021.

[24] I. Sandu, P. Bechtold, L. Nuijens, A. Beljaars, and A. Brown, "On the causes of systematic forecast biases in near-surface wind direction over the oceans," 2020, publisher: ECMWF. [Online]. Available: https://www.ecmwf.int/node/19545

[25] A. Stoffelen and J. Vogelzang, "Wind Bias Correction Guide, version 1.5," 2021. [Online]. Available: https://nwp-saf.eumetsat.int/site/download/documentation/scatterometer/Wind_Bias_Correction_Guide_v1.5.pdf

[26] G. King, M. Portabella, W. Lin, and A. Stoffelen, "Correlating extremes in wind and stress divergence with extremes in rain over the tropical atlantic," Institut de Ciències del Mar (ICM - CSIC), Tech. Rep. SAF/OSI/CDOP3/KNMI/SCI/RP/312. [Online]. Available: https://osi-saf.eumetsat.int/sites/osi-saf.eumetsat.int/files/inline-files/OSI_AVS15_02_Correlating_extremes_wind_and_rain_Tropical_Atlantic_Gregory_King.pdf

[27] M. Gade and A. Stoffelen, "An introduction to microwave remote sensing of the asian seas," in *Remote Sensing of the Asian Seas*, V. Barale and M. Gade, Eds. Springer International Publishing, 2019, pp. 81–101. [Online]. Available: https://doi.org/10.1007/978-3-319-94067-0_4

[28] M. A. Bourassa, T. Meissner, I. Cerovecki, P. S. Chang, X. Dong, G. De Chiara, C. Donlon, D. S. Dukhovskoy, J. Elya, A. Fore, M. R. Fewings, R. C. Foster, S. T. Gille, B. K. Haus, S. Hristova-Veleva, H. M. Holbach, Z. Jelenak, J. A. Knaff, S. A. Kranz, A. Manaster, M. Mazloff, C. Mears, A. Mouche, M. Portabella, N. Reul, L. Ricciardulli, E. Rodriguez, C. Sampson, D. Solis, A. Stoffelen, M. R. Stukel, B. Stiles, D. Weissman, and F. Wentz, "Remotely Sensed Winds and Wind Stresses for Marine Forecasting and Ocean Modeling," *Frontiers in Marine Science*, vol. 6, 2019, publisher: Frontiers. [Online]. Available: https://www.frontiersin.org/articles/10.3389/fmars.2019.00443/full

[29] J. de Kloe, A. Stoffelen, and A. Verhoef, "Improved use of scatterometer measurements by using stress-equivalent reference winds," *IEEE Journal of Selected Topics in Applied Earth Observations and Remote Sensing*, vol. 10, no. 5, pp. 2340–2347, 2017.

[30] M. Portabella and A. Stoffelen, "On Scatterometer Ocean Stress," *Journal of Atmospheric and Oceanic Technology*, vol. 26, no. 2, pp. 368–382, Feb. 2009, publisher: American Meteorological Society Section: Journal of Atmospheric and Oceanic Technology. [Online]. Available: https://journals.ametsoc.org/view/journals/atot/26/2/2008jtecho578_1.xml

[31] A. Stoffelen, "Scatterometry," Ph.D. dissertation, University of Utrecht, October 1998, ISBN 90-393-1708-9.

[32] "Tutorial on Satellite Derived Wind Products." [Online]. Available: https://resources.eumetrain.org/data/4/438/navmenu.php?tab=2&page=3.0.0

[33] R. Meyer, "ASCAT Level 1: Product Generation Specification," *EUMETSAT*, p. 186, 2016. [Online]. Available: https://www-cdn.eumetsat.int/files/2020-04/pdf_ten_990009-eps-ascat-pgs.pdf

[34] O. S. W. Team, "ASCAT Wind Product User Manual," *Ocean and Sea Ice SAF*, p. 28, Jan. 2021. [Online]. Available: https://scatterometer.knmi.nl/publications/pdf/ASCAT_Product_Manual.pdf

[35] M. Portabella, E. Makarova, A. Trindade, and G. Grieco, "ERA* Hourly Global Stress Equivalent Wind and Wind Stress from ESA WOC project. Ver. 2.0," 2022, dataset. [Online]. Available: https://www.worldoceancirculation.org/Products#/metadata/3aae3973-48c4-43e6-8b11-65440e595613

[36] R. Giesen and A. Stoffelen, "Hourly global ocean sea surface wind and stress from scatterometer and model product wind_glo_phy_l4_nrt_012_004, issue: 1.0." [Online]. Available: https://catalogue.marine.copernicus.eu/documents/QUID/CMEMS-WIND-QUID-012-004.pdf

[37] D. Cho, C. Yoo, J. Im, and D.-H. Cha, "Comparative assessment of various machine learning-based biascorrection methods for numerical weather prediction model forecasts of extreme air temperatures in urban areas," *Earth and Space Science*, vol. 7, no. 4, p. e2019EA000740, 2020. [Online]. Available: https://onlinelibrary.wiley.com/doi/abs/10.1029/2019EA000740

[38] V. Lebedev, V. Ivashkin, I. Rudenko, A. Ganshin, A. Molchanov, S. Ovcharenko, R. Grokhovetskiy, I. Bushmarinov, and D. Solomentsev, "Precipitation Nowcasting with Satellite Imagery," in *Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, ser. KDD '19. New York, NY, USA: Association for Computing Machinery, Jul. 2019, pp. 2680–2688. [Online]. Available: https://doi.org/10.1145/3292500.3330762

[39] R. Tang, Y. Ning, C. Li, W. Feng, Y. Chen, and X. Xie, "Numerical Forecast Correction of Temperature and Wind Using a Single-Station Single-Time Spatial LightGBM Method," *Sensors (Basel, Switzerland)*, vol. 22, no. 1, p. 193, Dec. 2021. [Online]. Available: https://www.ncbi.nlm.nih.gov/pmc/articles/PMC8749602/

[40] L. Han, M. Chen, K. Chen, H. Chen, Y. Zhang, B. Lu, L. Song, and R. Qin, "A Deep Learning Method for Bias Correction of ECMWF 24–240 h Forecasts," *Advances in Atmospheric Sciences*, vol. 38, no. 9, pp. 1444–1459, Sep. 2021. [Online]. Available: https://link.springer.com/10.1007/s00376-021-0215-y

[41] M. Chan, W. Wong, and K. Au-Yeung, "Machine learning in calibrating tropical cyclone intensity forecast of ECMWF EPS," *Meteorological Applications*, vol. 28, no. 6, 2021.

[42] S.-H. Park, J. Yoo, D. Son, J. Kim, and H.-S. Jung, "Improved calibration of wind estimates from advanced scatterometer metop-b in korean seas using deep neural network," *Remote Sensing*, vol. 13, no. 20, p. 4164, 2021. [Online]. Available: https://doi.org/10.3390/rs13204164

[43] J. Pathak, S. Subramanian, P. Harrington, S. Raja, A. Chattopadhyay, M. Mardani, T. Kurth, D. Hall, Z. Li, K. Azizzadenesheli, P. Hassanzadeh, K. Kashinath, and A. Anandkumar, "FourCastNet: A global data-driven high-resolution weather model using adaptive fourier neural operators," number: arXiv:2202.11214. [Online]. Available: http://arxiv.org/abs/2202.11214

[44] H. Setchell, "ECMWF Reanalysis v5," Feb. 2020. [Online]. Available: https://www.ecmwf.int/en/forecasts/dataset/ecmwf-reanalysis-v5

[45] CLS, "Global Total Surface and 15m Current (COPERNICUS-GLOBCURRENT) from Altimetric Geostrophic Current and Modeled Ekman Current Reprocessin," 2018, type: dataset. [Online]. Available: https://resources.marine.copernicus.eu/product-detail/MULTIOBS_GLO_PHY_REP_015_004/INFORMATION

[46] M.-H. Rio, S. Mulet, and N. Picot, "Beyond GOCE for the ocean circulation estimate: Synergetic use of altimetry, gravimetry, and in situ data provides new insight into geostrophic and Ekman currents: Ocean circulation beyond GOCE," *Geophysical Research Letters*, vol. 41, no. 24, pp. 8918–8925, Dec. 2014. [Online]. Available: http://doi.wiley.com/10.1002/2014GL061773

[47] J. M. Wallace, *Atmospheric Science: An Introductory Survey.* Elsevier Academic Press, 2006.

[48] "Vorticity," Feb. 2022, page Version ID: 1072765043. [Online]. Available: https://en.wikipedia.org/w/index.php?title=Vorticity

[49] X. Jin, C. Dong, J. Kurian, J. McWilliams, D. Chelton, and Z. Li, "SST-Wind Interaction in Coastal Upwelling: Oceanic Simulation with Empirical Coupling," *Journal of Physical Oceanography - J PHYS OCEANOGR*, vol. 39, p. 2957–2970, Nov. 2009.

[50] H. Wang, Z. Jianhua, M. Lin, Y. Zhang, and Y. Chang, "Evaluating chinese HY-2B HSCAT ocean wind products using buoys and other scatterometers," *IEEE Geoscience and Remote Sensing Letters*, vol. 17, no. 6, pp. 923–927, 2020.

[51] X. Xu and A. Stoffelen, "Improved rain screening for ku-band wind scatterometry," *IEEE Transactions on Geoscience and Remote Sensing*, vol. 58, no. 4, pp. 2494–2503, 2020.

[52] S. Russell and P. Norvig, *Artificial Intelligence: A Modern Approach, 4th US ed.* Pearson, 2021. [Online]. Available: http://aima.cs.berkeley.edu/

[53] M. P. Deisenroth, A. A. Faisal, and C. S. Ong, *Mathematics for Machine Learning.* Cambridge University Press, 2020.

[54] I. Goodfellow, Y. Bengio, and A. Courville, *Deep Learning.* MIT Press, 2016, http://www.deeplearningbook.org.

[55] C. M. Bishop, *Pattern Recognition and Machine Learning (Information Science and Statistics)*, 1st ed. Springer, 2006.

[56] K. Nyuytiymbiy. Parameters and hyperparameters in machine learning and deep learning. [Online]. Available: https://towardsdatascience.com/parameters-and-hyperparameters-aa609601a9ac

[57] A. Stoffelen and D. Anderson, "Scatterometer data interpretation: Estimation and validation of the transfer function cmod4," *Journal of Geophysical Research: Oceans*, vol. 102, no. C3, pp. 5767–5780, 1997. [Online]. Available: https://agupubs.onlinelibrary.wiley.com/doi/abs/10.1029/96JC02860

[58] A. Stoffelen, "Toward the true near-surface wind speed: Error modeling and calibration using triple collocation," *Journal of Geophysical Research: Oceans*, vol. 103, no. C4, pp. 7755–7766, 1998. [Online]. Available: https://agupubs.onlinelibrary.wiley.com/doi/abs/10.1029/97JC03180

[59] "sklearn.neural_network.MLPRegressor." [Online]. Available: https://scikit-learn/stable/modules/generated/sklearn.neural_network.MLPRegressor.html

[60] Keras: the python deep learning API. [Online]. Available: https://keras.io/

[61] "What is XGBoost?" [Online]. Available: https://www.nvidia.com/en-us/glossary/data-science/xgboost/

[62] H. Sahour, M. Sultan, M. Vazifedan, K. Abdelmohsen, S. Karki, J. Yellich, E. Gebremichael, F. Alshehri, and T. Elbayoumi, "Statistical applications to downscale GRACE- derived terrestrial water storage data and to fill temporal gaps," *Remote Sensing*, vol. 12 (3), p. 533, 2020.

[63] "Welcome to LightGBM's documentation! — LightGBM 3.3.2 documentation." [Online]. Available: https://lightgbm.readthedocs.io/en/v3.3.2/index.html

[64] "1.12. Multiclass and multioutput algorithms." [Online]. Available: https://scikit-learn/stable/modules/multiclass.html

[65] Multi-layer perceptron in TensorFlow - javatpoint. [Online]. Available: https://www.javatpoint.com/multi-layer-perceptron-in-tensorflow

[66] 1.17. neural network models (supervised). [Online]. Available: https://scikit-learn/stable/modules/neural_networks_supervised.html

[67] sklearn.neural_network.MLPRegressor. [Online]. Available: https://scikit-learn/stable/modules/generated/sklearn.neural_network.MLPRegressor.html

[68] M. Hardt and B. Recht, *Patterns, Predictions, and Actions.* Princeton University Press, 2022. [Online]. Available: https://mlstory.org/

[69] R. Shanmugamani, *Deep Learning for Computer Vision.* Packt Publishing, 2018. [Online]. Available: https://learning.oreilly.com/library/view/deep-learning-for/9781788295628/

[70] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," 2014. [Online]. Available: https://arxiv.org/abs/1412.6980

[71] A. Verhoef, J. Vogelzang, J. Verspeek, and A. Stoffelen, "AWDP User Manual and Reference Guide," 2020.

[72] "Scatterometer | NWP SAF." [Online]. Available: https://nwp-saf.eumetsat.int/site/software/scatterometer/

[73] "CDO - CDO - Project Management Service." [Online]. Available: https://code.mpimet.mpg.de/projects/cdo/wiki

[74] sklearn.compose.TransformedTargetRegressor. [Online]. Available: https://scikit-learn/stable/modules/generated/sklearn.compose.TransformedTargetRegressor.html

[75] "3.2. Tuning the hyper-parameters of an estimator." [Online]. Available: https://scikit-learn/stable/modules/grid_search.html

[76] "sklearn.model_selection.TimeSeriesSplit." [Online]. Available: https://scikit-learn/stable/modules/generated/sklearn.model_selection.TimeSeriesSplit.html

[77] Tree methods — xgboost 1.7.1 documentation. [Online]. Available: https://xgboost.readthedocs.io/en/stable/treemethod.html

[78] Parameters — LightGBM 3.3.3.99 documentation. [Online]. Available: https://lightgbm.readthedocs.io/en/latest/Parameters.html

[79] Welcome to SciKeras's documentation! — SciKeras 0.9.0 documentation. [Online]. Available: https://www.adriangb.com/scikeras/stable/

[80] J. Vogelzang and A. Stoffelen, "Quadruple collocation analysis of in-situ, scatterometer, and NWP winds," *Journal of Geophysical Research: Oceans*, vol. 126, no. 5, p. e2021JC017189, 2021, e2021JC017189 2021JC017189. [Online]. Available: https://agupubs.onlinelibrary.wiley.com/doi/abs/10.1029/2021JC017189

[81] A. Verhoef, J. Vogelzang, J. Verspeek, and A. Stoffelen, "PenWP User Manual and Reference Guide," 2018.

Appendix

A

Additional Tables

	Target grid	Half grid
Number of points longitude	2880	2880
Number of points latitude	1440	1440
First longitude coordinate	0.0625	0
Longitude increment	0.125	0.125
First latitude value	-89.9375	-90
Latitude increment	0.125	0.125

Table A.1: Target regular grid and half grid definitions

Variable	type	min	max
lon	sincos	NA	NA
lat	sin	NA	NA
eastward_wind	minmax	-20	20
northward_wind	minmax	-20	20
se_model_wind_divergence	minmax	-2.00E-04	2.00E-04
se_model_wind_curl	minmax	-2.00E-04	2.00E-04
se_model_speed	minmax	0	25
model_wind_to_dir	sincos	NA	NA
se_eastward_model_wind	minmax	-20	20
se_northward_model_wind	minmax	-20	20
msl	minmax	9.50E+04	1.05E+05
air_temperature	minmax	250	310
q	minmax	0	0.02
sst	minmax	270	305
uo	minmax	-1.5	1.5
vo	minmax	-1.5	1.5
sst_dx	minmax	-3.00E-04	3.00E-04
sst_dy	minmax	-3.00E-04	3.00E-04
current_speed	minmax	0	3
cos_sst_grad	copy		
cos_currents	copy		

Table A.2: Normalization of values

B

Software used in data preprocessing

Code B.1: Bufr2XmlNc run

```
genscat/tools/bufr2xmlnc/Bufr2XmlNc ascat_fullday_file.bufr
    ↪ output_file
--add_wind_div_rot
--add_asc_desc_flag
--adapt_to_cf_namelist
--select_date DATE
--add_uv_components
--get_modelwind_from_grib
--gribfile NWP_concatenated_file
--do_density_correction
```

Code B.2: nc_l2_to_l3 run

```
genscat/tools/nc_l2_to_l3/nc_L2_to_L3 L2_input_file
    ↪ L3_acending_output_file L3_descending_output_file
    ↪ l2_to_l3_modifications_file DATE
```

Code B.3: penwp run

```
penwp/execs/penwp_run -f input_buffer_file -nwpfl
    ↪ file_list_of_nwp_forecasts -noinv -noamb -verbosity 1
```

C

Configurations of models

Code C.1: XGBoost parameters

```
xgboost_params = {
        'early_stopping_rounds':  False,
        'booster':'gbtree',
        'tree_method':'hist',
        'n_estimators': 1500,
        'min_child_weight': 1,
        'gamma': 0.2,
        'subsample': .5,
        'colsample_bytree': 0.5,
        'max_depth': 10,
        'alpha': 0.2,
        'learning_rate': 0.01,
        'objective': 'reg:squarederror',
        'verbosity': 2
        }
```

Code C.2: MLP regressor parameters

```
mlp_params = {'activation': 'relu',
          'alpha': 0.005,
          'hidden_layer_sizes': (256, 128, 64, 32),
          'solver': 'sgd',
          'learning_rate_init': 0.01,
          'learning_rate': 'adaptive',
          'shuffle': True,
          'max_iter':500,
          'verbose': True,
          'tol': 1e-5,
          'early_stopping': True
```

```
        }
```

Code C.3: Tensorflow parameters

```
def get_reg(input_size, hidden_layer_sizes, dropout):
    model = keras.models.Sequential()
    model.add(keras.layers.Input(shape=(input_size,)))
    for hidden_layer_size in hidden_layer_sizes:
        model.add(keras.layers.Dense(hidden_layer_size, activation="
            ↪ relu"))
        model.add(keras.layers.Dropout(dropout))
    model.add(keras.layers.Dense(2))
    return model

epochs = 200
train_size = train_input.shape[0]
initial_learning_rate = 1e-4
final_learning_rate = 1e-5

learning_rate_decay_factor = (final_learning_rate /
    ↪ initial_learning_rate)**(1/epochs)
steps_per_epoch = int(train_size/batch_size)

lr_schedule = tf.keras.optimizers.schedules.ExponentialDecay(
                initial_learning_rate=initial_learning_rate,
                decay_steps=steps_per_epoch,
                decay_rate=learning_rate_decay_factor,
                staircase=True)

model = get_reg(input_size=train_input.shape[1], hidden_layer_sizes =
    ↪  (256, 128, 64, 32), dropout=0.25)

callbacks = [tf.keras.callbacks.EarlyStopping(monitor = 'val_loss',
    ↪ patience = 30, verbose = 1),
             tf.keras.callbacks.ModelCheckpoint(filepath=model_out +
                 ↪ 'model.{epoch:02d}-{val_loss:.4f}.h5'),
             tf.keras.callbacks.TensorBoard(log_dir=model_out + 'logs
                 ↪ ')]

adam = tf.keras.optimizers.Adam(learning_rate=lr_schedule)

model.compile(optimizer = adam,
              loss='mean_squared_error',
              metrics=['mean_squared_error', 'mean_absolute_error'],
               )

history = model.fit(train_input, # input data
          train_target,
          batch_size=batch_size,
          epochs=epochs,
```

```
        callbacks = callbacks,
        validation_data = (valid_input, valid_target),
        shuffle=True,
        validation_batch_size=1024,

    )
```