Predicting the Momentum Flux-Profile Relationship from Macro Weather Parameters

Emilio Dorigatti

May 28, 2018

Acknowledgments

The work of a lot of people, most of whom I do not know, was needed to enable me to this research project: the hundreds, perhaps thousands, of Open Source maintainers who created and take care of the Data Science libraries for the Python programming language, Python itself, operating systems, distributed computing platforms, and so on. Among them, I am extremely grateful to team behind the Hopsworks platform (Ismail et al., 2017), for giving me access to an astonishing amount of computational resources and their support in using them.

I cannot thank my family enough for supporting me in everything I did and moving mountains [??] so that I could focus on my studies free of worries [??].

This project marks the end of my years as a student. During all this time,

Summary

The study of climate heavily relies on simulations. For efficiency reasons, many important phenomena cannot be simulated, and have to be parametrized, i.e. their effect must be described based on macro parameters. The turbulence resulting from the interaction between the atmosphere and the surface is an example of such phenomena. One of the quantities of interest arising from turbulence is the momentum flux-profile relationship, which relates the transport of momentum (flux) and the change of wind speed with altitude (profile). This quantity can be computed following the Monin-Obukhov Similarity Theory (Obukhov, 1971). However, this theory requires parameters that are hard to measure, both in real life and in the simulations, is only applicable in a restricted range of conditions, and produces predictions that are accurate only up to 20-30% (Foken, 2006).

The goal of this thesis is to compute the momentum flux-profile relationship using only macro weather parameters, which are readily available in climate simulations; this is done using Data Mining techniques on 17 years of weather data collected from the Cabauw meteorological tower in the Netherlands. Moreover, we asses the impact of different sets of features on the prediction error.

Results show that even the simplest linear models are able to compete with the similarity theory, and complex methods such as gradient boosted trees can reduce the mean squared error by almost 50%. Furthermore, these methods are applicable to a much wider range of conditions compared to the similarity theory, while providing roughly the same predictive performance achieved by this theory *in its validity range*. These results are obtained using wind speed and air temperature at different levels, the soil temperature, and the net radiation at the surface; the improvement offered by the heat fluxes is significant, but of low magnitude. The soil heat flux, the dew point, and the hourly trend of the features do not have a tangible impact on the performance.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

This chapter briefly introduces the area for this project, and formally defines and motivates the problem that is tackled. Section 1.1 introduces Climate Science, Machine Learning and Data Mining, then section 1.2 formally defines the research questions that is work sets to answer, and section 1.3 explains why these questions are relevant and which benefits a successful answer would bring. Next, section 1.5 discusses potential ethical issues and sustainability aspects of this project, then section 1.6 explains the research methodology followed to answer the research questions, and, finally, section 1.7 describes how the remainder of the report is organized.

1.1 Background

Climatology, or climate science, studies the long-term average weather conditions. In the last few decades, climate scientists found evidence of ongoing changes in Earth's climate, most notably, a general increase in average temperature; in the long run, this and other changes can potentially have a devastating impact on life on our planet. Regardless of the causal effect of human activities on it, having a solid understanding of the climate allows us to find the best way to mitigate its change, and to preserve our planet.

Climate science is an extremely difficult field, for a number of reasons. First, climate is evident, by definition, only over long periods of time and large distances, making the usual scientific approach of testing hypotheses by conducting experiments inapplicable; instead, climate scientists have to rely on historical data. Second, the atmosphere is a complex and chaotic system, which must be described through systems of nonlinear differential equation. They can be used to create numerical simulations, and the resulting predictions compared to historical data to assess the accuracy of the theory. Furthermore, the chaotic character of the atmosphere makes it impossible to study parts of it in isolation from others, as small scale phenomena affect large scale ones, and vice versa. In spite of this, it is useful to split the atmosphere in vertical layers and horizontal zones, in order to differentiate among conditions and phenomena typically occurring in one area or the other.

The troposphere contains most of the air mass in the atmosphere, and occupies the lowest 10 to 12 kilometers of the atmosphere. This is where weather happens, and it can be divided in a number of sub-layers, the lowest of which is the *atmospheric boundary layer*: it is the region of the atmosphere that is affected by the conditions of the surface. Most human activities happen in this layer, and it is responsible for a large part of the diffusion and transport of aerosol such as, among others, pollutants. Yet, the physics governing the atmospheric boundary layer is not fully understood, and the theory is lacking.

In stark contrast to the traditional, top-down approach of science, recent developments in information technology made bottom-up approaches possible. In this new way of thinking, existing data is used to automatically infer the "best" explanation for the measurements at hand, the underlying laws that originated that data. The field that makes this possible is called *Machine Learning*: it takes advantage of several methods coming from statistics, information theory, optimization theory, etc., to make computers learn from examples. Together with Natural Language Processing and Automated

Planning, it is one of the three main branches of Artificial Intelligence, the sub-field of Computer Science that studies ways of making machines behave intelligently.

An even broader field, called *Data Mining*, is concerned with finding interesting patterns, structures, and regularities in large datasets, where the definition of interesting depends on the specific goals. Data Mining makes use of techniques from Machine Learning, as well as other statistical models, such as ANOVA, and visualization techniques.

1.2 Problem

One important issues in the study of the atmospheric boundary layer is the derivation of flux-profile relationships for wind and temperature: they essentially relate the transport of momentum and heat by the turbulence (flux) with the change of wind speed/temperature with altitude (profile). The flux of momentum is the rate of change of momentum (mass times velocity) per unit of area. Keeping the area constant, the momentum flux increases if the mass of the air increases (e.g. because of changes in temperature, pressure, or humidity) or if the speed of wind increases.

The state of the art relationships are defined by the Monin-Obukhov Similarity theory in terms of the instability parameter ξ, computed as the height above surface scaled by turbulence due to horizontal wind and vertical air movement due to variations in heat. Difficulties in measurements of relevant quantities make this theory accurate only up to 20-30% (Högström, 1996), and applicable in a restricted set of conditions.

Optis et al. (2014) shows that, at least in difficult conditions, several simulation models produce flux estimates that are highly inaccurate, presenting low correlation (mostly below 0.3) and errors that are as large as the fluxes themselves. On the contrary, (Ronda et al., 2003) and (Dimitrova et al., 2016) show that other simulation models are able to provide accurate estimates of the fluxes at the price of biased surface temperature estimates. Intuitively, this trade-off arises because of the deep relationship between the net radiation, the surface fluxes and the surface temperature: one can choose which two quantities to estimate accurately, and sacrifice accuracy on the remaining one. Basu and Lacser (2017) raises awareness on the fact that the lowest grid level used in many simulations recent simulations is too low, outside of the validity region of the Monin-Obukhov similarity theory, although no quantitative measure of the issues caused is presented. Since errors in the simulation accumulate over time, even slight improvements can greatly improve the accuracy of the results after years of simulated time.

Currently, limitations of the validity of the Monin-Obukhov similarity theory are not believed to be a likely explanation for the high scatter that is found in experimental studies, unless in highly stable conditions, where measurement challenges arise (Högström, 1996). With the availability of macro- and micro-meteorological data from specialized observation sites such as Cabauw, in the Netherlands, and the recent developments in Machine Learning, this conjecture can be finally put to the test. More specifically, we pose the following

Research Question 1: How can the data from the Cesar database be used to predict the momentum flux-profile relationship more accurately than the Monin-Obukhov similarity theory, by using only macro weather parameters?

Research Question 2: What impact do different macro-weather parameters have on the quality of the predictions obtained?

1.3 Purpose

Affirmative answers to the research questions would contribute to improve the quality of current global circulation models and large scale eddy simulations, allowing them to obtain more accurate predictions of the momentum flux-profile relationship, leading to more precise long term climate forecasts.

1.4 Goals

The first research question is answered by proposing one or more Data Mining methods, i.e. feature selection and extraction, and a Machine Learning model, that obtain sufficient predictive performance, whereas the second research question is answered by a comparison of the performance of the models when using certain subsets of features.

1.5 Ethics and Sustainability

Carrying out this project posed no ethical issues: data collection was mostly automated and performed by a third party, which provides free and open access to it, and privacy and confidentiality are not relevant issues since no test subjects were needed to perform the research.

The author strongly aligns with the Open Science movement, according to which scientific research should be made as open and accessible as possible, so that it can be examined, extended and improved by as many people as possible. This is what allowed mankind to become what it is today, and it should not change. For this reason, this work was performed entirely with Free and Open Source tools, providing a low barrier to entry, and focusing on reproducibility of the results. The entirety of the analysis and its historical record, from the very data to the generation of plots and tables, is openly accessible[1] and can be performed by anyone with enough computational resources.

http://www.un.org/sustainabledevelopment/health/

Some of the analysis performed in this work required a considerable amount of computational resources, in the order of hundreds of cores and terabytes of memory. Such resources are power hungry and require electricity to run and to be kept at an appropriate temperature [todo some stats about internet power usage etc], as well as proper support infrastructure, e.g. for networking. In an effort to reduce the energy requirements, only the necessary experiments were performed, and care was taken not to do unnecessary work, for example by reusing results of previous experiments where possible.

1.6 Research Methodology

The two most widely used research methodologies in data mining are CRISP-DM, the cross-industry standard process for data mining, by Chapman et al. (2000), and KDD, knowledge discovery in databases, by Fayyad et al. (1996). Despite the difference in terminology, they essentially describe the same process for extracting useful information, or *patterns*, from one or more raw data sources. This process is composed by the following steps (we use the nomenclature of CRISP-DM):

1. *Business Understanding:* Get familiar with the application domain, obtaining necessary background knowledge, then identify the goals of the process from the customer's perspective and translate them to data mining tasks;
2. *Data Understanding:* Obtain and clean the data that is necessary to conduct the data mining tasks, appropriately dealing with empty, missing and outlier values, and conducting simple analyses and visualizations to inspect the overall quality and gain further understanding of the data itself;
3. *Data Preparation:* Transform the data in the format needed to conduct the data mining tasks, e.g. by performing feature selection and/or extraction, such as joins, aggregates, etc.
4. *Modeling:* Conduct the data mining tasks, using the data prepared in step 3;
5. *Evaluation:* Interpret and evaluate the results obtained in step 4, both from a data mining perspective and from a business perspective, and quantify to what extent the goals defined in step 1 were achieved;
6. *Deployment:* Act on the outcome of the data mining tasks, report it and possibly take action based on the results and/or deploy models to support live decision making, etc.

[1] `https://github.com/e-dorigatti/mscthesis`

It is important to mention that these steps are not strictly sequential, but heavily interact with one another: insights gained from the data can give rise to new questions, the evaluation of the results can highlight flaws in earlier steps, etc.

1.7 Outline

The thesis is organized as follows: in chapter 2, we introduce the relevant background in climate science and machine learning, then, in chapter 3, we describe how the data is collected and used to answer the research question, chapter 4 contains the results of such investigation, and, finally, chapter 5 discusses the results, pointing out limitations and possible directions for expanding this work.

Chapter 2

Background

This chapter introduces the basic concepts the reader should be qualitatively familiar with, in order to understand the content of this thesis. It is assumed readers are already knowledgeable of simple mathematical concepts, such as calculus, linear algebra, statistics, and probability theory. We first describe fluid dynamics, with a focus on the boundary layer (section 2.1), then use that knowledge to describe the atmospheric boundary layer and the Monin-Obukhov similarity theory (section 2.2), and finally describe the Machine Learning algorithms that will be used to answer the research questions (section 2.3).

The most important section is 2.2.3, since it introduces the state of the art which this work tries to improve upon, but readers should feel free to skip the sections they are already familiar with.

2.1 Fluid Dynamics

Fluid dynamics is the discipline that studies the flow of fluids; it has several branches that study different fluids, such as aerodynamics (the study of air motion) and hydrodynamics (the study of water motion). These disciplines are routinely applied when designing cars, airplanes, ships, pipelines, etc.

2.1.1 Laminar and Turbulent flow

There are two distinct ways in which particles in a fluid can move: laminar flow and turbulent flow. In the former, all the particles move orderly, perhaps with a different speed, but all in the same direction, whereas in the latter the movement of particles is highly chaotic and unpredictable, and tends to dive rise to eddies of varying sizes. People are most familiar with the distinction between the two through the smoke rising from a cigarette, which starts smooth and becomes turbulent shortly thereafter, as in figure 2.1. The kind of flow in under specific conditions can be predicted using the Reynolds number Re, which is the ratio between inertia forces, favoring turbulent flow, and viscosity forces, stabilizing the fluid towards laminar motion:

$$Re = \frac{\rho u L}{\mu} = \frac{u L}{\nu}$$

With ρ the density of the fluid, u its velocity, L a characteristic linear dimension of the system under consideration, μ and ν the kinematic and dynamic viscosity of the fluid. The viscosity describes, intuitively, how much the molecules of the fluid tend to stick together and resist motion by generating drag. For example, water has low viscosity, and honey has high viscosity.

Since turbulence is random, it is usually studied in terms of the statistical properties of physical quantities through the Reynolds decomposition; given a quantity $a(s,t)$ which varies in space and time, we can compute its average

$$\overline{a}(s) = \frac{1}{T}\int_{T_0}^{T_0+T} a(s,t)dt$$

Figure 2.1: Smoke from a cigarette, and the transition from laminar to turbulent flow.

and the deviation from the average

$$a'(s,t) = a(s,t) - \overline{a}(s)$$

By definition, $\overline{a'} = 0$, which means that all the effects of turbulence are contained in a'. Common statistical properties such as variance and covariance are expressed respectively as $\overline{a'a'}$ and $\overline{a'b'}$.

2.1.2 The Boundary Layer

In the context of fluid dynamics, the boundary layer is the region where a fluid flows close to a solid surface. Imagine a laminar flow close to a solid surface; because of viscosity, the molecules flowing near the surface move slower, and, in the limit, the velocity of the molecules in direct contact with the surface is 0 (this is called the *non-slip condition*). Thus, the velocity of the fluid increases smoothly, continuously and monotonously with the distance from the solid, until it reaches the *free-flow* velocity, after which it stays constant. The region close to the surface, where the fluid moves slower, is called the *boundary layer*, and is the region where the viscosity of the fluid influences its motion. Its height δ can be defined when the local velocity surpasses a certain threshold, such as 99% of the free-flow velocity.

The variation of velocity with distance from the surface, $\partial\overline{u}/\partial z$, is called *shear*, and, together with viscosity, determines the materialization of turbulence in the flow. Every layer of fluid is squeezed between a faster moving layer above and a slower moving below; in high shear conditions, this causes high stress on the particles, and prevents them from moving orderly, thus leading to turbulent motion. Figure 2.2 shows turbulence forming close to the wall in a canal, where the water flows from right to left. Viscosity and the no-slip condition prevent this phenomenon to arise in a region very close to the solid surface, called the *laminar (or viscuous) sub-layer*, where we still find laminar motion.

The strength of the turbulence is proportional to $u_{rms} = (\overline{u'^2})^{1/2}$, which is, in turn, proportional to the shear. Again, because of the no-slip condition, u_{rms} is zero at $z = 0$, increases in the laminar sub-layer, and decreases to 0 at the end of the boundary layer, assuming laminar flow outside of it. Higher free-stream velocity generates higher shear, more turbulence, and a thinner laminar sub-layer. The strength of turbulence can be written in units of velocity, resulting in the *friction velocity*, computed as $u_* = (\tau/\rho)^{1/2} = (\nu \cdot \partial\overline{u}/\partial z)^{1/2}$, where τ is the shear stress, ρ is the density of the fluid, $\nu = \mu/\rho$ is the kinematic viscosity, and μ the dynamic viscosity. Therefore, the friction velocity increases with shear and viscosity, and decreases with density; it is proportional to the free-stream velocity and the turbulence strength, and inversely proportional to the height of the laminar sub-layer. Expressing the shear τ in terms of the horizontal kinematic momentum fluxes $\overline{u'w'}$ and $\overline{v'w'}$, we obtain an equivalent formulation of u_*:

$$u_* = \left(\overline{u'w'}^2 + \overline{v'w'}^2\right)^{1/4} \tag{2.1}$$

Figure 2.2: Turbulent boundary layer at the edge of a canal; water flows from right to left. Colors are inverted to make the patterns more visible.

The mean velocity $\overline{u}$ increases linearly within the laminar sub-layer, then logarithmically until the end of the boundary layer, thus the shear decreases further away from the surface. In the logarithmic sub-layer, the velocity is computed as $\overline{u}(z) = u_*(\log z - \log z_0)/\kappa$, where z_0 is the characteristic roughness of the surface, and κ is the von Karman's constant, whose value is around 0.4 Högström (1996). The characteristic roughness depends on the texture of the surface, and its relationship with the height δ_s of the laminar sub-layer; if the roughness scale is smaller than δ_s, the logarithmic velocity profile is not affected by the texture, because the laminar sub-layer completely covers the variations on the surface, and we have the so-called smooth turbulent flow. If, on the contrary, the bumps in the surface are larger than δ_s, the laminar sub-layer follow the profile of the surface, and the logarithmic velocity profile is altered depending on the texture, a regime called rough turbulent flow.

2.2 The Atmospheric Boundary Layer

The atmosphere is composed by air, which is behaves like a fluid. Therefore, close to the Earth's surface, in the region called *atmospheric boundary layer* (ABL), we find the same effects described in the previous section. Additionally, there are other phenomena that complicate things further, such as the temperature of the surface, which changes widely from day to night and from Summer to Winter, the rotation of the Earth, the varying roughness of the surface, due to cities and vegetation, etc. The effect of the surface on the first few hundred meters of the atmosphere is the main focus of *boundary layer meteorology*.

The height of the ABL typically varies between 100 and 1000 meters, highly depending on the conditions, and it is always turbulent. There are two main instabilities driving turbulence in the ABL:

- Shear instability: caused by shear, the mechanism described in the previous section. This happens at high Reynolds number, and, by using typical values for the ABL, we find Re well above 10^6.
- Rayleigh-Bernard (also known as buoyancy driven) instability: is caused by the decrease of potential density[1] with height, or, in other words, when warm fluid is below cold fluid; the warm fluid will rise, and the cold fluid will drop, a phenomenon called *convection*. During hot Summer days, the surface is much warmer than the air, thus the air close to the surface will heat and tend to rise.

Turbulence has the very important role of transport and mix of air properties, such as heat, moisture, particles, aerosols, etc. This is especially true in *unstable* conditions, when the air moving upwards (e.g. because it is warmer) is less dense than the air moving downwards; when the contrary happens, the ABL is called *stable*.

The ABL can be divided in two main sub-layers: the inner surface layer and the outer Ekmann layer. This distinction is mainly done based on the scale of the dominating turbulent eddies: they are much smaller than the height of the ABL in the surface layer, and of comparable size in the outer layer.

[1] the potential density is the density that a parcel of air would attain if brought at a standard reference pressure adiabatically, i.e. disallowing exchanges of heat with its surroundings. Potential density is useful to compare densities irrespectively of pressure, i.e. altitude

It is very important to have a macroscopic understanding of the turbulent processes in the ABL, because they happen at length and time scales too small to be simulated in global climate models. The process of expressing the result of turbulence as a function of large scale parameters is called parametrization; having realistic models is essential in order to conduct precise simulations of the global climate in the scale of tens or hundreds of years, because errors tend to accumulate and amplificate as the simulation goes on. Other fields that benefit from the study of the ABL are urban meteorology (interested in the dispersion of pollutants), agricultural meteorology (interested in the formation of frost and dew, the temperature of the soil, etc.), aviation (predict fog and strong winds), and so on.

2.2.1 Surface Fluxes

A flux measures the amount of a physical quantity that flows through a surface. In the context of boundary layer meteorology, we are interested in the flows through the surface of earth, because, through them, the surface and the atmosphere exchange energy; these fluxes are thus measured in $\mathrm{W\,m^{-2}}$. The main source of energy for the surface is short-wave radiation coming from the sun, and long-wave radiation coming from the atmosphere and the clouds. A small amount of long-wave radiation is emitted from the surface, therefore let the net radiative flux be R, positive when the surface gains energy.

The main fluxes by which the surface gains or loses energy to the atmosphere are called the turbulent flux of *sensible heat* H, also called kinematic heat flux, and the turbulent flux of *latent heat* λE, also called kinematic flux of water vapor/moisture. The difference between the two is that the former causes an actual change of temperature, whereas the latter does not affect temperature[2]. The main causes of sensible heat fluxes are conduction and convection, whereas the main cause of latent heat fluxes is water movement: condensation, evaporation, melting, etc.

The final flux of interest is the soil heat flux G, which is the heat transferred into or out of the deeper layer of the soil and not given back to the atmosphere. These four fluxes are linked by the surface energy balance equation:

$$R = H + \lambda E + G$$

which states that the total incoming energy R must be equal to the energy given back to the atmosphere $H + \lambda E$ (not counting long-wave radiation, which is accounted to in R) plus the energy absorbed by the surface G, assuming the temperature is constant. When the temperature is not constant, one can write a budget equation relating the changes in these quantities.

The turbulent fluxes H and λE are constant in the surface layer. Experimentally, the energy balance is not always achieved (Bosveld, 2018) because of the difficulty in measuring fluxes due to eddy correlation being inaccurate.

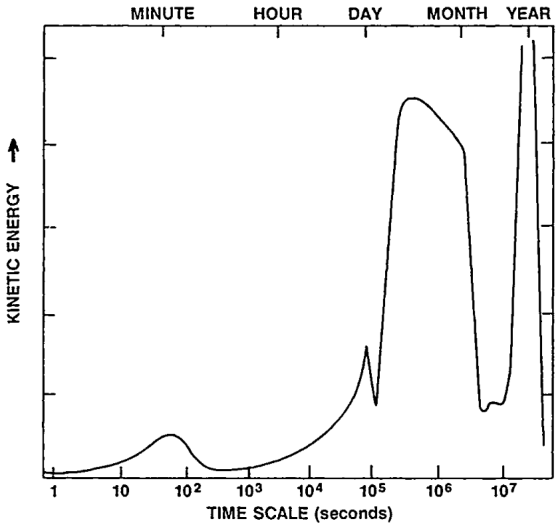
2.2.2 The Turbulence Kinetic Energy Budget

Kinetic energy is energy stored in form of movement: faster or heavier objects have more kinetic energy than slower or lighter ones. The Reynolds decomposition allows us to decompose the kinetic energy of turbulent flows in two terms: one caused by the mean flow, and one caused by turbulence. This decomposition can be justified by examining the temporal spectrum of kinetic energy, shown in figure 2.3. Four peaks are visible, corresponding to different sources of kinetic energy: turbulence, day-night cycle, synoptic scale variability, lows and highs passing, and seasons. Importantly, there are few sources of kinetic energy in the 30 minutes to one hour time scale; this so-called spectral gap allows us to separate between turbulence and other sources of fluctuations in the atmosphere.

From now on, we will use a coordinate system with the x axis aligned to the average horizontal wind direction, the y axis perpendicular to it, and the z axis pointing away from the surface. Then, we will use the letters u, v and w to denote the components of the wind along the axes x, y and z respectively; clearly, $\overline{v} = 0$. Eddy fluxes can then be described in terms of covariances: let θ denote the

[2] imagine a pot of boiling water; selecting a higher temperature on the stove will not increase the temperature of water above 100 °C, but will make it boil faster. The additional heat introduced in the system is dissipated through increased evaporation

Figure 2.3: Long term average of atmospheric kinetic energy at different time-scales. The peaks in the scale of days and months and years are due to the day-night and Summer-Winter cycles, the peaks in the monthly scale are due to baroclinic instability in the mid-latitude westerlies, and the peaks at one minute are due to convection and atmospheric turbulence (Scorer, 1992; Vinnichenko, 1970)



potential temperature[3], then $\overline{w'\theta'}$ is the turbulent heat flux, i.e. the sensible heat flux in the vertical due to turbulence. Usually the ABL is studied assuming homogeneous horizontal conditions, because they vary on a length scale larger than the height of the ABL. Because of this, the horizontal eddy correlations $\partial\overline{u'a'}/\partial x$ and $\partial\overline{v'a'}/\partial y$ are usually of negligible intensity, and are thus ignored; this is to say that the vertical fluxes are assumed to be horizontally constant. Note that this is not necessarily true when clouds are involved.

It is important to notice that turbulence is dissipative in nature. Consider a hot Summer day, where air is warmer close to the surface, and a circular eddy moving some air up and some down, so that the average motion is zero. The parcel of air moving up ($w' > 0$) ends up being warmer than its surroundings ($\theta' > 0$), while the one moving down ($w' < 0$) will be colder ($\theta' < 0$); the result is a net transport of heat through the eddy: $\overline{w'\theta'} > 0$. On the contrary, imagine a cold night, where the air close to the surface is colder; the same eddy would transport a colder parcel of air upwards, and a warmer one downwards. In both cases, the end result would be a net transport of heat without transport of mass; the eddy must lose energy, and thus dissipate over time.

Since turbulence changes over time, we are more interested in the change of kinetic energy, the *turbulent kinetic energy budget*. A full derivation is out of the scope of this work, but its final form (Högström, 1996) can be derived from prime physical principles, resulting in

$$\frac{\partial\overline{e'^2}}{\partial t} = \underbrace{\overline{u'w'}\frac{\partial\overline{u}}{\partial z}}_{P} - \underbrace{\frac{g}{T}\overline{w'\theta'}}_{B} + \underbrace{\frac{\partial}{\partial z}\frac{\overline{w'e'^2}}{2}}_{T_t} + \underbrace{\frac{1}{\rho}\frac{\partial\overline{p'w'}}{\partial z}}_{T_p} + \epsilon \tag{2.2}$$

where $e'^2 = u'^2 + v'^2 + w'^2$. The P term is the production due to shear, B is the production due to buoyancy, T_t is the turbulent transport of TKE by large-scale eddies, T_p is the transport due to pressure, and ϵ is molecular dissipation due to viscosity. P and B are the most prominent terms, and the transport terms are close to zero in neutral conditions (Högström, 1996).

The P term is always positive, as energy is taken from the mean flow to the turbulent one (in other words, turbulence can only generate kinetic energy). On the other hand, the contribution from

[3]the potential temperature is final temperature after bringing a parcel of air to a standard pressure adiabatically, i.e. not allowing exchange of temperature with the surroundings. It is a useful mean to compare temperatures irrespectively of pressure, i.e. altitude

buoyancy can be either positive or negative, depending on the difference of temperature between a parcel of air moved by the turbulence and the surrounding air. When $\overline{w'\theta'}$ is negative, the turbulence is moving cold air upwards and warm air downwards; these parcels of air will try to undo the effect of turbulence, thereby increasing the overall kinetic energy. A similar reasoning goes for when the heat flux is positive.

2.2.3 Monin-Obukhov Similarity Theory

One of the quantities of interest is the momentum flux-profile relationship, computed as follows:

$$\frac{\partial \overline{u}}{\partial z}\frac{\kappa z}{u_*} \tag{2.3}$$

where κ is the von Karman constant, which roughly equals 0.4. The profile is expressed by the $\partial\overline{u}/\partial z$ term, and the momentum flux is contained in the friction velocity u_* according its interpretation in equation 2.1.

One of the factors to distinguish laminar from turbulent flow is the length scale L of the system. This length scale for the ABL was derived by Obukhov (1971), and forms the basis of the similarity theory. According to this theory, the normalized wind profile of equation 2.3 can be expressed as an unique *universal* function of $\xi = z/L$, with

$$L = -\frac{u_*^3}{\kappa\frac{g}{\theta_v}\frac{Q}{\rho c_\rho}} = -\frac{u_*^3 T_v}{\kappa g\overline{w'\theta_v}} \tag{2.4}$$

where

- $g = 9.81\,\mathrm{m\,s^{-2}}$ the acceleration due to Earth's gravity
- θ_v virtual temperature[4], obtained as

$$\theta_v = \theta\frac{1 + r_v/\epsilon}{1 + r_v} = \theta(1 + 0.61 \cdot q) \tag{2.5}$$

  where θ is the air temperature, r_v is the mixing ratio, $q = r_v/(1 + r_v)$ the specific humidity, and ϵ is the ratio of the gas constants of dry air and water vapor, roughly 0.622.

- $T_v = \theta_v/g$ is the buoyancy parameter;
- ρ the air density, computed from the pressure P and the specific gas constant for dry air $R = 287.058\,\mathrm{J\,kg^{-1}\,K}$ as

$$\rho = \frac{P_0}{RT_v}$$

- c_ρ specific heat of dry air, $1005\,\mathrm{J\,kg^{-1}\,K^{-1}}$ at $300\,\mathrm{K}$
- Q the buoyancy flux, approximated by $H + 0.07\lambda E$ and measured in $\mathrm{W\,m^{-2}}$
- $\overline{w'\theta_v} = Q/\rho c_\rho$ the flux of virtual potential temperature, measured in $\mathrm{K\,m\,s^{-1}}$
- $T_* = -\overline{w'\theta}/u_*$

The stability parameter ξ is positive for stable conditions, where wind shear dominates the production of TKE, and negative for unstable conditions, where buoyancy is the main contributor to turbulence. It approaches 0 in the limit of neutral stratification (i.e. $\partial\overline{\theta}/\partial z = 0$), because the temperature flux goes to 0 causing L go to infinity.

[4]potential temperature of dry air if it had the same density as moist air. It allows to use formulas for dry air when the air is not dry.

The universal function must be determined experimentally. This is no easy task, and considerable effort has been devoted to it; one of the greatest difficulties lies in obtaining accurate and unbiased measurements, especially the fluxes. Högström (1988) is a meta-study that aggregates and improves many previous results, and suggests the following expressions:

$$\phi_m(\xi) = \begin{cases} (1 - 19.3\xi)^{-1/4} & -2 < \xi < 0 \\ 1 + 6\xi & 0 < \xi < 1 \end{cases} \tag{2.6}$$

The Monin-Obukhov similarity theory (MOST) is only applicable in the surface layer, at heights much larger than the aerodynamic roughness length, and with $|\xi| < 2$; intuitively, when $|\xi|$ is too large, the eddies do not feel the effect of the surface anymore, and, under very stable stratification ($\xi >> 1$), vertical motion is hampered, so the height z does not play a role. Even under ideal conditions, the predictions of this theory are accurate up to 10-20% (Foken, 2006). Some works, such as Yaglom (1977), argue that the poor agreement between data and experiments, and even among experiments themselves, can be largely attributed to difficulties in measuring the necessary quantities. Recent attempts (Grachev et al., 2007; Mcnaughton, 2006; Wilson, 2008; Hatfield et al., 2005) try to extend, improve, or altogether replace, this similarity theory or pars thereof.

2.3 Statistics and Machine Learning

The goal of Machine learning is to develop algorithms that allow computers to learn from examples. Learning is intended as the ability of inferring general rules from the available examples, so that new, previously unseen examples can be correctly characterized. The set of samples from which the computer is supposed to learn is called the *training set*, and each sample is a sequence of numbers describing its attributes, or *features*. There are three approaches in machine learning:

- *Supervised* learning: in this setting, the examples are composed of an input and a desired output, and the goal is to build a model that can correctly predict the output given the input. There are different algorithms depending on the type of output: *regression* algorithms predict continuous output, while *classification* algorithms predict discrete output.

- *Unsupervised* learning: in this setting, no output is available. The task of the algorithm is to figure out hidden relationships between the samples in the training set, for example whether they form clusters, or there are anomalous samples, or the correlation between features of the examples.

- *Reinforcement* learning: in this setting, the computer is free to act in an environment and to observe how the environment responds to its actions. Additionally, it receives a *reward* for every action it takes, and the goal of the computer is to learn a sequence of actions that maximizes the received reward. Reinforcement learning is often applied in robotics [citation] and game playing [alphazero citation].

A supervised machine learning model uses a set of parameters to compute the output value starting from the input features. The actual parameters values are learned from the training set in a process called *training*. This process is controlled by another set of parameters called hyper-parameters, whose value can be found from the training data as well. Whereas the parameters control the relationship between input and output, hyper-parameters control the "character" of the learning algorithm, such as how eager or conservative it is in learning minute details in the features. Learning too many details can be detrimental, because some differences can be due to noise, rather than actual differences.

The next sections describe the theory of learning, a general technique to estimate the parameters of a regression model, and introduce several machine learning algorithms for regression.

Notation: Scalars are denoted in *italic*, vectors (always column) and matrices in **bold**. The training set contains N training samples, indexed by n, and each sample is a pair of feature vector $\boldsymbol{x}_n \in \mathbb{R}^D$ and a target value $t_n \in \mathbb{R}$. The feature vectors are grouped in the $N \times D$ matrix $\boldsymbol{X} = [\boldsymbol{x}_1 \cdots \boldsymbol{x}_N]^\intercal$ and the target values in the $N \times 1$ vector $\boldsymbol{t} = [t_1, \ldots, t_n]^\intercal$. Models are parametrized by a parameter vector $\boldsymbol{\theta}$ and their output for the feature vector $\boldsymbol{x}_n$ is $f_n = f(\boldsymbol{x}_n; \boldsymbol{\theta})$. The vector containing both parameters and hyper-parameters is denoted with $\boldsymbol{\Theta}$.

2.3.1 Learning Theory

The goal of supervised learning is to use the training examples $D = (\boldsymbol{X}, \boldsymbol{t})$, independent and identically distributed according to an unknown distribution p_{XT}, to find a good prediction rule $\hat{f} : \mathcal{X} \to \mathcal{Y}$ among a family of available functions $\mathcal{F}$. In most practical cases, $\mathcal{X} = \mathbb{R}^D$ and $\mathcal{Y}$ is either $\mathbb{R}$ for regression or a subset of $\mathbb{N}$ for classification. The goodness of a prediction rule f is measured through a *loss function* $\ell : \mathcal{Y} \times \mathcal{Y} \to \mathbb{R}$ that tells, given a target value t and a guess $y = f(x)$, how much the guess is off. The *risk* of an estimator f is simply its expected loss:

$$R(f) = \mathbb{E}_{(x,t)\sim p_{XT}} \left[\ell(f(x), t) | f\right] \tag{2.7}$$

The ideal situation is to find the estimator f^* that has the lowest risk; unfortunately this is not possible, because the distribution p_{XT} is unknown. Since the training data is a sample from this distribution, we can compute the *empirical risk* of f on this set of examples instead:

$$\hat{R}(f) = \mathbb{E}_{(x,t)\sim D} \left[\ell(f(x), t) | f\right] = \frac{1}{N} \sum_{n=1}^{N} \ell(f(x_n), t_n) \tag{2.8}$$

We can now use the empirical risk as a surrogate for the true risk, selecting the estimator

$$\hat{f} = \operatorname*{argmin}_{f \in \mathcal{F}} \hat{R}(f) \tag{2.9}$$

as our best guess. This procedure is called *empirical risk minimization*. Note that $\hat{f}$ is a random function, because it depends on D, which is a random variable. An interesting question to ask is how good $\hat{f}$ actually is, or, in other words, what is its expected true risk $\mathbb{E}[R(\hat{f})]$ and how it compares to the lowest attainable risk $R(f^*)$. This latter quantity can be decomposed as

$$\mathbb{E}[R(\hat{f})] - R(f^*) = \left(\mathbb{E}[R(\hat{f})] - \inf_{f \in \mathcal{F}} R(f)\right) + \left(\inf_{f \in \mathcal{F}} R(f) - R(f^*)\right) \tag{2.10}$$

where the first term is the *estimation* error incurred by not selecting the best possible estimator in $\mathcal{F}$, and the second term is the *approximation* error caused by searching a good estimator in $\mathcal{F}$ and not somewhere else. They usually have opposite behavior:

- the estimation error is high when $\mathcal{F}$ is too complex for the data at hand, where complexity refers to the range of phenomena that can be accurately modeled by these functions. In other words, $\mathcal{F}$ contains many valid explanations that are equally good on the training set (they have low empirical risk $\hat{R}$), but are not accurate models of the underlying phenomenon p_{XT}, i.e. they do not generalize well (they have high risk R);
- the approximation error is high when $\mathcal{F}$ cannot adequately model the phenomenon that we are trying to describe, i.e. when it does not contain any good explanation for it.

This decomposition is often referred to as the *bias-variance decomposition*, where bias and variance refer to, respectively, approximation and estimation error.

A common way of building $\mathcal{F}$ is to fix the form of the models it contains, and allow their behavior to be controlled through a series of *parameters*. Then, equation 2.9 reduces to finding a good set of parameters; this process can also be governed by other parameters, called *hyper-parameters*. The fundamental problem is to find a class of functions that is powerful enough to model p_{XT}, but not too powerful so as to contain too many good explanations, because we would not be able to choose.

Equivalently, we want to find a good model, one that can explain the data we have, and generalize to new examples. This problem is known as *model selection*, and is important to distinguish between model selection for *identification*, and model selection for *estimation*. In the former case, the goal is to obtain a model and its parameters, to be used on new prediction problems, whereas the goal of the latter is to obtain a realistic estimate of the performance that can be obtained on new data. This problem can be approached in two ways: by directly estimating approximation and estimation errors using hold-out sets, or by penalizing complex models in order to favor simpler explanations, even though they might have slightly higher empirical risk. The next sections describe these two approaches.

2.3.2 Cross Validation

The idea behind hold-out methods is to partition the available data in two smaller sets D_T and D_V, usually of size 2/3 and 1/3 of the total, and use the *training* set D_T to choose $\hat{f}$ and D_V to estimate its risk. Since D is assumed to be a representative sample from p_{XT}, if the two partitions contain independent and identically distributed samples, the empirical risk on the *validation set* D_V can give us a glimpse on the generalization power of $\hat{f}$. This is because the validation set contains new samples that were not used to choose $\hat{f}$, thus the empirical risk on this set is an unbiased estimate of the true risk of the discovered model. This allows us not only to be confident about the performance of the estimator on unseen data, but also to compare different estimators. The problem of a single hold-out set is the variance of its estimate of the risk, which depends on the size of the validation set. This means that we are faced with a trade-off: use a lot of data to select a good estimator, but have high uncertainty in its estimated performance, or use less data and select a less powerful estimator, but have a more accurate picture of its performance.

The solution to this problem is to repeatedly perform this partitioning procedure so as to obtain many estimates of the risk, each on a different subset of validation samples, and average these results together. This can be done in a number of different ways:

- in random subsampling, the procedure above is simply repeated many times, by using two thirds random examples for training, and the remaining one third for validation;
- In bootstrapping (Efron and Tibshirani, 1993), the training set is created by taking $N = |D|$ examples *with replacement* from D, and using the remaining examples for validation. This means that the validation set contains on average approximately 36.8% of the samples in D, and the training set the remaining 63.2%, with many duplicates;
- In k-fold cross validation (Geisser, 1975), D is partitioned in k subsets, and each of them is used in turn as validation set, while the others are used for training. This produces k estimates of the true risk, coming from the k subsets.

Regardless of the method used, the final estimate of the performance is the average of the estimates obtained from the individual trials. Every method has different properties regarding both the bias and the variance of the estimates, and there is considerable controversy on which method should be used in which situation. For example, Kohavi (1995) recommends using 10-fold CV for comparing models, because, although its estimate of the performance is biased, it has lower variance compared to bootstrapping; however, Bengio and Grandvalet (2004) show that it is not possible to obtain an universal unbiased estimate of the variance of k-fold CV. Zhang and Yang (2015) further discusses this issue, and debunks some myths and commonly held misconceptions about CV, including the belief, consequent Kohavi (1995), that 10-fold cross validation is always the best choice. Generally, there is a tradeoff in choosing the value of k, as high values yield estimates with lower bias, but higher variance (Arlot et al., 2010), and are more computationally intensive.

Cross validation can be used to perform two distinct tasks: *estimation* and *identification*. The goal of the former is to accurately estimate the best performance attainable on a prediction problem, whereas the goal of the latter is to identify the parameters and hyper-parameters that give rise to the best possible estimators. Consequently, when the goal is estimation we want to eliminate bias, whereas in identification we need low variance. Estimation is best performed with nested CV(Stone, 1974; Varma and Simon, 2006), in which there is an inner CV loop done on every training fold of the

Algorithm 1 Nested cross validation with random hyper-parameter search.

Input:

D Dataset

K_O Number of outer folds

K_I Number of inner folds

R Number of random combinations to try

model Model to train

distrs Distribution of each hyper-parameter

Output: Summary statistics computed on each outer validation fold

for $k_o \leftarrow 1 \dots K_O$ **do** ▷ Outer K_O-fold cross validation
  Generate k_o-th outer fold (D_T^o, D_V^o) from the dataset D
  best_mse $\leftarrow$ inf
  best_params $\leftarrow \perp$
  for $r \leftarrow 1 \dots R$ **do** ▷ Find best hyper-parameters on D_T^o
    params $\leftarrow$ a random hyper-parameter combination from *distrs*
    params_mse $\leftarrow 0$
    for $k_i \leftarrow 1 \dots K_I$ **do** ▷ Evaluate *params* with K_I-fold CV
      Generate k_i-th inner fold (D_T^i, D_V^i) from D_T^o
      model $\leftarrow$ a new instance of the model with parameters *params*
      Train *model* on D_T^i
      mse $\leftarrow$ result of the evaluation of *model* on D_V^i
      params_mse $\leftarrow$ *params_mse* + *mse*
    end for
    if *params_mse* $<$ *best_mse* **then** ▷ MSE comparison on K_i-fold CV result
      best_mse $\leftarrow$ *params_mse*
      best_params $\leftarrow$ *params*
    end if
  end for
  model $\leftarrow$ a new instance of the model with parameters *best_params*
  Train *model* on D_T^o
  Evaluate *model* on D_V^o and compute evaluation metrics
end for
Compute summary of the metrics obtained in the outer loop

outer CV loop. This procedure is formalized in algorithm 1, where random hyper-parameter search (Bergstra and Bengio, 2012) is used in the inner CV loop to find good values for the hyper-parameters. Krstajic et al. (2014) advocates for repeating nested CV, since plain nested CV produces estimates with fairly high variance, but the computational cost of such procedure can be prohibitively high.

2.3.3 Parameter Estimation for Regression

In this section, we describe a general framework, rooted in Bayesian statistics, for estimating the parameters of a regression model, while controlling overfitting. An advantage of Bayesian methods is that they offer a sound theoretical foundation for model selection, without requiring repeated experiments to choose among candidate models, although this mathematical rigor is not free from practical difficulties (Wasserman, 2000; Chipman et al., 2001).

A common assumption in the regression setting is that the observations are corrupted by additive Gaussian white noise, i.e. $t_n = y_n + \epsilon_n$, where y_n is the "true" value, and $\epsilon_n \sim \mathcal{N}(0, \beta^{-1})$ is the noise. Let $f_n = f(\boldsymbol{x}_n; \boldsymbol{\theta})$ be the model's prediction for the sample $\boldsymbol{x}_n$, then we can write the probability of observing t_n, assuming that $f_n = y_n$, as:

$$p(t_n|\boldsymbol{x_n}, \boldsymbol{\Theta}) = \mathcal{N}(t_n|f(\boldsymbol{x_n}; \boldsymbol{\theta}), \beta^{-1}) \tag{2.11}$$

This probability is called *likelihood* of the observation t_n, under the model $f(\cdot;\cdot)$ with parameters $\boldsymbol{\theta}$ and hyper-parameters $\boldsymbol{\Theta}$ (only β in this specific example). Since the training data is assumed to be independent and identically distributed, the likelihood of the whole training set is

$$p(\boldsymbol{t}|\boldsymbol{X}, \boldsymbol{\Theta}) = \prod_{n=1}^{N} \mathcal{N}(t_n|f(\boldsymbol{x_n}; \boldsymbol{\theta}), \beta^{-1}) \tag{2.12}$$

and the predicted value t for a new sample $\boldsymbol{x}$ is distributed as

$$p(t|\boldsymbol{x}, \boldsymbol{X}, \boldsymbol{t}) = \int p(t|\boldsymbol{x}, \boldsymbol{\Theta}) \cdot p(\boldsymbol{\Theta}|\boldsymbol{t}, \boldsymbol{X}) \, \mathrm{d}\boldsymbol{\Theta} \tag{2.13}$$

In practice, it is not feasible to compute this integral, and its value is dominated by the values of $\boldsymbol{\Theta}$ close to the one that maximizes equation 2.12 anyways; this is the gist of *maximum likelihood estimation* (MLE). Note that, since the goal is to predict y_n, there is a high probability that the "true" parameters would *not* be the ones with maximum likelihood. When a model learns the noise in the training data, it cannot generalize well to new, unseen data, because the noise is random. This situation is known as *overfitting*, and tends to happen when the model is too complex relative to the amount of data available for training, and is related to high estimation error mentioned previously.

The risk of overfitting can be reduced with a number of *regularization* strategies. A widely used strategy consists in including a prior distribution on the parameters of the model, and maximizing their posterior distribution, computed using Bayes theorem:

$$\begin{aligned} p(\boldsymbol{\Theta}|\boldsymbol{t}, \boldsymbol{X}) &= \frac{p(\boldsymbol{X}, \boldsymbol{t}|\boldsymbol{\Theta}) \cdot p(\boldsymbol{\Theta})}{p(\boldsymbol{X}, \boldsymbol{t})} \\ &\propto p(\boldsymbol{t}|\boldsymbol{X}, \boldsymbol{\Theta}) \cdot p(\boldsymbol{X}|\boldsymbol{\Theta}) \cdot p(\boldsymbol{\Theta}) \\ &\propto p(\boldsymbol{t}|\boldsymbol{X}, \boldsymbol{\Theta}) \cdot p(\boldsymbol{\Theta}) \end{aligned} \tag{2.14}$$

where we removed $p(\boldsymbol{X}, \boldsymbol{t})$ and $p(\boldsymbol{X}|\boldsymbol{\Theta}) = p(\boldsymbol{X})$ because they are constant for a given dataset, and we are not interested in the exact probability, but where it reaches its maximum value. This parameter estimation procedure is called *maximum a posteriori* (MAP). Two commonly used prior distributions are the multivariate Laplace and the multivariate Normal, leading respectively to L1 and L2 regularization, when centered and symmetrical/spherical. Note that MAP reduces to MLE when we assume an uniform prior $p(\boldsymbol{\Theta})$.

The maximization of the posterior can be done conveniently by maximizing its logarithm; this gives expressions that are easier to handle analytically, and give less numerical problems when computed. The MAP problem can be formulated as follows:

$$\boldsymbol{\Theta}^* = \underset{\boldsymbol{\Theta}}{\operatorname{argmax}}\ \log p(\boldsymbol{t}|\boldsymbol{X}, \boldsymbol{\Theta}) + \log p(\boldsymbol{\Theta}) \tag{2.15}$$

If we assume a Gaussian likelihood like the one in equation 2.12, and a spherical Gaussian prior distribution $p(\boldsymbol{\theta}) = \mathcal{N}(\mathbf{0}, \lambda \boldsymbol{I})$, the MAP estimation of equation 2.14 becomes, after removing unnecessary constant terms,

$$\boldsymbol{\theta}^* = \underset{\boldsymbol{\theta}}{\operatorname{argmin}}\ L(\boldsymbol{\theta}) = \underset{\boldsymbol{\theta}}{\operatorname{argmin}} \sum_{n=1}^{N} \left(f(\boldsymbol{x}_n, \theta) - t_n\right)^2 + \lambda \boldsymbol{\theta}^\intercal \boldsymbol{\theta} \tag{2.16}$$

where L is the *loss function*, in this case the sum-of-squares error. It is customary to use the mean squared error instead of the sum of squares because it results in smaller numbers and is readily interpreted; being only a constant factor away from 2.16, it does not transcend the essence of MAP estimation. The parameter β can be estimated as well in a similar way, if necessary, and a fully Bayesian treatment allows to estimate λ, too. Depending on the model $f(\boldsymbol{x}_n, \boldsymbol{\theta})$, 2.16 can be solved analytically to yield a closed-form solution for $\boldsymbol{\theta}$.

When this is not possible, iterative optimization methods are employed. A widely used approach is called *gradient descent*: the gradient of a function computed at a given location "points" to the steepest direction where the function's value increases. By repeatedly following the gradient, it is possible to reach a local maximum, and, in the opposite direction, a local minimum:

$$\boldsymbol{\theta}_{n+1} := \boldsymbol{\theta}_n - \eta \cdot \nabla f(\boldsymbol{x}_n) \tag{2.17}$$

The series $\boldsymbol{\theta}_1, \ldots, \boldsymbol{\theta}_n$ is guaranteed to converge to the local optimum at a rate of $O(n^{-1})$ if certain conditions are met (Gitman et al., 2018). Nocedal and Wright (1999) describes more advanced optimization techniques that use the gradient and, possibly, the Hessian, such as Newton's. The gradient of $L(\boldsymbol{\theta})$ is

$$\nabla_{\boldsymbol{\theta}} \log p(\boldsymbol{\theta}|\boldsymbol{t}, \boldsymbol{X}) = 2 \sum_{n=1}^{N} \big(f(\boldsymbol{x}_n; \theta) - \boldsymbol{t}_n\big) \cdot \nabla_{\boldsymbol{\theta}} f(\boldsymbol{x}_n; \theta) + 2\lambda \boldsymbol{\theta} \tag{2.18}$$

and its Hessian is

$$\nabla^2_{\boldsymbol{\theta}} \log p(\boldsymbol{\theta}|\boldsymbol{t}, \boldsymbol{X}) = 2 \sum_{n=1}^{N} \Big[\big(\nabla_{\boldsymbol{\theta}} f(\boldsymbol{x}_n; \boldsymbol{\theta})\big) \big(\nabla_{\boldsymbol{\theta}} f(\boldsymbol{x}_n; \boldsymbol{\theta})\big)^\intercal + \big(f(\boldsymbol{x}_n; \boldsymbol{\theta}) - t_n\big) \nabla^2_{\boldsymbol{\theta}} f(\boldsymbol{x}_n; \theta) \Big] + 2\lambda \boldsymbol{I} \tag{2.19}$$

In some cases, the gradient and the Hessian can be approximated using a random subset of the training set, and, in extreme cases, a single sample. These variants are called *minibatch* gradient descent and *stochastic* gradient descent respectively. They both compute a noisy approximation to the true gradient, which can actually improve convergence and generalization of high-dimensional, non-convex loss functions such as those found in deep learning (Neelakantan et al., 2015; Smith and Le, 2017). Vanilla gradient descent can be greatly improved with a number of techniques, such as momentum (Rumelhart et al., 1986), adaptive learning rate (Duchi et al., 2011; Zeiler, 2012; Kingma and Ba, 2014), and so on, see Ruder (2016) for an overview.

2.3.4 Hyper-parameter Optimization

The previous section discussed some methods to find the best parameters for a model. The behavior of most model, though, is governed by other *hyper-parameters*, such as β in equation 2.11. In practice, finding good values for the hyper-parameters is often as important as fitting the model, since these hyper-parameters control the learning process itself.

A simple way to approach this problem is to choose some possible values for each hyper-parameter, try all the possible combinations, and pick the one that works best. Alternatively, one can sample each hyper-parameter from a probability distribution, and try many random combinations of values; Bergstra and Bengio (2012) showed that this latter method is surprisingly effective at this task, and

scales much better than the former. Unfortunately, this procedure can overfit, too, so it has to be paired with some resampling technique such as cross validation. The procedure is simply to test every hyper-parameter combination on every fold, so that 50 combinations tested with 10-fold cross validation require to fit the model 500 times.

More sophisticated techniques treat hyper-parameter optimization as a regression problem, and use supervised machine learning to predict the performance of a given hyper-parameter combination, and to guide the search accordingly (Bergstra et al., 2011). Clearly, one should use models that are not sensitive to their own hyper-parameter setting, or the problem is just moved! For this reason, popular algorithms that are used for this are evolutionary algorithms and Bayesian non-parametric models.

2.3.5 Ridge Regression

A linear regression model has the form $f(\boldsymbol{x}_n; \boldsymbol{\theta}) = \boldsymbol{\theta}^\intercal \boldsymbol{x}_n$; ridge regression is simply L2-regularized linear regression. This model is simple enough that the solution for equation 2.16 can be found analytically in closed form:

$$\boldsymbol{\theta}^* = (\boldsymbol{X}^\intercal \boldsymbol{X} + \lambda \boldsymbol{I})^{-1} \boldsymbol{X}^\intercal \boldsymbol{t} \tag{2.20}$$

Notice that the term *linear* in linear regression refers to linearity with respect to the parameters, not the features. In fact, new features can be added through the use of a possibly non-linear feature mapping $\phi : \mathbb{R}^D \to \mathbb{R}^M$, such that $\boldsymbol{x}'_n = \phi(\boldsymbol{x}_n)^\intercal = [\phi_1(\boldsymbol{x}_n), \ldots, \phi_M(\boldsymbol{x}_n)]^\intercal$. Then, the model parameters can be fitted to the augmented training set $\boldsymbol{\Phi} = \boldsymbol{X}'$, where $\boldsymbol{\Phi}$ is called the *design matrix*. A typical feature added is a bias $\phi_1(\boldsymbol{x}) = 1$.

2.3.6 k-Nearest Neighbors

The k-nearest neighbor model Stone (1977) predicts a value for a sample $\boldsymbol{x}$ by averaging the target values of the K training samples that are closest to $\boldsymbol{x}$, possibly weighted by distance or some other metric. Let $\boldsymbol{t}_{(1)}, \ldots, \boldsymbol{t}_{(K)}$ be the target values of the K training samples closest to $\boldsymbol{x}$ according to a distance metric $d : \mathbb{R}^D \to \mathbb{R}$, then we have

$$f(\boldsymbol{x}) = \sum_{k=1}^{K} w_{(k)} \cdot \boldsymbol{t}_{(k)} \tag{2.21}$$

Note that the k-nearest neighbors algorithm does not have parameters, only hyper-parameters: K itself, the distance function, and the weighting scheme. Typical distance functions are the Manhattan distance $d(\boldsymbol{u}, \boldsymbol{v}) = \sum_{d=1}^{D} |u_d - v_d|$ and the squared Euclidean distance $d(\boldsymbol{u}, \boldsymbol{v}) = \sum_{d=1}^{D} (u_d - v_d)^2$, and typical weights are uniform $w_{(k)} = 1/K$ and based on distance $w_{(k)} = d(\boldsymbol{x}, \boldsymbol{x}_{(k)}) / \sum_{k'=1}^{K} d(\boldsymbol{x}, \boldsymbol{x}_{(k')})$.

2.3.7 Gradient Boosted Trees

Gradient boosting (Breiman, 1997) is a general method of combining several weak predictors in order to obtain a stronger one. The final prediction for a sample is obtained as a weighted average of the predictions of the individual models:

$$f_M(\boldsymbol{x}) = \sum_{m=1}^{M} \beta_m h(\boldsymbol{x}; \boldsymbol{\theta}_m) \tag{2.22}$$

Each model parameters $\boldsymbol{\theta}_m$ and the weight β_m are found by improving the predictions of f_{m-1}:

$$(\beta_m, \boldsymbol{\theta}_m) = \underset{\boldsymbol{\theta}, \beta}{\operatorname{argmin}} \sum_{n=1}^{N} L\left(t_n, f_{m-1}(\boldsymbol{x}_n) + \beta h(\boldsymbol{x}_n, \boldsymbol{\theta})\right) \tag{2.23}$$

where L is the loss function, such as the squared error introduced previously. This optimization problem can be challenging, depending on the specific form of h and L. Actually, equation 2.23 is

equivalent to gradient descent in function space[5] (Mason et al., 1999; Friedman, 2001). This gradient can be computed explicitly on the N training points, but we do not know how to compute it on the other points that are not in the training set[6]. The fundamental idea is to find a model that can predict this gradient, by training it on the dataset. This model is exactly h with parameters $\boldsymbol{\theta}_m$:

$$\boldsymbol{\theta}_m = \underset{\boldsymbol{\theta}}{\operatorname{argmin}} \sum_{n=1}^{N} \left[-g_m(\boldsymbol{x}_n) - h(\boldsymbol{x}_n, \boldsymbol{\theta})\right]^2 \tag{2.24}$$

with g_m being the gradient of L with respect to f_{m-1}:

$$g_m(\boldsymbol{x}_n) = \frac{\partial L(t_n, f_{m-1}(\boldsymbol{x}_n))}{\partial f_{m-1}(\boldsymbol{x}_n)} \tag{2.25}$$

Then, the step size β_m is obtained by line search, i.e. minimizing the loss function in the direction of the gradient:

$$\beta_m = \underset{\beta}{\operatorname{argmin}} \sum_{n=1}^{N} L(t_n, \beta h(\boldsymbol{x}_n, \theta_m)) \tag{2.26}$$

This is now a tractable problem, since 2.25 can be computed easily, and both 2.24 and 2.26 are simple regression problems.

If $L(y_n, t_n) = (y_n - t_n)^2/2$, we have that $-g_m(\boldsymbol{x}_n) = f_{m-1}(\boldsymbol{x}_n) - t_n$, or, in other words, we are creating a new model by counteracting the errors of the current model! Other popular loss functions are the least absolute deviation, $L(y_n, t_n) = |y_n - t_n|$, in which case $-g_m = sign(t_n - f_{m-1}(\boldsymbol{x}_n))$, and the Huber loss:

$$L_\delta(t_n, y_n) = \begin{cases} \frac{1}{2}(y_n - t_n)^2 & |y_n - t_n| \leq \delta \\ \delta(|y_n - t_n| - \frac{1}{2}\delta) & |y_n - t_n| > \delta \end{cases} \tag{2.27}$$

which is robust to outliers as least absolute deviation, but does not penalize small errors too much, like least squares. In this case,

$$-g_m(\boldsymbol{x}_n) = \begin{cases} t_n - y_n & |t_n - y_n| \leq \delta \\ \delta \cdot sign(t_n - y_n) & |t_n - y_n| > \delta \end{cases} \tag{2.28}$$

In summary, the gradient boosting algorithm starts with a simple prediction f_0, such as the average of the true values, builds a model of the form in equation 2.22 by iteratively applying equations 2.24 and 2.26.

A popular choice is to use decision trees for h. Often, an additional *learning rate* parameter is introduced to shrink the β coefficients. Other hyper-parameters for this algorithm are M, the the number of boosting steps to perform, the number of examples to use to compute equations 2.24 and 2.26, the number of features used to build the trees, their maximum depth, and so on.

A popular extension to vanilla gradient boosting is *extreme gradient boosting* (Chen and Guestrin, 2016), which essentially adds both L1 and L2 regularization to the loss function, helping to reduce overfitting, and provides better scalability.

2.3.8 Gaussian Processes

A Gaussian Process (GP) is a collection of an infinite number of random variables, such that the joint distribution of any finite subset of them is Gaussian. Intuitively, this allows us to define a distribution over functions, since there is a random variable for every point of the input space. We can sample a function out of a GP by obtaining the random variables at the points of interest; their distribution is a (multivariate) Gaussian, with a certain mean and covariance, and a sample from this distribution

[5] this means that you actually modify the function itself, not its parameters, as we did in section 2.3.3. This is true provided that the parametric function h is able to represent the gradient

[6] this is the essence of gradient descent in function space, since a function is defined for every point (at least the functions we consider in Machine Learning are)

contains the values of the sampled function at the points we queried. This process is expressed as follows:

$$p(\boldsymbol{y}|\boldsymbol{X}) = \int p(\boldsymbol{y}|\boldsymbol{f})p(\boldsymbol{f}|\boldsymbol{X})\mathrm{d}f \tag{2.29}$$

Where $\boldsymbol{X}$ are the query points, $\boldsymbol{f}$ a function sampled from the GP, and $\boldsymbol{y}$ the value of the function at the query points. By definition of GP, all the distributions in equation 2.29 are Gaussian, since the Gaussian distribution is self-conjugate[7]. Simplistically, a GP is just a way of computing the mean and covariance of $\boldsymbol{f}$ given $\boldsymbol{X}$, i.e. $p(\boldsymbol{f}|\boldsymbol{X})$; $\boldsymbol{f}$ is then a multivariate Gaussian, which we can manipulate as usual.

Consider the setting of ridge regression described in section 2.3.5, where $\boldsymbol{y} = f(\boldsymbol{X};\boldsymbol{\theta}) = \boldsymbol{\Phi\theta}$, with $\boldsymbol{\Phi}$ being the design matrix. Recall that in section 2.3.3 we introduced the Maximum A Posteriori estimation method by using a prior distribution on the model parameters $\boldsymbol{\theta}$; this makes $\boldsymbol{y}$ a random variable itself. If we use an isotropic Gaussian for $\boldsymbol{\theta}$, we have that the distribution of $\boldsymbol{y}$ is again Gaussian with mean

$$\mathbb{E}[\boldsymbol{y}] = \mathbb{E}[\boldsymbol{\Phi\theta}] = \boldsymbol{\Phi}\mathbb{E}[\boldsymbol{\theta}] = \boldsymbol{0} \tag{2.30}$$

and covariance

$$\mathrm{cov}[\boldsymbol{y}] = \mathbb{E}[\boldsymbol{y}\boldsymbol{y}^\intercal] = \boldsymbol{\Phi}^\intercal\mathbb{E}[\boldsymbol{\theta}^\intercal\boldsymbol{\theta}]\boldsymbol{\Phi} = \alpha^{-1}\boldsymbol{\Phi}^\intercal\boldsymbol{\Phi} = \boldsymbol{K} \tag{2.31}$$

Where α is the precision of the prior on $\boldsymbol{\theta}$. This is a simple GP with zero mean, and whose covariance between any two points can be computed as the dot product of their features.

In general, we can use any function k to compute the covariance between two input points, as long as the resulting matrix $\boldsymbol{K}$, with $K_{ij} = k(\boldsymbol{x}_i, \boldsymbol{x}_j)$ is symmetric and positive semi-definite[8]. We call any such function *kernel*, a function $k : \mathbb{R}^d \times \mathbb{R}^d \to \mathbb{R}$ that, intuitively, expresses the similarity of any two input vectors. Mercer's theorem guarantees that for every symmetric positive-definite kernel there exists a basis function ϕ, such that the kernel corresponds to the dot product in the space generated by ϕ. In practice, this means that we do not have to explicitly define ϕ, and that we can even use basis functions that create an infinite dimensional space. Commonly used kernels are the radial basis function kernel:

$$k(\boldsymbol{x}, \boldsymbol{x}') = \exp\left(-\frac{\|\boldsymbol{x} - \boldsymbol{x}'\|^2}{2\sigma^2}\right) \tag{2.32}$$

where $\|\boldsymbol{x}-\boldsymbol{x}'\|^2 = \sum_d(\boldsymbol{x}_d-\boldsymbol{x}'_d)^2$ is the squared Euclidean distance, and which incidentally corresponds to an infinite dimentional basis function, and the linear kernel:

$$k(\boldsymbol{x}, \boldsymbol{x}') = \sigma^2\boldsymbol{x}^\intercal\boldsymbol{x}' \tag{2.33}$$

Furthermore, certain operations on a kernel generate another valid kernel, and kernels can be combined in certain ways to produce new kernels. For example, the product of a kernel with a positive constant is a valid kernel, the sum and product of two kernels are valid kernels, etc. Generally, a kernel can have parameters, which we denote, as usual, as $\boldsymbol{\theta}$.

In order to use a Gaussian Process for inference (in the regression setting), we need to compute the predictive distribution $p(\boldsymbol{y}|\boldsymbol{X}, \boldsymbol{t}, \boldsymbol{X}^*, \boldsymbol{\theta})$, where $\boldsymbol{X}^*$ is the matrix containing the vectors for which we want to predict the value $\boldsymbol{y}$. We know that the predictive distribution is Gaussian, therefore its mean $\boldsymbol{\mu_y}$ and covariance $\boldsymbol{\Sigma_y}$ can be obtained relatively easily, resulting in:

$$\boldsymbol{\mu_y} = K_{\boldsymbol{\theta}}(\boldsymbol{X}^*, \boldsymbol{X})K_{\boldsymbol{\theta}}(\boldsymbol{X}, \boldsymbol{X})^{-1}\boldsymbol{t} \tag{2.34}$$

[7] referring to equation 2.14, the prior is said to be conjugate to the likelihood if the posterior has the same algebraic form of the prior; the Gaussian distribution is self-conjugate, because a Gaussian prior with a Gaussian likelihood results in a Gaussian posterior. Conjugate distributions are mostly used for algebraic convenience.

[8] this means that $\boldsymbol{K}^\intercal = \boldsymbol{K}$ and $\boldsymbol{v}^\intercal\boldsymbol{K}\boldsymbol{v} \geq 0$ for any $\boldsymbol{X}$ (used to compute $\boldsymbol{K}$) and $\boldsymbol{v}$. This condition is necessary for $\boldsymbol{K}$ to be a valid covariance matrix

$$\boldsymbol{\Sigma_y} = K_{\boldsymbol{\theta}}(\boldsymbol{X}^*, \boldsymbol{X}^*) - K_{\boldsymbol{\theta}}(\boldsymbol{X}^*, \boldsymbol{X})^\intercal K_{\boldsymbol{\theta}}(\boldsymbol{X}, \boldsymbol{X})^{-1} K_{\boldsymbol{\theta}}(\boldsymbol{X}, \boldsymbol{X}^*) \tag{2.35}$$

where $K_{\boldsymbol{\theta}}(\boldsymbol{A}, \boldsymbol{B})$ is a matrix whose (i, j) cell equals to $k(A_i, B_j; \boldsymbol{\theta})$. From these formulas, note that in order to use GPs, one needs to remember the kernel matrix, of size N^2 and invert it, an operation which has a computational complexity of $O(N^3)$; this makes GPs, and any method based on kernels, hard to scale up.

Up until now, we assumed the kernel was fixed, and treated $\boldsymbol{\theta}$ as hyper-parameters. We could use one of the cross validation methods described in section 2.3.2 to find their values, but the Bayesian framework that we used to derive GP allows us to find $\boldsymbol{\theta}$, too, by maximizing $\log p(\boldsymbol{t}|\boldsymbol{X}, \boldsymbol{\theta})$[9] with respect to $\boldsymbol{\theta}$. This function often has many local maxima, and is usually optimized by means of gradient descent.

Finally, the derivative of a GP is itself a GP, whose mean is obtained simply by deriving equation 2.34 (McHutchon, 2013):

$$\frac{\partial \boldsymbol{y}}{\partial \boldsymbol{X}_*} = \frac{\partial K_{\boldsymbol{\theta}}(\boldsymbol{X}_*, \boldsymbol{X})}{\partial \boldsymbol{X}_*} K_{\boldsymbol{\theta}}(\boldsymbol{X}, \boldsymbol{X})^{-1} \boldsymbol{t} \tag{2.36}$$

We invite the reader to refer to (Bishop, 2006, Chapter 6) for an in-depth treatment of kernel methods, including Gaussian Processes.

2.3.9 Robust Effect Size

The effect size is a statistic used to compare the means of two populations, usually called *control* and *treatment*, that is computed as the standardized difference of the means:

$$\delta = \frac{\mu_t - \mu_c}{\sigma} \tag{2.37}$$

where μ_t and μ_c are the means of the treatment and control groups respectively, and σ is a normalizer that can be computed in essentially three ways: when the standard deviations of the control and treatment populations can be assumed to be equal, then σ should be the pooled standard deviation[10], otherwise it should be the standard deviation of either the control or treatment group. Generally, the effect size is computed to be positive when the treatment group represent an improvement over the control group.

Recently, there is a departure from the standard procedure of reporting p-values, test statistics and critical values resulting from hypothesis testing towards reporting effect sizes and their confidence intervals, in addition or in alternative to the usual statistics, see for example Wilkinson (1999) and *Standards for Reporting on Empirical Social Science Research in AERA Publications* (2006). However, effect sizes can be as misleading as significance testing when the assumptions underlying its expression are violated; commonly used effect size measures, such as Cohen's d [citation], assume both populations are normally distributed and have the same standard deviation[11]. Alternatives that do not assume equal standard deviation exist (Shieh, 2013; Algina et al., 2006; Bonett, 2008), but they all require normality, and generally use approximate distributions of the standard error and/or the CI (or both).

In order not to assume normality, we compute the CI for the effect size using the bootstrap percentile method. This method estimates the population value of any statistic θ and its CI using a single sample of N observations, by extracting M new samples (with replacement) of size N from it and computing θ on each resample. The population statistic is estimated by the average of the M sample statistics, and the extremes of the (central) $100(1-\alpha)\%$ CI are respectively the $100(\alpha/2)$ and $100(1-\alpha/2)$ percentiles of those values. Usually M is in the order of 10^2 or 10^3, enough to guarantee a certain degree of stability in these estimates.

The sample effect size of equation 2.37 is not robust to outliers when it is computed with the sample means and standard deviations. Since we do not assume normality outliers are expected, therefore we

[9] obtained as in equation 2.29, usually called log-marginal-likelihood, since we marginalized $\boldsymbol{f}$

[10] the average standard deviation of the control and treatment groups, weighted by the degrees of freedom (one less than the number of observations) of each group

[11] a property called *homoscedasticity* (contrary: heteroscedasticity)

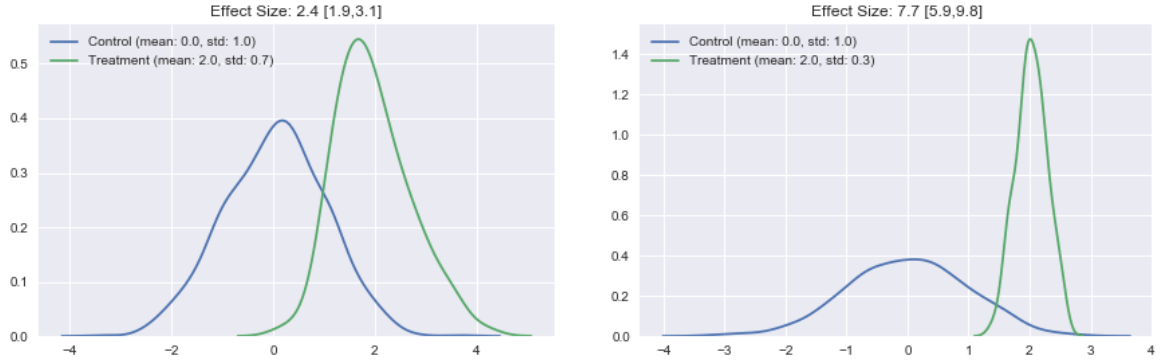


Figure 2.4: Graphical explanation of why the effect size is normalized with the standard deviation of the treatment, when the control has a mean of 0 and standard deviation of 1 and the treatment has a mean of 2; inarguably, the treatment in the right figure the best of the two. Of course, one could perform the same experiment but keeping the treatment fixed and varying the standard deviation of the control: in that case, the effect size would not change. Refer to the main text for a discussion.

use measures of location that are not heavily affected by them: the sample trimmed mean and the sample Winsorized standard error. The $100\gamma\%$ trimmed mean $\overline{X}_\gamma$ of a sample $X_1, \ldots, X_N$ is simply the mean of that sample after discarding the $100\gamma\%$ largest and the $100\gamma\%$ smallest observations[12]; a common value for γ is 0.2. The standard error of the sample trimmed mean is obtained from the Winsorized sample standard deviation; let $X_{(\gamma)}$ and $X_{(1-\gamma)}$ be the 100γ and $100(1-\gamma)$ percentiles, and define the Winsorized observations as follows:

$$W_n = \begin{cases} X_{(\gamma)} & \text{if } X_n \leq X_{(\gamma)} \\ X_{(1-\gamma)} & \text{if } X_n \geq X_{(1-\gamma)} \\ X_n & \text{otherwise} \end{cases} \tag{2.38}$$

then we have the Winsorized sample mean

$$\overline{X}_W = \frac{1}{N}\sum_{n=1}^{N} W_n \tag{2.39}$$

the Winsorized sample variance

$$\hat{\sigma}_W^2 = \frac{1}{N-1}\sum_{n=1}^{N}(W_n - \overline{X}_W)^2 \tag{2.40}$$

and the estimated variance of the trimmed mean

$$\mathrm{Var}[\overline{X}_\gamma] = \frac{\hat{\sigma}_W^2}{N(1-2\gamma)^2} \tag{2.41}$$

The standard error of $\overline{X}_\gamma$ is the usual $(\mathrm{Var}[\overline{X}_\gamma]/N)^{1/2}$. Plugging everything into equation 2.37 and normalizing with the treatment Winsorized sample standard error gives the sample effect size

$$\hat{\delta} = (\overline{T}_\gamma - \overline{C}_\gamma) \cdot \sqrt{\frac{N}{\mathrm{Var}[\overline{T}_\gamma]}} = (1-2\gamma)\frac{\overline{T}_\gamma - \overline{C}_\gamma}{\hat{\sigma}_W} \tag{2.42}$$

[12] this is common practice, for example, when scoring in sports such as the modern Olympiads, where the smallest and largest scores are discarded.

where $T_1, \ldots, T_N$ and $C_1, \ldots, C_M$ are the *independent* samples of treatment and control groups respectively. See Wilcox and Keselman (2003) and Wilcox (2010)for an introduction to robust statistical methods, including the ones just described.

Figure 2.4 shows the effect size computed according to 2.42, and tries to provide an intuitive explanation as to why we normalize with the treatment standard deviation. Note that we cannot use an average (weighted or unweighted) as normalizer since it corresponds to assuming homogeneity of *population* variances, an assumption that we wanted to avoid in the first place. Since the goal is to assess the improvement caused by the treatment, it would not make sense to normalize with the standard deviation of the control group, because it would not allow us to discriminate between treatments with the same mean but different standard deviations: given two treatments with the same mean, we prefer the one with the smaller standard deviation. Of course, this reasoning can be applied in the opposite direction: normalizing by the control standard deviation would give a larger effect size when the control group has a smaller standard deviation, which is certainly a desirable property, since a control group with a smaller standard deviation is more clearly separated from the treatment. However, we consider the control group to be fixed, and we will never need to compare two control groups against the same treatment.

Chapter 3

Method

This chapter describes how the data is collected (section 3.1) and used to re-create the flux-profile relationships (section 3.2), as well as their prediction based on the Monin-Obukhov similarity theory (section 3.3). Then, we describe the main contribution of this thesis, namely how their prediction can be improved using machine learning techniques (section 3.4). Finally, we describe how the evaluation is performed (section 3.5), and explicitly state the necessary criteria for a successful answer to the research questions (section 3.6).

3.1 Data Collection and Preparation

The Cabauw Experimental Site for Atmospheric Research[1] (Cesar) is a consortium formed by eight Dutch institutes and universities, which collaborate to operate and maintain an observatory for micrometeorological conditions near the village of Cabauw, the Netherlands. The data collected characterizes the state of the atmosphere and the soil, and their interaction via radiation and surface fluxes.

The observatory is surrounded by fields and no urban agglomerations is present within 15 kilometers; the land is flat with changes of altitude within a few meters over 20 kilometers. The main mast is 213 meters high and offers measurement levels every 20 meters; at each level there are three booms of length 10 meters that allow observations as unobstructed by the main mast as possible, although some disturbance remains. There are three additional smaller masts of height 10 and 20 meters located close to the main mast, in order to obtain undisturbed measurements at the lower levels, and facilities to perform soil and surface observations.

The main focus of this project is on the wind profile and the turbulent fluxes of sensible and latent heat. Additional variables, such as temperature and humidity, are needed to compute quantities of interest, most importantly the Obukhov length, and as possible predictors for the universal functions. There is one measurement for each variable every ten minutes, and missing measurements are gap-filled with a number of techniques. The data collected is always visually validated by an operator, which marks suspect or invalid sections of data.

Following, we describe the instruments used to collect the measurements (section 3.1.1), the techniques used to impute missing or invalid measurements (section 3.1.2), and the criteria applied to filter low quality data (section 3.1.3).

3.1.1 Measurement

The Cesar observatory provides full information regarding data collection[4], see in particular Bosveld (2018), what follows is a brief summary of the sections in that document that are relevant for this work.

[1] http://www.cesar-database.nl/

[4] http://projects.knmi.nl/cabauw/insitu/observations/documentation

(a) The main mast

(b) Sonic anemometer and optical open-path sensor at the 180 m level

Figure 3.1: The Cabauw main mast (3.1a, from the Cesar Observatory website[2]) and some measurement instruments it supports (3.1b, from the KNMI website[3])

.

Wind Speed Measurement

Wind speed and direction are measured at heights of 200, 140, 80, 40, 20 and 10 meters, in either two or all three booms available. The wind vane that measures direction has a resolution of 1.5°, and the cup anemometer that measures wind speed has an accuracy of the largest between 1% and $0.1\,\mathrm{m\,s^{-1}}$. Monna (1978) studied the threshold sensitivity of both instruments, and fount it lower than $0.5\,\mathrm{m\,s^{-1}}$, even though the measurements are inaccurate up to $3\,\mathrm{m\,s^{-1}}$.

For every ten minutes interval, the measurement comes from the instrument that is best exposed to the wind, and less affected by the obstruction caused by the mast. Corrections are then applied to the raw measurements to further attenuate the disturbance by the tower, following Wessels (1983).

Eddy Correlation

The eddy correlation technique is used to compute the turbulent surface fluxes of sensible and latent heat H and λE, as well as the momentum flux, starting from fluctuations in wind, temperature, humidity, and CO_2 content.

These measurements are obtained with a sonic anemometer and and an optical open-path sensor. Sonic anemometers measure the wind speed by leveraging the fact that the speed of sound in free air is affected by the speed of the air itself; since the speed of sound is known, the wind speed can be easily recovered from the time a sound impulse takes to travel a short distance. By measuring the wind velocity along three orthogonal paths, the full wind vector can be recovered. This measure can then be used to compute the temperature, by leveraging the fact that the speed of sound is affected by the temperature of the medium it travels in. Optical open-path sensors quantify the amount of water vapor and carbon dioxide in the air by emitting a ray of infrared light and measuring its intensity 10 to 20 centimeters further. H_2O and CO_2 molecules in the air absorb electromagnetic radiation at known frequencies, thus the concentration of water vapor and carbon dioxide can be inferred by measuring the attenuation at these wavelengths. Some of these instruments are shown in figure **??**.

All these instruments can take up to 100 measurements per second, necessary in order to compute the turbulence. The eddy correlation technique measures fluxes by computing their sample covariance with the vertical wind speed. Let w_t be the vertical wind speed at time t, then the turbulent vertical flux for the quantity a is computed as follows:

[4] http://www.cesar-observatory.nl/
[4] https://www.knmi.nl/research

$$F_a = \frac{1}{T_2 - T_1} \sum_{t=T_1}^{T_2} (w_t - \overline{w})(a_t - \overline{a})$$

where $\overline{w}$ and $\overline{a}$ are the averages of w_t and a_t for $T_1 \leq t \leq T_2$. The fluxes in the Cesar database are computed every ten minutes, with 10 measurements per second.

The eddy correlation technique is far from perfect, see e.g. (Lee et al., 2004; Loescher et al., 2006). Other methods can be used to compute H and u_*; for example, Holtslag (1987) uses an iterative method to estimate these quantities from L, starting from an initial guess and refining the result, alternating improvements of u_* and H with improvements of L.

Air Temperature and Dew Point Measurement

Both air temperature and dew point[5] are measured at heights of 200, 140, 80, 40, 20, 10 and 1.5 meters, and the dew point is computed from the air temperature and the humidity obtained with the open-path sensor described previously. The instruments at the highest four levels are located on the main tower, whereas 20 and 10 meters are measured in the smaller mast, and 1.5 meters is measured by the weather station. The resolution of the instruments is of 0.1 °C, whereas the accuracy is 0.1 °C for temperature and 3.5% of relative humidity for the dew point (1.5% after 2014). Low wind speed and high irradiation can result result in a few 0.1 °C overestimate of the temperature (Meijer, 2000), whereas the humidity can be over estimated when drying after dew, fog, or rain (Bosveld, 2018).

Net Radiation Measurement

The Cabauw observatory measures both incoming and outgoing long- and short-wave radiation, and the net radiation is simply the combination of these four components. The basic operating principle of an instrument that measures radiation is to have a device coated with paint that is highly absorbing towards a certain range of frequencies; the temperature difference between the surface of this sensor and the body of the whole instrument generates an electrical potential that is proportional to the radiation absorbed by the sensor, after appropriate corrections. The instruments that measure short-wave radiation are called *pyranometers*, while long-wave radiation is measured by *pyrgeometer*. The instrument used to measure the net radiation has a pair or pyrgeometers, one facing up and one facing down, to measure the net long-wave radiation, and a pair of pyranometers arranged similarly to measure net short-wave radiation (see figure 3.2a).

This instrument is positioned at 1.5 m height, and is ventilated and heated to prevent formation of dew, that can make the measurements invalid.

Surface Soil Heat Flux

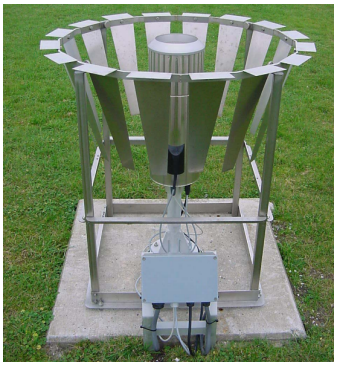
The surface soil heat flux, measured in $\mathrm{W\,m^{-2}}$, is extrapolated from the average fluxes measured at depths of 5 cm and 10 cm; under the assumption of homogeneous soil, the amplitude of the flux in the frequency domain decays exponentially with depth. In order to increase the temporal resolution, the amplitude of the surface soil heat flux is then related to the amplitude of the soil temperature gradient, obtained as the difference between the soil temperature at 0 cm and 2 cm. An inverse Fourier transform converts this surface soil heat flux into the time domain (Holtslag and Bruin, 1988, Appendix A). The heat flux is computed by dividing the temperature difference at the top and bottom sides of a plate-shaped sensor[7] with the horizontal area of the sensor. A *thermocouple* is a device that generates a voltage proportional to this temperature difference, and self-calibrating sensor converts this voltage into a heat flux measurement in $\mathrm{W\,m^{-2}}$ (Gainsford, 2006).

[5] the temperature the air needs to be to reach a relative humidity of 100%

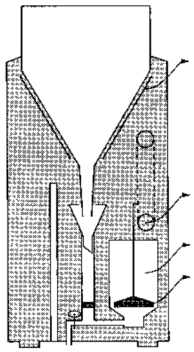
[6] `https://en.wikipedia.org/wiki/File:Hukseflux_netto_radiometer_nr01_photo.jpg`

[7] because of conduction, the temperature of the sensor would tend to homogenize; the higher the difference in temperature, the more energy is hitting the warmer side.

(a) Net Radiometer



(b) Rain Gauge with Wind Shield



(c) Rain Gauge Schematic

Figure 3.2: Some instruments used at the Cabauw observatory. Credits: picture a - Wikimedia Commons[6], picture b - Wauben (2004), picture c - Babington (2000).

Rain Amount

The KNMI electronic rain gauge (Babington, 2000; Wauben, 2004) records the average precipitation intensity of the last 12 seconds in $\mathrm{mm\,s^{-1}}$ every 12 seconds, and the rain amount, in mm, is derived for every 10 minutes interval. The rain gauge is composed by a heated container in which precipitation accumulates and which is emptied every 12 seconds; the flow of outgoing liquid is measured by a floater connected to a potentiometer, and its value divided by 12 gives the precipitation intensity. The accuracy of the precipitation meter is 0.2 mm, and the precision is 0.1 mm. Figure 3.2b shows the rain gauge with the wind shield (Wauben, 2004), and figure 3.2c shows a diagram of the instrument (Babington, 2000).

3.1.2 Gap Filling

With gap-filling, missing measurements are replaced by synthetic values (Bosveld, 2014). The gap-filling method depends on the missing parameter and the duration of the period where data is not available. There are two classes of parameters: forcing parameters, which include wind, temperature, specific humidity, incoming radiation and rain, and validation parameters, which include the surface fluxes, outgoing radiation, and friction velocity.

For less than two hours of missing measurements, both forcing and evaluation parameters are gap-filled by interpolation of nearby values. For longer periods, forcing parameters are derived by transforming measures obtained from the nearby site of De Bilt, which are themselves gap-filled, if necessary. Evaluation parameters are computed with a vegetation model that uses the forcing parameters as input. The gap-filling procedure is performed by the Cesar consortium, and we do not impute the remaining missing values, since we believe there is a good reason why these values were not gap-filled in the first place.

3.1.3 Data Filtering

Following other works in this field, such as Korrell et al. (1981) and Högström (1988), we exclude all the records where any of the following conditions applies:

- The sensible heat flux H is smaller than $10\,\mathrm{W\,m^{-2}}$;
- The friction velocity u_* is smaller than $0.1\,\mathrm{m\,s^{-1}}$;
- The wind speed $\overline{u}$ is lower than $1\,\mathrm{m\,s^{-1}}$.

The reason for this is that the measures are too inaccurate to be of use.

3.2 Momentum Flux-Profile Relationship

Since the turbulent fluxes and the friction velocity are measured at the surface level, we can compute the momentum flux-profile relationships only at 10, 20 and 40 meters, because these quantities can be assumed constant only within the surface layer. It is very hard to know the exact height of the surface layer, because it depends on the weather and no exact formulas are known, but it is usually assumed to be 10% as high as the boundary layer. Based on the typical height of the boundary layer, the surface layer is often higher than 40 meters and lower than 80. JW Verkaik (2006) indeed reports that a large number of observations from Cabauw at 20 m are inside the surface layer, the 100 m level is already outside of the surface layer, and Korrell et al. (1981) used the observations at 50 m in their analysis, but not those at 100 m. Moreover, Braam (2008) reported that the observations at 60 m are outside of the surface layer only in the early morning; the height of the boundary layer starts growing at around 6 AM, and reaches 600 m between 9.30 AM and 10.30 AM, and reaches its maximum height, between 1800 m and 2000 m, at the end of the day Unfortunately, there is no mention of the 40 m level in Braam (2008). Note that these measures were obtained in May, therefore they are likely to be smaller in Winter, and larger in the Summer. This evidence seems to support the fact that 40 m is inside the surface layer most of the time.

3.2.1 Obukhov Length

The Obukhov length is computed as in equation 2.4, reported here:

$$L = -\frac{u_*^3 T_v}{\kappa g \overline{w'\theta_v}}$$

The flux of virtual potential temperature can be computed following the formulas in section 2.2.3, as the data contains all the necessary quantities; u_* is given, as well as the specific humidity and the pressure. The Obukhov length is computed at each height level using the corresponding air temperature measurement, and the fluxes measured at the surface.

3.2.2 Gradients

The momentum flux-profile relationship is in equation 2.3, reported here for the reader's convenience:

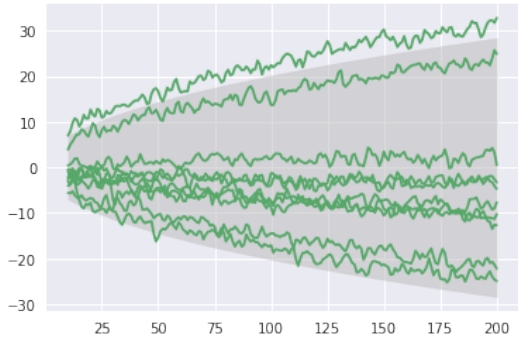
$$\phi_m(\xi) = \frac{\partial \overline{u}}{\partial z}\frac{kz}{u_*}$$

In order to compute it from the data, we need to compute the derivative of the wind speed with respect to the altitude. In general, the derivative can be obtained by fitting a model on the observations, and computing the derivative using the model. The simplest option is to use a piecewise linear function that passes through the measurements; let the observations be sorted by height and y_i the measurement at height z_i, then for $z \in [z_i, z_{i+1}]$ we have:

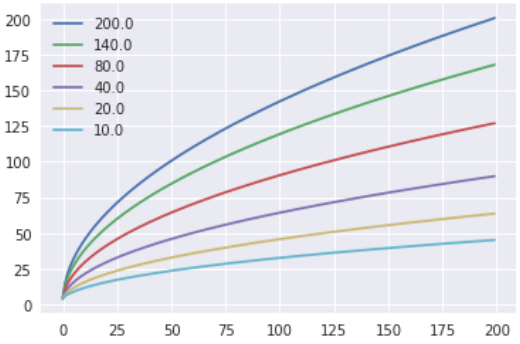
$$f(z) = y_i + (z - z_i)\frac{y_{i+1} - y_i}{z_{i+1} - z_i} \tag{3.1}$$

The derivative of this function at height z_i is then the average of the slope of the segments that start and end at z_i:

$$\begin{aligned} f'(z_i) &= \lim_{h\to 0} \frac{f(z_i + h) - f(z_i - h)}{2h} \\ &= \frac{1}{2} \cdot \lim_{h\to 0} \left(\frac{f(z_i + h)}{h} - \frac{f(z_i - h)}{h} \right) \\ &= \frac{1}{2} \cdot \lim_{h\to 0} \left(\frac{y_i}{h} + \frac{y_{i+1} - y_i}{z_{i+1} - z_i} - \frac{y_i}{h} + \frac{y_i - y_{i-1}}{z_i - z_{i-1}} \right) \\ &= \frac{1}{2} \cdot \left(\frac{y_{i+1} - y_i}{z_{i+1} - z_i} + \frac{y_i - y_{i-1}}{z_i - z_{i-1}} \right) \end{aligned} \tag{3.2}$$



(a) Prior distribution and some samples; the grey area is the 95% CI.



(b) Values of the kernel: each line corresponds to a different value of z_1, and z_2 is on the ordinate.

Figure 3.3: Behavior of the kernel in equation 3.5, where the altitude is on the x axis, and the predicted value on the y axis. Notice the approximate logarithmic profile with which the predicted value changes with altitude; this is caused by the square root term in the kernel, and emulates the effect of the $\ln z$ term in equation 3.3.

In order to compute this derivative at the lowest measured level $z_1 = 10\,\mathrm{m}$, we can exploit the no-slip condition and introduce an artificial observation $y_0 = 0\,\mathrm{m\,s^{-1}}$ at z_0. The value of the *rougness length* z_0 depends on the properties of the surface, and, although no unambiguous value is known for the Cabauw observatory, its value is likely between 10^{-1}m and 10^{-2}m (JW Verkaik, 2006; Baas et al., 2017), therefore it is reasonable to conclude that its effect is negligible on the final gradient.

A more complicated model, introduced by Nieuwstadt (1984), is

$$f(z) = a + bz + cz^2 + d\ln z \tag{3.3}$$

The model is linear in its parameters, therefore equation 2.20 can be used to compute the coefficients, using the feature mapping $\phi(z) = \left[1, z, z^2, \ln z\right]^\intercal$ and regularization parameter $\lambda = 0$. Once the coefficients are known, the gradient is trivial to compute:

$$f'(z) = b + 2cz^2 + d\frac{1}{z} \tag{3.4}$$

After removing the observations for which this model has a $R^2 < 0.9$, Nieuwstadt (1984) estimates the uncertainty of this gradient to be around 30%. Note that some works, such as Yage et al. (2006), do not use the squared term, and others, such as Steeneveld et al. (2005) add squared and even cubic logarithmic terms $\ln^2 z$ and $\ln^3 z$.

Finally, a third method of computing the gradient is to fit a Gaussian process to the profile using the following kernel:

$$k(z_1, z_2) = \exp\left(-\frac{(z_1 - z_2)^2}{2\sigma_0^2}\right) + \sqrt{\sigma_1^2 + z_1 z_2} + \sigma_2^2 \mathbb{1}[z_1 = z_2] + \sigma_3^2 \tag{3.5}$$

where $\mathbb{1}[P]$ is an indicator function whose value is 1 if the predicate P is true, and 0 otherwise, and the parameter σ_2 controls the noise in the measurement. Unfortunately, this noise is not constant, and does not depend on the altitude level; for example, for the wind speed u, the precision of the instrument is $\max(0.1, 0.01 \cdot u)$. Therefore, all the parameters must be found by optimizing the marginal likelihood. Figure 3.3 shows the prior distribution of a Gaussian process with this kernel, as well as the value of the kernel for some choices of z_1 and z_2.

Our final goal is to obtain the derivative of the predictive posterior mean, which can be done using equation 2.36; for this we need to derive equation 3.5 with respect to z_1:

$$\frac{\partial k(z_1, z_2)}{\partial z_1} = \frac{z_2 - z_1}{\sigma_0^2} \exp\left(-\frac{(z_1 - z_2)^2}{2\sigma_0^2}\right) + \frac{z_2}{2\sqrt{\sigma_1^2 + z_1 z_2}} + \sigma_2^2 \mathbb{1}[z_1 = z_2] \tag{3.6}$$

3.3 Monin-Obukhov Similarity Theory

Even though there is agreement on form of the momentum flux-profile relationship, there is still debate on the exact values of the coefficients, with different experiments resulting in different values (Högström, 1988). In order to ensure a fair comparison with the results of this work, we fit the universal functions to the data from the Cesar database. Their general form is:

$$\phi(\xi) = \begin{cases} a + b\xi & \xi \geq 0 \\ a(1 - c^2\xi)^d & \xi < 0 \end{cases} \tag{3.7}$$

Where a is close to 1, b is positive, and d is negative. Since d is negative, the base of the power must be positive, hence the squared c. Following the approach outlined in section 2.3.3, the coefficients a, b, c and d can be found by minimizing the L2-regularized mean squared error using equations 2.18 and 2.19; in this case, the parameter vector is $\boldsymbol{\theta} = [a, b, c, d]^\intercal$. The gradient of 3.7 is:

$$\nabla_\theta \phi(\xi)|_{\xi \geq 0} = \begin{bmatrix} 1 \\ \xi \\ 0 \\ 0 \end{bmatrix} \qquad \nabla_\theta \phi(\xi)|_{\xi < 0} = \begin{bmatrix} \tau^d \\ 0 \\ -2acd\xi\tau^{d-1} \\ a\tau^d \ln \tau \end{bmatrix} \tag{3.8}$$

where $\tau = 1 - c^2\xi$. The Hessian of 3.7 when $\xi \geq 0$ is simply 0, because it is a linear function in all parameters, while, in the negative case, we have:

$$\nabla^2_\theta \phi(\xi)|_{\xi < 0} = \begin{bmatrix} 0 & 0 & -2cd\xi\tau^{d-1} & \tau^d \ln \tau \\ 0 & 0 & 0 & 0 \\ -2cd\xi\tau^{d-1} & 0 & \frac{2ad\xi\tau^d}{c^4\xi^2 + \tau}\left(2c^2 d\xi - c^2\xi - 1\right) & -2ac\xi\tau^{d-1}\left(d \ln \tau + 1\right) \\ \tau^d \ln \tau & 0 & -2ac\xi\tau^{d-1}\left(d \ln \tau + 1\right) & \tau^d \ln^2 \tau \end{bmatrix} \tag{3.9}$$

Analytical computation of the Hessian allows us to use the Newton conjugate gradient descent algorithm, which provides super-linear convergence rate, unlike other conjugate gradient methods whose rate of convergence is only linear (Nocedal and Wright, 1999). This model is then fitted to the data using L2 regularization, and evaluated as the other regression models.

3.4 Model Fitting

In this section, we discuss how we use the data from the Cesar database to predict ϕ_m.

3.4.1 Features

All the features come from the Cesar database. The predictors are partitioned in five sets:

F1: soil temperature, altitude z, wind at z, temperature at z, wind at 10 meters, temperature at 10 meters, wind at 20 meters, temperature at 20 meters, wind at 40 meters, temperature at 40 meters;

F2: Soil heat flux;

F3: Net radiation;

F4: Rain amount, dew point;

F5: Turbulent kinetic heat flux H, turbulent latent heat flux λE;

These sets are used cumulatively in the order they are listed, meaning that F2 is used in conjunction with F1, F3 with F2 and F1, and so on. Features that can be computed from others, such as the virtual temperature (equation 2.5), are not included. The reason is that the less inputs the models require, the more useful and "agile" they can be when used as a component in a larger system, such as climate

simulations: reducing the amount of computations makes the simulation faster and reduces numerical errors, since many approximate constants have to be used. Moreover, one of the objectives of this work is to quantify the effect of the feature sets on the prediction accuracy; using derived features would actually make this process difficult, since the point of it is to find the fundamental quantities of importance.

This division was decided based on the knowledge of a domain expert, so that the level of a feature reflects both its expected impact on the performance and how desirable it is to include it. An example of the former reasoning is with F4, where the effect of moisture is expected to be already captured by the soil heat flux (in F2), and to be generally negligible in all but the most extreme conditions. An example of the latter reasoning is with F5, because turbulent fluxes are hard to measure accurately (see section 3.1.1), and current simulation models are known to be quite inaccurate in their estimation of these fluxes (Optis et al., 2014). Similarly, the friction velocity was not included, because the point of predicting ϕ_m is to use it to estimate u_* from the wind gradient, which is readily measured both in real life and in simulations. JW Verkaik (2006) has shown that the direction of the wind affects the universal functions at Cabauw, because of the different covers of the surface, changing the roughness length, and disturbances by the main mast, preventing accurate wind speed measurement. Nonetheless, we decided not to include the wind direction in the features, as this is very specific to the Cabauw observatory, and would reduce the generality of our models.

We also create a second version of each feature set, augmented with the hourly trend of each variable, except for the altitude z. The reason for including the trend is that it can give an indication of, for example, the time of the day, or other complex phenomena for which there is no measurement. The interval for the trend (one hour) is used because it is enough to capture local variations, but not too large so as to contain irrelevant information. The hourly trend is computed simply as the difference between the current value and the value measured one hour before, divided by one hour. Given the goal of this work, namely to produce a model to be used in climate simulations, it would not make sense to use future values to compute the trend.

Finally, each feature is centered and standardized so as to have zero mean and unit standard deviation:

$$x'_{i,j} = \frac{x_{i,j} - \mu_j}{\sigma_j} \tag{3.10}$$

This method is not robust to outliers, since they heavily affect mean and standard deviation; this can be prevented by subtracting the median and normalizing with the interquartile range instead. These two methods give similar results in our datasets, therefore we follow equation 3.10, since it it the most widely used in practice. Obviously, every μ_j and σ_j are computed only on the training data, and used to normalize both training and testing data.

3.4.2 Estimators

The models that we use are ridge regression, k-nearest neighbors, gradient boosted trees, and neural networks. Due to the size of the dataset, we cannot use kernel-based algorithms such as SVM and Gaussian Processes, as they require $\Theta(N^2)$ memory to store the kernel matrix, and $O(N^3)$ time to invert it.

All models but neural networks are fitted using nested cross validation with random hyper-parameter search. This procedure detailed in section 3.5, but we list here the distributions of the hyper-parameters that we use in the random search:

- Ridge regression and Monin-Obukhov estimator: the only hyper-parameter to tune is the regularization coefficient, for which we use a $\log_{10}$-uniform distribution[8] from 10^{-6} to 10^{1}.
- k-nearest neighbors: the hyper-parameters to tune are the number of neighbors, chosen uniformly from 1 to 15, the distance function, either L1 or L2 norm, and the weights of the neighbors, either uniform or directly proportional to the distance to the query point.

[8] a random variable X has a $\log_\beta$-uniform distribution from β^a to β^b if $\log_\beta X$ is uniformly distributed between a and b. Equivalently, if Y is uniform between a and b, then β^Y is $\log_\beta$-uniform.

Table 3.1: Distribution of the hyper-parameters for gradient boosted trees

Hyper-parameter	Distribution	Values
Number of estimators	Log_{10}-uniform	From 10^1 to 10^4
Learning rate	Log_{10}-uniform	From 10^{-3} to 10^0
Number of features	Uniform	All possible values
Maximum depth	Uniform	From 4 to 12
Sub-sampling factor	Uniform	From 25% to 100%
Loss function	Uniform	Least squares, least absolute deviation, Huber loss (equation 2.27)
δ (for the Huber loss)	Uniform	From 1% to 99%

- Gradient boosted trees: the distributions are shown in table 3.1. Note that δ, for the Huber loss, refers to the percentile of the residuals, so that $\delta = 0.5$ uses L2 loss for the smallest 50% and L1 loss for the largest 50%.

We use log-uniform distributions for parameters with a high range of possible values to reflect the intuition that small values are as likely as large values, and to avoid the samples being dominated by the larger values. Samples from an uniform distribution from 10^0 to 10^3 are ten times more likely to be from $[10^2, 10^3]$ than from $[10^1, 10^2]$, and, on average, only one in 100 samples will be in $[10^0, 10^1]$. Log_{10}-uniform distributions do not have this problem, and a sample is equally likely to belong to any of the three intervals.

3.5 Performance Evaluation

The goal of performance evaluation is to obtain a realistic and unbiased estimate of the performance of a model on unseen data. Because the Monin-Obukhov similarity theory is only valid in the $-2 \leq \xi \leq 1$ range, we will use this data as the primary target for evaluation. We will also evaluate the models on the full dataset, to see what kind of performance can be expected outside of the typical range where the similarity theory is employed.

3.5.1 Evaluation Procedure

The two research questions require different evaluation procedures. The first question essentially entails finding what kind of performance is possible to achieve: this is best done using nested cross-validation, so that this estimate is not biased. The second research question is about comparing alternatives: in this case, we use repeated cross-validation to obtain estimates with lower variance. In both cases we use random search (Bergstra and Bengio, 2012) to optimize the hyper-parameters of the models.

For nested cross-validation, we use 10 outer folds and 10 inner folds, and try 10 random hyper-parameter combinations. Since we have 2 datasets and $5 \cdot 2$ feature sets, every model is fit a total of $2 \cdot 5 \cdot 2 \cdot (10 \cdot 10 \cdot 10 + 10) = 20200$ times. For repeated cross-validation, we use 10 folds and 25 hyper-parameters combinations, resulting in 250 fittings.

todo how many repeats? better be fixed!

According to the model and its hyper-parameters, fitting can take from a few seconds up to more than 24 hours; in order to make the process feasible, we implemented algorithm 1 using Apache Spark (Zaharia et al., 2016). Since the number of stragglers is very low[9], running multiple nested cross validation jobs at the same time can reduce the duration of the whole process by five to ten times.

A fundamental assumption underlying hold-out evaluation methods is that the samples in the training set are independent and identically distributed, so that the distribution in the two partitions are equal. This is not our case, since there is a inherent time dependency in the data, meaning that samples obtained close in time are very similar. This can be confirmed by training and evaluating a

[9] for gradient boosted trees, the slowest method, the median fitting time in our experiments is around half a minute, the third quartile is around ten minutes, and the maximum is between 20 and 30 hours

k-nearest neighbors classifier with $k = 1$ on random splits: the resulting mean squared error is in the order of 10^{-3}, which is clearly unrealistic. To circumvent this problem, the CV folds are created on *months*: all the samples in a given month and year are either in the training set or in the validation set. We choose months instead of days because the weather often does not change significantly in the span of 24 hours, whereas in a month there are around three weeks worth of samples that are independent from the conditions at the start and end of the month.

3.5.2 Evaluation Metrics

The main evaluation metric is the mean squared error, since that is what we are optimizing for, but we compute other metrics in the outer cross validation loop to get a more complete idea of the performance of the estimators:

- Mean Squared Error (MSE);
- Mean Absolute Error (MAE);
- Median Absolute Error (mAE);
- Mean Absolute Percent Error (MAPE);
- Median Absolute Percent Error (mAPE);
- R^2 Score:

$$1 - \frac{\sum (f_n - t_n)^2}{\sum (t_n - \bar{t})^2}$$

Where f_n is the predicted value for the test sample $\boldsymbol{x}_n$ with true value t_n, the squared error is $(f_n - t_n)^2$, the absolute error is $|f_n - t_n|$, the absolute percent error is $100|1 - f_n/t_n|$. We present both mean and median scores because the former are heavily skewed by outliers. We also present both absolute and percent errors because the latter are easier to interpret, but tend to explode when the true value is very small,[10] a condition that frequently happens in our dataset.

3.6 Success Criteria

The differences needed to answer the research questions are quantified using the effect size as described in section 2.3.9. We prefer the effect size over the mean MSE because the effect size allows us to consider the standard deviation of the MSE, as well as its mean; this could lead us to prefer models with slightly worse average MSE, but more consistent and reliable performances (i.e. smaller standard deviation). Since we are looking for an improvement, we compute the right 95% confidence interval for the population effect size. In other words, besides providing a point estimate for the population effect size δ_{pop}, we also provide an interval $[\delta_{low}, \infty)$ that contains it 95% of the times. We say the treatment is better if $\delta_{low} > 0$, and the best treatment is the one with the largest δ_{low}.

A successful answer to the first research question entails finding a model whose mean squared error on any feature set, with or without trend, is smaller than the mean squared error of the MOST estimator introduced in section 3.3. The second research question is answered by comparing the mean squared error obtained on the ten feature sets (F1 to F5, with and without trend). This comparison is done using the model that achieved the best performance when answering the first research question, and, in order to increase the reliability of our estimates, we repeat the cross validation procedure 5 times for each combination of features and trend.

Note that we cannot use the output of a model to obtain the importance of the features, as this method is not reliable when some of them are correlated (Strobl et al., 2007, 2008), which is a very relevant issue in our case. Trivially, wind speed and temperature at the different levels are very strongly correlated (Pearson's r is above 0.95), and there are other, more complex, correlations, such

[10] predicting 1 for a true value of 100 gives a percent error of 99%, but predicting 100 for a true value of 1 gives, somewhat unfairly, a percent error of 9900%, whereas their absolute error is identical.

as between the net radiation and the soil heat flux, or between the soil temperature and the dew point (Pearson's r is respectively 0.75 and 0.79). In spite of the correlations, all these variables contribute to the flux-profile relationships, often in very nuanced ways.

Chapter 4

Results

This chapter presents the results obtained following the procedures outlined in chapter 3. Section 4.1 describes the dataset and shows some of its peculiarities, then section 4.2 identifies the best method to compute the gradient of the profile, finally section 4.3 shows the momentum flux-profile relationship obtained on the dataset and its predictions from the estimators we discussed previously.

All the processing was done using the Python programming language[1] and the usual Data Science library stack, in no particular order: Pandas (McKinney, 2010), scikit-learn (Pedregosa et al., 2011), SymPy (Meurer et al., 2017), Matplotlib (Hunter, 2007), Seaborn[2], NumPy (Walt et al., 2011), Jupyter Notebook (Kluyver et al., 2016), IPython (Pérez and Granger, 2007), Spark (Zaharia et al., 2016), with the crucial support of Hopsworks (Ismail et al., 2017)

4.1 Exploratory Data Analysis

The dataset consists of 17 years of measurements, from January 2001 to December 2017, for a total of 3,436,416 observations. The Cesar database contains one tar-gzipped archive for every month, whose content is a single data file in the netCDF format (Unidata, 2018). The archives contain a subset of related variables, and we used the datasets containing surface fluxes, tower meteorological profiles, meteorological surface data, and soil heat, for a total of 816 datasets covering 204 months.

Unfortunately, most of these observations contain unreliable measurements, as detailed in section 3.1.3, leaving only 1,561,973 usable records. Additionally, the turbulent fluxes measured in March 2016 exhibit a much wider range than the March measurements of other years. Roughly 15 to 20% of the measurements in that month are suspicious; since the Cesar database contains a separate dataset every month, we decided to completely exclude the dataset of March 2016, fearing for a systematic error somewhere in the process. Moreover, 6 measurements of the turbulent latent heat are way above the acceptable range, 4,538 do not have a dew point, and 247,656 do not have a soil temperature; this leaves 1,308,362 usable records. Table 4.1 summarizes the weather parameters that were used to compute the similarity functions and/or as features for the models, and table 4.2 presents the number of observations that were gap-filled with each method, for the measurements for which this information was provided.

Figure 4.1 shows the turbulent latent heat flux versus hour of day both in Winter and Summer, whereas figure 4.2 shows the friction velocity as a function of air temperature and wind speed. The dependency of u_* with the wind speed of figure 4.2b is quite complicated: there are three clear regions with different constant of proportionality and correlation coefficient. The group where the two quantities are almost perfectly correlated corresponds to imputed values for the friction velocity, derived from the wind speed. The three other groups differ widely in the values of the fluxes and radiation; for example, fitting an ordinary least squares model to 1000 bootstraps of the observations with negative net radiation yields an average proportionality coefficient of 0.051 with standard deviation 1.52×10^{-5}, whereas the same procedure on the complementary set yields an average of 0.046 with

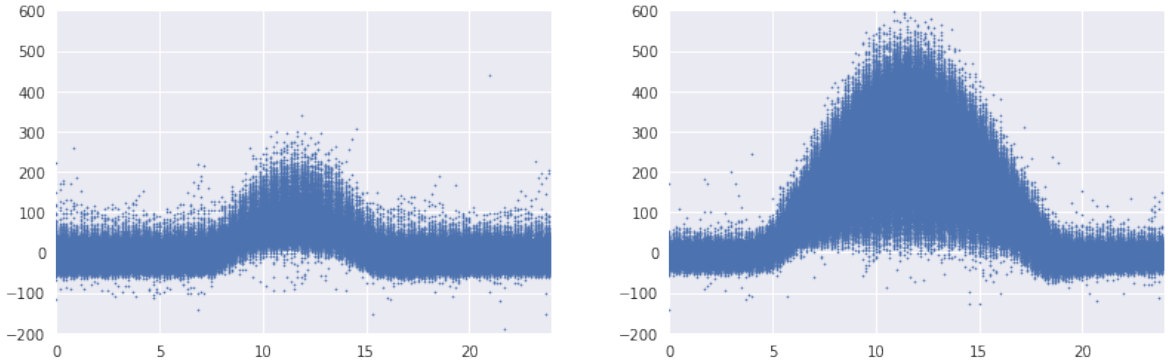
[1] `https://www.python.org/`
[2] `https://doi.org/10.5281/zenodo.883859`

Table 4.1: Basic descriptive statistics of the features in the Cesar database, except for the air density and virtual air temperature, which were computed by us. Note that the relative humidity was computed by the Cesar consortium using values from other measurements, and 16,833 records (1.3%) have a relative humidity above 100%.

	Mean	**Std.**	**Min.**	**25%**	**50%**	**75%**	**Max.**
H	3.7	47.3	-460.0	-25.8	-14.7	28.2	479.6
λE	74.7	114.9	-534.8	-4.9	23.8	132.9	6,940.8
u_*	0.3	0.1	0.1	0.2	0.3	0.4	1.8
Wind Speed	5.9	2.6	1.0	4.0	5.4	7.2	29.4
Air Temp.	284.7	6.4	259.6	280.1	284.8	289.4	306.9
Soil Temp.	284.4	5.7	271.9	279.6	284.3	289.1	302.6
Dew Point	280.3	6.1	188.0	276.6	280.6	284.7	296.7
Spec. Hum.	6.6	2.5	-0.9	4.8	6.4	8.3	17.8
Rel. Hum.	76.5	15.2	0.0	66.9	79.2	88.3	133.7
Air Press.	1,009.6	69.0	960.9	1,008.3	1,014.8	1,020.7	1,046.4
Rain Amount	0.0	0.1	0.0	0.0	0.0	0.0	15.1
CO_2 **Flux**	-0.1	0.3	-7.5	-0.2	0.0	0.1	6.1
Soil Heat Flux	2.9	15.6	-77.8	-6.5	-0.7	8.9	1,139.0
Net Rad.	81.8	167.2	-158.5	-35.4	-3.0	172.5	7,775.4
Air Dens.	0.3	0.1	0.1	0.2	0.3	0.3	2.8
Virt. Air Temp.	1,443.8	454.9	124.8	1,104.2	1,391.0	1,753.2	3,506.0

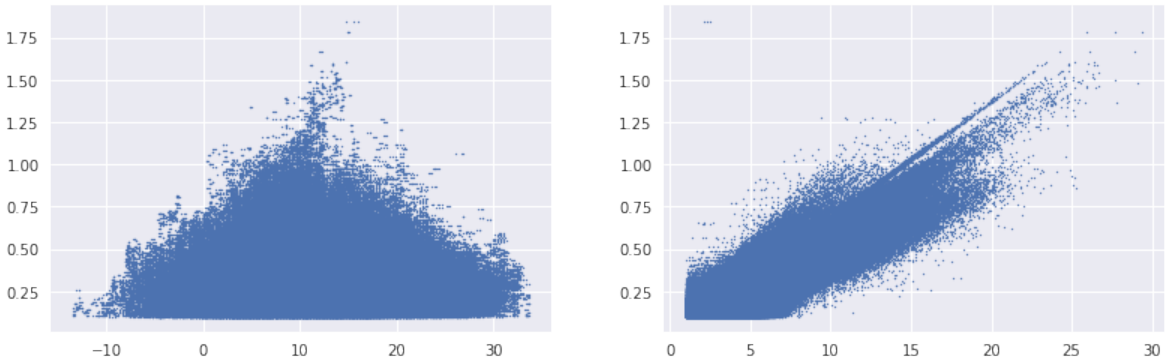
Table 4.2: Number of measurements by gap filling method, for the quantities that have this information. Due to inconsistencies in the Cabauw Documentation (Bosveld, 2014), we cannot be sure of the meaning of the columns, but this is our interpretation: 0 - Unknown, 2 - Cabauw In situ (i.e. from the masts, not gapfilled), 3 - Automatic Weather Station in Cabauw (also not gapfilled, but sometimes secondary source), 4 - Computed from profiles, 5 - Interpolated, 6 - Cabauw-based model, 7 - De Bilt-based model. As the net radiation cannot be obtained from the profiles, we think that sources 4 and 5 for this quantity are mistakes, and should respectively be 6 and 7.

	0	**2**	**3**	**4**	**5**	**6**	**7**
H	-	1,192,346	-	-	17,291	97,720	1,005
λE	-	1,180,434	-	-	18,503	97,846	11,579
u_*	-	1,174,593	-	132,461	465	618	225
Wind Speed	-	1,302,308	2,338	-	879	331	2,506
Air Temp.	-	1,298,075	-	-	2,419	5,959	1,909
Spec. Hum.	-	1,292,626	-	-	3,149	10,693	1,894
Air Press.	7,096	-	1,298,205	-	654	-	2,407
Rain Amount	7,096	1,181,167	117,844	-	346	-	1,909
CO_2 **Flux**	-	964,749	-	-	48,983	287,855	6,775
Soil Heat Flux	-	1,232,822	-	-	-	72,256	3,284
Net Radiation	-	1,274,106	-	20,962	1,715	-	11,579



(a) Winter (b) Summer

Figure 4.1: Turbulent latent heat flux versus hour of day in Winter (left) and Summer (right). One can see both the difference in day duration, and the effect of increased temperature on evaporation and transpiration. The sensible heat flux follows a very similar pattern, with lower absolute values.



(a) Temperature (b) Wind

Figure 4.2: Friction velocity versus temperature (left) and wind speed (right). There is a clear subgroup where the wind and u^* are almost perfectly correlated; this is because of the gap-filling technique discussed in section 3.1.2. Additionally, there are three subgroups with different regression line fit; they differ widely in fluxes, radiation, and altitude (since u_* is computed at the surface, the intercept is different, because wind speed tends to be positively correlated with altitude).

standard deviation 1.40×10^{-5}. Note that this dependency might be related to the observed imbalance in the surface energy budged, usually around 15% during day time and 100% during night time at the Cabauw observatory; Bosveld (2018) notes that the imbalance seems to depend on the speed of wind.

Figure 4.3 shows the result of the t-SNE dimensionality reduction algorithm (van der Maaten and Hinton, 2008) applied to 250,000 samples (19.2% of the total), using the same features of table 4.1, plus z, L, ξ, the wind and virtual temperature gradients, ϕ_m, the month of the year and hour of day. The 250,000 samples were chosen uniformly at random and standardized with equation 3.10 before fitting, but the colors were computed using the raw features on a perceptually uniform color scale, after excluding the top and bottom 1% values. Overall, the points cluster in a relatively uniform blob, which is not surprising once you consider that the measurements were taken continuously, meaning that all physical quantities change smoothly, thus all points close in time are also close in feature space.

The colors reveal some structure, though: the left-hand side of the blob contain measurements taken in the Summer, and the right-hand side contains measurement in Winter (see the air temperature in figure 4.3e and month in figure 4.3v). Moreover, unstable conditions are clustered in a stripe, centered vertically, that runs from left to right (see the fluxes of figures figs. 4.3b, 4.3c and 4.3t, the radiation in figure 4.3u, and the hour of day in figure 4.3w). In figure 4.3h we also a see cluster on the top with

Table 4.3: Pearson correlation coefficient between the gradients predicted by the different methods, divided by level. The FD gradient is poorly correlated with the other two methods at the 10 meters level, but is very well correlated with the LG gradient at the other levels. The LG and GP gradients disagree on the lower levels, but it appears their predictions converge as altitude increases.

z	FD / GP	LG / FD	LG / GP
10	0.192	0.480	0.317
20	0.645	0.991	0.640
40	0.863	0.981	0.835

all the observations at 40 meters, whereas the two lower levels are mixed somewhat uniformly in the bottom part of the plot.

Finally, figure 4.3x shows the result of k-means clustering in the standardized feature space, using $k = 2$ (blue/orange) and $k = 4$ (the four shades). This clustering is consistent with the qualitative division above, with the dark orange and blue clusters corresponding to Summer and Winter respectively, light orange corresponding to unstable conditions with considerable vertical movement of air, and light blue corresponding to strong horizontal winds with little movement in the vertical. Stably stratified conditions seem to occur at the boundary of the blob, where ϕ_m takes the highest values (see figure 4.3l). This is also not surprising, since stable stratification tend to happen at night throughout the year, unlike free convection or z-less scaling, which only happen in specific conditions.

Talking about speed of wind, figure 4.2b shows some measurements exceeding $25\,\mathrm{m\,s^{-1}}$; according to the Beaufort scale, holding umbrellas and walking become hard at $15\,\mathrm{m\,s^{-1}}$. Those measurements are not issues with the data: figure 4.4 shows the highest wind speed present in the Cesar database to date, corresponding to cyclone Jeanett.

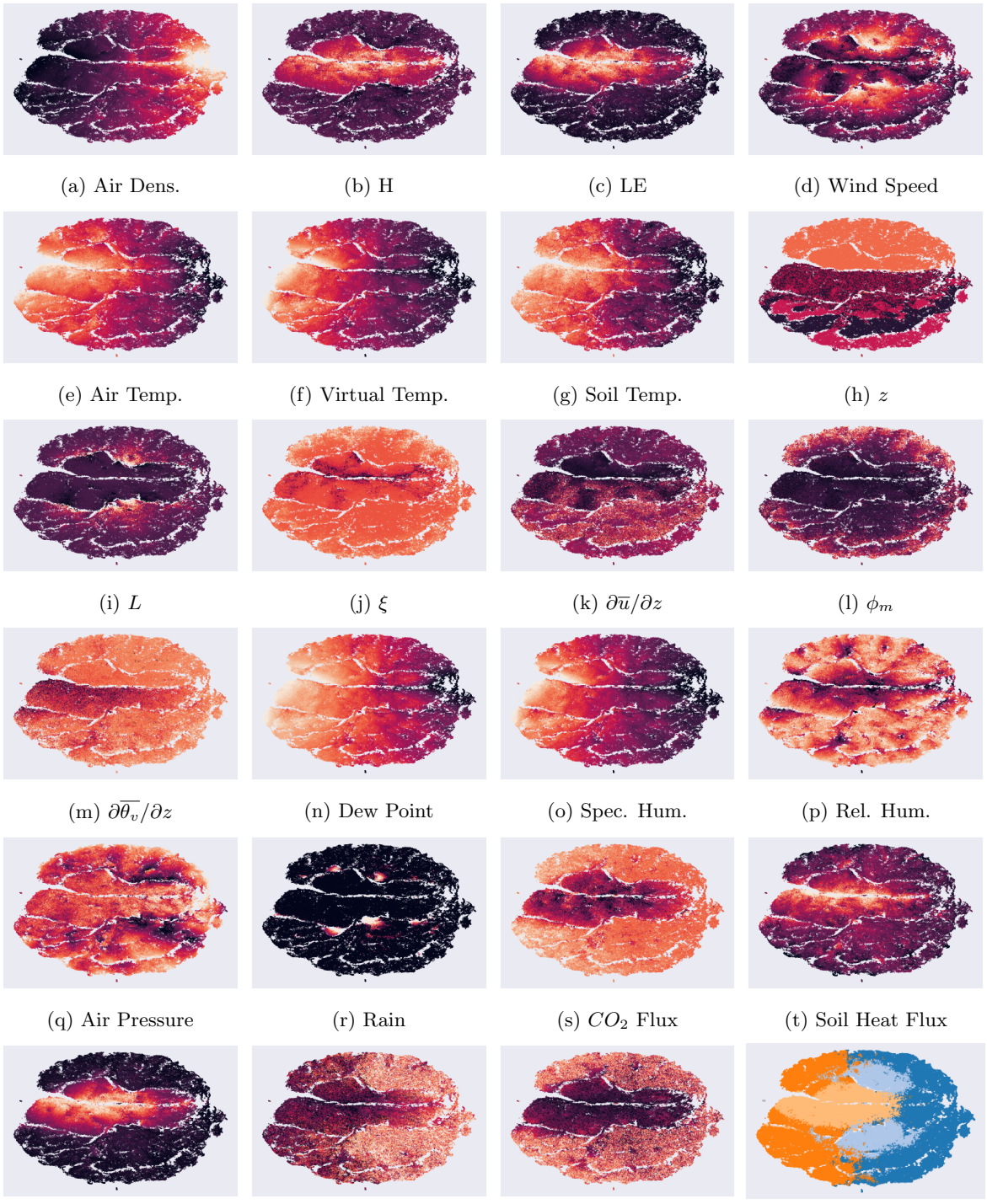
4.2 Gradient Computation

Section 3.2.2 introduced three different ways to compute the gradient, but we need to choose one to compute the universal functions. In this section, we refer to the finite differences method in equation 3.2 as *FD*, the logarithm-based model of equation 3.4 as LG and the Gaussian Process-based model of equation 2.36 as *GP*. Table 4.3 shows the Pearson correlation coefficient between the three methods, separately for every altitude level. It is clear that the FD gradient is inadequate at the lowest level; this can be explained by considering that equation 3.2 with $z = 10$ reduces to the wind speed at 20 meters divided by 20. This method appears to be too naive, especially since it does not take the roughness length into consideration, but, as JW Verkaik (2006) showed, finding a value for z_0 is far from trivial. In spite of this, the FD and the Log gradients strongly agree on all levels, and are quite different from the gradients by GP

Since LG and GP. predict the wind speed too, we can compare their predictions to the measured wind speed. Notice that these methods are fitted on the wind speed at all levels, from 10 to 200 meters, but we evaluate them only on the three lowest levels, since that is where we compute the universal functions. Table 4.4 shows the squared error between the modeled wind and the true wind speed at the three levels for both models. It is interesting to see that both models are quite accurate at all levels; this allows us to conclude that FD is not reliable at 10 meters.

Figure 4.5 shows the error as a function of the wind speed. It is interesting to see that GP becomes more and more reliable as the wind speed increases; not only does its error decrease, so does the variance of the error. It is possible to give an intuitive explanation for the error of LG: the measurement error of the wind speed is the maximum between $0.1\,\mathrm{m\,s^{-1}}$ and 1%, therefore the error of Log can be attributed almost entirely to uncertainty in the data, whereas GP seems to be overfitting.

Given that the final goal of finding the gradient is to compute the universal functions, for which we already have more or less agreed upon expressions (equation 2.6), it makes sense to compare the error of these expressions when predicting the universal functions (equation 2.3) computed with the gradients coming from the three different methods; the best method is, then, the one with the smallest error. Table 4.5 contains these errors, both with all the data, and by considering only $2 \leq \xi \leq 1$. FD



(a) Air Dens. (b) H (c) LE (d) Wind Speed

(e) Air Temp. (f) Virtual Temp. (g) Soil Temp. (h) z

(i) L (j) ξ (k) $\partial\overline{u}/\partial z$ (l) ϕ_m

(m) $\partial\overline{\theta_v}/\partial z$ (n) Dew Point (o) Spec. Hum. (p) Rel. Hum.

(q) Air Pressure (r) Rain (s) CO_2 Flux (t) Soil Heat Flux

(u) Net Rad. (v) |Month − 6| (w) |Hr. Day − 12| (x) k-Means

Figure 4.3: Results of the t-SNE dimensionality reduction algorithm; every figure uses a different feature to color the points (darker points have smaller values), except figure 4.3x, where the colors represent the result of 2-means clustering (blue/orange) and 4-means clustering (the four shades). The colors are consistent, in the sense that light and dark shades of blue are placed in the same cluster by 2-means clustering, ditto for orange. Figures 4.3v and 4.3w use respectively the month and hour of day as feature values, but are colored using the values transformed with the formula in their caption. The points were drawn in random order. (Best viewed in color)

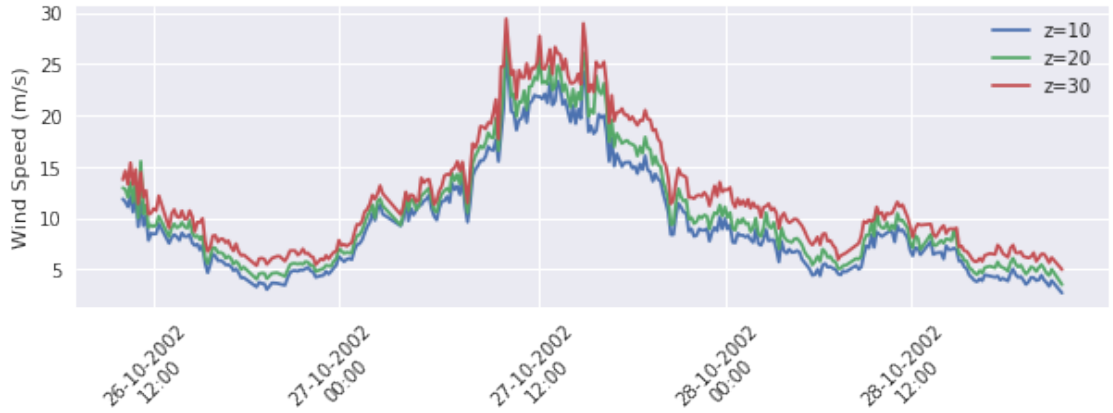


Figure 4.4: Wind speed measurements during Cyclone Jeanett, which struck north-west Europe in October 2002, causing 33 fatalities. The top of the mast registered winds at $35\,\mathrm{m\,s^{-1}}$; although, in absolute terms, destructive, this cyclone registered winds up to $42.5\,\mathrm{m\,s^{-1}}$ in other parts of Europe!

Table 4.4: Absolute percent error of the wind speed modeled by the LG and GP methods at each altitude level. The altitude does not affect the error, and the GP wind speed is closer to the true wind speed than the LG model in the vast majority of cases, although the difference is negligible, often less than a few percents.

	z	LG	GP	z	LG	GP	z	LG	GP
Mean		1.2250	1.0329		2.2605	1.4217		1.3878	1.0364
Std.		1.4033	1.1169		2.3432	1.4658		1.4783	1.1208
Min.		0.0000	0.0000		0.0000	0.0000		0.0000	0.0000
25%	10	0.3714	0.3249	20	0.7447	0.4549	40	0.4744	0.3037
50%		0.8217	0.7608		1.6251	1.0595		1.0263	0.7265
75%		1.5505	1.4179		2.9642	1.9693		1.8413	1.4143
Max.		53.7541	212.3985		106.5623	192.9434		80.3531	43.4657

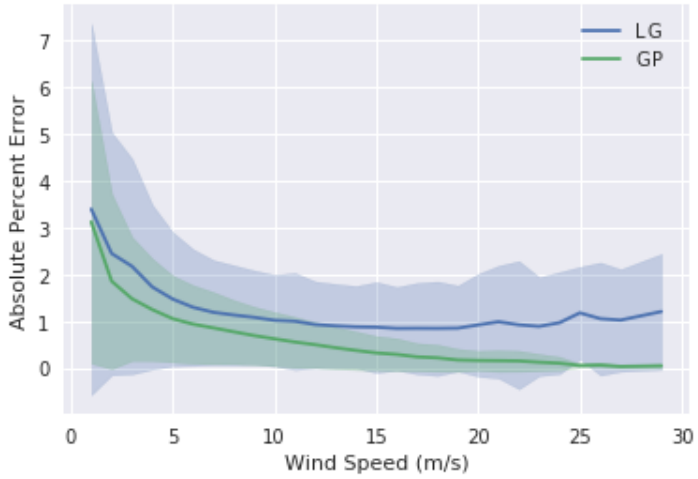
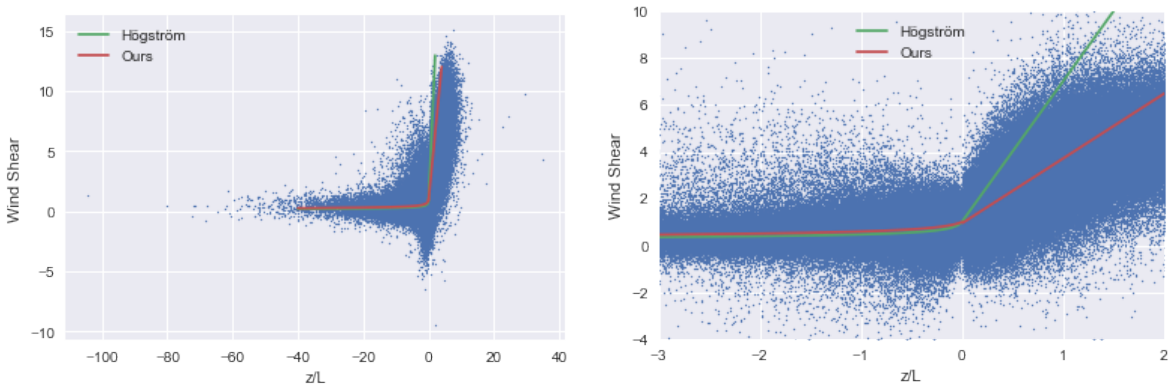


Figure 4.5: Mean absolute percent error of the wind modeled by the LG and GP models as a function of the true wind speed; the lines are the average, and the shaded regions enclose the average plus or minus the standard deviation. Both quantities are computed by binning the wind by its integral part. Notice that slightly less than 5% are above $11\,\mathrm{m\,s^{-1}}$, and only 656 observations, or 0.04% of the total, are above $20\,\mathrm{m\,s^{-1}}$ ($72\,\mathrm{km\,h^{-1}}$)

Table 4.5: Mean squared error of ϕ_m (coefficients according to equation 2.6) when predicting the empirical flux-profile relationship (equation 2.3), computed with the three methods of calculating the gradient. The errors in second line (F.D. w/o z=10) were computed after discarding the data at the 10 meters level, because we argued the F.D. method is not reliable at that level. We show the error both in the range where the Monin-Obukhov similarity theory is known to be valid (84% of the samples), and the error using all data.

Method	MSE ($-2 \leq \xi \leq 1$)	MSE (full dataset)
F.D.	1.994	9.322
F.D. (w/o z=10)	0.627	11.832
Log	0.660	8.498
G.P.	1.213	8.899



(a) Full dataset

(b) MOST Dataset

Figure 4.6: Wind shear ϕ_m versus the instability parameter ξ, with the prediction according to equation 2.6 in green, and our predictions, following equation 3.7 with the coefficients being the median values in table 4.7, in red. Refer to the main text for a discussion of the difference.

is slightly better than LG if we do not consider the 10 m level, but way worse if we include it. GP is significantly worse than FD in the MOST validity region, but only slightly worse on the whole dataset; this suggests that GP is weak when $|\xi|$ is small.

To conclude, although FD (equation 3.2) is the most accurate way of computing the profile gradient, it does not allow us to use data at the 10 m level. Considering that LG (equation 3.4) is just slightly worse, but allows us to include that level (increasing the data available by 50%!), all the results presented from now on make use of it.

4.3 Flux-Profile Relationship

Figure 4.6 shows the wind shear versus the instability parameter, together with the commonly accepted flux-profile relationship in green and the relationship fitted on our dataset in red. Qualitatively, the scatter plot looks as expected, going to zero as ξ decreases, linearly increasing in the $0 \leq \xi \leq 1$ interval, and leveling after that, possibly approaching the $\xi^{1/3}$ form proposed by Grachev et al. (2007). In the next sections, we refer to the interval $-2 \leq \xi \leq 1$ as the *MOST validity range*, or *MOST dataset*; it contains 1,297,841 samples, 83.6% of the total.

The performance of the MOST estimator is shown in table 4.6; we see a considerable increase in the MSE when the full dataset is used; this is not surprising, considering that, with $\xi > 1$, ϕ_m tends to level off, and there are a considerable number of outliers in the range $-20 \leq \xi \leq -2$ (see figure 4.6a) that cannot be fitted by the MOST estimator. Figure 4.6 shows ϕ_m as a function of ξ as computed in our dataset, as well as the predictions of equation 2.6 and the fitted MOST estimator.

Table 4.6: Evaluation metrics for the MOST estimator on both datasets obtained with nested cross-validation. The first number is the average over the 10 outer folds, followed by the standard deviation in parentheses.

Metric	Full	MOST
MSE	0.64 (0.04)	0.32 (0.02)
R^2	0.71 (0.01)	0.61 (0.01)
MAE	0.53 (0.01)	0.39 (0.01)
mAE	0.37 (0.01)	0.28 (0.01)
MAPE	257.80 (207.39)	178.30 (68.99)
mAPE	28.48 (0.81)	24.44 (0.81)

Table 4.7: Values of the coefficients of the MOST estimator of equation 3.7 fitted in the MOST validity range. The minimum values for c and d seem to be outliers, as well as the next smallest value, but the other 8 values are closely clustered together within an interval of about 0.15.

	Mean	Std.	Min.	25%	50%	75%	Max.
a	0.94	0.00	0.94	0.94	0.94	0.95	0.95
b	2.77	0.01	2.76	2.77	2.77	2.78	2.78
c	2.45	0.47	1.25	2.61	2.65	2.67	2.71
d	-0.30	0.11	-0.60	-0.26	-0.26	-0.26	-0.25

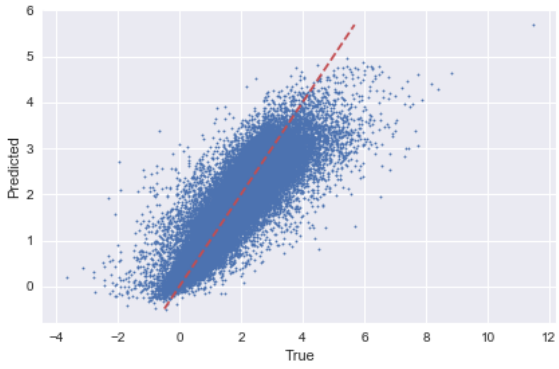
Table 4.6 shows the values of the coefficients in equation 3.7, when fitted to the region of data where the Monin-Obukhov similarity theory is valid. We note a big difference with the values recommended in Högström (1988), as well as most studies on the topic. However, Bouwman (1990) studied the stable, nocturnal boundary layer at Cabauw, and reported values varying between 0.81 and 0.95 for a and between 2.7 and 3.2 for b, depending on the direction of the wind; there is no known explanation for this discrepancy. Although Bouwman (1990) used the same method that we applied to find the coefficients, no goodness of fit statistics, such as MSE or R^2, were reported.

4.4 First Research Question

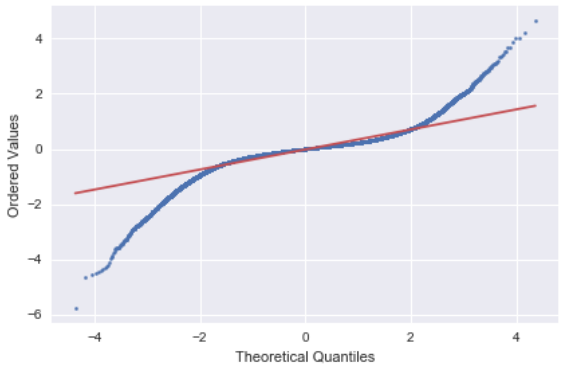
The first research question asked *how can the from the Cesar database be used to predicthe momentum flux-profile relationship more accurately than the Monin-Obukhov similarity theory, by using only macro weather parameters?*

Table 4.8 shows descriptive statistics of the MSE obtained by the best combination of features and trend, for each estimator and for each datasets, as well as the effect size comparing it with the MOST baseline. The scores were obtained with the nested cross validation procedure, and are also shown in figure 4.9. It is very interesting to compare the kNN and GBT estimators: kNN manages to get a larger effect size when compared to MOST, despite the fact that its best performance corresponds to GBT's worst. This can be explained by the different scatter of the data points: kNN is more consistent, with only a couple of outliers driving up the standard deviation, while GBT's performances are more evenly scattered (and the rounding to two decimal digits in the table is deceiving). Consider the samples with the full dataset: the interquartile range for kNN is 0.025, whereas for GBT it is 0.055, and the Winsorized standard deviation over 10^4 bootstraps is respectively 0.017 and 0.027. This difference is large enough to counteract the difference in means, and produce a larger effect size.

Figure 4.7 shows the predictions obtained with the GBT estimator on one of the outer validation folds, and a probability plot of the residuals. It is clear that the error is largely determined by outliers: the mean squared error of these predictions is 0.155, but only 17% of the squared errors is larger than that, and the median is 0.028. Moreover, in figure 4.7a we see a tendency to overestimate small values of ϕ_m and underestimate large ones.



(a) Predicted value (y-axis) versus true value (x-axis), MSE=0.155 and $R^2 = 0.803$



(b) Residuals and normal fit, $R^2 = 0.850$ and Shapiro-Wilk's $W = 0.850$

Figure 4.7: Illustration of the predictions and their residuals obtained by the gradient boosted trees on one of the outer validation folds (107,932 samples).

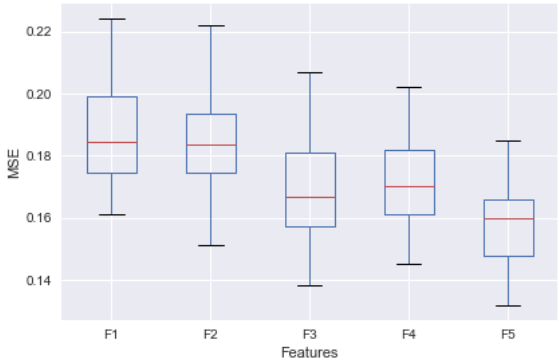


Figure 4.8: MSE obtained by the five feature sets without trend on the MOST dataset.

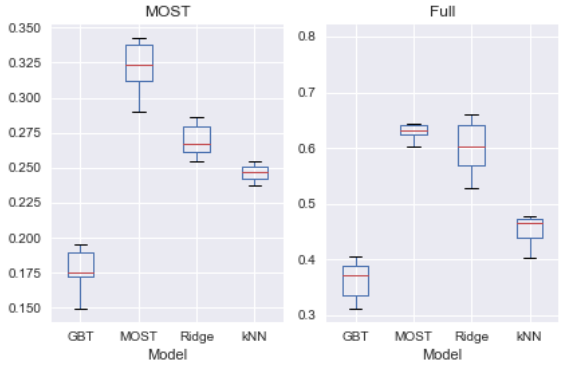


Figure 4.9: MSE obtained by the best models, for the MOST dataset (left) and the full dataset (right).

Table 4.8: Descriptive statistics of the best MSE achieved by each estimator, and effect size comparing it (treatment) with the MOST baseline (control); the only model which is not significantly better than the MOST baseline is Ridge on the MOST dataset.

Dataset	Model	Feat.	Trend	Mean	Std.	Min.	50%	Max.	Effect Size
Full	**MOST**	-	-	0.64	0.04	0.58	0.63	0.71	-
	GBT	F4	Y	0.36	0.03	0.31	0.37	0.40	**[4.7, 7.0**
	kNN	F1	Y	0.45	0.02	0.40	0.46	0.48	**[4.7, 9.3**
	Ridge	F5	N	0.62	0.08	0.53	0.60	0.80	[-0.1, 0.5
MOST	**MOST**	-	-	0.32	0.02	0.29	0.32	0.34	-
	GBT	F3	Y	0.18	0.01	0.15	0.18	0.19	**[6.6, 10.2**
	kNN	F2	N	0.25	0.01	0.22	0.25	0.25	**[4.8, 11.0**
	Ridge	F1	N	0.27	0.02	0.24	0.28	0.29	**[1.9, 4.3**

Table 4.9: Effect sizes comparing the MSE scores of each feature set with and without trend; the upper effect size uses the trend as treatment, whereas the lower effect size uses the trend as control. Using the trend does not improve the performance.

Dataset	Model	F1	F2	F3	F4	F5
MOST	**GBT**	[-0.5, -0.1 [-0.2, 0.1	[-0.3, -0.0 [-0.3, 0.0	[-1.0, -0.5 **[0.2, 0.5**	[-0.4, -0.1 [-0.3, 0.0	[-0.6, -0.3 **[0.0, 0.3**

Table 4.10: Effect sizes comparing the MSE scores of all pairs of feature sets; in light of table 4.9, the trend is not included in the features. The control is on rows, and the treatment is on columns. The only feature sets that do not bring an improvement are F2 over F1 and F4 over F3.

Dataset	Model	Features	F1	F2	F3	F4	F5
MOST	**GBT**	**F1**	-	[-0.2, 0.2	**[0.6, 1.0**	**[0.6, 1.0**	**[1.4, 2.0**
		F2	[-0.4, -0.1	-	**[0.5, 0.8**	**[0.5, 0.8**	**[1.3, 1.8**
		F3	[-1.3, -0.9	[-1.4, -0.9	-	[-0.6, -0.2	**[0.3, 0.7**
		F4	[-1.2, -0.8	[-1.1, -0.8	[-0.2, 0.2	-	**[0.5, 0.9**
		F5	[-1.9, -1.4	[-2.0, -1.5	[-0.8, -0.5	[-1.2, -0.8	-

4.5 Second Research Question

The second research question asked *What impact do different macro-weather parameters have on the quality of the predictions obtained?*

Table 4.9 compares each feature set with and without trend, for the MOST dataset and the GBT estimator. Essentially, the table shows that the features augmented with trend do not provide a significant improvement over those without trend; it is interesting to relate this result with table 4.8, where the largest effect size when comparing GBT with MOST is actually obtained with the trend: this is another example of differing standard deviations counteracting the difference in means.

Figure 4.8 shows the MSE obtained by the GBT estimator on the MOST dataset for each feature set; given that the trend does not provide additional benefits, we compare the feature sets without the trend. This data is used to compute the effect sizes shown in table 4.10. The only feature sets that do not improve over the previous sets are F2 (soil heat flux) over F1, and F4 (rain amount and dew point) over F3; in all other cases, adding features gives a significant improvement. Figure 4.8 might give the impression that the samples are normally distributed, but a Shapiro-Wilk normality test on F1 gives a p-value of 0.0519 ($W = 0.954$), and a p-value of 0.140 ($W = 0.965$) on F3; we feel there is not enough evidence to deem *all* five groups normally distributed, hence our use of the non-parametric bootstrap CI method to compare them.

Chapter 5

Conclusion

The Monin-Obukhov similarity theory is an important component in many climate simulation models. In this work we investigated whether it is possible to improve the predictions of this theory by using macro weather parameters instead of parameters describing turbulence, and we tried to identify the importance of the features.

As shown in table 4.8, we could improve the predictions of the similarity theory even with a simple regularized linear model, and more complex methods can reduce the MSE by almost 50%. Table 4.10 shows that, besides the heat fluxes, the net radiation is an important predictor, whereas the dew point and the soil heat flux do not contribute significantly to the predictive performance. Furthermore, 4.9 shows that no useful information can be obtained from knowing the value of a feature one hour earlier than the prediction.

5.1 Discussion

We can comment on several aspects of this work, highlighting limitations and potential alternative approaches. An obvious, but perhaps easy to ignore, fact is that the models presented are black boxes that cannot be easily interpreted. This approach is very different from the traditional methodology in Physics and related fields, in which theories offer predictions by *explaining* the phenomena at hand.

5.1.1 Features

The features used for prediction play a very important role, since they affect the ability to generalize to different conditions, and the ease with which the model can be applied to new situations. This is especially noteworthy since no feature engineering was performed, except for standardization. This means, for example, that the model could have incorporated biases coming from the instruments used at Cabauw, thereby hurting performance when using samples coming from different instruments (newer version, different maker, different measuring principle, etc.). The features that are computed with the eddy correlation method are particularly susceptible to this issue: since the shortcomings of such procedure are well known, it is likely that better methods will be developed in the future.

Another thing to keep in mind is that the models were trained using only a subset of the features available in the dataset; undoubtedly, using all of them would reduce the error attained by the models. The reason for this is that we hoped to reduce the amount of computation needed for the project by answering both research questions with the results of the nested cross validation procedure. However, the variance of these results proved to be too high to produce significant confidence intervals, so we performed the repeated cross validation procedure to obtain results with smaller variance, leading to narrowed confidence interval[1]. Ideally, the first research question should have been answered with all the features.

Although our experiments showed that measurements in the past do not improve the performance, the method we used was quite crude. Advanced methods explicitly based on time series analysis, such

[1] this is not cheating

as ARIMA, could be tried. We performed some informal experiments with recurrent neural networks, but the results are discouraging, since they had difficulties even in properly fitting the dataset (see appendix B)

trend could be better included using some sort of time series prediction together with the features we considered in this work, but preliminary experiments (3 layers gru + 4 layers fully connected) do not seem to work well)

richardson number instead of obukhov length

Feature importance: Even though we showed that the dew point and the soil heat flux are not important predictors, we cannot rule out the possibility that the model was able to figure out the information contained in these features using features from earlier sets, or, on the contrary, needed features from later sets make sense of them. For example, the soil heat flux could have had an impact only when the model could relate it to the turbulent fluxes in F5. Similarly, perhaps the model was able to figure out the dew point by relating the the air temperature and the net radiation with the soil heat flux (to estimate humidity). The opposite is also possible, namely these measures could have been not informative enough, and the model needed more features, such as the virtual temperature, to make sense of them.

Micro parameters: One of the biggest obstacles in applying the similarity theory lies in accurately measuring the quantities it relies on. Examples are the von Karman's constant κ, the roughness length z_0, the turbulent fluxes H and λE, the friction velocity u_*, and so on. The advantage of using only macro parameters to predict ϕ_m is that accurately measuring these quantities becomes less important, in the sense that we can force models to find explanations that do not rely on them. Of course, such models would likely have inferior performances, depending on the influence of these parameters on ϕ_m.

Another downside of not using micro parameters is that the resulting models do not generalize well in conditions that are different to those presented in the training set, especially if it contains measures coming from a single location, such as in our case. Training with data coming from different locations certainly produces more general models, as long as the data contains features that are informative and general enough to allow the model to differentiate between the different conditions. For example, the properties of the surface, such as its roughness, have a large impact on turbulence; what our model thinks is the normal behavior of turbulence might in fact be largely determined by the surface around the measurement site. But our model is completely unaware of the effect of the surface, so it might not be able to give accurate predictions at sites with different surface characteristics. See Beljaars (1982) for a discussion on how the obstacles around the Cabauw tower affect the flux-profile relationships measured there; in particular, the turbulence behaves differently depending on the wind direction. A training set with relevant features and diverse samples can alleviate this problem. For example, features could be extracted from satellite images of the surroundings; a very crude approach is to classify the surface into one of several categories, and use a lookup table with pre-computed roughness lengths.

5.1.2 Training Samples

Surface layer: Although the Cabauw tower takes measurements until 200 m, one of the issues we had to deal with is that the height of the surface layer is difficult to determine. We chose to use the measures at 10, 20 and 40 meters only, which comprise only half of the available data. There is evidence that some of the samples at 40 meters are anomalous: in figure 4.3h, the samples at 40 meters are clearly separated from the other two levels, which are well mixed, and in figure 4.6a there are some outliers in the range $-20 < \xi < 0$ that deviate significantly from the assumed relationship. These outliers mostly belong to the 40 m level. On the other hand, it is very likely that in some situations the samples at 80 m, and perhaps even 100 m, are in the surface layer.

Extreme weather events: Another interesting hurricane

Altitude levels: can generate new training samples at more altitude levels, since we have a model for wind and temp (need to interpolate spec. humidity too), according to the simulation that uses this model

5.1.3 Performance

This is a notable achievement, since the chosen model will be just one of the many components of climate simulations, which are notoriously very slow and computationally intensive. A linear model has the potential to compete with the efficiency of MOST, depending on the number of features, since it is fundamentally a dot product

even though we predicted phim, the actual interest is in u*. we predict phim to have a direct means of comparing with state of the art (?), Weber (1999) computes u* from the standard deviations obtaining rmse between 0.05 and 0.1

incorporate uncertainty in the model (e.g. quantile regression), then use this in the simulations to get climate uncertainty (monte carlo by sampling from prediction would be a quick and dirty way of implementing this, although it would make the simulation quite slow)

5.1.4 Deployment

todo

5.2 Related Work

todo (Hurley and Luhar, 2009) predicts ustar, fluxes, wind speed etc., would be interesting to compute phim with their predictions and compare

(Holtslag and van Westrhenen, 1991) predicts abl parameters (ustar, fluxes etc) from macro parameters obtained from simulations. quite cool we should also try to predict using only inputs from simulations

(Nieuwstadt, 1978) predicts ustar from profiles (choose ustar that minimizes difference between measured and predicted profile), give only r=0.84, r=0.9 and r=0.89 for different wind directions, very similar to our sqrt(r), but they needed to use z0. they also give a method to compute confidence intervals (we can do that too!!!)

(Yaglom, 1977) argues that

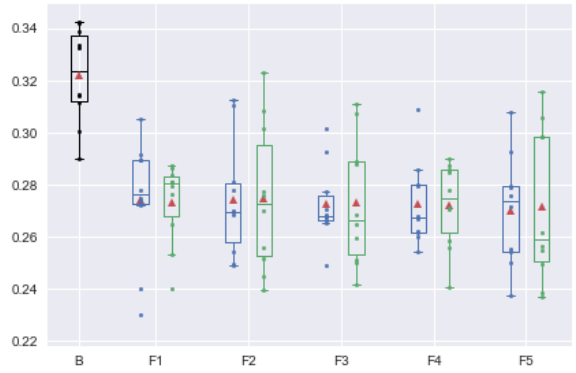
5.3 Limitations and Future Work

todo future work: see appendix, also test on other dataset, try inside simulations similarly to (Holtslag and van Westrhenen, 1991)

Appendix A

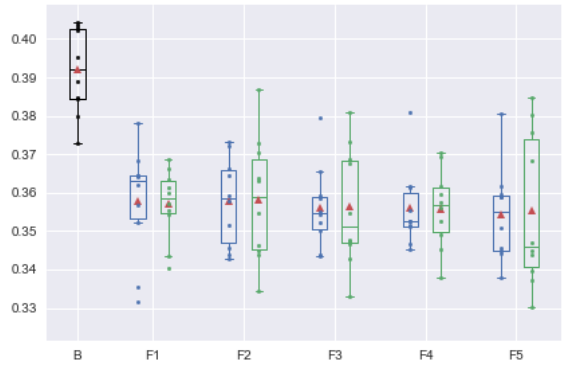
Detailed Results

The averaged evaluation metrics for the Ridge, kNN and GBT estimators are presented in tables A.1, A.3 and A.5, while figures A.1, A.3 and A.5 show boxplots of these metrics for the MOST validity range, and figures A.2, A.4 and A.6 do so for the whole dataset. Finally, tables A.2, A.4 and A.6 list the hyper-parameters obtained in each outer fold that generated the best results shown in table 4.8.

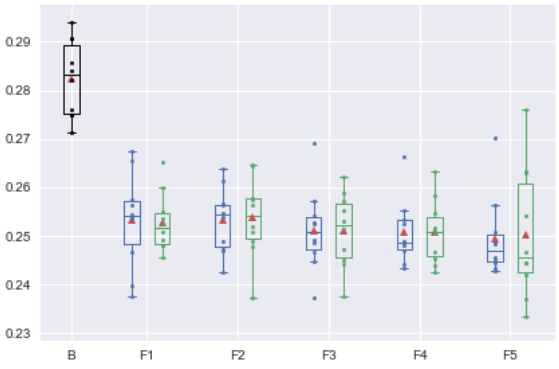


(a) Mean Squared Error

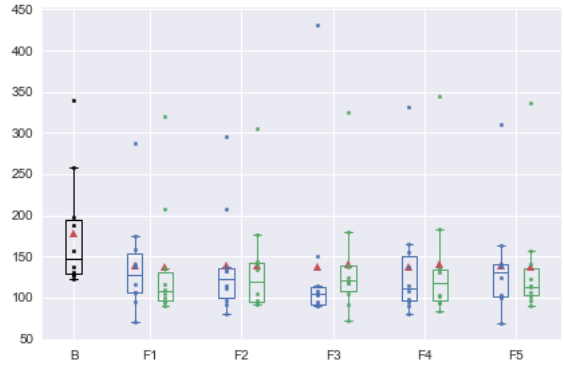
(b) R^2 Score



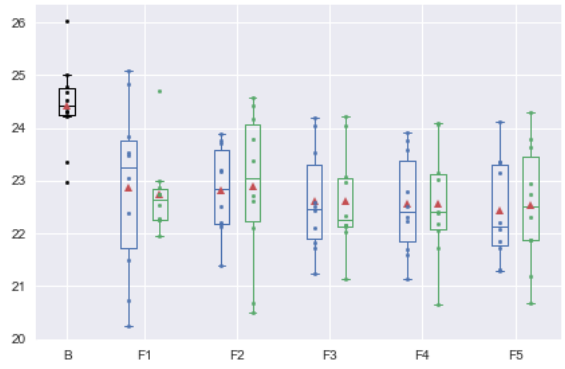
(c) Mean Absolute Error



(d) Median Absolute Error

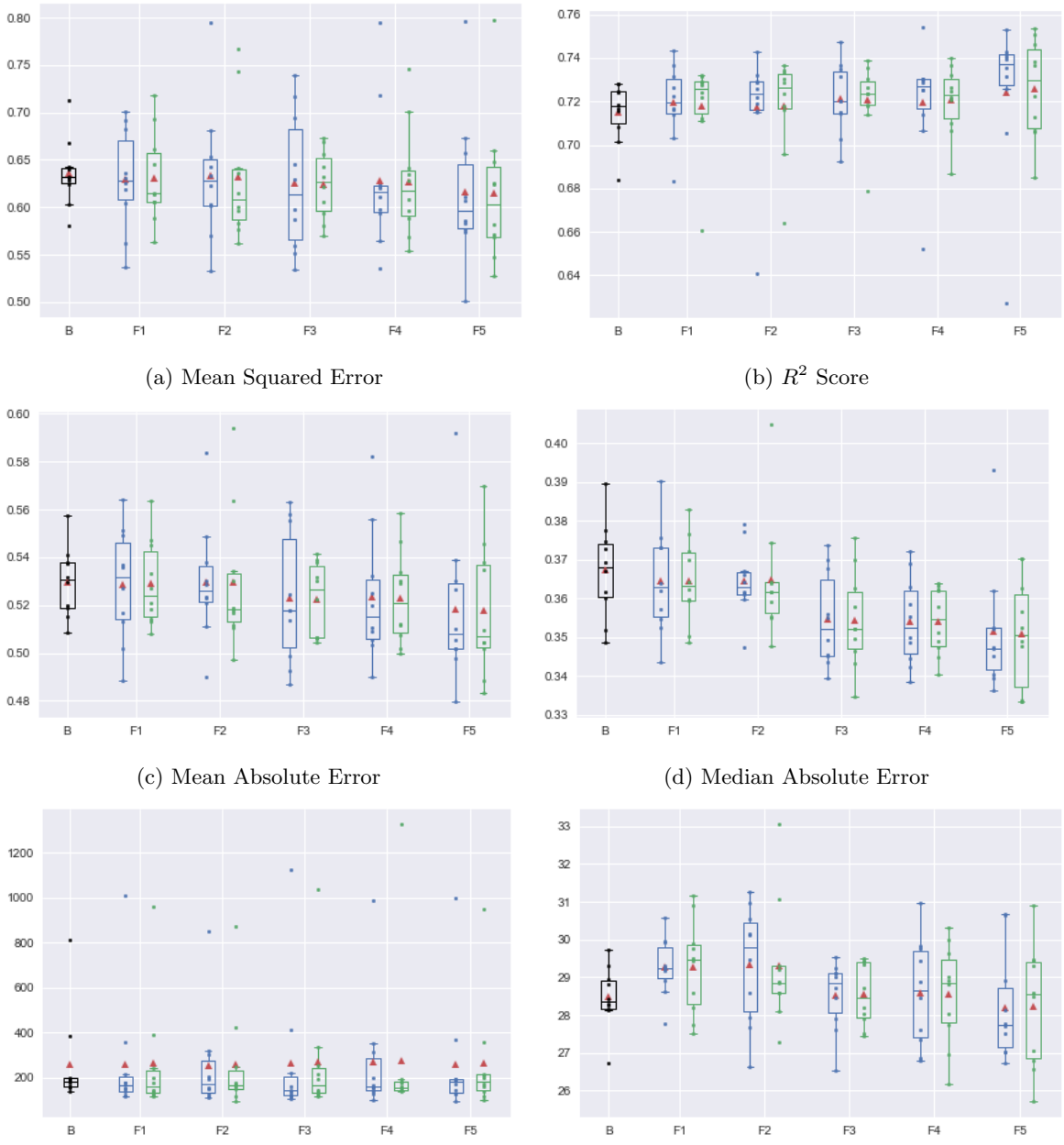


(e) Mean Absolute Percent Error



(f) Median Absolute Percent Error

Figure A.1: Evaluation metrics of the Ridge Regression Estimator in the MOST validity range; the feature sets with trend are blue, those without trend are green, and he leftmost black box is the MOST estimator baseline.



(a) Mean Squared Error

(b) R^2 Score

(c) Mean Absolute Error

(d) Median Absolute Error

(e) Mean Absolute Percent Error

(f) Median Absolute Percent Error

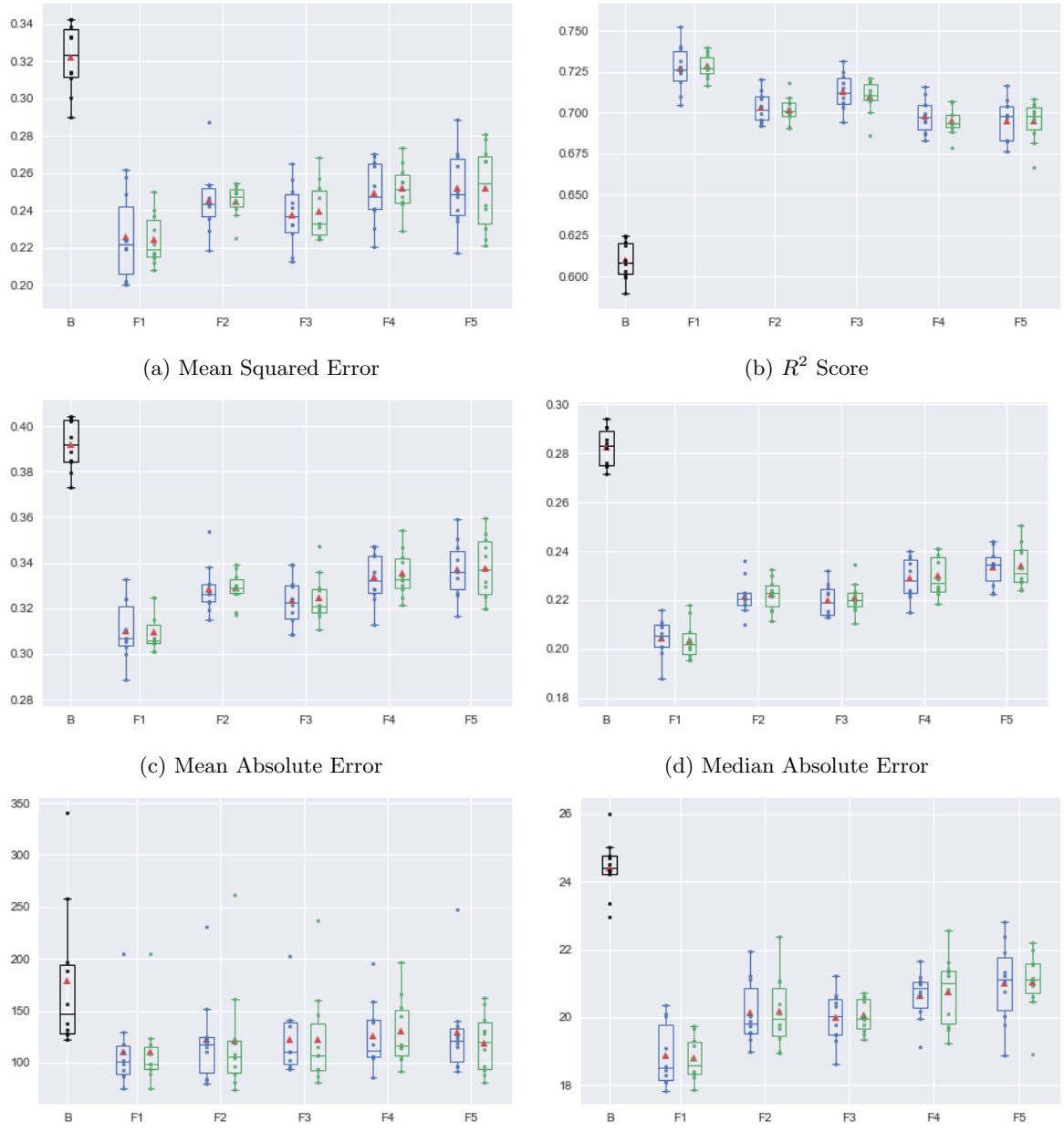
Figure A.2: Evaluation metrics of the Ridge Regression Estimator on all the data; the feature sets with trend are blue, those without trend are green, and he leftmost black box is the MOST estimator baseline.

Table A.1: Evaluation metrics for the Ridge estimator.

Metric	Dataset	Trend	F1	F2	F3	F4	F5
MSE	Full	N	0.63 (0.05)	0.63 (0.07)	0.62 (0.04)	0.63 (0.06)	0.62 (0.08)
		Y	0.63 (0.05)	0.63 (0.07)	0.63 (0.07)	0.63 (0.08)	0.62 (0.08)
	MOST	N	0.27 (0.02)	0.27 (0.03)	0.27 (0.02)	0.27 (0.02)	0.27 (0.03)
		Y	0.27 (0.02)	0.27 (0.02)	0.27 (0.01)	0.27 (0.02)	0.27 (0.02)
R^2	Full	N	0.72 (0.02)	0.72 (0.02)	0.72 (0.02)	0.72 (0.02)	0.73 (0.02)
		Y	0.72 (0.02)	0.72 (0.03)	0.72 (0.02)	0.72 (0.03)	0.72 (0.04)
	MOST	N	0.67 (0.02)	0.67 (0.03)	0.67 (0.02)	0.67 (0.02)	0.67 (0.02)
		Y	0.67 (0.02)	0.67 (0.01)	0.67 (0.02)	0.67 (0.02)	0.67 (0.01)
MAE	Full	N	0.53 (0.02)	0.53 (0.03)	0.52 (0.02)	0.52 (0.02)	0.52 (0.03)
		Y	0.53 (0.02)	0.53 (0.02)	0.52 (0.03)	0.52 (0.03)	0.52 (0.03)
	MOST	N	0.36 (0.01)	0.36 (0.02)	0.36 (0.02)	0.36 (0.01)	0.36 (0.02)
		Y	0.36 (0.01)	0.36 (0.01)	0.36 (0.01)	0.36 (0.01)	0.35 (0.01)
mAE	Full	N	0.36 (0.01)	0.36 (0.02)	0.35 (0.01)	0.35 (0.01)	0.35 (0.01)
		Y	0.36 (0.01)	0.36 (0.01)	0.35 (0.01)	0.35 (0.01)	0.35 (0.02)
	MOST	N	0.25 (0.01)	0.25 (0.01)	0.25 (0.01)	0.25 (0.01)	0.25 (0.01)
		Y	0.25 (0.01)	0.25 (0.01)	0.25 (0.01)	0.25 (0.01)	0.25 (0.01)
MAPE	Full	N	262.89 (258.33)	258.83 (235.59)	269.43 (278.64)	278.46 (368.15)	262.95 (252.08)
		Y	262.27 (271.17)	253.28 (222.37)	268.17 (314.75)	270.60 (265.59)	261.74 (270.25)
	MOST	N	137.37 (73.04)	138.23 (65.42)	140.78 (70.93)	140.17 (77.45)	137.71 (72.83)
		Y	139.18 (60.62)	139.73 (65.11)	137.92 (104.84)	137.57 (73.89)	138.81 (66.28)
mAPE	Full	N	29.27 (1.25)	29.30 (1.63)	28.55 (0.83)	28.56 (1.32)	28.23 (1.64)
		Y	29.28 (0.78)	29.34 (1.55)	28.53 (0.92)	28.60 (1.42)	28.21 (1.43)
	MOST	N	22.74 (0.77)	22.90 (1.46)	22.63 (0.95)	22.58 (1.05)	22.54 (1.17)
		Y	22.87 (1.63)	22.81 (0.85)	22.62 (1.00)	22.56 (0.96)	22.45 (0.98)

Table A.2: Hyper-parameters and MSE for each outer fold obtained by the best combination of trend and features in each dataset by the Ridge estimator.

Dataset	Features	Trend	Hyper-Param.	1	2	3	4	5	6	7	8	9	10
Full	**F5**	**Y**	**(MSE)**	0.501	0.583	0.576	0.610	0.586	0.607	0.796	0.575	0.657	0.673
			Alpha	7.423	0.007	3.197	1.007	5.845	3.796	0.981	0.041	3.181	2.143
MOST	**F3**	**Y**	**(MSE)**	0.266	0.293	0.265	0.267	0.302	0.270	0.268	0.249	0.268	0.277
			Alpha	4.598	2.839	8.458	0.566	4.173	4.536	0.547	5.280	6.344	0.105



(a) Mean Squared Error

(b) R^2 Score

(c) Mean Absolute Error

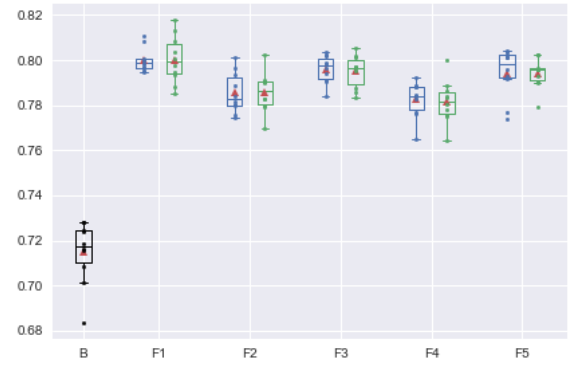
(d) Median Absolute Error

(e) Mean Absolute Percent Error

(f) Median Absolute Percent Error

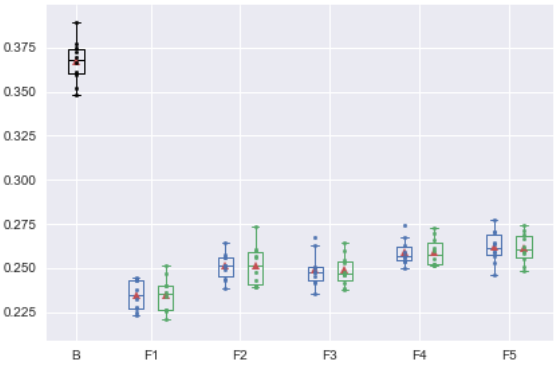
Figure A.3: Evaluation metrics of the k-Nearest Neighbors Regressor Estimator in the MOST validity range; the feature sets with trend are blue, those without trend are green, and he leftmost black box is the MOST estimator baseline.

(a) Mean Squared Error

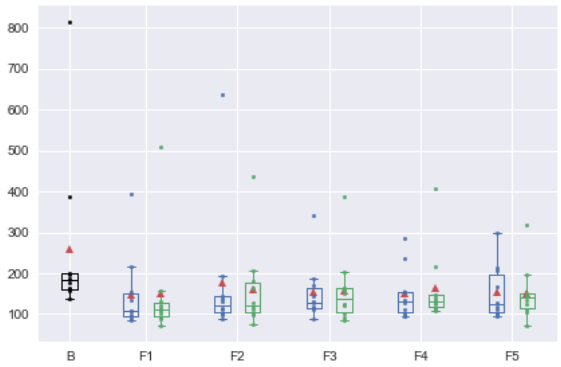


(b) R^2 Score

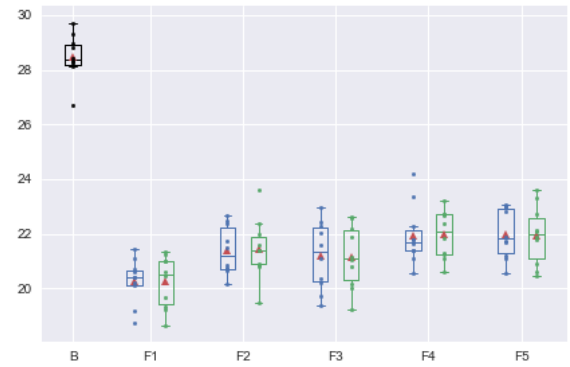
(c) Mean Absolute Error



(d) Median Absolute Error



(e) Mean Absolute Percent Error



(f) Median Absolute Percent Error

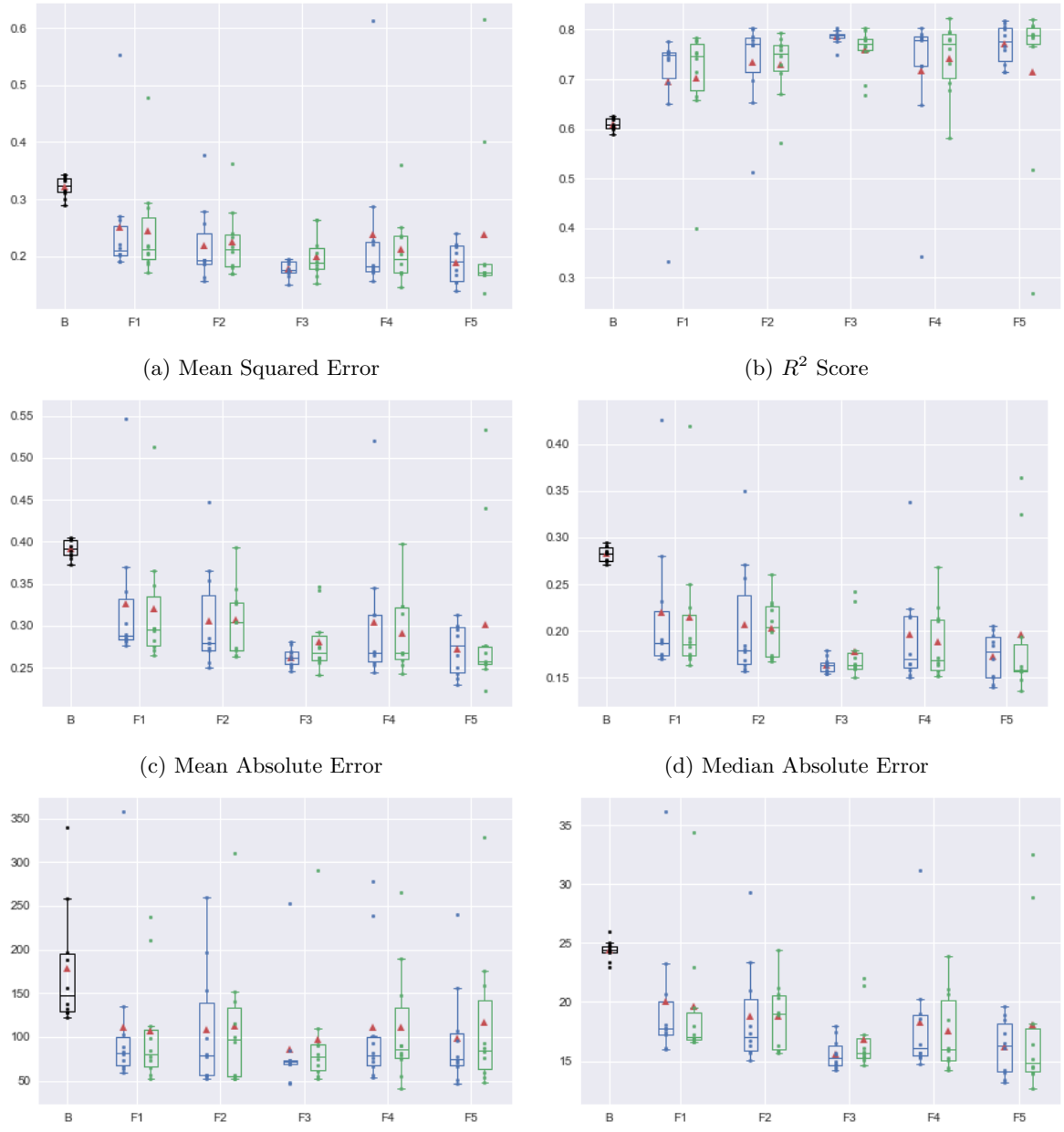
Figure A.4: Evaluation metrics of the k-Nearest Neighbors Regressor Estimator on all the data; the feature sets with trend are blue, those without trend are green, and he leftmost black box is the MOST estimator baseline.

Table A.3: Evaluation metrics for the k-Nearest Neighbors estimator.

Metric	Dataset	Trend	F1	F2	F3	F4	F5
MSE	Full	N	0.45 (0.05)	0.48 (0.04)	0.46 (0.04)	0.49 (0.04)	0.46 (0.05)
		Y	0.45 (0.02)	0.48 (0.03)	0.46 (0.03)	0.49 (0.04)	0.46 (0.05)
	MOST	N	0.22 (0.01)	0.25 (0.01)	0.24 (0.02)	0.25 (0.01)	0.25 (0.02)
		Y	0.23 (0.02)	0.25 (0.02)	0.24 (0.02)	0.25 (0.02)	0.25 (0.02)
R^2	Full	N	0.80 (0.01)	0.79 (0.01)	0.79 (0.01)	0.78 (0.01)	0.79 (0.01)
		Y	0.80 (0.01)	0.79 (0.01)	0.80 (0.01)	0.78 (0.01)	0.79 (0.01)
	MOST	N	0.73 (0.01)	0.70 (0.01)	0.71 (0.01)	0.69 (0.01)	0.69 (0.01)
		Y	0.73 (0.01)	0.70 (0.01)	0.71 (0.01)	0.70 (0.01)	0.70 (0.01)
MAE	Full	N	0.41 (0.02)	0.43 (0.02)	0.42 (0.02)	0.43 (0.01)	0.43 (0.02)
		Y	0.41 (0.01)	0.43 (0.01)	0.42 (0.02)	0.43 (0.02)	0.43 (0.02)
	MOST	N	0.31 (0.01)	0.33 (0.01)	0.32 (0.01)	0.34 (0.01)	0.34 (0.01)
		Y	0.31 (0.01)	0.33 (0.01)	0.32 (0.01)	0.33 (0.01)	0.34 (0.01)
mAE	Full	N	0.23 (0.01)	0.25 (0.01)	0.25 (0.01)	0.26 (0.01)	0.26 (0.01)
		Y	0.23 (0.01)	0.25 (0.01)	0.25 (0.01)	0.26 (0.01)	0.26 (0.01)
	MOST	N	0.20 (0.01)	0.22 (0.01)	0.22 (0.01)	0.23 (0.01)	0.23 (0.01)
		Y	0.20 (0.01)	0.22 (0.01)	0.22 (0.01)	0.23 (0.01)	0.23 (0.01)
MAPE	Full	N	149.67 (128.39)	162.01 (104.52)	158.82 (88.71)	163.85 (91.11)	150.99 (67.73)
		Y	147.68 (95.02)	176.15 (164.41)	153.01 (71.99)	149.90 (63.20)	153.62 (67.03)
	MOST	N	110.32 (35.89)	121.57 (54.94)	122.51 (47.61)	130.45 (33.21)	119.11 (29.13)
		Y	110.71 (36.57)	122.25 (44.17)	122.75 (33.56)	126.06 (32.63)	129.07 (44.68)
mAPE	Full	N	20.25 (0.97)	21.44 (1.09)	21.16 (1.15)	21.97 (0.88)	21.94 (1.07)
		Y	20.28 (0.82)	21.38 (0.89)	21.20 (1.24)	21.94 (1.08)	22.00 (0.92)
	MOST	N	18.79 (0.65)	20.18 (1.08)	20.06 (0.50)	20.77 (1.06)	21.06 (0.93)
		Y	18.89 (0.95)	20.14 (0.96)	19.99 (0.78)	20.66 (0.73)	21.02 (1.21)

Table A.4: Hyper-parameters and MSE for each outer fold obtained by the best combination of trend and features in each dataset by the k-Nearest Neighbors estimator.

Dataset	Features	Trend	Hyper-Param.	1	2	3	4	5	6	7	8	9	10
Full	**F1**	**Y**	k	36	40	23	45	33	43	19	41	30	24
			(MSE)	0.481	0.417	0.434	0.468	0.444	0.459	0.462	0.401	0.462	0.455
			Dist.	1	1	1	2	1	1	1	1	1	1
			Weight	Unif.	Dist.	Dist.	Dist.	Dist.	Dist.	Dist.	Dist.	Dist.	Unif.
MOST	**F2**	**N**	k	19	19	18	33	18	32	16	17	21	18
			(MSE)	0.255	0.241	0.237	0.249	0.254	0.244	0.249	0.225	0.245	0.252
			Dist.	1	1	2	1	1	1	1	1	1	2
			Weight	Unif.	Dist.	Dist.	Dist.	Dist.	Dist.	Dist.	Unif.	Dist.	Dist.



(a) Mean Squared Error

(b) R^2 Score

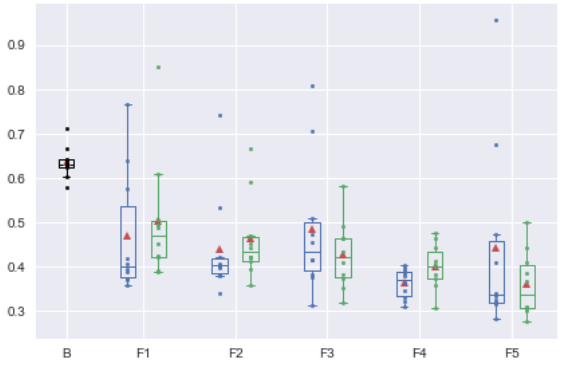
(c) Mean Absolute Error

(d) Median Absolute Error

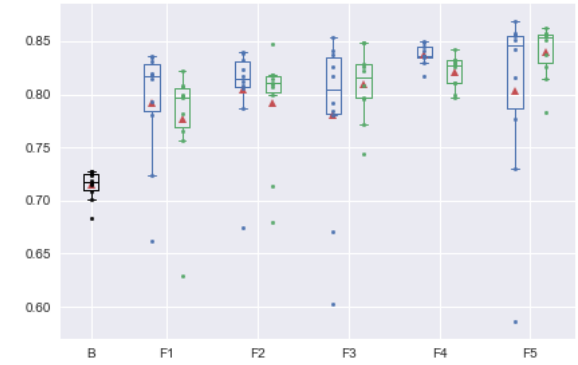
(e) Mean Absolute Percent Error

(f) Median Absolute Percent Error

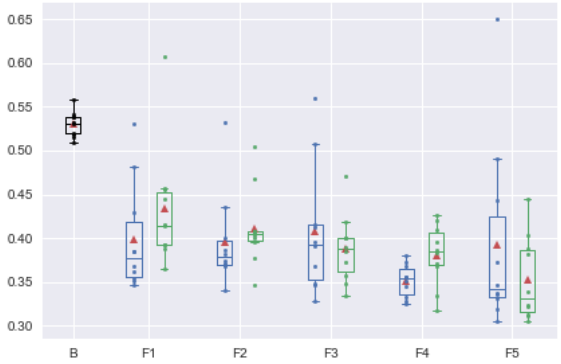
Figure A.5: Evaluation metrics of the Gradient Boosted Trees Estimator in the MOST validity range; the feature sets with trend are blue, those without trend are green, and he leftmost black box is the MOST estimator baseline.



(a) Mean Squared Error

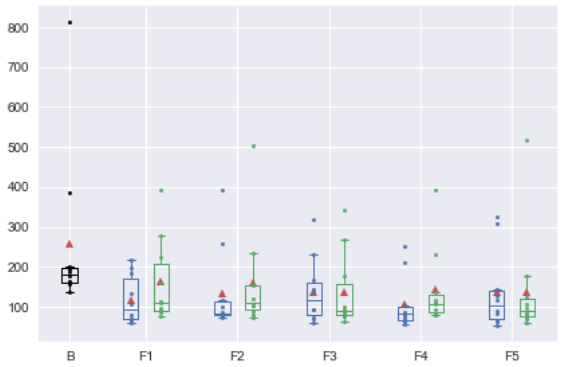


(b) R^2 Score



(c) Mean Absolute Error

(d) Median Absolute Error



(e) Mean Absolute Percent Error

(f) Median Absolute Percent Error

Figure A.6: Evaluation metrics of the Gradient Boosted Trees Estimator on all the data; the feature sets with trend are blue, those without trend are green, and he leftmost black box is the MOST estimator baseline.

Table A.5: Evaluation metrics for the Gradient Boosted Trees Estimator estimator.

Metric	Dataset	Trend	F1	F2	F3	F4	F5
MSE	Full	N	0.50 (0.14)	0.46 (0.09)	0.43 (0.08)	0.40 (0.05)	0.36 (0.07)
		Y	0.47 (0.14)	0.44 (0.12)	0.49 (0.16)	0.36 (0.03)	0.44 (0.21)
	MOST	N	0.25 (0.09)	0.22 (0.06)	0.20 (0.04)	0.21 (0.06)	0.24 (0.15)
		Y	0.25 (0.11)	0.22 (0.07)	0.18 (0.01)	0.24 (0.14)	0.19 (0.04)
R^2	Full	N	0.78 (0.06)	0.79 (0.05)	0.81 (0.03)	0.82 (0.02)	0.84 (0.03)
		Y	0.79 (0.06)	0.80 (0.05)	0.78 (0.08)	0.84 (0.01)	0.80 (0.09)
	MOST	N	0.70 (0.12)	0.73 (0.07)	0.76 (0.05)	0.74 (0.07)	0.71 (0.18)
		Y	0.70 (0.13)	0.73 (0.09)	0.79 (0.01)	0.72 (0.14)	0.77 (0.04)
MAE	Full	N	0.43 (0.07)	0.41 (0.04)	0.39 (0.04)	0.38 (0.04)	0.35 (0.05)
		Y	0.40 (0.06)	0.40 (0.05)	0.41 (0.07)	0.35 (0.02)	0.39 (0.11)
	MOST	N	0.32 (0.08)	0.31 (0.04)	0.28 (0.04)	0.29 (0.05)	0.30 (0.10)
		Y	0.33 (0.08)	0.31 (0.06)	0.26 (0.01)	0.30 (0.08)	0.27 (0.03)
mAE	Full	N	0.26 (0.07)	0.24 (0.04)	0.22 (0.03)	0.22 (0.04)	0.20 (0.04)
		Y	0.22 (0.04)	0.22 (0.04)	0.23 (0.06)	0.19 (0.02)	0.23 (0.09)
	MOST	N	0.21 (0.08)	0.20 (0.03)	0.18 (0.03)	0.19 (0.04)	0.20 (0.08)
		Y	0.22 (0.08)	0.21 (0.06)	0.16 (0.01)	0.20 (0.06)	0.17 (0.03)
MAPE	Full	N	162.52 (104.39)	161.47 (128.93)	137.02 (95.56)	143.56 (98.91)	138.85 (137.36)
		Y	118.51 (60.72)	134.46 (106.10)	138.86 (82.25)	107.69 (67.06)	138.16 (98.96)
	MOST	N	106.73 (64.65)	112.76 (78.44)	97.33 (70.36)	111.27 (69.51)	116.23 (85.87)
		Y	111.10 (89.45)	108.63 (71.50)	85.95 (59.66)	111.78 (79.21)	98.26 (58.87)
mAPE	Full	N	21.98 (4.51)	20.60 (2.29)	19.18 (2.93)	18.91 (2.24)	17.28 (3.20)
		Y	19.33 (2.90)	19.49 (2.43)	19.95 (4.57)	16.82 (1.21)	19.60 (6.28)
	MOST	N	19.59 (5.58)	18.84 (2.90)	16.84 (2.66)	17.52 (3.34)	18.08 (6.89)
		Y	20.04 (6.06)	18.85 (4.49)	15.62 (1.29)	18.23 (4.92)	16.18 (2.36)

Table A.6: Hyper-parameters and MSE for each outer fold obtained by the best combination of trend and features in each dataset by the Gradient Boosted Trees estimator.

Dataset	Features	Trend	Hyper-Param.	1	2	3	4	5	6	7	8	9	10
			(MSE)	0.322	0.391	0.379	0.404	0.331	0.311	0.347	0.364	0.386	0.399
			Alpha	0.990	0.403	0.856	0.107	0.835	0.688	0.914	0.872	0.331	0.852
			Learning Rate	0.018	0.024	0.017	0.059	0.018	0.044	0.046	0.054	0.045	0.065
Full	**F4**	**Y**	**Loss**	huber	ls	ls	huber	ls	ls	lad	huber	lad	huber
			Max. Depth	8	8	8	7	8	9	9	7	4	8
			Max. Features	0.813	0.210	0.793	0.802	0.322	0.560	0.415	0.373	0.904	0.782
			Num. Estimators	780	735	493	542	461	345	711	690	589	375
			Subsample	0.936	0.357	0.931	0.852	0.262	0.934	0.316	0.394	0.477	0.804
			(MSE)	0.149	0.172	0.172	0.187	0.191	0.178	0.194	0.164	0.173	0.195
			Alpha	0.446	0.722	0.133	0.231	0.089	0.189	0.678	0.052	0.963	0.224
			Learning Rate	0.045	0.075	0.085	0.041	0.012	0.031	0.014	0.097	0.050	0.016
MOST	**F3**	**Y**	**Loss**	ls	huber	ls	huber	lad	huber	ls	huber	lad	lad
			Max. Depth	8	9	8	9	7	8	9	9	8	9
			Max. Features	0.954	0.736	0.180	0.506	0.847	0.740	0.583	0.572	0.390	0.291
			Num. Estimators	391	100	790	400	385	528	336	793	498	509
			Subsample	0.960	0.331	0.454	0.813	0.555	0.348	0.958	0.717	0.550	0.412

Appendix B

Prediction with Deep Neural Networks

The reason why this is not included in the main report is that time and resources constraint prevented a formal and thorough evaluation of these results. That said, this approach is definitely promising and worth pursuing in the future.

Methods: The features were obtained in the same way as for the other estimators discussed in the main report. The validation set used all the samples belonging to 18 months, and a new random validation set was created for every experiment performed. The configuration of the networks was obtained manually:

- **Architecture:** Fully connected layers only, number of neurons halved at every layer. The first layer with 1 neuron was the output layer, and the first hidden layer had either 128, 256 or 512 neurons, resulting in 7 to 9 hidden layers;
- **Activation Function:** linear for the output layer, PReLU (He et al., 2015) for the hidden layers;
- **Regularization:** Dropout ($p = 0.5$, only when the first layer had 256 neurons or more);
- **Optimizer:** Adam (Kingma and Ba, 2014) with inital learning rate 10^{-3};
- **Batch Size:** 1024 (1165 batches per epoch);
- **Learning Rate Schedule:** Divide by 10 if the validation loss had not decreased by at least 10^{-4} in the last 10 epochs;
- **Early Stopping:** if the validation error did not improve in the last 25 epochs, or if the learning rate was reduced below 10^{-6}.

Table B.1: MSE obtained by neural networks compared to the MOST estimator and gradient boosted trees. The features included the trend.

Dataset	**MOST**	**GBT**	**F1**	**F2**	**F3**	**F4**	**F5**
Full	0.64	0.36	0.31	0.29	0.26	0.24	0.21
MOST	0.32	0.18	-	-	-	-	0.13

Results: The results are shown in table B.1. Neural networks show promising results, but one needs to keep in mind the anecdotal nature of these results, since they are a point estimate, and not an average of different validation scores.

Other experiments were performed using all the observations in the hour preceding the sample to predict, and using up to three GRU layers, followed by up to four fully connected layers, with the other parameters as detailed above. The performances were discouraging, with the network struggling to go below 0.15/0.20 training MSE and 0.7/0.8 validation MSE on F4 and full dataset.

Appendix C

Prediction with an Ensemble of Regularized Linear Models

We can relax the constraints needed to answer the research questions, in particular the second one, and perform additional feature engineering. In order to minimize the computational resources needed, we apply Ridge linear regression to the MOST dataset.

Methods: Instead of restricting ourselves to the five feature sets, we use all the quantities shown in table 4.1, except for the turbulent heat fluxes, and, in light of the results shown in the main report, we do not use the trend. We add polynomial features, then standardize the dataset, apply PCA, and fit the Ridge estimator. This whole procedure, including feature engineering, is performed separately for a number of random subsets of samples (with replacement) and features, and the final prediction is the average of the predictions obtained by the different models. We then optimize the parameters of every steps with random search and cross validation. This means that, although all the steps share the same hyper-parameters, they are fit on different subsets of the full dataset, and thus have different parameters. This improves generalization, since the best performing hyper-parameters were able to extract good features and use them to obtain good predictions on different independent subsets. Table C.1 shows the distribution of the hyper-parameters, which were optimized with 10-fold cross validation by sampling 250 values, as well as the optimal values found in the procedure.

Results: The average MSE obtained on the 10 validation folds is 0.187 with a standard deviation of 0.022, which is just slightly worse than the best results obtained by the gradient boosted trees on F3. All the other metrics are within one standard deviation, too.

Discussion: The linear models use all the available features, and the effect of PCA is only to decorrelate the features. However, they use few samples and weak regularization; in spite of this, the MSE

Table C.1: Distribution of the hyper-parameters used in the random search and optimal value found.

	Hyper-Parameter	Distribution	Values	Optimal
Base Estimator	Polynomial Degree	Uniform	1 or 2	2
	Polynomial Interactions	Uniform	Yes or No	Yes
	PCA Target Dimension	Uniform	From 2 to 19	19
	Ridge Regularization	Log_{10}-uniform	From 10^{-7} to 10^{1}	2×10^{-5}
Bagging Estimator	Number of Base Estimators	Uniform	From 5 to 100	13
	Samples per Base Estimator	Uniform	From 5% to 25%	11%
	Features per Base Estimator	Uniform	From 4 to 20	20

on the training folds is almost identical to the MSE on the validation folds, indicating that the linear models do not overfit and suggesting that the ensemble is not needed to reduce variance.

Appendix D

More Hyper-parameter Optimization

By foregoing nested cross-validation and focusing on a particular dataset (MOST) and set of features (table 4.1 without H and λE), we can reduce the computational resources needed for more involved hyper-parameter tuning procedures.

Methods: We optimized the hyper-parameters of extreme gradient boosted trees (Chen and Guestrin, 2016) using both HyperBand (Li et al., 2017) and the Python library GPyOpt[1]. For Hyperband, we use the default parameters suggested in the paper, i.e. $\eta = 3$ and $R = 81$, with each unit of budget corresponding to 15 trees. The Bayesian optimization procedure is based on Gaussian Processes with the RBF kernel, using the integrated expected improvement as acquisition function. Integration is performed over the posterior of the hyper-parameters, approximated with MCMC sampling (Snoek et al., 2012), in order to improve the accuracy of the proposed hyper-parameters. The dataset was randomly split in three smaller sets: a training set of 162 months, a validation set of 20 months, and a test set of 20 months. The training set was used to train the models, and the hyper-parameter optimization procedure optimized the scores on the validation set; the test set is used at the end to evaluate the optimal configuration. Table D.1 shows the distribution of the hyper-parameters used in the search, and the optimal values found by the two methods.

[1] `http://github.com/SheffieldML/GPyOpt`

Table D.1: Distribution of the hyper-parameters used and best values found by the two optimization procedures.

Hyper-parameter	**Distribution**	**Values**	**Best (Hyperband)**	**Best (GPyOpt)**
Number of Trees	Log_{10}-Uniform	From 10^1 to 10^3	1000	1000
Learning Rate	Log_{10}-Uniform	From 10^{-4} to 10^0	8.86×10^{-3}	4.34×10^{-2}
Max. Depth	Uniform	From 2 to 12	12	12
Min. Child Weight	Uniform	From 1 to 10	9	10
Subsample	Uniform	From 10% to 100%	64.6%	49.3%
γ (Min. loss decrease)	Uniform	From 0 to 1	0.194	0
α (L1 regularization)	Log_{10}-Uniform	From 10^{-20} to 10^0	1.87×10^{-11}	1.70×10^{-7}
λ (L2 regularization)	Log_{10}-Uniform	From 10^{-20} to 10^0	2.46×10^{-10}	5.96×10^{-5}
Number of features	Uniform	From 10% to 100%	22.7%	12.5%
Loss	Uniform	MSE or MAE	MAE	MAE

Results: The MSEs of the best configuration found by Hyperband, after 200 steps, were 0.08250, 0.1613, and 0.1676 on the training set, validation set, and test set respectively. The scores of best configuration found by GPyOpt, after 18 steps, were 0.04756, 0.1606, and 0.1670. These results are very similar to what was obtained with random search and nested cross validation.

Bibliography

Algina, J., Keselman, H. J. and Penfield, R. D. (2006), 'Confidence intervals for an effect size when-variances are not equal', *Journal of Modern Applied StatisticalMethods* **1**(2), 2–13.

Arlot, S., Celisse, A. et al. (2010), 'A survey of cross-validation procedures for model selection', *Statistics surveys* **4**, 40–79.

Baas, P., Van de Wiel, B., van der Linden, S. and C. Bosveld, F. (2017), 'From near-neutral to strongly stratified: Adequately modelling the clear-sky nocturnal boundary layer at cabauw', **166**.

Babington, P. (2000), *Handbook for the Meteorological Observation*, Koninklijk Nederlands Meteorologisch Instituut.

Basu, S. and Lacser, A. (2017), 'A cautionary note on the use of monin–obukhov similarity theory in very high-resolution large-eddy simulations', *Boundary-Layer Meteorology* **163**(2), 351–355.
URL: *https://doi.org/10.1007/s10546-016-0225-y*

Beljaars, A. C. M. (1982), 'The derivation of fluxes from profiles in perturbed areas', *Boundary-Layer Meteorology* **24**(1), 35–55.
URL: *https://doi.org/10.1007/BF00121798*

Bengio, Y. and Grandvalet, Y. (2004), No unbiased estimator of the variance of k-fold cross-validation, *in* S. Thrun, L. K. Saul and B. Schölkopf, eds, 'Advances in Neural Information Processing Systems 16', MIT Press, pp. 513–520.

Bergstra, J., Bardenet, R., Bengio, Y. and Kégl, B. (2011), Algorithms for hyper-parameter optimization, *in* 'Proceedings of the 24th International Conference on Neural Information Processing Systems', NIPS'11, Curran Associates Inc., USA, pp. 2546–2554.

Bergstra, J. and Bengio, Y. (2012), 'Random search for hyper-parameter optimization', *J. Mach. Learn. Res.* **13**(1), 281–305.

Bishop, C. M. (2006), *Pattern Recognition and Machine Learning (Information Science and Statistics)*, Springer-Verlag New York, Inc., Secaucus, NJ, USA.

Bonett, D. G. (2008), 'Confidence intervals for standardized linear contrasts of means.', *Psychological Methods* **13**(2), 99–109.

Bosveld, F. (2014), 'A quality controlled and gap-filled data set for cabauw for the period 2001 - today'.

Bosveld, F. C. (2018), Cabauw in-situ observational program 2000 - now: Instruments, calibrations and set-up, Technical report, Koninklijk Nederlands Meteorologisch Instituut.

Bouwman, P. (1990), Flux-profile relationships in the nocturnal boundary layer, Technical report, Koninklijk Nederlands Meteorologisch Instituut.

Braam, M. (2008), Determination of the surface sensible heat flux from the structure parameter of temperature at 60 m height during day-time, Technical Report TR-303, Koninklijk Nederlands Meteorologisch Instituut.

Breiman, L. (1997), Arcing the edge, Technical Report 486, Statistics Department, University of California, Berkeley.

Chapman, P., Clinton, J., Kerber, R., Khabaza, T., Reinartz, T., Shearer, C. and Wirth, R. (2000), Crisp-dm 1.0 - step-by-step data mining guide.

Chen, T. and Guestrin, C. (2016), Xgboost: A scalable tree boosting system, *in* 'Proceedings of the 22Nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining', KDD '16, ACM, New York, NY, USA, pp. 785–794.

Chipman, H., George, E. I. and McCulloch, R. E. (2001), *The Practical Implementation of Bayesian Model Selection*, Vol. Volume 38 of *Lecture Notes–Monograph Series*, Institute of Mathematical Statistics, Beachwood, OH, pp. 65–116.

Dimitrova, R., Silver, Z., Zsedrovits, T., Hocut, C. M., Leo, L. S., Di Sabatino, S. and Fernando, H. J. S. (2016), 'Assessment of planetary boundary-layer schemes in the weather research and forecasting mesoscale model using materhorn field data', *Boundary-Layer Meteorology* **159**(3), 589–609.

Duchi, J., Hazan, E. and Singer, Y. (2011), 'Adaptive subgradient methods for online learning and stochastic optimization', *J. Mach. Learn. Res.* **12**, 2121–2159.

Efron, B. and Tibshirani, R. J. (1993), *An Introduction to the Bootstrap*, number 57 *in* 'Monographs on Statistics and Applied Probability', Chapman & Hall/CRC, Boca Raton, Florida, USA.

Fayyad, U. M., Piatetsky-Shapiro, G. and Smyth, P. (1996), Advances in knowledge discovery and data mining, American Association for Artificial Intelligence, Menlo Park, CA, USA, chapter From Data Mining to Knowledge Discovery: An Overview, pp. 1–34.

Foken, T. (2006), '50 years of the monin–obukhov similarity theory', *Boundary-Layer Meteorology* **119**(3), 431–447.

Friedman, J. H. (2001), 'Greedy function approximation: A gradient boosting machine.', *Ann. Statist.* **29**(5), 1189–1232.

Gainsford, P. (2006), *HFP01SC - Self Calibrating Heat Flux Sensor*, Hukseflux Thermal Sensors.

Geisser, S. (1975), 'The predictive sample reuse method with applications', *Journal of the American Statistical Association* **70**(350), 320–328.

Gitman, I., Dilipkumar, D. and Parr, B. (2018), 'Convergence Analysis of Gradient Descent Algorithms with Proportional Updates', *ArXiv e-prints* .

Grachev, A. A., Andreas, E. L., Fairall, C. W., Guest, P. S. and Persson, P. O. G. (2007), 'Sheba flux–profile relationships in the stable atmospheric boundary layer', *Boundary-Layer Meteorology* **124**(3), 315–333.

Hatfield, J., Baker, J., Prueger, J. H. and Kustas, W. P. (2005), Aerodynamic methods for estimating turbulent fluxes, *in* 'Micrometeorology in Agricultural Systems', American Society of Agronomy, Crop Science Society of America, and Soil Science Society of America.
URL: *https://doi.org/10.2134/agronmonogr47.c18*

He, K., Zhang, X., Ren, S. and Sun, J. (2015), 'Delving deep into rectifiers: Surpassing human-level performance on imagenet classification', *CoRR* **abs/1502.01852**.

Högström, U. (1988), 'Non-dimensional wind and temperature profiles in the atmospheric surface layer: A re-evaluation', *Boundary-Layer Meteorology* **42**(1), 55–78.

Högström, U. (1996), 'Review of some basic characteristics of the atmospheric surface layer', *Boundary-Layer Meteorology* **78**(3), 215–246.

Holtslag, A. (1987), Surface fluxes and boundary layer scaling: models and applications, Technical report, Koninklijk Nederlands Meteorologisch Instituut.

Holtslag, A. A. M. and Bruin, H. A. R. D. (1988), 'Applied modeling of the nighttime surface energy balance over land', *Journal of Applied Meteorology* **27**(6), 689–704.

Holtslag, A. A. M. and van Westrhenen, R. M. (1991), *Diagnostic Derivation of Boundary Layer Parameters from the Outputs of Atmospheric Models*, Springer US, Boston, MA, pp. 329–337.

Hunter, J. D. (2007), 'Matplotlib: A 2d graphics environment', *Computing In Science & Engineering* **9**(3), 90–95.

Hurley, P. and Luhar, A. (2009), 'Modelling the meteorology at the cabauw tower for 2005', *Boundary-Layer Meteorology* **132**(1), 43–57.

Ismail, M., Gebremeskel, E., Kakantousis, T., Berthou, G. and Dowling, J. (2017), Hopsworks: Improving user experience and development on hadoop with scalable, strongly consistent metadata, *in* '2017 IEEE 37th International Conference on Distributed Computing Systems (ICDCS)', pp. 2525–2528.

JW Verkaik, A. H. (2006), 'Wind profiles, momentum fluxes and roughness lengths at cabauw revisited'.

Kingma, D. P. and Ba, J. (2014), 'Adam: A method for stochastic optimization', *CoRR* **abs/1412.6980**.

Kluyver, T., Ragan-Kelley, B., Pérez, F., Granger, B., Bussonnier, M., Frederic, J., Kelley, K., Hamrick, J., Grout, J., Corlay, S., Ivanov, P., Avila, D., Abdalla, S. and Willing, C. (2016), Jupyter notebooks – a publishing format for reproducible computational workflows, *in* F. Loizides and B. Schmidt, eds, 'Positioning and Power in Academic Publishing: Players, Agents and Agendas', IOS Press, pp. 87 – 90.

Kohavi, R. (1995), A study of cross-validation and bootstrap for accuracy estimation and model selection, *in* 'Proceedings of the 14th International Joint Conference on Artificial Intelligence - Volume 2', IJCAI'95, Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, pp. 1137–1143.

Korrell, A., Panosky, H. and Rossi, R. (1981), 'Wind profiles at the boulder tower'.

Krstajic, D., Buturovic, L. J., Leahy, D. E. and Thomas, S. (2014), 'Cross-validation pitfalls when selecting and assessing regression and classification models', *Journal of Cheminformatics* **6**(1), 10.
URL: *https://doi.org/10.1186/1758-2946-6-10*

Lee, X., Massman, W. and Law, B. (2004), *Handbook of Micrometeorology: A Guide for Surface Flux Measurement and Analysis*, Vol. 29 of *Atmospheric and Oceanographic Sciences Library*, Kluwer Academic Publishers.

Li, L., Jamieson, K., DeSalvo, G., Rostamizadeh, A. and Talwalkar, A. (2017), 'Hyperband: Bandit-based configuration evaluation for hyperparameter optimization', *ICLR 2017: 5th International Conference on Learning Representations* .

Loescher, H. W., Law, B. E., Mahrt, L., Hollinger, D. Y., Campbell, J. and Wofsy, S. C. (2006), 'Uncertainties in, and interpretation of, carbon flux estimates using the eddy covariance technique', *Journal of Geophysical Research: Atmospheres* **111**(D21).

Mason, L., Baxter, J., Bartlett, P. and Frean, M. (1999), Boosting algorithms as gradient descent, *in* 'Proceedings of the 12th International Conference on Neural Information Processing Systems', NIPS'99, MIT Press, Cambridge, MA, USA, pp. 512–518.

McHutchon, A. (2013), Differentiating gaussian processes.
URL: *http://mlg.eng.cam.ac.uk/mchutchon/DifferentiatingGPs.pdf*

McKinney, W. (2010), Data structures for statistical computing in python, *in* S. van der Walt and J. Millman, eds, 'Proceedings of the 9th Python in Science Conference', pp. 51 – 56.

Mcnaughton, K. G. (2006), 'On the kinetic energy budget of the unstable atmospheric surface layer', *Boundary-Layer Meteorology* **118**(1), 83–107.

Meijer, E. (2000), Evaluation of humidity and temperature measurement of vaisala's hmp243 plus pt100 with two reference psychrometers, Technical report, Koninklijk Nederlands Meteorologisch Instituut.

Meurer, A., Smith, C. P., Paprocki, M., Čertík, O., Kirpichev, S. B., Rocklin, M., Kumar, A., Ivanov, S., Moore, J. K., Singh, S., Rathnayake, T., Vig, S., Granger, B. E., Muller, R. P., Bonazzi, F., Gupta, H., Vats, S., Johansson, F., Pedregosa, F., Curry, M. J., Terrel, A. R., Roučka, v., Saboo, A., Fernando, I., Kulal, S., Cimrman, R. and Scopatz, A. (2017), 'Sympy: symbolic computing in python', *PeerJ Computer Science* **3**, e103.

Monna, W. (1978), Comparative investigation of dynamic properties of some propeller vanes, Technical report, Koninklijk Nederlands Meteorologisch Instituut.

Neelakantan, A., Vilnis, L., Le, Q. V., Sutskever, I., Kaiser, L., Kurach, K. and Martens, J. (2015), 'Adding gradient noise improves learning for very deep networks', *CoRR* **abs/1511.06807**.

Nieuwstadt, F. (1978), 'The computation of the friction velocity u* and the temperature scale t* from temperature and wind velocity profiles by least-square methods', *Boundary-Layer Meteorology* **14**(2), 235–246.
URL: *https://doi.org/10.1007/BF00122621*

Nieuwstadt, F. T. M. (1984), 'The turbulent structure of the stable, nocturnal boundary layer', **41**, 2202–2216.

Nocedal, J. and Wright, S. J. (1999), *Numerical Optimization*, Springer, New York, NY, USA.

Obukhov, A. M. (1971), 'Turbulence in an atmosphere with a non-uniform temperature', *Boundary-Layer Meteorology* **2**(1), 7–29.
URL: *https://doi.org/10.1007/BF00718085*

Optis, M., Monahan, A. and C. Bosveld, F. (2014), 'Moving beyond monin-obukhov similarity theory in modelling wind-speed profiles in the lower atmospheric boundary layer under stable stratification', **153**, 497–514.

Pedregosa, F., Varoquaux, G., Gramfort, A., Michel, V., Thirion, B., Grisel, O., Blondel, M., Prettenhofer, P., Weiss, R., Dubourg, V., Vanderplas, J., Passos, A., Cournapeau, D., Brucher, M., Perrot, M. and Duchesnay, E. (2011), 'Scikit-learn: Machine learning in python', *J. Mach. Learn. Res.* **12**, 2825–2830.

Pérez, F. and Granger, B. E. (2007), 'IPython: a system for interactive scientific computing', *Computing in Science and Engineering* **9**(3), 21–29.

Ronda, R. J., Haarsma, R. J. and Holtslag, A. A. M. (2003), 'Representing the atmospheric boundary layer in climate models of intermediate complexity', *Climate Dynamics* **21**(3), 327–335.

Ruder, S. (2016), 'An overview of gradient descent optimization algorithms', *CoRR* **abs/1609.04747**.

Rumelhart, D. E., Hinton, G. E. and Williams, R. J. (1986), Parallel distributed processing: Explorations in the microstructure of cognition, vol. 1, MIT Press, Cambridge, MA, USA, chapter Learning Internal Representations by Error Propagation, pp. 318–362.

Scorer, R. S. (1992), 'Atmospheric data analysis, roger daley, cambridge atmospheric and space science series, cambridge university press, cambridge, 1991. no. of pages: xiv + 457. isbn 0521 382157', *International Journal of Climatology* **12**(7), 763–764.

Shieh, G. (2013), 'Confidence intervals and sample size calculations for the standardized mean difference effect size between two normal populations under heteroscedasticity', **45**.

Smith, S. L. and Le, Q. V. (2017), 'A bayesian perspective on generalization and stochastic gradient descent', *CoRR* **abs/1710.06451**.

Snoek, J., Larochelle, H. and Adams, R. P. (2012), Practical bayesian optimization of machine learning algorithms, *in* 'Proceedings of the 25th International Conference on Neural Information Processing Systems - Volume 2', NIPS'12, Curran Associates Inc., USA, pp. 2951–2959.
URL: *http://dl.acm.org/citation.cfm?id=2999325.2999464*

Standards for Reporting on Empirical Social Science Research in AERA Publications (2006), *Educational Researcher* **35**(6), 33–40.

Steeneveld, G. J., Holtslag, A. A. M. and Debruin, H. A. R. (2005), 'Fluxes and gradients in the convective surface layer and the possible role of boundary-layer depth and entrainment flux', *Boundary-Layer Meteorology* **116**(2), 237–252.

Stone, C. (1977), 'Consistent nonparametric regression', *The Annals of Statistics* **5**(4), 595–620.

Stone, M. (1974), 'Cross-validatory choice and assessment of statistical predictions', *Journal of the royal statistical society. Series B (Methodological)* pp. 111–147.

Strobl, C., Boulesteix, A.-L., Kneib, T., Augustin, T. and Zeileis, A. (2008), 'Conditional variable importance for random forests', *BMC Bioinformatics* **9**(1), 307.

Strobl, C., Boulesteix, A.-L., Zeileis, A. and Hothorn, T. (2007), 'Bias in random forest variable importance measures: Illustrations, sources and a solution', *BMC Bioinformatics* **8**(1), 25.

Unidata (2018), 'Network common data form (netcdf), version 4.6.0 [software]'.

van der Maaten, L. and Hinton, G. (2008), 'Visualizing data using t-SNE', *Journal of Machine Learning Research* **9**, 2579–2605.

Varma, S. and Simon, R. (2006), 'Bias in error estimation when using cross-validation for model selection', *BMC Bioinformatics* **7**(1), 91.
URL: *https://doi.org/10.1186/1471-2105-7-91*

Vinnichenko, N. K. (1970), 'The kinetic energy spectrum in the free atmosphere1 second to 5 years', *Tellus* **22**(2), 158–166.

Walt, S. v. d., Colbert, S. C. and Varoquaux, G. (2011), 'The numpy array: A structure for efficient numerical computation', *Computing in Science and Engg.* **13**(2), 22–30.

Wasserman, L. (2000), 'Bayesian model selection and model averaging', *Journal of Mathematical Psychology* **44**(1), 92 – 107.

Wauben, W. (2004), 'Precipitation amount and intensity measurements using a windscreen'.

Weber, R. O. (1999), 'Remarks on the definition and estimation of friction velocity', *Boundary-Layer Meteorology* **93**(2), 197–209.

Wessels, H. (1983), Distortion of the wind field by the cabauw meteorological tower, Technical report, Koninklijk Nederlands Meteorologisch Instituut.

Wilcox, R. R. (2010), *Fundamentals of Modern Statistical Methods: Substantially Improving Power and Accuracy*, 2nd edn, Springer New York.

Wilcox, R. R. and Keselman, H. J. (2003), 'Modern robust data analysis methods: Measures of central tendency.', *Psychological Methods* **8**(3), 254–274.

Wilkinson, L. (1999), 'Statistical methods in psychology journals: Guidelines and explanations.', *American Psychologist* **54**(8), 594–604.

Wilson, J. D. (2008), 'Monin-obukhov functions for standard deviations of velocity', *Boundary-Layer Meteorology* **129**(3), 353–369.

Yage, C., Viana, S., Maqueda, G. and Redondo, J. M. (2006), Influence of stability on the flux-profile relationships for wind speed,?m, and temperature,?h, for the stable atmospheric boundary layer, *in* 'Nonlinear Processes in Geophysics', Vol. 13, European Geosciences Union (EGU), pp. 185–203.

Yaglom, A. M. (1977), 'Comments on wind and temperature flux-profile relationships', *Boundary-Layer Meteorology* **11**(1), 89–102.

Zaharia, M., Xin, R. S., Wendell, P., Das, T., Armbrust, M., Dave, A., Meng, X., Rosen, J., Venkataraman, S., Franklin, M. J., Ghodsi, A., Gonzalez, J., Shenker, S. and Stoica, I. (2016), 'Apache spark: A unified engine for big data processing', *Commun. ACM* **59**(11), 56–65.

Zeiler, M. D. (2012), 'Adadelta: An adaptive learning rate method', *CoRR* **abs/1212.5701**.

Zhang, Y. and Yang, Y. (2015), 'Cross-validation for selecting a model selection procedure', *Journal of Econometrics* **187**(1), 95–112.