



UNIVERSITAT^{DE}
BARCELONA

Treball final de grau

**GRAUS SIMULTANIS EN MATEMÀTIQUES I
EN ENGINYERIA INFORMÀTICA**

**Facultat de Matemàtiques i Informàtica
Universitat de Barcelona**

**Sub-seasonal to seasonal climate
forecasting using Machine
Learning**

Autor: Sergi Bech Sala

Directors: Dr. Jordi Vitrià & Llorenç Lledó (BSC)

Realitzat a: Barcelona Supercomputing Center (BSC-CNS)

Barcelona, 24 de gener de 2022

Abstract

The main topic of this work is the study and the application of Machine Learning (ML) techniques to improve probabilistic forecasts of two-meter temperature and total precipitation at sub-seasonal scales (i.e. several weeks ahead) for the whole globe. We analyze the performance of a number of Machine Learning methods and finally we combine the best models to obtain the optimal prediction at each latitude, longitude, and for each lead time.

In addition, the results of this work have been presented to an open prize challenge launched by the World Meteorological Organization (WMO) to improve current forecasts of precipitation and temperature from state-of-the-art numerical weather and climate prediction models 3 to 6 weeks into the future using Artificial Intelligence.

Resum

El tema principal d'aquest treball és l'estudi i l'aplicació de diferents tècniques d'aprenentatge automàtic per tal de millorar a escala global les prediccions sub-estacionals (i.e. varies setmanes d'antelació) probabilístiques de temperatura a dos metres i precipitació total acumulada. Hem estudiat el rendiment de diferents models d'aprenentatge automàtic i finalment els hem combinat per tal d'obtenir un model final que ens proporciona la millor predicció per cada latitud, longitud i horitzó de pronòstic.

Adicionalment, els resultats d'aquest treball s'han presentat a una competició oberta organitzada per l'Organització Meteorològica Mundial (WMO) amb l'objectiu de fer servir la intel·ligència artificial per millorar les prediccions de la setmana 3 fins la setmana 6 obtingudes a partir dels millors models numèrics de previsió meteorològica i climàtica actuals.

Resumen

El tema principal de este trabajo es el estudio i aplicación de diferentes técnicas de aprendizaje automático con el objetivo de mejorar a escala global las predicciones sub-estacionales (i.e varias semanas de antelación) probabilísticas de temperatura a dos metros i precipitación total acumulada. Hemos estudiado el rendimiento de diferentes modelos de aprendizaje automático y finalmente los hemos combinado para obtener un modelo que nos proporcione la mejor predicción para cada latitud, longitud y horizonte de pronóstico.

Adicionalmente, los resultados de este trabajo se han presentado en un concurso abierto organizado por la Organización Meteorológica Mundial (WMO) con el objetivo de usar la inteligencia artificial para mejorar las predicciones de la semana 3 hasta la semana 6 obtenidas a partir de los mejores modelos numéricos de previsión meteorológica i climática actuales.

Acknowledgements

Primer de tot vull agrair a tota la gent amb qui he estat treballant durant la meva estada al Barcelona Supercomputing Center (BSC-CNS) i en especial a en Llorenç Lledó i a en Lluís Palma per guiar-me i ajudar-me en tot moment, ja que sense ells no hauria estat possible realitzar aquest treball. A més, també els hi vull donar les gràcies per tots els nous coneixements que m'han ensenyat sobre el món de les prediccions climàtiques i sobre computació amb superordinadors.

Per part de la UB vull agrair a en Jordi Vitrià per la seva ajuda i els seus consells sobre la realització d'aquest treball.

Finalment, també vull agrair als meus amics i a la meva família en especial als meus pares per tot el suport que sempre m'han donat al llarg d'aquest treball i de tota la carrera.

Contents

1	Introduction	1
1.1	Motivation	1
1.2	Thesis Organization	2
1.3	Temporal Horizons of Climate Predictions	3
1.4	Numerical Weather and Climate Predictions	4
1.5	Machine Learning Based Predictions	5
1.6	Contributions	6
2	State of the art	7
2.1	Data-driven Deep Learning Sub-seasonal Predictions with a Surrogate Atmosphere-only Model	7
2.2	Model Output Post-processing with Deep Learning	8
2.3	Sub-seasonal Climate Forecasting via Machine Learning	8
3	Theoretical Background	11
3.1	Numerical Weather	11
3.1.1	The Governing System of Equations of Climate Predictions	11
3.1.2	Approximations of the Equations	12
3.1.3	Numerical Solutions to the Equations	14
3.2	Machine Learning	18
3.2.1	Logistic Regression	18
3.2.2	Decision Trees	21
3.2.3	Ensemble Methods	23
3.3	Forecast Verification	28
3.3.1	Standard Metrics	28
3.3.2	Skill Scores	31
4	Experiments	33
4.1	Details of the WMO S2S-AI Challenge to Improve Sub-seasonal Predictions	33
4.1.1	Submissions	33
4.1.2	Verification Metrics	35
4.1.3	Possible Approaches	35
4.2	Datasets	36
4.2.1	Datasets Coordinates	36

4.2.2	Training Datasets	36
4.2.3	Forecast Datasets (ML Model Input Data for Prediction)	40
4.3	Methodology	41
4.3.1	Data Preprocessing	42
4.4	Methods	44
4.4.1	Climatology	44
4.4.2	Raw ECMWF Forecasts	44
4.4.3	Logistic Regression	45
4.4.4	Random Forest	46
4.5	Train-Test-Validation Strategy	47
4.6	Combination of Methods	48
4.6.1	In-sample Skill by Year	48
4.6.2	Final Model Forecasts	48
4.6.3	Challenge Results	50
4.7	Implementation	50
4.8	Post Challenge and Further Work	52
4.8.1	Unique Model	52
4.8.2	Teleconnection Indices as Predictors	52
4.8.3	XGBoost and Hyperparameter Tunning	53
4.8.4	Quantile Random Forest	53
4.8.5	Adding More Dynamical Models	54
5	Conclusions	55
6	Appendix	57
	Bibliography	61

Chapter 1

Introduction

People is constantly checking weather forecasts either on television or on the internet to know if it is going to be a rainy, warm or cold week and specially to know if the weather will be good enough to go out. Weather forecasts are also very relevant for many socio-economic sectors that are affected by atmospheric variability. In general, we are interested in what is likely to happen in the next 24 hours and up to 10 days ahead, which is what current numerical weather prediction models are able to anticipate. Forecasts at longer lead times can be produced with climate models: seasonal forecasts tell us what will likely happen in the coming seasons, and sub-seasonal forecasts tell us what is likely to happen between two weeks to two months from now.

Compared to other time ranges, sub-seasonal predictions have received much less attention so far, maybe due to the false belief that they are the least useful or because of its greater difficulty. However, putting some effort trying to improve this predictions is going to be very valuable to our society. So, in this work we focus on how to improve the skill of sub-seasonal predictions using machine learning.

1.1 Motivation

Many socio-economic sectors such as energy, agriculture, health, logistics or tourism are affected by the atmospheric conditions at weekly timescales. For instance, in July 2013 a heat wave hit Germany. As temperatures went up, more energy was needed for cooling buildings, but simultaneously the winds slowed down and the many wind farms in the country could not keep up with electricity demand during several days. In March 2018 a cold spell called the Beast from the East brought low temperatures and heavy snowfall across all Europe, increasing energy demand. Events like this show that renewable energy is sensitive to variations in atmospheric conditions. If we want to transition to sustainable energy sources we need better climate predictions weeks in advance.

Extreme weather and climate events like high winds, flooding, heavy snowfall, heat waves, droughts, and wildfires can cause a lot of economic losses, damaging infrastructures such as roads, buildings, railways or telecommunication towers, but more importantly posing human lives at risk. Those events also threaten the security of water, food

and energy supply. The agriculture sector needs better sub-seasonal predictions in order to make early decisions and avoid as much as possible the damaging effects of bad weather on food production. Water resource allocation and management would also benefit from better predictions.

If we add the fact that climate change is increasing the variability of the weather and making extreme events stronger and more frequent, now more than ever, sub-seasonal predictions are very important to anticipate extreme weather and climate events weeks ahead. Better predictions will allow to develop early warning systems that will help the humanitarian sector to react systematically saving lives and minimizing socio-economic risks.

Nowadays, with growing popularity and rapid advances in the machine learning and deep learning fields it is the perfect moment to put more effort trying to improve the skill of sub-seasonal predictions. The main idea is to develop new models that combine state-of-the-art sub-seasonal predictions with Artificial Intelligence methods.

1.2 Thesis Organization

This work is organized in five chapters:

1. **Introduction:** This chapter contains an overview on some important topics about climate predictions in order to better understand our objective and the essentials about forecasts and how we can use machine learning to improve them. We also describe the main contributions of this work.
2. **State of the Art:** The second chapter collects some interesting papers related to this work and we give a very brief explanation in order to have an idea of the state-of-the-art methods.
3. **Theoretical Background:** The third chapter contains some theoretical background to put in context and better understand this thesis. It is divided in three sections that are related to our work: Numerical Weather and Climate Prediction, Machine Learning and Forecast Verification.
4. **Experiments:** The fourth chapter gives all the details about the problem that we want to solve, the datasets we have used, the methodology and methods we have developed and the results we have obtained.
5. **Conclusions:** The last chapter summarizes all the work and give conclusions of the research done.

The project schedule for this work is represented with a Gantt chart in figure 1.1.

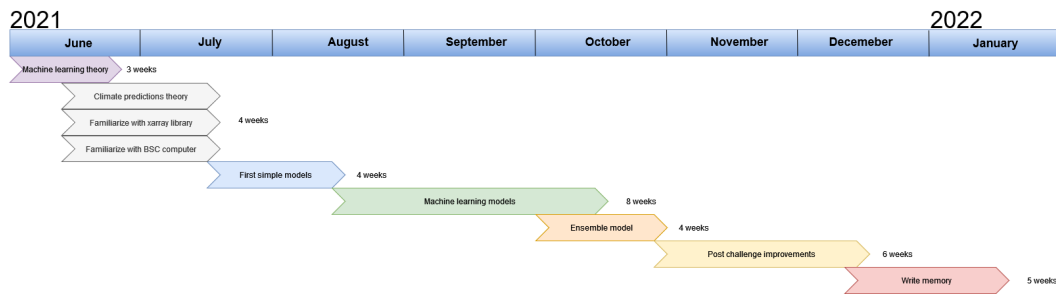


Figure 1.1: Gantt Chart showing the planification of this work.

1.3 Temporal Horizons of Climate Predictions

For our work we are only interested in sub-seasonal predictions but in order to understand our problem it is useful to clarify the different temporal horizons of weather and climate science and the different sources of variability that affect each time range and allow making predictions. Depending on the time scale of the predictions, they can be split into different categories: weather forecasts, climate predictions and climate projections.

- **Weather forecasts** provide hourly or daily information from 24 hours up to 15 days ahead. They deal with meteorological phenomena in very high detail at any particular time and location. For example, they can predict tomorrow's temperature in Barcelona at eight o'clock. However, they quickly lose accuracy after a week or so. The principal source of predictability for weather predictions is the current atmospheric conditions, so the results are heavily dependent on the initial atmospheric conditions.
- **Climate predictions** provide time-aggregated information (e.g. weekly, monthly or annual means) and extend from two weeks to seasons and even up to decades. Climate predictions rely on physical processes that occur in our planet at slower timescales than weather variability—such as the evolution of sea ice, sea surface temperature or soil moisture—to deduce the likely impacts on surface variables that affect our lives. So, the interest of climate predictions is not the particular temperature for a particular day and place, but in how the average temperature in that location will behave. The results of climate predictions are less sensitive to initial atmospheric conditions, and the initial state and evolution of the ocean, land surface and sea ice is used as boundary conditions that force the atmosphere.
 - In this time range we have the **sub-seasonal predictions** which cover from two weeks up to two months ahead. It is considered a difficult time range since the lead time is sufficiently long so that much of the memory of the atmospheric initial conditions is lost and it is too short for the variability of the ocean to have a strong influence. However, potential sources of predictability for this time range are the Madden Julian oscillation (MJO), El Niño/Southern Oscillation (ENSO) evolution, soil moisture, snow cover and sea ice, stratosphere–troposphere interactions, ocean conditions and other tropical-extratropical teleconnections.

- **Climate projections** extend further in the future, from decades up to centuries. The objective of climate projections is to predict the global evolution of Earth's climate and study phenomena such as climate change. These predictions require scenario hypotheses which are based on future projected levels of greenhouse gases and socio-economic development. These scenarios are the predominant factor in order to provide plausible descriptions of how the future may evolve.

For each of these time scales we can differentiate between real-time operational predictions and retrospective forecasts also called hindcasts:

- **Operational forecasts** are produced regularly using state-of-the-art models, and are timely disseminated to assist decision making and produce early warning.
- **Hindcasts** are retrospective forecasts, i.e. predictions initialized in the past but using the state-of-the-art operational model. Since the observations for the past are already available, hindcasts are used to understand the performance of the operational models by comparing the past forecasts with the corresponding observations in order to determine how well they match the observed results. This information is very valuable to know if the model performs well or not because testing the model accuracy provides information about the quality of future forecasts. Hindcasts are also useful to determine known biases and limitations of the operational predictions, which can then be corrected with statistical or machine learning techniques.



Figure 1.2: The temporal horizons of weather and climate science. Image source: BSC Earth System Services spreadsheet

1.4 Numerical Weather and Climate Predictions

Weather forecasts and climate predictions are produced with mathematical models that simulate the evolution of the atmosphere, the ocean, the land surface, and the cryosphere and all its interactions, given the current state as initial conditions. The use of an atmospheric model to make predictions is called Numerical Weather Prediction (NWP) whereas the usage of coupled atmosphere-ocean-land-cryosphere models is called Numerical Climate Prediction or simply Climate Modelling. These weather and climate models use systems of differential equations based on the laws of physics, and in order to run they require a very powerful supercomputer. However, even with a very powerful computer the models have the problem of being very dependant on the density and the quality of

the current observations. Another problem of these numerical models is that it is impossible to find an exact solutions to the system of equations due to the chaotic nature of the partial differential equations that govern the atmosphere. This limitation causes small errors to spread and grow a lot, limiting the performance of longer-term predictions. In order to mitigate this problem a method called ensemble forecasting is used to obtain useful results further into the future.

Ensemble Forecasting

Ensemble forecasting is a form of Monte Carlo analysis and the principal idea of this method is making a set (ensemble) of forecasts with the aim of giving an indication of the range of possible future states of the atmosphere. Different ensemble members are produced by using slightly different but equally plausible initial conditions, and also by slightly tuning the model parameters within the range of plausible values. Analyzing the distribution of results of these multiple simulations can reduce the two principal sources of uncertainty in forecast models:

- The sensitive dependence on initial conditions due to observational errors and the chaotic nature of the differential equations.
- The approximate numerical mathematical methods to solve the system of differential equations.

1.5 Machine Learning Based Predictions

Machine learning methods can be very useful in order to improve the performance of sub-seasonal predictions. Although there may be multiple different ways to apply machine learning in order to make or improve weather or climate predictions here we describe two general approaches:

- **Data-driven:** Use observed conditions to train a model that predicts future states. The model learns what variables can explain future climate behaviour, i.e. predictability sources, without modelling physical processes. In other words, with this approach we are not using the numerical weather predictions and we only rely on using machine learning in order to make predictions.
- **Recalibration:** Start from forecasts made with mathematical models and improve or re-calibrate them using machine learning post-processing methods. The model learns the weaknesses of the numerical weather predictions and corrects them. In other words, with this approach we are combining the numerical weather predictions and the machine learning techniques in order to make predictions.

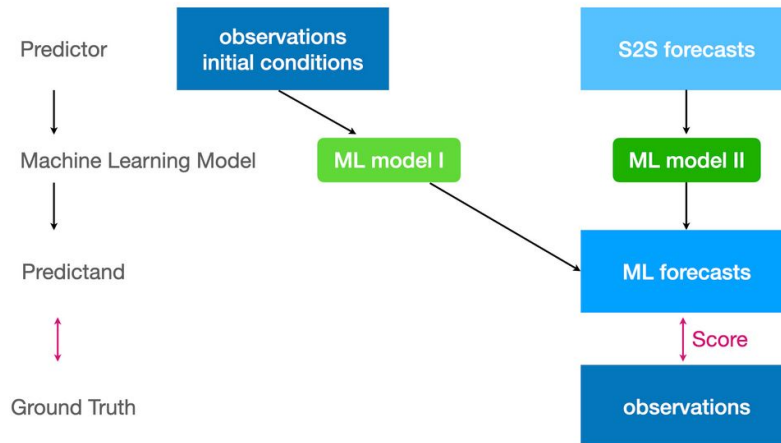


Figure 1.3: Scheme of the two possible ML approaches for weather forecasting and climate predictions. Image source: <https://s2s-ai-challenge.github.io/>

1.6 Contributions

The main objectives and contributions that this project accomplishes are the following:

- We study the performance of different machine learning techniques applied to sub-seasonal predictions of two-meter temperature and total precipitation.
- We provide probabilistic forecasts of bi-weekly aggregated temperature and precipitation for weeks 3-4 and weeks 5-6 during 2020 and at a global scale.
- We make an ensemble of different machine learning models in order to provide a final method that selects the best model at each grid point (each latitude and longitude).
- We develop a general, reproducible and commented *python* implementation with all source code openly available. The implementation combines *sklearn*, *xarray* and *dask* in a way that can make easier the process to apply machine learning on gridded global climate data with dimensions time, longitude, latitude and with outputs in longitude and latitude.
- We submitted our final model to the S2S-AI-Challenge organized by the World Meteorological Organization (WMO) and we obtained the second place.

Chapter 2

State of the art

In this chapter we summarize the existing literature that uses machine learning or deep learning in order to make sub-seasonal predictions, either from observations or by post-processing dynamical predictions from numerical model output.

It is important to note that the field of sub-seasonal predictions with artificial intelligence is still growing. The high-dimensionality of the data, the strong spatio-temporal relationships and the limited number of training samples available are three of the main challenges to overcome. Some of the state-of-the-art methods are very difficult to reproduce with our constraints and time limitation.

2.1 Data-driven Deep Learning Sub-seasonal Predictions with a Surrogate Atmosphere-only Model

Starting with sub-seasonal forecasting with deep learning (i.e. no dynamical model employed here) we have the work presented by Jonathan A. Weyn et al. (2021) [14]. They produce an ensemble prediction by training a surrogate model that uses a Deep Learning Weather Prediction model to predict the time evolution of six atmospheric variables. The surrogate model is initialized many times to produce an ensemble. This model is very computationally efficient and uses convolutional neural networks (CNNs) on a cubed sphere grid to produce global forecasts.

The ensemble forecasts for lead times of 4 and 5-6 weeks obtained with this model performs slightly worse than the European Centre for Medium Range Weather Forecasts (ECMWF) numerical predictions. At shorter lead times (weather timescales), the ECMWF ensemble performs much better. However, unlike numerical weather predictions, this method can generate very large ensembles at a fraction of the computational cost so it is a very promising approach in order to improve skill of sub-seasonal predictions in future works.

2.2 Model Output Post-processing with Deep Learning

The paper from Keran Chen et al. (2020) [2] proposes a model output deep learning (MODL) method for post-processing in order to improve the skill of temperature forecasts in Tianjin area. Even though the objective of this paper is weather forecasting up to 10 days it is still interesting to see how they improve forecasts using deep learning. This MODL is a post-processing method based on deep convolutional neural network, which directly learns the mapping relationship between the forecast output by numerical model and the observation temperature in order to obtain more accurate temperature forecasts. The MODL method is compared with the ECMWF forecast, Model Output Statistics (MOS) methods, and Model Output Machine Learning (MOML) methods and the accuracy is in general higher.

Another paper that presents a MODL is the one by Michael Steininger et al. (2020) [11]. In this case the time range objective is climate projections. They propose a CNN architecture named convMOS specifically designed for reducing errors in climate models outputs and apply it to the Germany regional climate model.

These two papers are a clear example that deep learning is used for weather predictions and climate projections but it is still not very used for climate predictions.

2.3 Sub-seasonal Climate Forecasting via Machine Learning

Analysis of Different Machine Learning and Deep Learning Methods

This article by Sijie He et al. (2020) [5] studies a variety of machine learning and deep learning approaches for sub-seasonal forecasting. They show that even though deep learning models provide good results and have great potential for improvements are not always better than machine learning methods, e.g. XGBoost and random forest. This paper also explains some challenges of sub-seasonal predictions and provides very useful information on which predictors have more importance for the models.

Data-driven and Post-process Machine Learning

Another important paper related to machine learning is the paper by Jessica Hwang et al. (2020) [6] where they present an ensemble of two nonlinear regression models in order to improve sub-seasonal forecasting in the Western U.S. The first nonlinear regression post-process sub-seasonal forecasts integrating also some meteorological measurements and pruning irrelevant predictors using a multi task feature selection process. The second model is a data-driven weighted local linear regression which only uses historical observations of the target variable. Another important contribution is that they provide to the community the dataset they constructed in order to train all the models.

The Importance of Machine Learning for Sub-seasonal Predictions

Finally, related to machine learning and sub-seasonal forecasts there is an opinion article by Judah Cohen et al. (2018) [3] which highlights the reasons why government-sponsored forecast centers should dedicate more efforts and attention to machine learning methods. The article says that machine learning is very powerful at mining data and better at recognizing patterns and it shows many examples where machine learning methods beat the state of the art dynamical numerical models. Finally, they argue that forecast centers should put more effort in hybrid techniques utilizing both state of the art dynamical models and updated machine learning methods to improve sub-seasonal predictions.

Chapter 3

Theoretical Background

In this chapter we start explaining the theoretical basis of numerical weather prediction in order to have a better understanding of how dynamical weather and climate predictions are produced and why machine learning can improve their weaknesses. Next, we focus on explaining and constructing several machine learning methods that we have implemented to solve our problem. Finally, we give some theory about forecast quality assessment.

3.1 Numerical Weather

3.1.1 The Governing System of Equations of Climate Predictions

We start describing the governing system of equations called primitive equations which are the basis for numerical weather and climate prediction models. It is important to say that these equations cannot be solved analytically so they must be converted into a form that can be solved with numerical methods.

The equations 3.1, 3.2 and 3.3 correspond to spherical Earth momentum equations and state that the rate of change of momentum of a body is proportional to the resultant force acting on the body, and is in the same direction as the force. Equation 3.4 is the thermodynamic energy equation which take account various effects on temperature. The equation 3.5 is the continuity equation for total mass which states that mass is neither gained nor destroyed and equation 3.6 states the same but applied for water vapor. Finally, equation 3.7 is the ideal gas law which relates temperature, pressure and density of a gas. More equations can be added to this system if needed to model other physical quantities for example, cloud water, cloud ice and the different types of precipitation.

$$\frac{\partial u}{\partial t} = -u \frac{\partial u}{\partial x} - v \frac{\partial u}{\partial y} - w \frac{\partial u}{\partial z} + \frac{uv \tan \theta}{a} - \frac{uw}{a} - \frac{1}{\rho} \frac{\partial p}{\partial x} - 2\Omega(w \cos \theta - v \sin \theta) + Fr_x \quad (3.1)$$

$$\frac{\partial v}{\partial t} = -u \frac{\partial v}{\partial x} - v \frac{\partial v}{\partial y} - w \frac{\partial v}{\partial z} + \frac{u^2 v \tan \theta}{a} - \frac{uv}{a} - \frac{1}{\rho} \frac{\partial p}{\partial y} - 2\Omega u \sin \theta + Fr_y \quad (3.2)$$

$$\frac{\partial w}{\partial t} = -u \frac{\partial w}{\partial x} - v \frac{\partial w}{\partial y} - w \frac{\partial w}{\partial z} - \frac{u^2 + v^2}{a} - \frac{1}{\rho} \frac{\partial p}{\partial z} + 2\Omega u \cos\theta - g + Fr_z \quad (3.3)$$

$$\frac{\partial T}{\partial t} = -u \frac{\partial T}{\partial x} - v \frac{\partial T}{\partial y} + (\gamma - \gamma_d)w + \frac{1}{c_p} \frac{dH}{dt} \quad (3.4)$$

$$\frac{\partial \rho}{\partial t} = -u \frac{\partial \rho}{\partial x} - v \frac{\partial \rho}{\partial y} - w \frac{\partial \rho}{\partial z} - \rho \left(\frac{\partial u}{\partial x} + \frac{\partial v}{\partial y} + \frac{\partial w}{\partial z} \right) \quad (3.5)$$

$$\frac{\partial q_v}{\partial t} = -u \frac{\partial q_v}{\partial x} - v \frac{\partial q_v}{\partial y} - w \frac{\partial q_v}{\partial z} + Q_v \quad (3.6)$$

$$P = \rho RT \quad (3.7)$$

The independent variables u , v , and w are the Cartesian velocity components, p is pressure, ρ is density, T is temperature, q_v is specific humidity, Ω is the rotational frequency of Earth, θ is latitude, a is the radius of Earth, γ is the lapse rate of temperature, γ_d is the dry adiabatic lapse rate, c_p is the specific heat of air at constant pressure, g is the acceleration of gravity, H represents a gain or loss of heat, Q_v is the gain or loss of water vapor through phase changes, and Fr is a generic friction term in each coordinate direction.

3.1.2 Approximations of the Equations

As we have said the primitive equations can only be solved using numerical methods and even so are very computationally demanding. It is common for operational models to use some approximations or remove some terms to the equations in order to achieve a model that it is still accurate but more efficient to solve numerically.

Introduction to Reynold's Equations

The system of equations that we have just described apply to all scales of motion, even on waves and turbulence that are too small to affect weather processes. The above equations are typically modified so that they only apply to larger non-turbulent motions. To do that we have to split all the dependent variables into mean and turbulent parts where the mean is defined as a time average over a grid cell. For example:

$$\begin{aligned} u &= \bar{u} + u' \\ v &= \bar{v} + v' \\ w &= \bar{w} + w' \\ T &= \bar{T} + T' \\ \rho &= \bar{\rho} + \rho' \end{aligned}$$

Now, we have to substitute these expressions into equations 3.1-3.7. For example, the expanded form for the first term on the right side of equation 3.1 is:

$$u \frac{\partial u}{\partial x} = (\bar{u} + u') \frac{\partial}{\partial x} (\bar{u} + u') = \bar{u} \frac{\partial \bar{u}}{\partial x} + \bar{u} \frac{\partial u'}{\partial x} + u' \frac{\partial \bar{u}}{\partial x} + u' \frac{\partial u'}{\partial x} \quad (3.8)$$

Then, if we want the equations belong to the mean motion (non-turbulent weather scales) we have to apply an averaging operator to all the terms. For example, for the term 3.8 we have:

$$\overline{u \frac{\partial u}{\partial x}} = \overline{u} \frac{\partial \overline{u}}{\partial x} + \overline{u} \frac{\partial u'}{\partial x} + \overline{u'} \frac{\partial \overline{u}}{\partial x} + \overline{u'} \frac{\partial u'}{\partial x} \quad (3.9)$$

Then, using the Reynold's postulates (Reynolds 1895, Bernstein 1966) which state that for variables a and b :

$$\begin{aligned} \overline{a'} &= 0 \\ \overline{\overline{a}} &= \overline{a} \\ \overline{\overline{ab}} &= \overline{\overline{a}} \overline{\overline{b}} = \overline{a} \overline{b} \\ \overline{\overline{ab'}} &= \overline{\overline{a}} \overline{\overline{b'}} = \overline{a} \overline{b'} = 0 \end{aligned}$$

we can simplify equation 3.9 as follows:

$$\overline{u \frac{\partial u}{\partial x}} = \overline{u} \frac{\partial \overline{u}}{\partial x} + \overline{u'} \frac{\partial u'}{\partial x} \quad (3.10)$$

Now, applying the averaging process to all terms in the equations 3.1 in particular, we represent each dependent variable by the sum of a resolved mean and an unresolved turbulent component, and then apply the averaging operator we obtain the following equation:

$$\frac{\partial \overline{u}}{\partial t} = -\overline{u} \frac{\partial \overline{u}}{\partial x} - \overline{v} \frac{\partial \overline{u}}{\partial y} - \overline{w} \frac{\partial \overline{u}}{\partial z} - \frac{1}{\overline{\rho}} \frac{\partial \overline{\rho}}{\partial x} - \overline{u'} \frac{\partial u'}{\partial x} - \overline{v'} \frac{\partial u'}{\partial y} - \overline{w'} \frac{\partial u'}{\partial z} - \overline{\rho'} \frac{\partial \rho'}{\partial x} + \overline{otherterms} \quad (3.11)$$

Finally, more transformations and simplifications can be done in order to obtain a more simplified equation but showing this transformation it is enough to understand that now the dependent variable represent only nonturbulent motions. Analogous transformations can be done to the other equations to express only the nonturbulent motions of the dependent variable.

Other Approximations

Here we are going to give two more examples of other approximations that can be done in order to simplify some of the primitive equations:

- **Hydrostatic approximation:** Because sound waves are generally not important for meteorological predictions, it is better to use a form of the equations that does not admit them. One possible approach is to use the hydrostatic approximation where we have to replace the complete third equation of motion 3.3 by the following one containing only gravity and vertical pressure gradient terms.

$$\frac{\partial p}{\partial z} = -\rho g$$

- **Boussinesq approximation:** This approximation also is used to filter sound waves in the equation 3.5 and is obtained by substituting the following equality into 3.5:

$$\frac{\partial u}{\partial x} + \frac{\partial v}{\partial y} + \frac{\partial w}{\partial z} = 0$$

3.1.3 Numerical Solutions to the Equations

The primitive equations are usually solved at points defined by a quasi-regular, three dimensional spatial grid. The term quasi-regular means that the points are not exactly equally spaced, for example when using latitude-longitude coordinates or adaptive grids where we have more resolution in areas of strong gradients. The time axis is also defined by discrete and evenly spaced intervals of time, or time-steps. Given this framework, one common way to approximate the equations numerically is by using finite-difference methods. Here we are going to give a brief example on how to solve on a grid a slightly simplified version of the first equation of motion 3.1. Note that this equation is a nonlinear, non-homogeneous, partial differential equation that cannot be solved analytically.

$$\frac{\partial u}{\partial t} = -u \frac{\partial u}{\partial x} - v \frac{\partial u}{\partial y} - w \frac{\partial u}{\partial z} - \frac{1}{\rho} \frac{\partial p}{\partial x} + fv + Fr_x \quad (3.12)$$

In order to solve the equation 3.12, it will be represented using a simple three-point centered difference approximation in time and space:

$$\frac{\partial}{\partial y} f(x, y, z, t) = \frac{f_{i,j+1,k}^\tau - f_{i,j-1,k}^\tau}{y(j+1) - y(j-1)} = \frac{f_{j+1}^\tau - f_{j-1}^\tau}{2\Delta y} \quad (3.13)$$

where f is any dependent variable; τ defines a discrete point in the time axis; i, j, k define coordinates on the x, y, z space axes respectively and Δy is the distance between two adjacent points on the y axis.

Using the finite-difference method (3.13) we can transform the equation 3.12 in to the next solvable equation:

$$\frac{\partial u}{\partial t} = \frac{u_{i,j,k}^{\tau+1} - u_{i,j,k}^{\tau-1}}{2\Delta t} = -u_{i,j,k}^\tau \frac{u_{i+1,j,k}^\tau - u_{i-1,j,k}^\tau}{2\Delta x} - v_{i,j,k}^\tau \frac{u_{i,j+1,k}^\tau - u_{i,j-1,k}^{\tau-1}}{2\Delta y} \quad (3.14)$$

$$-w_{i,j,k}^\tau \frac{u_{i,j,k+1}^\tau - u_{i,j,k-1}^{\tau-1}}{2\Delta z} - \frac{1}{\rho_{i,j,k}^\tau} \frac{p_{i+1,j,k}^\tau - p_{i-1,j,k}^{\tau-1}}{2\Delta x} + fv_{i,j,k}^\tau + Fr_{x(i,j,k)}^{\tau-1} \quad (3.15)$$

where Δx and Δy are often assumed to be the same and are chosen in order to have a sufficient number of grid points to represent the meteorological feature of our interest which can vary depending on the particular application of our model. There is also some criteria in order to choose the best time step to have numerical stability. For example, the Courant-Friedrichs-Lewy criterion requires that $\frac{U\Delta t}{\Delta x} \leq 1$, where U is the horizontal speed of the fastest wave on the grid which can be estimated using different techniques depending on the model. Note that this single equation has to be solved for each time step at each grid point so the number of time steps and the grid resolution control the computational requirements. In addition, because the model atmosphere cannot extend to infinity it is necessary to define some upper-boundary conditions for the models.

Now, after applying the finite-difference method, the resultant equation can be solved for each grid point for $u_{i,j,k}^{\tau+1}$ and the right side is evaluated based on the values of the dependent variables from the two previous time iterations τ and $\tau - 1$. Note that because atmospheric modeling is an initial value problem we have to specify the state of the dependant variables at the beginning of the integration of the equations. Correctly initializing the model is a very important step that has consequences for forecast accuracy. For example, one can assume that in general the quality of the forecasts cannot be better than the quality of initial conditions. It is also important to notice that to integrate in time we need two time iterations τ and $\tau - 1$ so we have to do the first step using a forward difference method.

Types of Grids

It is important to consider that in order to solve the equations different grids or map projections can be used and each one have its pros and cons. Some commonly used map projections are the cylinder (Mercator), right-circular cone (Lambert conformal) and plane (polar stereographic) and each one may be better depending on the particular application of the numerical model. For example, figure 3.1 shows that the points on the polar stereographic grid (computational grid) are equidistant but the physical distance is generally different which is a disadvantage of using this grid.

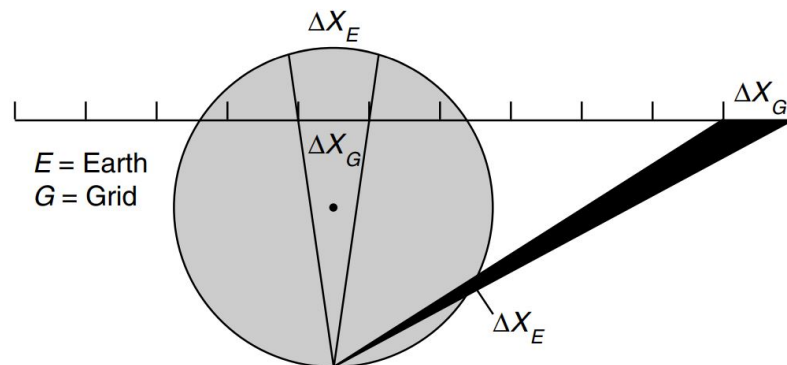


Figure 3.1: Distance relationship between two points on Earth and on the polar stereographic grid. Image Source: [13]

Another different type of grids are the latitude-longitude grids which as you can see at figure 3.2 have the problem that the longitudinal distance between grid points becomes progressively smaller as the meridians converge at the poles. This problem requires to take small time steps to maintain numerical stability which leads to a lot more computations.

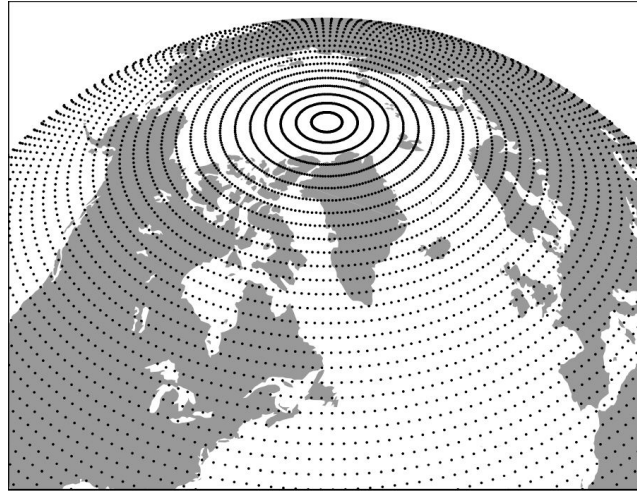


Figure 3.2: Image of a latitude–longitude grid for a part of the sphere. Image source:[13]

There are a lot more different type of grids like spherical geodesic grids, grids that are combinations of different projections, grids with gradual changes in resolution, etc... In order to see more information about this topic check the bibliography [13].

Truncation Error

In a grid point model if we approximate the equations using finite-difference the derivative is not perfectly estimated because we have truncation errors which can lead to wrong modelling of some physical interactions. The magnitude of such errors depends on how good is our approximation and a common way to quantify this errors is to use the Taylor's series polynomial:

$$f(x) = f(a) + (x - a) \frac{\partial f(a)}{\partial x} + \frac{(x - a)^2}{2!} \frac{\partial^2 f(a)}{\partial x^2} + \frac{(x - a)^3}{3!} \frac{\partial^3 f(a)}{\partial x^3} + \dots$$

$$+ \dots + \frac{(x - a)^n}{n!} \frac{\partial^n f(a)}{\partial x^n} + R(n, x)$$

Where f can be any meteorological variable in the derivative terms of equations 3.1-3.6 and the series can be written for any dependent variable. Note that for a series truncated at n terms there is a residual R that defines the truncation error. Next, we are going to give an example on how to define the truncation error for two-point and three-point finite-differences:

For the two-point approximation let $x = a + \Delta x$ and truncate the series dropping

second-order and higher order terms to obtain:

$$f(a + \Delta x) = f(a) + (\Delta x) \frac{\partial f(a)}{\partial x}$$

Now, we just have to solve for the derivative to obtain what is called the forward-in-space differencing formula which has a first order accuracy because we truncate the second order and higher terms.

$$\frac{\partial f(a)}{\partial x} = \frac{f(a + \Delta x) - f(a)}{\Delta x} \quad (3.16)$$

The backward-in-space formula is obtained doing an analogous process letting $x = a - \Delta x$.

The three-point approximation can be obtained writing the backward and forward series and subtracting them just as follows:

$$\begin{aligned} f(a + \Delta x) &= f(a) + (\Delta x) \frac{\partial f(a)}{\partial x} + \frac{(\Delta x)^2}{2!} \frac{\partial^2 f(a)}{\partial x^2} + \frac{(\Delta x)^3}{3!} \frac{\partial^3 f(a)}{\partial x^3} + \dots \\ f(a - \Delta x) &= f(a) - (\Delta x) \frac{\partial f(a)}{\partial x} + \frac{(\Delta x)^2}{2!} \frac{\partial^2 f(a)}{\partial x^2} - \frac{(\Delta x)^3}{3!} \frac{\partial^3 f(a)}{\partial x^3} + \dots \end{aligned}$$

and subtracting the two series to obtain:

$$f(a + \Delta x) - f(a - \Delta x) = 2\Delta x \frac{\partial f(a)}{\partial x} + 2 \frac{(\Delta x)^3}{3!} \frac{\partial^3 f(a)}{\partial x^3} + \dots$$

and solving for the first order derivative:

$$\frac{\partial f(a)}{\partial x} = \frac{f(a + \Delta x) - f(a - \Delta x)}{2\Delta x} - \frac{(\Delta x)^2}{3!} \frac{\partial^3 f(a)}{\partial x^3} - \dots \quad (3.17)$$

Finally, if we truncate equation 3.17 after the first term on the right we obtain a three-point approximation to the derivative which we say it has second order accuracy because we ignore the third-order and higher terms.

$$\frac{\partial f(a)}{\partial x} = \frac{f(a + \Delta x) - f(a - \Delta x)}{2\Delta x} \quad (3.18)$$

Now, one way of calculating the truncation error of the series is to compute the ratio between the value of the derivative from 3.18 and the exact value of the derivative. This ratio defines the truncation error in the finite-difference approximation.

Finally, it is important to say that there are other artifacts due to the numerical approximation of the equations, for example linear stability and aliasing which we are not going to cover on this work and for more information just check the bibliography [13].

Spectral Methods

Another different approach to grid points methods are the spectral methods which consist in transforming the equations into spectral form by substituting the dependent variables with finite expansions using double Fourier series or Fourier-Legendre functions.

Such models are initialized by first transforming the dependent variables from physical space (grid point values) to the transform space (expansion coefficients) and then solving the ordinary differential equations integrating them forward in time using conventional finite differencing. Finally, in order to obtain interpretable forecasts we have to transform back the solutions into physical space.

In order to improve the spectral methods also exists some pseudospectral models that treat some processes in spectral space and some other in the grid point space, this approach is also called transform method because it involves transforming from one space to another at every time step.

3.2 Machine Learning

In this chapter we are going to explain the details of some actual machine learning methods that can be used to improve sub-seasonal predictions. First, we introduce some notation that it is going to be useful for the next sections:

- We put $m \in \mathbb{N}$ to represent the number of training examples and $n \in \mathbb{N}$ the number of features.
- We define X as the training input dataset and Y as our output (target) dataset. So the notation for one single training example would be $(x^{(i)}, y^{(i)})$ where $x^i \in \mathbb{R}^n$ is the feature vector, $y^{(i)} \in \mathbb{R}$ is the observation and $i \in 1, 2, 3, \dots, m$.
- It is common to add a feature with value 1 in our feature vector in order to work always with the noise or bias factor. Often we just write $x \in \mathbb{R}^{n+1}$ in order to refer to the expanded vector $(1, x)$ where $x \in \mathbb{R}^n$ is the feature vector of one training sample.
- We write Θ to represent the parameters of the model where $\Theta \in \mathbb{R}^k$ with $k \in \mathbb{N}$. To refer to one single parameter we write θ_l where $\theta_l \in \Theta$ and $l \in 0, 1, 2, 3, \dots, k - 1$.
- We use $h_\theta(x)$ to refer to the hypothesis function of the model. For example, the hypothesis function for a one variable linear regression would be $h_\theta(x) = \theta_0 + \theta_1 x$.
- We put $J(\Theta)$ as the cost function we want to minimize/maximize for the specific problem.

3.2.1 Logistic Regression

Logistic regression is a machine learning or statistic method that can be used to model the probability of an event to belong to a certain class for example, rain/no rain, win/lose, or even can be extended to model several classes of events for example, below-normal/normal/above-normal conditions. So, logistic regression can be used to solve classification problems either binary classification or multiclass classification where we label the output classes with numbers $\{0, 1, 2, \dots\}$.

Binary Classification

To begin with we are going to explain how the logistic regression algorithm for binary classification is constructed and then we are going to explain how to extend it for multi-class problems. For the moment, we can assume that we only have two classes as outputs, that is $y^{(i)} \in \{0, 1\}$.

Given an input value we are interested in obtaining the probabilities to belong at each class so, we want that the output of our hypothesis function take values between 0 and 1, that is we want $h_\theta(x) \in [0, 1]$. In order to achieve that we are going to choose the following hypothesis function where we assume that we have the bias factor and one parameter for each feature, that is $\Theta \in \mathbb{R}^k$ with $k = n + 1$ and $x \in \mathbb{R}^{n+1}$ is the expanded features vector:

$$h_\theta(x) = g(\Theta^T x) = \frac{1}{1 + e^{-\Theta^T x}} \quad (3.19)$$

where $g(z) = \frac{1}{1+e^{-z}}$ is the sigmoid function (logistic function) and $\Theta^T x$ is the notation of the scalar product between the parameters vector and the expanded vector.

Since sigmoid function take values between (0, 1) it is clear that our hypothesis function it is going to take values between [0, 1] as we want (see fig. 3.3).

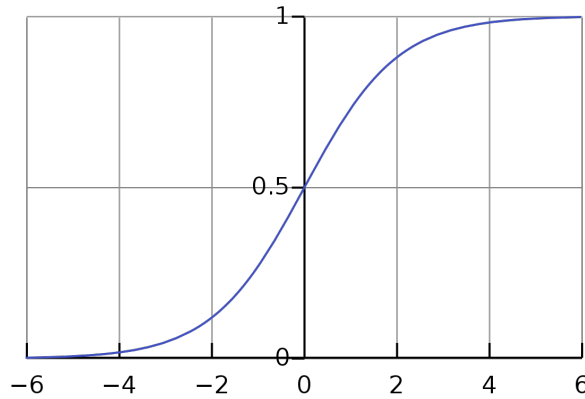


Figure 3.3: Graphic of the sigmoid function.

In order to better understand what our hypothesis function is doing, we can think that it gives the probability of $y = 1$ conditioned to x was observed (feature vector x parameterized by parameter θ), that is $P(y = 1|x;\theta) = h_\theta(x)$. We also have that $1 - h_\theta(x) = P(y = 0|x;\theta)$ and we can take these two equations and compress them into only one:

$$P(y|x;\theta) = h_\theta(x)^y (1 - h_\theta(x))^{1-y} \quad (3.20)$$

Now, we are going to use the maximum likelihood estimation as our cost function:

$$J(\Theta) := \mathcal{L}(\Theta) = P(Y|X;\Theta) = \prod_{i=1}^m P(y^{(i)}|x^{(i)};\Theta) = \prod_{i=1}^m h_\theta(x^{(i)})^{y^{(i)}} (1 - h_\theta(x^{(i)}))^{1-y^{(i)}}$$

Our objective is to find the value of Θ that maximizes the likelihood of the parameters and in order to make the computations easier we are going to take the log likelihood:

$$l(\Theta) := \ln(\mathcal{L}(\Theta)) = \sum_{i=1}^m y^{(i)} \ln(h_{\theta} x^{(i)}) + (1 - y^{(i)}) \ln(1 - h_{\theta} x^{(i)}) := J(\Theta)$$

Now we can use the gradient ascent algorithm in order to find the parameters Θ that maximizes $l(\Theta)$, that means we are going to update the parameters according to:

$$\theta_l = \theta_l + \alpha \frac{\partial}{\partial \theta_l} l(\Theta)$$

where $\alpha > 0$ is known as the learning rate. It is important to note that with the choose of the logistic function to represent our hypothesis function we guaranteed that the likelihood function has only one global maximum.

Our last step to find the parameters that better fit the model and build our logistic regression algorithm is to compute $\frac{\partial}{\partial \theta_l} l(\Theta)$. With a bit tedious computations one can see that:

$$\frac{\partial}{\partial \theta_l} l(\Theta) = \sum_{j=1}^m (y^{(j)} - h_{\theta}(x^{(j)})) x_l^{(j)}$$

Once we have already found the parameters that better fit our model (trained the model) in order to predict a new input x we just have to apply the hypothesis function $h_{\theta}(x)$ with the obtained parameters and apply a decision boundary like:

$$\begin{aligned} \text{Predict } y &= 1 \quad \text{if } h_{\theta}(x) \geq 0.5 \\ \text{Predict } y &= 0 \quad \text{if } h_{\theta}(x) < 0.5 \end{aligned}$$

Multiclass Classification: One vs All

Now, that we have seen the approach of binary classification with logistic regression we are going to talk about multiclass classification using the one versus all approach. Note that there is another approach to multiclass classification using a generalization of logistic regression called softmax regression which tries to minimize the cross entropy function, i.e. a measure of the difference between two probability distributions.

The one versus all approach is based on training a classifier for each class k in order to predict the probability of $y = k$. Let's suppose that we have n different classes, that is $k = \{1, 2, 3, 4, \dots, n\}$. The first step is to train a logistic regression classifier for each class to obtain k different classifiers. Each classifier predicts the probability of being class k versus the probability of being the rest $n - 1$ classes:

$$h_{\theta}^{(k)}(x) = P(y = k | x, \theta)$$

where $k = \{1, 2, 3, 4, \dots, n\}$

Once all the classifiers are trained, to predict the class for a new given input x we just have to choose the class k such that:

$$\max_k h_{\theta}^{(k)}(x)$$

3.2.2 Decision Trees

A decision tree is a nonlinear supervised machine learning algorithm that can be used for classification or regression problems. This algorithm uses a tree model formed by branches and leaves with the main idea to use the features or observations about an item to get some conclusions about the target value. Here, we are going to mainly focus in classification decision trees that is, the target value can only take a discrete number of values. We use leaves to represent the class labels and branches to represent the features that lead to those class label. In general, what a classification decision tree is doing is a greedy, top-down recursive partitioning of the input space into regions. As an example, figure 3.4 shows a decision tree for forecasting.

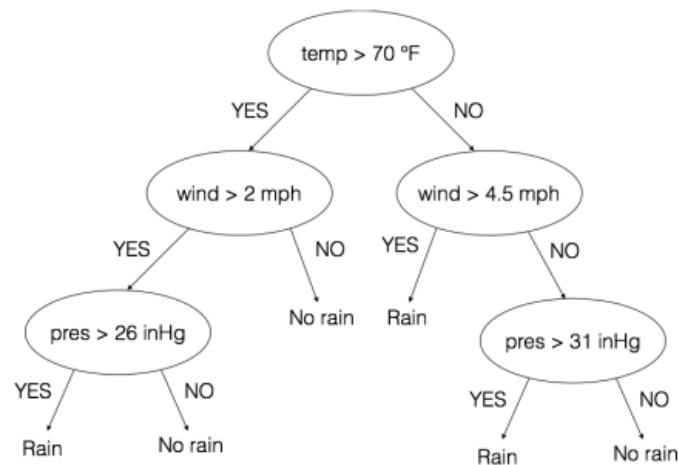


Figure 3.4: Decision tree example for predicting rain/no rain. It uses temperature, wind and pressure features in order to partition the space and make decisions.

In order to define our problem in a more formal way, we can say that we have a parent region R_p and at each step we are looking for a split $S_p := (R_1, R_2)$ into children regions R_1 and R_2 such that (see figure 3.5):

$$R_1 = \{x \mid x_j < t, x \in R_p\}$$

$$R_2 = \{x \mid x_j \geq t, x \in R_p\}$$

where x_j is the j feature of x and $t \in \mathbb{R}$ is the threshold we are using.

Note that in classification decision trees in order to give predictions we are predicting the majority class of the split region. For example, looking at the figure 3.5 for a point that is inside R_{22} region we are going to predict the "red" class.

Now, the more natural question one can ask is how good is a split and how can we choose the splits. To do that, we can start defining a loss function, $L(R)$, on a region R . One simple approach for a binary classification problem is to define $L(R)$ as the number of miss-classification examples in our region. Once we have defined our loss function we

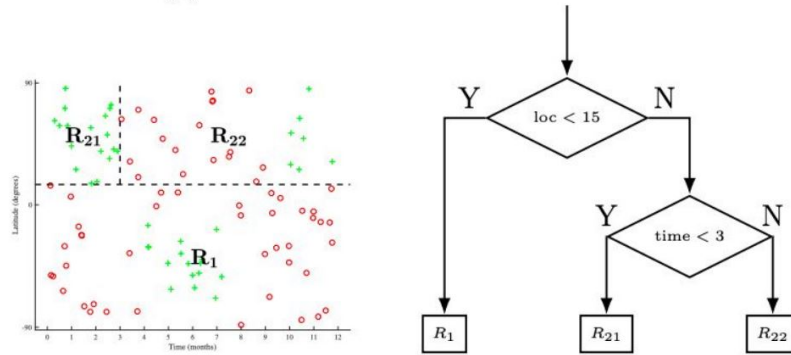


Figure 3.5: Example of how a decision tree is splitting the space into child regions.

want to pick a split that decreases the loss as much as possible, that is:

$$\max_{j,t} L(R_p) - (L(R_1) + L(R_2))$$

where the first term is the parent loss and the second term is the children loss. However, we have the problem that this miss-classification loss is not sensitive enough, so we have to find a better loss function. Instead, we can define the cross-entropy loss for discrete variables.

Given C classes, we can define \hat{p}_c to be the proportion of examples in R that are of class c and using this definition the cross-entropy loss function can be defined as:

$$L_{cross} = - \sum_c \hat{p}_c \log_2 \hat{p}_c$$

Another loss function that is widely used for decision trees is the Gini loss function which is also a strictly concave function as the cross-entropy loss:

$$L_{Gini} = \sum_c \hat{p}_c (1 - \hat{p}_c)$$

After we have defined the loss function we have to find the best split that decreases the loss as much as possible. In fact, considering these two loss functions the optimization process can be done with a greedy algorithm but also exists some methods that are a bit more efficient.

Regularization of Decision Trees

One important thing we must be careful with decision trees is overfitting. For example, if we grew our tree without ever stopping we can end up having a separate region for each single data point. Decision trees can be considered high-variance models and in order to avoid overfitting we are interested in regularizing them. In general, in order to solve this problem we can go through some heuristics:

- **Minimum Leaf Size:** Stop splitting after hitting a minimum leaf size, for example if we only have five examples in that region then stop splitting.

- **Maximum Depth of the Tree:** Stop splitting after reaching a maximum depth of the tree.
- **Maximum Number of Nodes:** Stop splitting after reaching a maximum number of nodes.
- **Minimum Decrease in Loss:** Stop splitting when the decrease in loss is very small. It is important to note that if we only use this heuristic we may stop too early because we are not considering higher order interactions between our variables. For example, the latitude can give us no information but combined with the month can produce a decrease in loss.

3.2.3 Ensemble Methods

As we have said one negative side of the decision trees is that they are high variance methods and also that they are not very good with additive structures. Now, let's see the idea why ensemble methods that is, methods by which we can aggregate the output of trained models, helps to improve the performance of decision trees.

Suppose we have random variables X_i for $0 \leq i < n$ that are independent and identically distributed (*i.i.d.*). Assume the variance of X_i is $Var(X_i) = \sigma^2$ for all X_i then, the variance of the mean is:

$$Var(\bar{X}) = Var\left(\frac{1}{n} \sum_i X_i\right) = \frac{\sigma^2}{n}$$

The problem is that the independence assumption is oftentimes not correct but if we drop the independence assumption and we just assume *i.d.* and instead we assume that the correlation between X_i 's is ρ . Then, we can write:

$$Var(\bar{X}) = Var\left(\frac{1}{n} \sum_i X_i\right) \tag{3.21}$$

$$= \frac{1}{n^2} \sum_{i,j} Cov(X_i, X_j) \tag{3.22}$$

$$= \frac{n\sigma^2}{n^2} + \frac{n(n-1)\rho\sigma^2}{n^2} \tag{3.23}$$

$$= \rho\sigma^2 + \frac{1-\rho}{n}\sigma^2 \tag{3.24}$$

Where in 3.23 we use the definition of Pearson correlation coefficient $\rho_{X,Y} = \frac{Cov(X,Y)}{\sigma_x\sigma_y}$ and that $Cov(X, X) = Var(X)$.

Now, if we consider each random variable to be the error of a given model in order to reduce the variance we can make use of formula 3.24. One can see that the idea is to have as many as models as possible in order to have a bigger n and decrease the second term and also, we want to make sure that our models are as decorrelated as possible so that ρ value goes down and the first term goes down. Decreasing the two terms leads to an overall decrease in variance of the error of the ensemble.

With this idea in mind there are several different ways to ensemble and generating de-correlated models:

- Use different algorithms.
- Use different training sets.
- Bagging (e.g. Random Forest).
- Boosting (e.g. *XGBoost*).

Bagging

Bagging stands for Bootstrap Aggregation and is a variance reduction ensembling method. Bootstrap is a method used in statistics to measure the uncertainty of some estimator.

Suppose that we have a true population P that we want to compute an estimator (e.g. mean) and we have our training set S sampled from P ($S \sim P$). We can find an approximation of our estimator computing it on S but to know what the error is respect to the true value we would need independent training sets S_1, S_2, \dots all sampled from P .

What bagging does is assume that our population is our training example, that is $S = P$ so now we can obtain new samples of our population by just generating new samples from S with replacement ($Z \sim S, |Z| = |S|$). This samples are called bootstrap samples and in fact we can generate as many examples as we want Z_1, Z_2, \dots, Z_M .

Now, what we want to do in practice is to aggregate this bootstrap samples, that means that we are going to take a bunch of bootstrap samples, Z_m , train separate machine learning models on each one and then classify by majority vote in the case of classification forests (note that for bagged regression forests the prediction for a new point is the averaged response of all decision trees).

Going back to the previous variance formula:

$$\text{Var}(\bar{X}) = \rho\sigma^2 + \frac{1-\rho}{M}\sigma^2$$

where now we put M as the number of bootstrap samples we take. What we are doing with bagging is creating less correlated predictors so we are decreasing ρ as we want. Also, since we can take as many as bootstrap samples as we want we can also drive down the second term. It is important to note that increasing the samples with bootstrapping does not cause to overfit because ρ is insensitive to M and therefore the higher the M , less overall variance. Another advantage for bagging is that it can be shown that each bootstrap sample only contains $\frac{2}{3}$ of S so we can use the other $\frac{1}{3}$ as an estimate of error called out-of-bag error.

The counterpart of using bagging is that the bias is slightly increased because of each bootstrap set not having the full training set but in general, since decision trees are high-variance low-bias methods are ideal for bagging.

Random Forest

If our dataset contains one very strong predictor then our bagged trees would always use that feature to split which ends up to more correlated trees. Random forest is just a type of bagged decision tree that introduces more randomization into each individual

decision tree with the idea to further decorrelate the random variables and decrease ρ even more. In order to do that what random forest does is at each split consider only a fraction of our total features because if we force the method using different features we are decreasing the correlation between our models. The downside we have is that there is an increase in bias due to we are restricting the feature space but again, since we are working with low bias methods this is often not a problem.

Quantile Random Forest

Random forest like all other types of decision trees can be used also for regression problems in order to provide an accurate approximation of the conditional mean of a dependent variable, i.e. $E(Y|X = x)$. However, it can be shown that random forest provide information about the full conditional distribution of the dependent variable. With a generalization of random forest called quantile regression forests we can obtain an estimation of the quantiles.

The conditional distribution of a function is given by:

$$F(y|X = x) = P(Y \leq y|X = x)$$

where $y \in \mathbb{R}$.

For a continuous distribution function we can define the α -quantile $Q_\alpha(x)$ as:

$$Q_\alpha(x) = \inf\{y : F(y|X = x) \geq \alpha\} \quad (3.25)$$

The quantile give us more information about the distribution of Y as a function of the predictor variable X than the conditional mean alone. This led to the development of machine learning methods known as quantile regression and also quantile random forests.

Quantile regression objective is to estimate the conditional quantiles from data. This problem can be formulated as an optimization problem in a similar way one can construct the linear regression or the logistic regression. Let L_α be the loss function where $0 < \alpha < 1$ and defined by the weighted absolute deviations:

$$L_\alpha(y, q) = \begin{cases} \alpha|y - q| & \text{if } y > q \\ (1 - \alpha)|y - q| & \text{if } y \leq q \end{cases} \quad (3.26)$$

Then, quantile regression minimized the expected loss $E(L_\alpha)$:

$$Q_\alpha(x) = \arg \min_q E\{L_\alpha(Y, q)|X = x\}.$$

In order to minimize that loss function we can use the gradient descent algorithm.

One possible application of quantile regression is that can be used to build prediction intervals. For example, a 95% prediction interval for the value of Y is given by:

$$I(x) = [Q_{.025}(x), Q_{.975}(x)]$$

This interval tells us that a new observation of Y , for $X = x$ is with a high probability (95%) in the interval $I(x)$. So, quantile regression can offer us a way of judging the reliability of predictions. Another application of quantile regression is the outlier detection

using for example the conditional interquartile range.

Now, the idea of quantile random forest is the same as regression random forest but instead of approximating the conditional mean, approximating the full conditional distribution, which can be rewrite as:

$$F(y|X = x) = P(Y \leq y|X = x) = E(1_{\{Y \leq y\}}|X = x)$$

The key difference that allows us compute the conditional distribution approximation is that for each node in each tree, quantile regression forests keeps the value of all observations that fall into this node and uses this information to build the conditional distribution. In contrast, regression forests keeps only the mean of the observations for each node in each tree.

Finally, in order to estimate $\hat{Q}_\alpha(x)$ we just have to put our estimation $\hat{F}(y|X = x)$ into definition 3.25.

Boosting

Boosting methods decrease the bias of our model training sequentially some high bias, low variance simple models called weak learners or weak classifiers. The idea with boosting decision trees is that each tree can only make one decision before making a prediction, these are known as decision stumps. After training one weak learner and make predictions then, we train a new decision stump that tries to improve the missclassifications from the previous one. So, in this method learners are learned sequentially with early learners fitting simple models and then analyzing the data for errors. When an input is misclassified, its weight is increased so that the next classifier is more likely to classify it correctly. At the end, we output a combination of these weak learners as an ensemble classifier in order to obtain a better performing model.

Gradient Boosting

Gradient boosting algorithms are a type of boosting algorithms that at each step in order to find the next weak learner parameter's and weighting make use of gradient descent or Newton-Raphson algorithm to solve this minimization problem. One of the most natural things to do would be to take the derivative of the loss function and perform gradient descent.

However, we cannot make arbitrary moves in the input space because we are restricted to taking steps in our model class so we can only add in parameterized weak learners. So, what gradient boosting algorithms do is compute the gradient at each training point with respect to the current predictor (typically a decision stump):

$$g_i = \frac{\partial L(y_i, f(x_i))}{\partial f(x_i)}$$

where $L(y, f(x_i))$ is the loss function and $f(x_i)$ is the ensemble classifier or the decision stump. Then, we train a new regression predictor to match this gradient and use it as the gradient step.

One of the most popular gradient boosting algorithms is *XGBoost* which can have slightly different implementations but here we show as an example a generic *XGBoost* algorithm.

Suppose that we have a training set $\{(x_i, y_i)\}_{i=1}^N$, a differentiable loss function $L(y, F(x))$, a number of weak learners M and a learning rate α .

Algorithm 1 XGBoost

1. Initialize the model with a constant value:

$$\hat{f}_{(0)}(x) = \arg \min_{\theta} \sum_{i=1}^N L(y_i, \theta)$$

2. For $m = 1$ to M :

- (a) Compute the gradients and Hessians:

$$\hat{g}_m(x_i) = \left[\frac{\partial L(y_i, f(x_i))}{\partial f(x_i)} \right]_{f(x)=\hat{f}_{(m-1)}(x)}$$

$$\hat{h}_m(x_i) = \left[\frac{\partial^2 L(y_i, f(x_i))}{\partial f(x_i)^2} \right]_{f(x)=\hat{f}_{(m-1)}(x)}$$

- (b) Fit a weak learner using the training set $\{x_i, -\frac{\hat{g}_m(x_i)}{\hat{h}_m(x_i)}\}_{i=1}^N$ by solving the next optimization problem:

$$\hat{\phi}_m = \arg \min_{\phi \in \Phi} \sum_{i=1}^N \frac{1}{2} \hat{h}_m(x_i) \left[-\frac{\hat{g}_m(x_i)}{\hat{h}_m(x_i)} - \phi(x_i) \right]^2$$

$$\hat{f}_m(x) = \alpha \hat{\phi}_m(x)$$

- (c) Update the model:

$$\hat{f}_{(m)}(x) = \hat{f}_{(m-1)}(x) + \hat{f}_m(x)$$

3. Output:

$$\hat{f}(x) = \hat{f}_{(M)}(x) = \sum_{m=0}^M \hat{f}_m(x)$$

3.3 Forecast Verification

Verification or validation consists in evaluating the quality of the forecast. There are various ways to do it but in general all the methods involve comparing the variables predicted by the model with real observations of those variables. For more information about verification check [15].

Before explaining some verification methods it is useful to define some basic terminology about forecasts performance:

- **Accuracy:** A measure of the average degree to which pairs of forecast values and observation correspond. Scalar measures of accuracy summarize the overall quality of the forecasts in the form of a single number.
- **Bias:** A measure of the correspondence between the average of a forecast variable and the average of the observations.
- **Reference Forecast:** Easily available, non-model-based dataset that can be interpreted as a simple benchmark to beat. For example, the climatology (forecast the historical mean) or persistence (forecast the last observed value) methods are typically used.
- **Skill:** The accuracy of a forecast relative to a reference forecast.

3.3.1 Standard Metrics

Accuracy Measures for Continuous Variables

These measures apply to variables that are continuous, that is that can take many values within a realistic range. For example, if we want to predict the temperature value itself.

First of all we have the bias which is the same as the Mean Error (ME) also known as systematic error:

$$ME = Bias = \frac{1}{n} \sum_{k=1}^n (x_k - o_k) = \bar{x} - \bar{o}$$

where x_k is the k -th forecast of n and o_k is the corresponding observed value.

One possible accuracy measure is the arithmetic average of the absolute difference between the forecast and real observations (MAE). This measure computes the average magnitude of the forecast error:

$$MAE = \frac{1}{n} \sum_{k=1}^n |x_k - o_k| \quad (3.27)$$

A perfect forecast have a MAE of zero so the closer the value is to zero better is the forecast.

Another very common accuracy measure is the Mean-Square Error (MSE), which is the averaged squared difference between the forecast and the observation and it is defined as:

$$MSE = \frac{1}{n} \sum_{k=1}^n |x_k - o_k|^2 \quad (3.28)$$

Because the errors are squared this measure will be more sensitive to large errors than the MAE. Sometimes it is useful to use the square root of MSE because has the same physical dimensions as the forecast and observations, that is $RMSE := \sqrt{MSE}$. It is important to note that all this measures represent systematic and random components to the error.

Accuracy Measures for Discrete Variables

This measures apply when our predictand is a yes-no condition, for example if we have to answer if it is going to rain or if the temperature will exceed some threshold. This problem can be represented using a 2x2 contingency table as the one in figure 3.6.

(a)

		Observed		
		Yes	No	
Forecast	Yes	a	b	$a+b$
	No	c	d	$c+d$
		$a+c$	$b+d$	$n = a+b+c+d$

Marginal totals for forecasts (pointing to the right column)

Marginal totals for observations (pointing to the bottom row)

Sample size (pointing to the bottom-right cell)

Figure 3.6: Contingency table for forecast and observed variables. Image source: [13]

Suppose we have n forecast–observation pairs, then:

- a represents the number of times that an observed event was correctly forecast (called hits).
- b is the number of times that no event occurred but the forecast predict an occurrence (called false alarms).
- c is the number of times that an observed event is forecast to not occur (called misses).

- d is the number of times that an event was correctly forecast to not occur (called a correct negative).

Now, we are going to define some common accuracy measures that are based on this contingency table. The Proportion Correct represents the fraction of forecasts that correctly predict event or no-event and is defined as:

$$PC = \frac{a + d}{n} \quad (3.29)$$

One disadvantage of this measure is that gives the same importance to a correct positive and a correct negative. An alternative score is the Threat Score (TS), which is useful when the yes-event is much less frequent than the no-event. This measure is also called Critical Success Index (CSI) and is defined as:

$$TS = CSI = \frac{a}{a + b + c} \quad (3.30)$$

The bias is a comparison between the average forecast and the average observation:

$$B = \frac{a + b}{a + c}$$

The False-Alarm Ratio is the fraction of yes forecasts that are wrong and can be defined as follows:

$$FAR = \frac{b}{a + b}$$

We also have the false-alarm rate:

$$F = \frac{b}{b + d}$$

which is the ratio of false alarms to the total number of nonoccurrences of the event. Finally, we can also define the hit rate or the probability of detection (POD) which is defined as:

$$H = POD = \frac{a}{a + c}$$

Accuracy Measures for Probabilistic Forecasts

Probabilistic forecasts are the ones that gives the probability of an event occurring, with a value between 0 and 1. One accuracy measure to verify probabilistic forecasts is the Brier score which is essentially the mean squared error of the probability forecasts, considering that the observation is 1 if the event occurs, and that the observation is 0 if the event does not occur. The brier score averages the squared differences between pairs of forecast probabilities and the subsequent binary observations:

$$BS = \frac{1}{N} \sum_{k=1}^N (y_k - o_k)^2 \quad (3.31)$$

where y_k is the probability that was forecast, o_k the actual outcome of the event and N is the number of forecasting instances. It is important to note that this score takes on a value between zero and one with the lower the score is the better are the predictions.

The Brier score is appropriate for binary and categorical outcomes that can be structured as true or false. However, the original definition by Brier is applicable to multi-category forecasts and ordinal variables which can take on three or more values.

$$BS = \frac{1}{N} \sum_{k=1}^N \sum_{i=1}^R (y_{ki} - o_{ki})^2$$

where R is the number of possible classes in which the event can fall, N the overall number of instances of all classes, y_{ki} is the predicted probability for class i and o_{ki} is 1 if it is i -th class in instant k and 0 otherwise. Note that, this original formulation is also valid for binary forecasts but it has twice the value of the score given by formulation 3.31.

The ranked probability score (RPS) is a measure for probabilistic multiple-category events forecasts. It answers the question of how well the probability forecast predict the category that the observation fell into. RPS score is the sum of squared differences between the components of the cumulative forecast and observation vector (one-hot encoded vector):

$$RPS = \sum_{m=1}^M [(\sum_{k=1}^m y_k) - (\sum_{k=1}^m o_k)]^2 \quad (3.32)$$

where y_k is the forecast probabilities, M the number of categories and o_k the k -th component of the observation vector which is 1 if the observation belongs to class k and 0 otherwise. It is important to note that the perfect RPS score is 0 and all forecasts that are less than perfect receive scores that are positive numbers, so the RPS has a negative orientation.

Equation 3.32 for RPS is only for a single forecast-event pair so in order to evaluate a collection of n forecasts requires averaging RPS values for each forecast-event pair:

$$\langle RPS \rangle = \frac{1}{n} \sum_{k=1}^n RPS_k \quad (3.33)$$

3.3.2 Skill Scores

As we have defined before the skill is the accuracy of the forecast model in relation to a reference forecast. The Skill Score (SS) is defined as a percentage improvement over the reference forecast and can be written as follows:

$$SS_{ref} = \frac{A - A_{ref}}{A_{perf} - A_{ref}} \cdot 100$$

where A is the accuracy of a forecast, A_{ref} is the accuracy of the reference forecast and A_{perf} is the accuracy of a perfect forecast. Note that if our forecast accuracy is less than the reference forecast accuracy then the skill is negative. Skill scores for discrete variables

are also based on the contingency table. One of the most common is the Heidke Skill Score (HSS) which is defined as:

$$HSS = \frac{2(ad - bc)}{(a + c)(c + d) + (a + b)(b + d)}$$

The accuracy measure (A) is based on the proportion correct (3.29) and the A_{ref} is the proportion correct that would be obtained by random forecast. Another common skill score is the Gilbert Skill Score (GSS) also called Equivalent Threat Score (ETS) which is based on the TS accuracy (3.30):

$$GSS = ETS = \frac{a - a_{ref}}{a - a_{ref} + b + c}$$

where $a_{ref} = \frac{(a+b)(a+c)}{n}$.

Skills score for continuous variables are based on MAE (3.27), MSE (3.28) or RMSE. As a reference, climatology or persistence are generally used. Using the MSE, the accuracies for climatology and persistence can be defined as follows:

$$MSE_{Clim} = \frac{1}{n} \sum_{k=1}^n (\bar{o} - o_k)^2$$

$$MSE_{Pers} = \frac{1}{n} \sum_{k=1}^n (o_{k-1} - o_k)^2$$

where o_k is the observation, \bar{o} is the climatological mean of the observed variable, and o_{k-1} is the previous value of the variable. Now, the skill score can be written as:

$$SS = \frac{MSE - MSE_{ref}}{0 - MSE_{ref}} = 1 - \frac{MSE}{MSE_{ref}}.$$

Two of the most common skill scores for probabilistic forecasts are the Brier Skill Score (BSS) and the Ranked Probability Skill Score (RPSS). The first one as the name indicates it is based on the Brier score (3.31):

$$BSS = \frac{BS - BS_{ref}}{0 - BS_{ref}} = 1 - \frac{BS}{BS_{ref}}$$

since $BS_{perf} = 0$

Since also $RPS_{perf} = 0$ then, ranked probability skill score for a collection of RPS values relative to the RPS computed from the climatological probabilities (gives equal probabilities to all possible events) can be computed as:

$$RPSS = \frac{\langle RPS \rangle - \langle RPS_{Clim} \rangle}{0 - \langle RPS_{Clim} \rangle} = 1 - \frac{\langle RPS \rangle}{\langle RPS_{Clim} \rangle} \quad (3.34)$$

RPSS measure is widely used as a measure of the predictive skill for probabilistic forecasts issued for categorical events (predictions distributed in ordered categories or terciles). The RPSS takes into account the ordering of categories using cumulative probabilities and ranges between $-\infty$ and 1. Scores below 0 are defined as unskillful, those equal to 0 are equal to the prediction given by the climatology, and scores above 0 is an improvement upon climatology. A score of 1 is a perfect forecast.

Chapter 4

Experiments

In this chapter we explain our approach to improve sub-seasonal predictions with machine learning. First, we give all the details about the WMO challenge and the datasets we have used. Then, we explain our methodology and approach to the problem and finally we give information about methods and improvements we have tried after the challenge.

4.1 Details of the WMO S2S-AI Challenge to Improve Sub-seasonal Predictions

This thesis has been done with the idea to improve the skill of sub-seasonal predictions by participating in the S2S-AI-challenge ([Challenge website](#)) organized by the World Meteorological Organization. This means, that the objectives, conditions and constraints of our problem are aligned with the main objective of the challenge which is to use artificial intelligence techniques to improve week 3 plus week 4 (week 3-4) and week 5 plus week 6 (week 5-6) sub-seasonal global probabilistic two-meter temperature (t2m) and total precipitation tercile forecasts issued in the year 2020.



Figure 4.1: Organizers and sponsors of the S2S-AI-challenge.

4.1.1 Submissions

Submissions for the challenge must provide predictions for weeks 3-4 and weeks 5-6 for all 53 forecasts issued on Thursdays in 2020 (weekly initialized). The target is the tercile-based categorical probabilities (called tercile probabilities) of biweekly aggregated data, that is, the probability of the below-normal, normal, and above-normal categories of aggregated data. We only focus on giving predictions over land points and for two different variables: two-meter temperature and total precipitation.

	Two-meter temperature	Total precipitation
Abbreviation	t2m	tp
Unit	Kelvin (K)	Kg/m^2
Description	Temperature at 2m height averaged for the date given.	Total precipitation accumulated from forecast time until valid time.
Week 3-4 aggregation	Mean [day 15, day 28]	Day 29 – day 15
Week 5-6 aggregation	Mean [day 29, day 42]	Day 42 - day 28

Table 4.1: Detailed information about the two variables we concern.

Aggregation Example

To make it even more clear we put an example of how our target aggregated data can be computed for the two variables.

- **Temperature:** For t2m our week 3-4 aggregation is the mean temperature from day 15 to day 28. In the same way our week 5-6 aggregation is the mean temperature from day 29 to 42.
- **Total Precipitation:** For tp our week 3-4 aggregation is computed adding accumulated precipitation from day 15 to day 28. In the same way our week 5-6 aggregation is computed adding accumulated precipitation from day 29 to 42. If we have total precipitation accumulated from forecast time until valid time the way to compute our aggregation is to subtract *day 29 – day 15* data and *day 42 – day 28* in order to compute week 3-4 and week 5-6 aggregations respectively.

As you can see in the graphic 4.2 if we consider our start date as day 1 in order to compute our week 3-4 temperature aggregation we have to add all the temperature values from day 15 to 28 and divide by 14 to obtain the mean value.

Our final target is to predict the probability of these biweekly aggregations of being below-normal, normal, and above-normal categories also called the tercile probability.

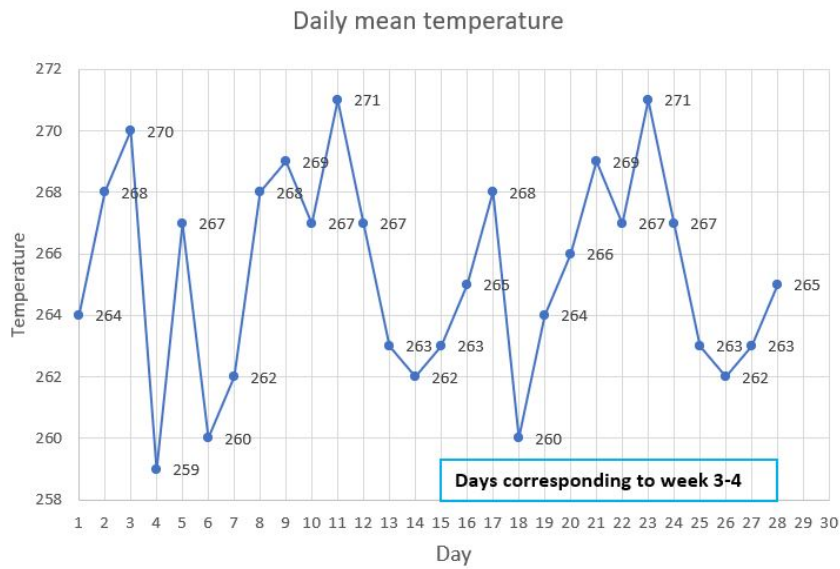


Figure 4.2: Graphic that shows daily mean temperature values from day 1 to day 28.

4.1.2 Verification Metrics

Since we have to give probabilistic forecasts the score function we use to evaluate how good is a single forecast is the RPS between the machine learning model forecasts and the ground truth which is calculated as 3.32 setting $M = 3$ because we have three categories: below-normal/normal/above-normal. Then, since we have to give a collection of 53 different forecasts (we have 53 start dates) we have to average each RPS value as in equation 3.33 with $n = 53$.

Then, this averaged RPS is compared to climatology forecast (reference forecast) in the RPSS which is calculated as in equation 3.34.

Since we have to give forecasts for each grid cell over land globally, the RPSS is computed independently for every land point (we called gridded RPSS). It is important to note that for each grid point we also have to compute the RPSS for both variables and both lead times.

Finally, the gridded RPSS is spatially averaged over [90N-60S] land points and averaged over both variables (t2m and tp) and lead times (week 3-4 and week 5-6) resulting in a unique final score value.

4.1.3 Possible Approaches

As we explained in chapter 1 there are two general approaches for sub-seasonal predictions. In our case we decide to go for the model output post-processing machine learning approach also known as recalibration, that is we are going to use sub-seasonal dynamical forecast and apply a machine learning method in order to improve or correct them.

4.2 Datasets

The challenge organizers provide some prepared datasets (with biweekly aggregated data already computed) to train our models and they provide also some tools to download some more datasets and predictor variables in case we need them. However, we are restricted to use publicly available datasets or to provide access to any own pre-processed datasets. In this section, we are going to explain all the details about the datasets we used.

4.2.1 Datasets Coordinates

First of all we explain all the coordinates that are present in our multi-dimensional datasets:

- **Start date (forecast time in the challenge files):** the start date is the initialization date, i.e. the date and time at which numerical integrations start from the observed initial conditions. Typically, the operational model outputs are made publicly available during the start date or shortly after.
- **Valid time:** valid time (or valid period) is the time period for which a forecast is valid. In our case we have forecasts valid for week 3-4 after initialization, and week 5-6 after initialization. The valid time for a specific forecast is usually specified as calendar dates, for example: the forecast is valid for the week starting the 10th January.
- **Lead time:** the lead time (sometimes referred to as forecast horizon or forecast period) is the time interval between the start date and the start of the valid time. Since we are working with biweekly aggregate values in our datasets the coordinate lead time corresponds to 14 days for the weeks 3-4 forecasts and 28 days for the weeks 5-6 forecasts. Notice that:

$$\text{start of valid period} = \text{start date} + \text{lead time}$$

- **Latitude and Longitude:** geographic coordinates of the center of each grid cell, measured in degrees from the equator and the Greenwich meridian. In our case, they span a global grid with a spatial resolution of 1.5x1.5 degrees.
- **Realization (or number of ensemble member):** an integer number to identify the different ensemble members made with perturbed initial conditions.

4.2.2 Training Datasets

For training the models, we use the European Centre for Medium-Range Weather Forecasts (ECMWF) hindcasts and the corresponding categorical observations divided in three classes: below-normal, normal and above-normal conditions. We also use a number of teleconnection indices as predictors, which were downloaded from a public source and pre-processed by us.

Challenge Training Datasets

These datasets are the ones we use from the challenge organizers.

ECMWF Hindcast Data:

- **Forecast start dates:** from 2000/01/02 to 2019/12/31, corresponding to the weekly Thursdays in 2020.
- **Lead times:** 14 and 28 days.
- **Parameters:** t2m, tp.
- **Realization:** 11 members.

Climate Prediction Center (CPC) Categorical Observations:













- **Forecast start dates:** from 2000/01/02 to 2019/12/31, corresponding to the weekly Thursdays in 2020.
- **Lead times:** 14 and 28 days.
- **Parameters:** t2m, tp.
- **Category:** below normal/normal/above normal in hot-one-encoded format. For example, if the observation corresponds to category above normal we have [0,0,1].

► Dimensions:	(forecast_time: 1060, latitude: 121, lead_time: 2, longitude: 240, realization: 11)		
▼ Coordinates:			
forecast_time	(forecast_time)	datetime64[ns]	20...
latitude	(latitude)	float64	90...
lead_time	(lead_time)	timedelta64[ns]	14 ...
longitude	(longitude)	float64	0.0...
realization	(realization)	int64	0 1...
valid_time	(lead_time, forecast_time)	datetime64[ns]	da...
▼ Data variables:			
t2m	(lead_time, realization, forecast_time, latitude, longitude)	float32	da...
tp	(lead_time, realization, forecast_time, latitude, longitude)	float32	da...
▼ Attributes:			





Figure 4.3: ECMWF Hindcast data.

► Dimensions: (category: 3, forecast_time: 1060, latitude: 121, lead_time: 2, longitude: 240)

▼ Coordinates:

category	(category)	<U12 'belo...	 
forecast_time	(forecast_time)	datetime64[ns] 2000...	 
latitude	(latitude)	float64 90.0 ...	 
lead_time	(lead_time)	timedelta64[ns] 14 d...	 
longitude	(longitude)	float64 0.0 1...	 
valid_time	(lead_time, forecast_time)	datetime64[ns] dask...	 

▼ Data variables:

t2m	(category, lead_time, forecast_time, latitude, longitude)	float32 dask...	 
tp	(category, lead_time, forecast_time, latitude, longitude)	float32 dask...	 

► Attributes: (0)

Figure 4.4: Climate Prediction Center Categorical Observations

Teleconnections Training Datasets

Datasets not provided by the challenge organizers that correspond to other predictor variables that we use in some models.

Monthly observations of teleconnections indices:

- **Valid Date:** from 2000/01/01 to 2019/12/01.

Hindcast of Stratospheric Polar Vortex (weekly data):

- **Start dates:** from 2000/01/01 to 2019/12/01.
- **Lead times:** 14 and 28 days.
- **Realization:** 11 members.

Hindcast of Madden Julian Oscillation (weekly data):

- **Start dates:** from 2000/01/01 to 2019/12/01.
- **Lead times:** 14 and 28 days.
- **Parameters:** rmm1, rmm2, amplitude, phase.

► Dimensions: (date: 240)

▼ Coordinates:

date	(date)	datetime64[ns]	2000-01-01 ... 2019-12-01		
-------------	--------	----------------	---------------------------	--	--

▼ Data variables:

ENSO	(date)	float64	-1.79 -1.53 -1.26 ... 0.74 0.51		
IOD	(date)	float64	-0.06 0.071 0.24 ... 0.958 0.312		
QBO	(date)	float64	7.53 6.96 8.12 ... 5.0 4.19 4.72		
AAO	(date)	float64	1.273 0.6197 0.1331 ... -1.84 -1.36		
NAO	(date)	float64	0.19 1.48 0.4 ... -1.03 0.16 1.02		
EA	(date)	float64	-1.97 0.41 -0.68 ... 0.67 0.07 0.81		
WP	(date)	float64	-0.21 -0.41 0.01 ... -0.12 0.74		
PNA	(date)	float64	-1.7 1.2 0.95 ... -0.97 -0.07 -0.12		
EAWR	(date)	float64	0.65 0.1 -1.13 ... -0.52 -0.59 0.15		
SCA	(date)	float64	-1.02 -1.16 -0.92 ... 1.45 0.84		
POL	(date)	float64	-0.18 -1.55 -1.52 ... 0.98 -0.43		
Sunsports	(date)	float64	133.1 165.7 217.7 ... 0.4 0.5 1.5		

Figure 4.5: Monthly Hindcast Teleconnections data.

► Dimensions: (member: 11, forecast_time: 1060, lead_time: 2)

▼ Coordinates:

member	(member)	int64	0 1 2 3 4 5 6 7 8 9 10		
forecast_time	(forecast_time)	datetime64[ns]	2000-01-02 ... 2019-12-31		
lead_time	(lead_time)	timedelta64[ns]	14 days 28 days		
valid_time	(lead_time, forecast_time)	datetime64[ns]	2000-01-16 ... 2020-01-28		

▼ Data variables:

spv	(lead_time, forecast_time, member)	float64	23.17 25.75 19.06 ... 17...		
------------	------------------------------------	---------	-----------------------------	--	--

► Attributes: (0)

Figure 4.6: Stratospheric Polar Vortex Hindcast data.

► Dimensions: (forecast_time: 1060, lead_time: 2)

▼ Coordinates:

forecast_time	(forecast_time)	datetime64[ns]	2000-01-02 ... 2019-12-31		
lead_time	(lead_time)	timedelta64[ns]	14 days 28 days		
valid_time	(lead_time, forecast_time)	datetime64[ns]	2000-01-16 ... 2020-01-28		

▼ Data variables:

rmm1	(lead_time, forecast_time)	float64	-0.1049 -0.1435 ... 1.374 0.7377		
rmm2	(lead_time, forecast_time)	float64	-0.1874 -0.145 ... 0.1436 0.1848		
ampl	(lead_time, forecast_time)	float64	0.4603 0.337 ... 1.392 0.7665		
phase	(lead_time, forecast_time)	float64	3.143 3.786 2.643 ... 4.714 5.0		

► Attributes: (0)

Figure 4.7: Madden Julian Oscillation Hindcast data.

4.2.3 Forecast Datasets (ML Model Input Data for Prediction)

To produce 2020 tercile probability predictions we require the ECMWF 2020 forecasts and for some of the models with more predictors also the 2020 teleconnection forecasts.

ECMWF 2020 forecasts:

- **Start dates:** from 2020/01/02 to 2020/12/31, weekly every Thursday in 2020.
- **Lead times:** 14 and 28 days.
- **Parameters:** t2m, tp.
- **Realization:** 51 members.

Observations of Monthly Teleconnection indices (prior to the forecast start date):

- **Valid Date:** from 2020/01/01 to 2020/12/01.

Forecasts of Stratospheric Polar Vortex:

- **Start dates:** from 2020/01/02 to 2020/12/31, weekly every Thursday in 2020.
- **Lead times:** 14 and 28 days.
- **Realization:** 51 members.

Forecasts of Madden Julian Oscillation:

- **Start dates:** from 2020/01/02 to 2020/12/31, weekly every Thursday in 2020.
- **Lead times:** 14 and 28 days.
- **Parameters:** rmm1, rmm2, amplitude, phase.

► Dimensions:	(forecast_time: 53, latitude: 121, lead_time: 2, longitude: 240, realization: 51)		
▼ Coordinates:			
forecast_time	(forecast_time)	datetime64[ns]	20...
latitude	(latitude)	float64	90...
lead_time	(lead_time)	timedelta64[ns]	14 ...
longitude	(longitude)	float64	0.0...
realization	(realization)	int64	0 1...
valid_time	(lead_time, forecast_time)	datetime64[ns]	da...
▼ Data variables:			
t2m	(lead_time, realization, forecast_time, latitude, longitude)	float32	da...
tp	(lead_time, realization, forecast_time, latitude, longitude)	float32	da...
► Attributes:	(8)		

Figure 4.8: EMCWF 2020 forecasts.

4.3 Methodology

To begin with, it should be mentioned that our approach to the challenge is based on combining four different methods in order to produce 2020 predictions.

Temperature (t2m)	Total Precipitation (tp)
Climatology model	Climatology model
Raw ECMWF model	Raw EMCWF model
Logistic regression with ECMWF ensemble mean as predictor	
Random forest with ECMWF members (realization) as predictors	

Table 4.2: Summary of the methods employed for each variable. Note that for tp we only use two models because we realize on cross validation that the others methods are unskillful.

The general scheme of the methodology we use to prevent data leakage and overfitting is the following:

1. Train the models in leave-one-year-out cross-validation (LOYO CV), i.e., removing one year for each training iteration. As we have 20 years of hindcast, this is a 20-fold CV.
2. Make predictions of the excluded year and compute RPSS (repeat for each fold).
3. Compute the median of RPSS over all training iterations (20-fold).
4. Train again each model using the full hindcast period (2000-2019) and predict 2020.
5. Use the cross-validation median RPSS computed in step 3 to select the best model prediction at each grid point.

All the code is well documented in the *BSC_contribution.ipynb* notebook and all the auxiliary functions needed are in the script *utils.py* ([Link to code repository](#)).

4.3.1 Data Preprocessing

First of all, we decide to split the forecast time dimension (i.e. the start date) into year and week coordinates in order to work more easily with the data and machine learning models.

Before:

The screenshot shows the metadata for an xarray.Dataset. The dimensions are forecast_time: 1060, latitude: 121, lead_time: 2, longitude: 240, and realization: 11. The coordinates include forecast_time (datetime64[ns]), latitude (float64), lead_time (timedelta64[ns]), longitude (float64), realization (int64), and valid_time (datetime64[ns]). The data variables are t2m and tp, both of type float32, with dimensions (lead_time, realization, forecast_time, latitude, longitude).

xarray.Dataset				
► Dimensions:	(forecast_time: 1060, latitude: 121, lead_time: 2, longitude: 240, realization: 11)			
▼ Coordinates:				
forecast_time	(forecast_time)	datetime64[ns]	20...	
latitude	(latitude)	float64	90...	
lead_time	(lead_time)	timedelta64[ns]	14 ...	
longitude	(longitude)	float64	0.0...	
realization	(realization)	int64	0 1...	
valid_time	(lead_time, forecast_time)	datetime64[ns]	da...	
▼ Data variables:				
t2m	(lead_time, realization, forecast_time, latitude, longitude)	float32	da...	
tp	(lead_time, realization, forecast_time, latitude, longitude)	float32	da...	

Figure 4.9: Hindcast data before splitting the forecast time dimension.

After:

The screenshot shows the metadata for an xarray.Dataset after splitting the forecast time dimension. The dimensions are latitude: 121, lead_time: 2, longitude: 240, realization: 11, year: 20, and week: 53. The coordinates include latitude (float64), lead_time (timedelta64[ns]), longitude (float64), realization (int64), valid_time (datetime64[ns]), year (int64), and week (int64). The data variables are t2m and tp, both of type float32, with dimensions (lead_time, realization, latitude, longitude, year, week).

xarray.Dataset				
► Dimensions:	(latitude: 121, lead_time: 2, longitude: 240, realization: 11, year: 20, week: 53)			
▼ Coordinates:				
latitude	(latitude)	float64	90.0 ...	
lead_time	(lead_time)	timedelta64[ns]	14 da...	
longitude	(longitude)	float64	0.0 1...	
realization	(realization)	int64	0 1 2 ...	
valid_time	(lead_time, year, week)	datetime64[ns]	dask...	
year	(year)	int64	2000 ...	
week	(week)	int64	0 1 2 ...	
▼ Data variables:				
t2m	(lead_time, realization, latitude, longitude, year, week)	float32	dask...	
tp	(lead_time, realization, latitude, longitude, year, week)	float32	dask...	

Figure 4.10: Hindcast data after splitting the data. We have 53 weeks and 20 years corresponding to 2000-2019.

Computation of Anomalies

In our first approach we tried to train a model for each week of the year, resulting in only 20 training examples (2000-2019) and a different model for each week. We saw that the model performance was so bad because we had insufficient data in order to apply machine learning methods so we decided that we want to train the models using all the weeks of the year with the idea to enlarge training samples. Training the models using all weeks provide us with $53 \times 20 = 1060$ training samples and a unique model to make

predictions for all 53 weeks. However, in order to do it we need to remove the seasonal cycle of the atmospheric variables. The process we did to remove the seasonal cycle is to compute ECMWF hindcast anomalies with respect to its own biweekly lead-time-dependant climatology. The data anomalies are computed first averaging data over year and realization dimensions (this is called a hindcast climatology) and then subtracting this hindcast climatology to the initial hindcast data. The 2020 forecast anomalies are computed subtracting the hindcast climatology to the initial 2020 forecast data in the same way.

The graphics 4.11 and 4.12 show the temperature values before and after computing anomalies respectively.

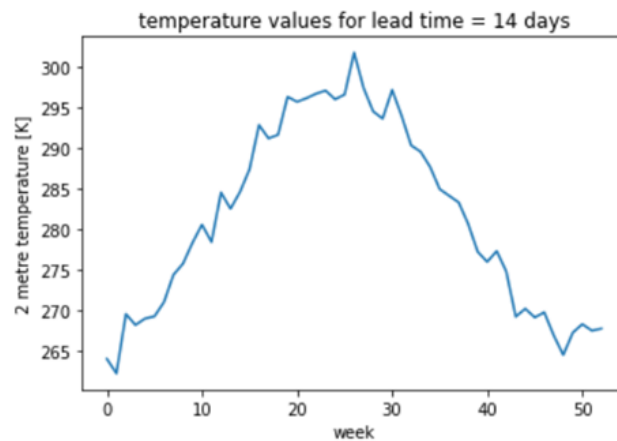


Figure 4.11: Temperature values for one year.

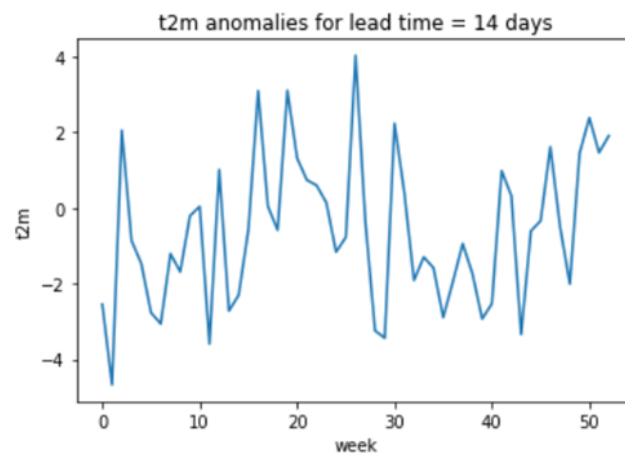


Figure 4.12: Temperature anomaly values for one year.

4.4 Methods

We have employed four machine learning/statistical methods that correct the ECMWF model output of the target variable. All methods are trained separately for each grid point and lead time but using all weeks of the year together by computing anomalies.

4.4.1 Climatology

Climatology consist in simply issue a probability of $\frac{1}{3}$ of observing each tercile category. No training is needed. No predictors are used. We include this method to make sure that we do not perform worse than climatology forecast.

Model Performance

If we consider the RPSS formula (see 3.34) it is clear that the RPSS of this model is going to be 0 for all points because in this case the "machine learning model" and the reference forecast are the same.

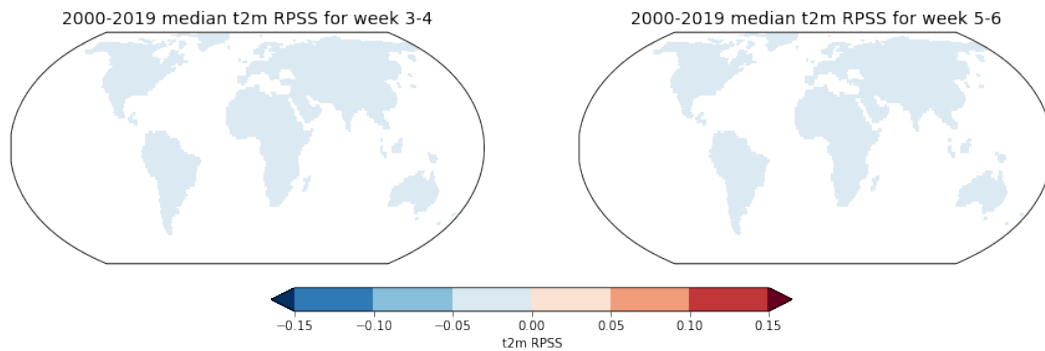


Figure 4.13: This figure shows climatology model temperature RPSS for all grid points and both lead times. The RPSS value is the median value of the 20-fold cross validation results.

4.4.2 Raw ECMWF Forecasts

This method consists in counting the number of ensemble members exceeding the tercile edges and using the proportion of members in each group as the class probability. The tercile edges are computed using the hindcast for the 2000-2019 period (2020 excluded to prevent data leakage), separately for each week of the year but for all members at once. Despite the method being described as *raw*, using tercile edges from the hindcast acts as an implicit bias adjustment. From a machine learning perspective, the training of this method consists in determining the tercile edges. Therefore, we have a separate model for each week of the year. We include this method to ensure that we do not perform worse than the ECMWF model.

Model Performance

We can see that this model performs well in some regions for weeks 3-4, more specifically in the tropics area. For week 5-6 we can see that the model is unskillful for the majority of the points.

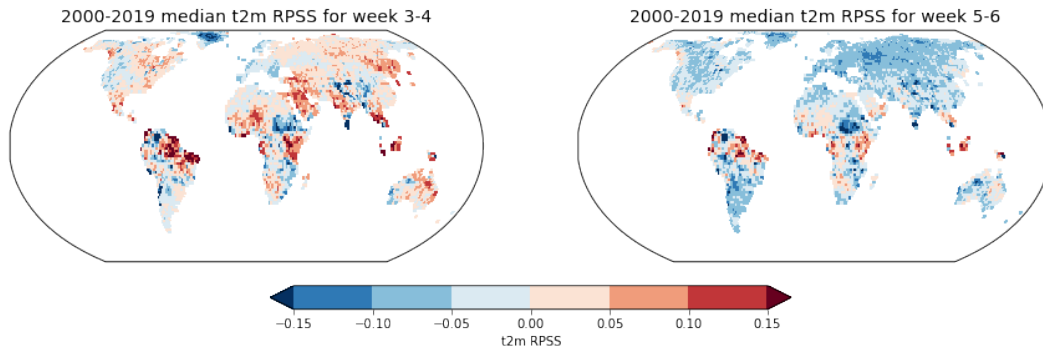


Figure 4.14: This figure shows RAW model temperature RPSS for all grid points and both lead times. The RPSS value is the median value of the 20-fold cross validation results.

		area	northern_extratropics	tropics	southern_extratropics
		lead_time			
t2m	14 days		-0.01	0.02	-0.00
	28 days		-0.05	-0.02	-0.05
tp	14 days		-0.11	-0.51	-0.05
	28 days		-0.12	-0.53	-0.07

Figure 4.15: This table shows raw model t2m and tp RPSS averaged for three different areas and for both lead times.

4.4.3 Logistic Regression

We model the probabilities of observing each tercile category as a Logistic Regression (LR) on the ECMWF ensemble mean (as in Hamill et al. 2004 [4]). The logistic regression hypothesis function in our case is (see equation 3.19):

$$h_{\theta}(x) = \frac{1}{1 + e^{\theta_0 + \theta_1 x}}$$

where x is the ensemble mean and θ_0, θ_1 are the parameters of the model.

In order to have larger samples for training (i.e., determine the parameters) we train one single model for all weeks of the year using ensemble mean anomalies with respect

to the climatological mean. A multiclass implementation of logistic regression that uses *one-versus-all* has been used to obtain probabilities for the three classes that sum up to one.

Model Performance

We can see that this model performs pretty good for weeks 3-4 for the majority of the points. For weeks 5-6 we have less unskillful regions than with the previous methods.

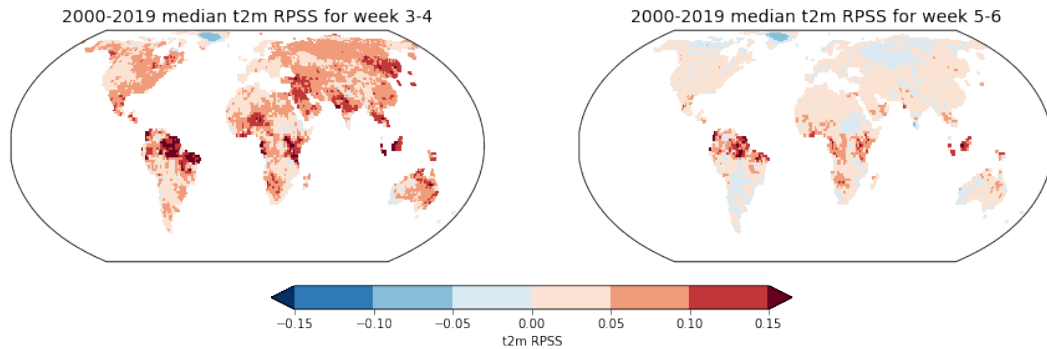


Figure 4.16: This figure shows logistic regression model temperature RPSS for all grid points and both lead times. The RPSS value is the median value of the 20-fold cross validation results.

	area	northern_extratropics	tropics	southern_extratropics	
	lead_time				
t2m	14 days		0.05	0.07	0.04
	28 days		0.00	0.03	0.00

Figure 4.17: This table shows logistic regression model t2m RPSS averaged for three different areas and for both lead times.

4.4.4 Random Forest

We employ a Random Forest (RF) classification algorithm with 11 sorted ensemble members as predictors and the observed category as the target. All the weeks of the year are trained together by using anomalies. For the 2020 forecasts, we subset 11 out of the 51 members available by picking sorted members 1, 6, 11, 16, ..., 51. The random forest (James et al. 2013 [8]) trains 100 bagged trees of depth 4 with a random pre-selection of the split variables. The forecasted class probabilities are obtained by analyzing in each tree the proportions of the training data that ended up in the same leaf as the forecast predictors and averaging for all trees.

Model Performance

We can see that random forest performs similarly to logistic regression for weeks 3-4 but is a bit worse for weeks 5-6.

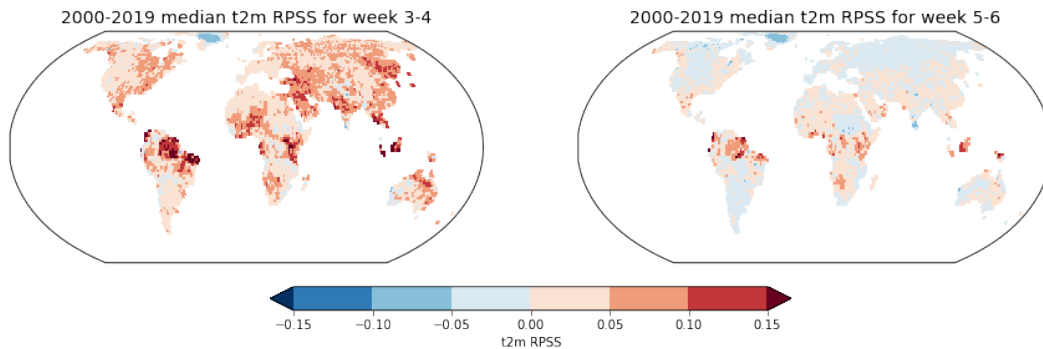


Figure 4.18: This figure shows random forest model temperature RPSS for all grid points and both lead times. The RPSS value is the median value of the 20-fold cross validation results.

		area	northern_extratropics	tropics	southern_extratropics
		lead_time			
t2m	14 days		0.04	0.06	0.03
	28 days		-0.01	0.02	-0.00

Figure 4.19: This table shows random forest model t2m RPSS averaged for three different areas and for both lead times.

4.5 Train-Test-Validation Strategy

We employ K-fold cross-validation (James et al. 2013 [8]) to split the hindcast period into train/validation sets. Specifically, as the test dataset (i.e. the final verification) is a complete year (2020), we employ leave-one-year-out cross-validation to obtain forecast quality estimates for one-year periods. As we have 20 years of hindcast, this corresponds to a 20-fold CV.

In other words, we train each method 20 times, setting aside one year from the training and reserving it for prediction and verification. Although we do not tune any hyperparameter of the methods, the validation set helps to understand the year-to-year variability of the performance. It also helps to prevent overfitting during the multi-method combination at each grid point.

4.6 Combination of Methods

The performance of each method in 2020 can be estimated as a random draw from the 20 performance results obtained in the cross-validation. At each grid point, we want to select the method that performs better. As the RPSS distribution for the 20 hindcast years is skewed, we decide to use the median RPSS instead of the mean RPSS to select the best method.

4.6.1 In-sample Skill by Year

In order to test how our final model perform we compute the averaged RPSS for the in-sample years 2000-2019. That is, we make predictions for each year from 2000-2019 using the cross validation median RPSS to select the best model at each point. Figure 4.20 shows RPSS values for in-sample years 2000-2019 and also the RPSS value obtained for 2020.

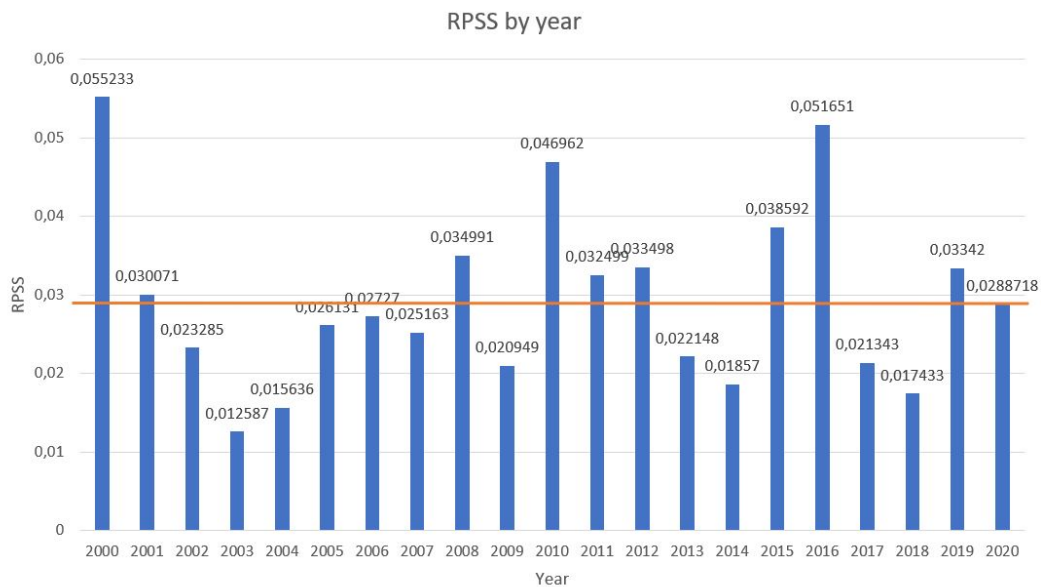


Figure 4.20: In-sample RPSS by year from 2000 to 2019 and 2020 RPSS. This RPSS correspond to a unique value because it is the score averaged over all grid point and both variables and lead times.

4.6.2 Final Model Forecasts

As a final step, we train each method with all the hindcast data (2000-2019) and predict 2020. Then, we select the best method at each grid point based on the hindcast CV results. Note that this selection of the best method it has to be done at each grid point for all 53 forecasts and both variables and lead times.

Multimodel Performance for 2020

The graphics 4.21 and 4.22 shows our ensemble of models performance for 2020 temperature predictions. We can see that we have good scores in tropic areas for both lead times. Also we can see some good results for northern extratropics.

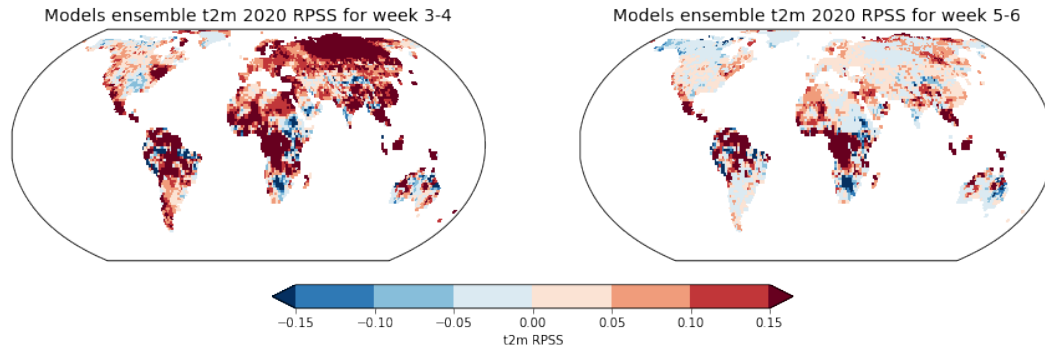


Figure 4.21: This figure shows ensemble of models t2m RPSS for all grid points and both lead times. The RPSS value is the verification for 2020 predictions.

		area	northern_extratropics	tropics	southern_extratropics
		lead_time			
t2m	14 days		0.08	0.11	0.04
	28 days		0.02	0.07	-0.00

Figure 4.22: This table shows ensemble of models t2m RPSS averaged for three different areas and for both lead times.

Decision Map

We use the 2000-2019 median RPSS in order to plot which is the best model at each grid point, that is the decision map for our final model. We can see that for weeks 3-4 we nearly always have a method that makes predictions better than climatology and in general logistic regression is the method with better performance. Figure 4.23 shows the temperature best model at each grid point for weeks 3-4 and weeks 5-6 based on the 2000-2019 median RPSS.

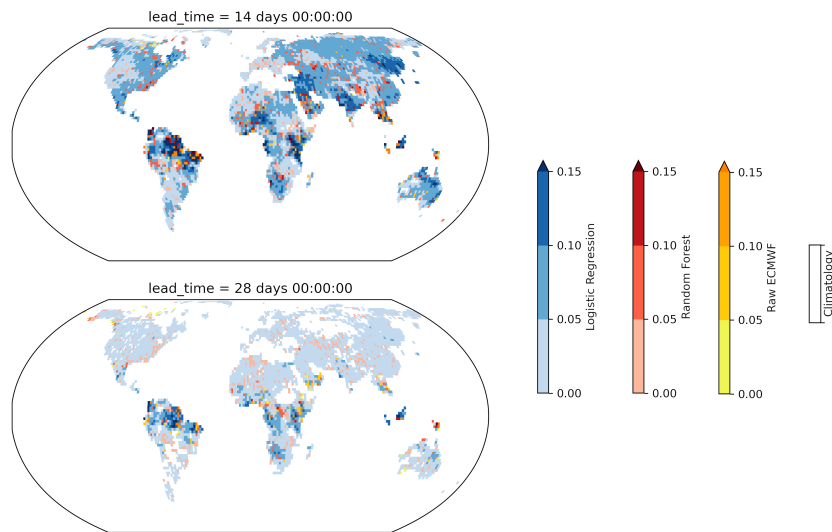


Figure 4.23: Multi-model graphic that shows the best model based on the CV results with the correspondent t2m RPSS for each grid point and both lead times.

4.6.3 Challenge Results

We presented our multi-model ensemble approach to the challenge and we obtained the second place. We also have passed the expert peer review of the method and the code implementation with very good reviews. Figure 4.24 shows the final score for the top three submissions.

4.7 Implementation

In order to give essential details about the implementation it is important to say that this work has been done at Barcelona Supercomputer Center (BCS-CNS), so I had access to the supercomputer computation resources. All the code is done using *Python* with *Jupyter Notebook* environment and run it using a BSC machine called *power9*. We use *xarray* and *dask* python packages in order to work more efficiently with labelled multi-dimensional arrays and with *netCDF* and *Zarr* datasets formats. The random forest and logistic regression classifiers are from *scikit-learn* python library.

([Link to project main code](#))

Train/Predict Parallelization

One important challenge we encounter is to parallelize all computations for each grid point and in order to do that we combine *xarray* and *dask* to parallelize train/predict computations with the *apply_ufunc* function.

The most interesting part of our implementation is the way we use *xarray apply_ufunc* to use *scikit-learn* classifiers. For example, to train the models, we have an atomic function

	Project	User	Submission tag	RPSS	Visibility
1	s2s-ai-challenge-template	Bertrand Denis, David Landry, Jordan Gierschendorf, Arlan Dirkson	submission-methods-7	0.0459475	
2	s2s-ai-challenge-BSC	Sergi Bech, Llorenç Lledó, Lluís Palma, Carlos Gomez, Andrea Manrique	submission-ML_models	0.0288718	
3	s2s-ai-challenge-uconn	Shanglin Zhou, Adam Bienkowski	submission-tmp3-0.0.1	0.00620681	

Figure 4.24: Scores for the top three submissions to the challenge.

that implements the case of training the model for one concrete grid point. Then, we use *apply_ufunc* in order to call this function for every land point with the *dask* option to parallelize the computations. Figure 4.25 and 4.26 show an example of the training code:

```
# Atomic function that trains a random forest.
# We train a classifier for each lat, lon.
def atomic_function_training_rf(dataset, obs):
    obs = np.asarray(obs).reshape(dataset.shape[1]*dataset.shape[2])
    dataset = dataset.reshape((dataset.shape[0],dataset.shape[1]*dataset.shape[2]))
    dataset = (np.sort(dataset, axis=0)) #Sort hindcast members
    dataset = dataset.T
    try:
        clf = RandomForestClassifier(max_depth=4, random_state=0).fit(dataset,obs)
        return clf
    except:
        return None
```

Figure 4.25: This is the base case function. Here we make some data reshapes in order to train the classifier with all weeks of the year together.

Focusing on figure 4.26 note that the parameter *input_core_dims* indicates the dimensions that we do not want to iterate, which in this case are: realization, year, and week from the *X_train* dataset and year, week from the *y_train*. We do not want to iterate these dimensions because our predictors are the 11 members (realization) and as we have said we train a model at each grid point using all years and weeks of the year. The *vectorize = True* parameter indicates that we want to iterate over all other dimensions.

```
#Train a classifiers for each grid point and lead_time and store it in a data array
all_classifiers = xr.apply_ufunc(
    atomic_function_training_rf, X_train.t2m,
    y_train.t2m,
    input_core_dims = [["realization", "year", "week"], ["year", "week"]], vectorize = True,
    dask = 'parallelized',
    output_dtypes=[object]).compute()
```

Figure 4.26: Code example of how to use *apply_ufunc* in order to train the random forest classifiers for each grid point and lead time.

Using this configuration, the atomic function training receives for every land point and for both lead times the training dataset with dimensions realization, year, week and the observation dataset with dimensions year, week.

4.8 Post Challenge and Further Work

The deadline to present contributions to the challenge was November 1st 2020, but after the challenge we continue working on the project to try to improve the results and discover if there are other methods that perform better. In this section we, are going to describe briefly other methods that we have tested and also we described some further work ideas that we thought would be interesting to try.

4.8.1 Unique Model

One approach we tried is to use all grid points as training data along with all years and weeks to have a unique global random forest method that makes predictions for every grid point. We train separately for the two lead times. As with this unique model we lose the spatial information, our idea was to add latitude, longitude, and week of year as predictors. However, since latitude and longitude coordinates are two features representing a three-dimensional space we have the problem that the longitude coordinate goes all around, which means the two most extreme values are actually very close together. So, a better way to put these features would be mapping them to x, y, and z coordinates.

Before adding the spatial information we tried how the model performs with only members as predictors and we saw that it performs so bad so we decided to not further develop this approach because it will take more time than we had. However, we believe that this method could be a good idea but it requires to add properly the spatial and local information we lose training one unique model for all grid points.

4.8.2 Teleconnection Indices as Predictors

To improve our random forest method we tried to add teleconnection indices that give information of the large-scale atmospheric circulation as predictors for example, ENSO, Antarctic oscillation (AAO), North Atlantic Oscillation (NAO), Indian Ocean Dipole (IOD), Madden Julian Oscillation (MJO) and also we add information about soil moisture and stratospheric polar vortex. It is important to note that our ENSO, AAO, NAO and IOD

data correspond to monthly observations and since the effects of these predictors to climate are retarded we put the predictors lagged (e.g. the January teleconnections data is a predictor for all weeks of February). The MJO, soil moisture and stratospheric polar vortex data correspond to daily data so we just have to preprocess the datasets computing the biweekly aggregations for week 3-4 and week 5-6.

After training the model we saw that these predictors do not make a big impact but there is a little RPSS improvement in some regions and a worsening in others. Figure 4.27 shows the performance for 2020 of random forest with teleconnections predictors against the random forest without teleconnections. Note that positive values is where random forest with teleconnections is better and negative values is where random forest with only members as predictors performs better.

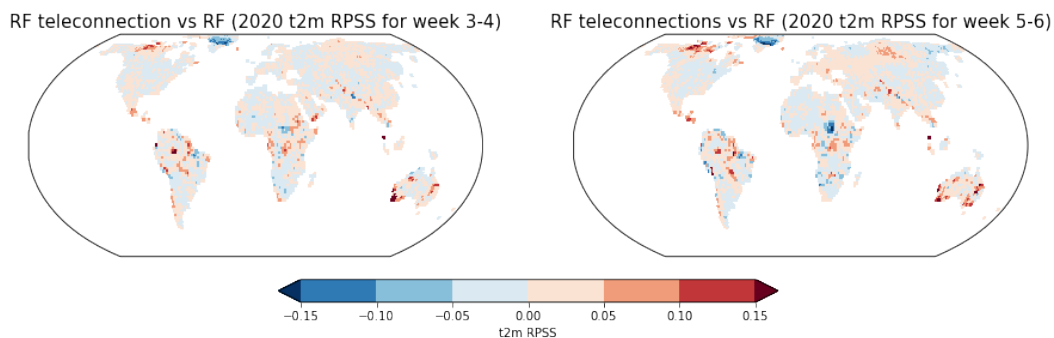


Figure 4.27: RPSS comparison of Random Forest with teleconnections and ensemble members as predictors against Random Forest with only ensemble members as predictors.

4.8.3 XGBoost and Hyperparameter Tunning

Another method we have tried it is *xgboost* but after some testing with hyperparameters that *makes sense* we saw that performs so similar as random forest.

We had the idea to try to do hyperparameter optimization but since we have a model for each grid point it would be so computationally hard to find the best hyperparameters for each land point and for both lead times and variables. One possible approach to decrease the computational cost would be to group grid points into some adjacent regions that are climatology similar and then find the best hyperparameters for these regions.

4.8.4 Quantile Random Forest

Another idea we had was to not use categorical observations as our ground truth and instead use the raw biweekly aggregated values of temperature and total precipitation in order to have a problem where we can apply quantile random forest method to predict the 3-quantiles (i.e. terciles) for 2020 forecast. Then, we can categorize into below-normal, normal and above-normal the forecast data using the predicted terciles as edges.

We develop this method but it not performs as good as we expected and the computational cost turned out to be bigger than with classification random forest.

4.8.5 Adding More Dynamical Models

In order to have more data and try to improve the results we analyze the possibility to add more members from other datasets, but we encountered that the data from other centers has very different constraints and it would be difficult to adapt it to our challenge:

- **ECCC hindcast data:** The problem on using the Canadian model is that the data only covers lead times from 0 to 32 days so it would only be useful for the weeks 3-4 (15 to 28 days).
- **NCEP hindcast data:** The United States hindcast dates differ from ECMWF and ECCC because are from 1999/01/07 to 2010/12/30 corresponding to the weekly Thursdays in 2010.
- **BOM:** The Australian center data has a grid resolution less than 1.5.
- **CNRM:** The French model has the issue that after day 1/3/2016 (included) the model only runs up to day 32 so it would only be useful for the weeks 3-4 (15 to 28 days).

Chapter 5

Conclusions

The hypothesis of this work was that machine learning techniques can improve the skill of sub-seasonal climate predictions either with data driven methods or with model output post-processing techniques that is, using machine learning to improve the predictions done with dynamical models. Consequently, our objective was to develop a model output machine learning method and test it making predictions for 2020.

The model we propose for the challenge consists in a multimodel approach with 4 methods trained locally: climatology, raw (de-biased), logistic regression and random forest. Then, for each grid point and lead time the best method based on the RPSS score using a leave-one-year-out cross validation from 2000 to 2019 is chosen for the final 2020 predictions.

The models were trained using the ECMWF hindcast data and the Climate Prediction Center categorized observations as ground truth. We compute temperature and precipitation anomalies in order to train with all weeks of the year.

The multimodel method got a final RPSS score of 0.0288718 which is the top two score of the challenge and also beats the benchmark from the ECMWF dynamical model which have a RPSS score of -0.0016 . The method and the code got good reviews and the approval of the expert peer review of the challenge.

The code provides a very general implementation using *xarray*, *dask* and *sklearn* that with further work can even lead to a small python package which will make it easier doing machine learning with *sklearn* on gridded global climate data with dimensions time, longitude and latitude.

Looking at the results we can conclude that our method perform better for weeks 3-4 nearly always beating the climatology. For weeks 5-6 we beat climatology at tropics region but in northern and southern extratropics regions we perform equal or worse than the climatology. About the variables we always perform better for temperature which is common because in general total precipitation is more difficult to predict. The best method at each grid point is in general logistic regression but random forest also has good performance.

Summarizing, we see that machine learning can improve the skill of sub-seasonal predictions but there is still room for improvement and some further work that can be done in order to obtain better and more reliable predictions.

Chapter 6

Appendix

In this section we show some interesting RPSS comparisons of our machine learning models performance against the 2020 ECMWF recalibrated predictions which we called 2020 ECMWF benchmark. This plots are very helpful to see in which regions we perform better or in which regions our model is worse.

2020 ECMWF Benchmark

First of all we plot how the 2020 ECMWF benchmark performs against climatology in order to see in which regions it has skill compared to our models.

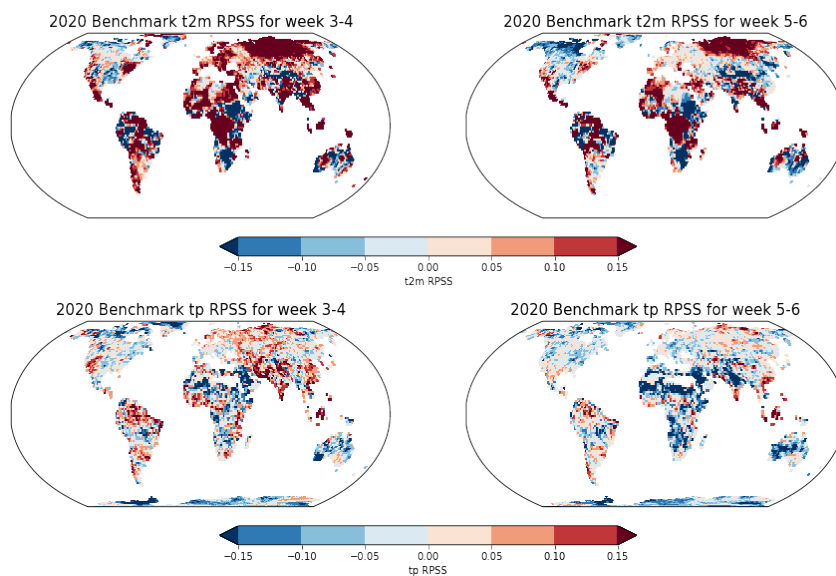


Figure 6.1: 2020 ECMWF benchmark RPSS for t2m and tp.

Logistic Regression vs 2020 ECMWF Benchmark

The positive regions (red) are the ones where logistic regression performs better while the negative regions (blue) are the ones where the benchmarks performs better.

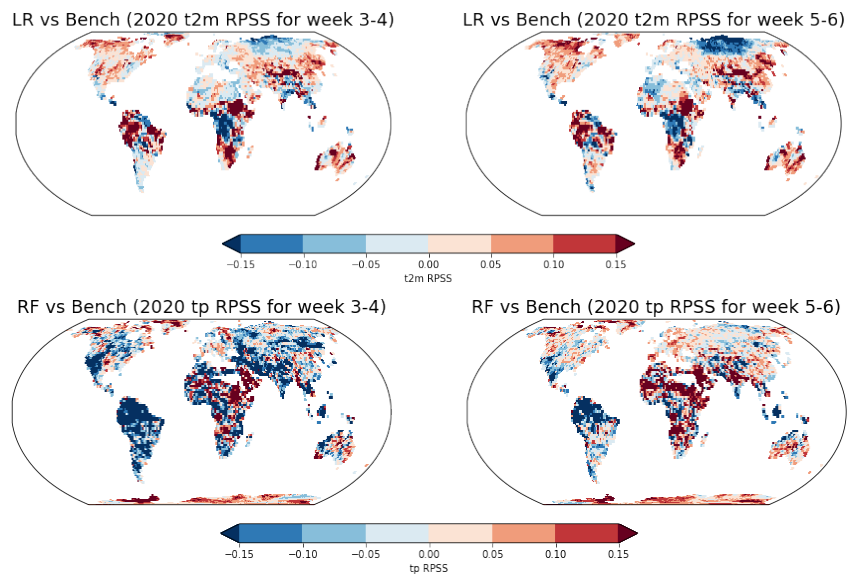


Figure 6.2: Logistic Regression vs 2020 ECMWF benchmark RPSS for t2m and tp.

Random Forest vs 2020 ECMWF Benchmark

The positive regions (red) are the ones where random forest performs better while the negative regions (blue) are the ones where the benchmarks performs better.

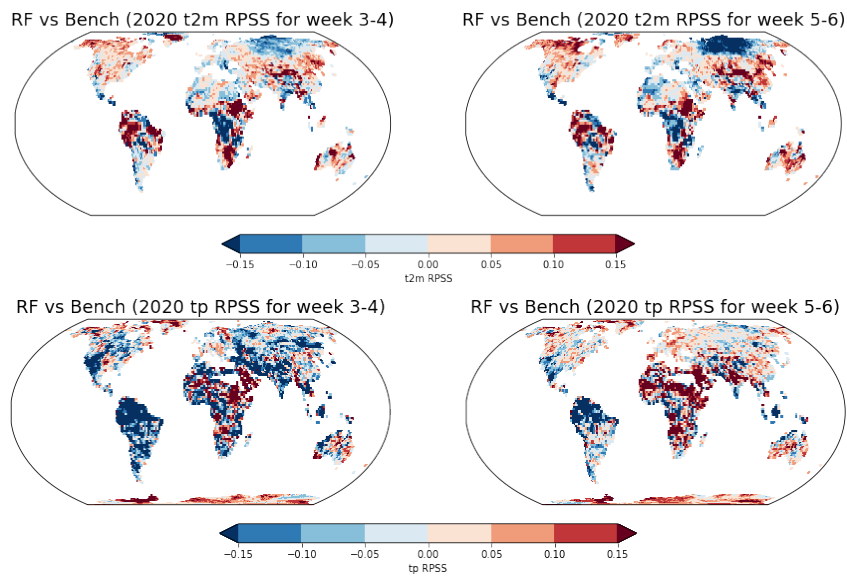


Figure 6.3: Random Forest vs 2020 ECMWF benchmark RPSS for t2m and tp.

Random Forest with Teleconnections vs 2020 ECMWF Benchmark

The positive regions (red) are the ones where random forest unique model performs better while the negative regions (blue) are the ones where the benchmarks performs better.

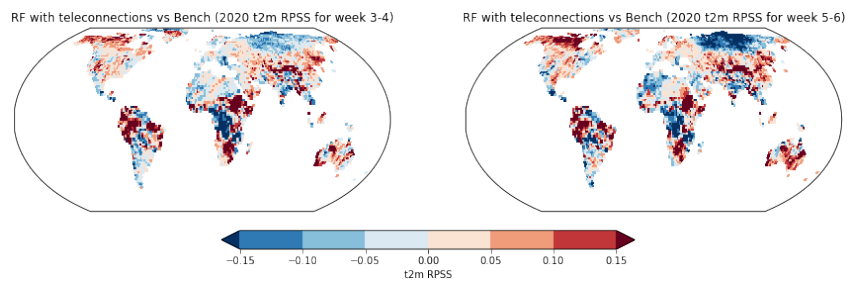


Figure 6.4: Random Forest with Teleconnections vs 2020 ECMWF benchmark RPSS for t2m.

Random Forest Unique Model vs 2020 ECMWF Benchmark

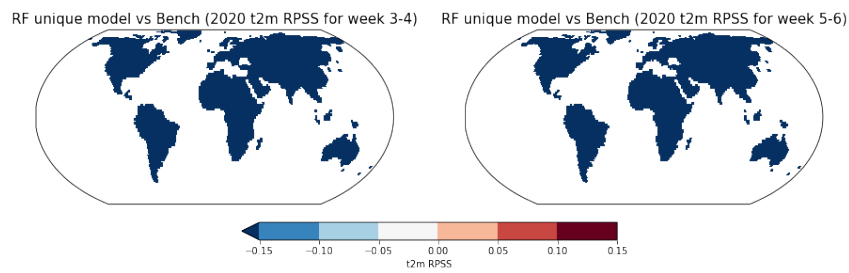


Figure 6.5: Random Forest Unique Model vs 2020 ECMWF benchmark RPSS for t2m.

Logistic Regression vs Random Forest

The positive regions (red) are the ones where logistic regression performs better while the negative regions (blue) are the ones where the random forest with only members as predictors performs better.

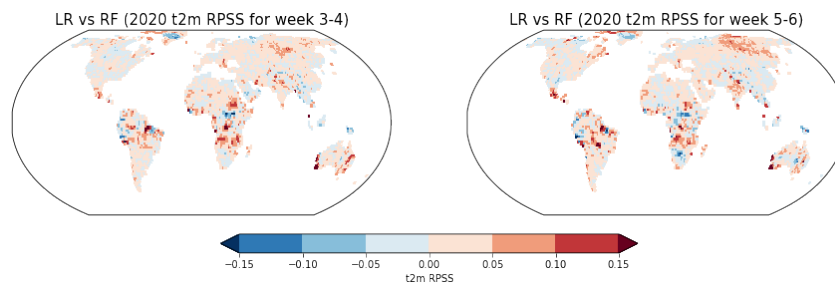


Figure 6.6: Logistic Regression vs Random Forest RPSS for t2m.

Bibliography

- [1] Tom R. Andersson and et. al., *Seasonal Arctic sea ice forecasting with probabilistic deep learning*, preprint 2021.
- [2] Keran Chen, Ping Wang, Xiaojun Yang, Nan Zhang and Di Wang, *A Model Output Deep Learning Method for Grid Temperature Forecasts in Tianjin Area*, 2020.
- [3] Judah Cohen, Dim Coumou, Jessica Hwang, Lester Mackey, Paulo Orenstein, Sonja Totz, Eli Tziperman, *S2S reboot: An argument for greater inclusion of machine learning in subseasonal to seasonal forecasts*, 2018, DOI: 10.1002/wcc.567.
- [4] Hamill, T. M., Whitaker, J. S., Wei, X. (2004). *Ensemble Reforecasting: Improving Medium-Range Forecast Skill Using Retrospective Forecasts*. In *Monthly Weather Review* (Vol. 132, Issue 6, pp. 1434–1447). American Meteorological Society.
- [5] He, S., Li, X., DelSole, T., Ravikumar, P., and Banerjee, A. (2021). *Sub-Seasonal Climate Forecasting via Machine Learning: Challenges, Analysis, and Advances.*, Proceedings of the AAAI Conference on Artificial Intelligence, 35(1), 169-177. Retrieved from <https://ojs.aaai.org/index.php/AAAI/article/view/16090>
- [6] Jessica Hwang, Paulo Orenstein, Judah Cohen, Karl Pfeiffer, Lester Mackey, *Improving Subseasonal Forecasting in the Western U.S. with Machine Learning*, arXiv:2006.07972v2 [cs.LG] 24 Jun 2020.
- [7] Gareth James, Daniela Witten, Trevor Hastie and Robert Tibshirani, *An Introduction to Statistical Learning with Applications in R*, 2017, Springer New York Heidelberg Dordrecht London.
- [8] James, G., Witten, D., Hastie, T., and Tibshirani, R. (2013). *An Introduction to Statistical Learning*. Springer New York.
- [9] K. McGuffie and A. Henderson-Sellers, *A Climate Modelling Primer, Third Edition*, 2005 John Wiley and Sons, Ltd ISBN: 0-470-85750-1 (HB); 0-470-85751-X (PB)
- [10] Nicolai Meinshausen, *Quantile Regression Forests*, *Journal of Machine Learning Research*, 2006.
- [11] Michael Steininger, Daniel Abel, Katrin Ziegler, Anna Krause, Heiko Paeth and Andreas Hotho, *Deep Learning for Climate Model Output Statistics*, 2020.

-
- [12] John M. Wallace and Peter V. Hobbs, *Atmospheric science: An introductory survey*, University of Washington (2005).
- [13] Thomas T. Warner, *Numerical weather and climate prediction*, National Center for Atmospheric Research, Boulder, Colorado (2010).
- [14] Jonathan A. Weyn, Dale R. Durran, Rich Caruana, Nathaniel Cresswell-Clay, *Sub-seasonal forecasting with a large ensemble of deep-learning weather prediction models*, 2021 <https://doi.org/10.1029/2021MS002502>.
- [15] Daniel S. Wilks, *Statistical Methods in the Atmospheric Sciences*, Cornell University, Ithaca, NY, United States (2018).
- [16] Daniel S. Wilks, *Statistical Postprocessing of Ensemble Forecasts*, Cornell University, Ithaca, NY, United States (2018).