NATIONAL AND KAPODISTRIAN UNIVERSITY OF ATHENS

MASTER THESIS

Large scale corn phenological stage prediction with random forests - The US corn belt

Author:
Evangelos PANAGIOTOPOULOS

Supervisor:
Samis TREVEZAS

Department of Mathematics

03-10-2023

NATIONAL AND KAPODISTRIAN UNIVERSITY OF ATHENS

Abstract

Fucalty of Science

Department of Mathematics

Master In Statistics And Operational Research

Large scale corn phenological stage prediction with random forests - The US corn belt

by Evangelos PANAGIOTOPOULOS

This thesis aims to address a problem from the agricultural sciences, namely, predicting corn's phenological stage percentages using large-scale data. Current state-of-the-art research on the problem includes Hidden-Markov Models and Generalized Linear Mixed-Effects Models. In this thesis, the problem is viewed from a machine learning perspective. In particular, we investigate how the Random Forest (RF) algorithm as well as some of its variants can be implemented in our case.

We first introduce the problem and contextualize it within the machine learning framework. We then study the induction of decision trees, the building block of RF, covering both the univariate and the multivariate case. Furthermore, we describe the Random Forest algorithm and present different sampling, splitting, and aggregation options, including subject-level bootstrapping, Extremely Randomized Trees, and Historical Random Forests. Finally, we compare their results with each other upon the specific task of predicting the phenological stage percentages of corn crops in the USA and, more specifically, in the state of Nebraska.

Περίληψη

Πρόβλεψη των φαινολογικών σταδίων καλαμποκιού με χρήση τυχαίων δασών και δεδομένων μεγάλης κλίμακας - Η ζώνη καλαμποκιού των ΗΠΑ

Αυτή η διπλωματική εργασία στοχεύει να αντιμετωπίσει ένα πρόβλημα από τις γεωργικές επιστήμες, δηλαδή, την πρόβλεψη των ποσοστών των φαινολογικών σταδίων του καλαμποκιού χρησιμοποιώντας δεδομένα μεγάλης κλίμακας. Σύγχρονη έρευνα γύρω από το πρόβλημα περιλαμβάνει τα Κρυμμένα Μαρκοβιανά Μοντέλα και τα Γενικευμένα Γραμμικά Μοντέλα Τυχαίων Επιδράσεων. Σε αυτή τη διπλωματική εργασία, το πρόβλημα αντιμετωπίζεται από τη σκοπιά της μηχανικής μάθησης. Συγκεκριμένα, διερευνούμε πώς ο αλγόριθμος των Τυχαίων Δασών (ΤΔ) καθώς και ορισμένες από τις παραλλαγές του μπορούν να εφαρμοστούν στην περίπτωσή μας.

Αρχικά περιγράφουμε το υπό μελέτη πρόβλημα και το εντάσσουμε στο πλαίσιο της μηχανικής μάθησης. Στη συνέχεια, μελετάμε την κατασκευή των δέντρων απόφασης, των δομικών στοιχείων των ΤΔ, τόσο για τη μονομεταβλητή όσο και για την πολυμεταβλητή περίπτωση. Επιπλέον, περιγράφουμε τον αλγόριθμο των Τυχαίων Δασών και παρουσιάζουμε διαφορές επιλογές για τη δειγματοληψία, τον διαχωρισμό και τη συνάθροιση, συμπεριλαμβανομένου του bootstrapping σε επίπεδο υποκειμένων, τα Άκρως Τυχαία Δέντρα και τα Ιστορικά Τυχαία Δάση. Τέλος, συγκρίνουμε τα αποτελέσματά τους μεταξύ τους ως προς την ικανότητα πρόβλεψης των ποσοστών φαινολογικού σταδίων των καλλιεργειών καλαμποκιού στις ΗΠΑ και, πιο συγκεκριμένα, στην πολιτεία της Νεμπράσκα.

Acknowledgements

As this Master's thesis comes to an end, I would like to express my heartfelt gratitude to Professor Samis Trevezas for his unwavering support, invaluable guidance, unending patience, and steadfast trust in me throughout the journey of this work. His mentorship has been instrumental in shaping my research and academic growth, and I am truly grateful for the wisdom he shared.

I extend my sincere thanks to Ioannis Oikonomidis for his generosity in providing the essential data and sharing a portion of the code, as well as for his insightful advice regarding the intricate aspects of the problem under investigation. His contributions significantly enriched the quality of this research.

Finally, I dedicate this thesis to my beloved parents, who have passed away. Their love, encouragement, and unwavering belief in my abilities have been the driving force behind my academic pursuits. Though they are no longer with us, their memory continues to inspire me every day. This work is a tribute to their enduring love and support, and I hope it would make them proud.

Contents

Introduction

People have always considered it necessary to be able to both make accurate predictions and extract knowledge from data. This twofold objective was most often either a result of the desire to improve the quality of life or practical, daily life issues. In recent years, the field of machine learning has increasingly met this need. The term machine learning was first introduced in 1959 by Arthur Samuel, an IBM employee and pioneer in the field of computer gaming and artificial intelligence. Today, machine learning is considered a branch of artificial intelligence encompassing various techniques and algorithms that leverage sample data to generate precise predictions in a computationally efficient manner, without the need for explicit programming. Consequently, machine learning methods have witnessed a notable surge in their application across diverse domains, including meteorology, medicine, economics, agriculture, and others.

In agriculture, during the last few years, there has been an upward trend in the use of machine learning methods in the field of precision agriculture (PA). PA is a farm management approach that uses information technology (IT) to ensure that crops and soil receive exactly what they need for optimum health and productivity. Specifically, it aims to fully record the state of the arable land under study to better distribute fertilizers, improve the irrigation system and therefore minimize production costs and maximize net profit. To achieve this, remote sensing (RS) is used. RS is the acquisition of information about an object or phenomenon without making physical contact with it, in contrast to in-situ or on-site observation. Currently, the term generally refers to using satellite - or aircraft-based sensor technologies to detect and classify objects on Earth (Remote Sensing).

This thesis is highly motivated by the need to address a problem from the agricultural sciences, namely the prediction of the crop phenological stage percentages using large-scale data. From now on, this problem will be referred to as the Crop Phenological Stage Percentages Prediction (CPSPP) problem. It is worth noting that this problem has already been studied by Shen et al. (2013) who used a non-homogeneous hidden Markov model to predict the phenological stages. This approach was later improved by Ghamghami et al. (2020) who posed the problem within the Bayesian framework and more recently, by Oikonomidis and Trevezas (2023) who used Generalized Linear Mixed-Effects Models (GLMEMs). The aim of this thesis is to study the previous problem from the point of view of machine learning, and examine which of the various approaches discussed in this thesis is more suitable in this context.

Chapter 1

Corn Phenological Stage Percentages Prediction Problem

1.1 Corn

Since this dissertation deals with the problem of predicting the percentages of phenological stages in maize crops, we consider it necessary to first present some basic facts about the physiology of the plant as well as to point out the reasons why today's maize crops (and so the study of these) is of utmost importance.

Corn stands as one of the most extensively cultivated plants on a global scale. In recent years, there has been a notable increase in the annual production of maize worldwide, reaching a record of 1.2 billion tons in 2021 (Grain Market Report). Maize holds the distinction of being the predominant grain crop across the Americas, with the United States alone producing 384 million metric tons in 2021 (Maize). As per the Food and Agriculture Organization (FAO), the United States held the position of the largest corn producer globally in 2020, contributing 31% to the total production, followed by China at 22.4%. Within the United States, Iowa is the top producing state with Illinois and Nebraska following (World Population Review).

It comes as no surprise that corn holds such significance, given its multifaceted role in the daily lives of people and its versatile applications. Primarily, corn serves as a vital source of animal feed. Additionally, a considerable portion of corn is consumed directly by humans or in the form of its derivatives such as corn oil and corn flour. The notable utilization of corn extends to the production of bio-fuels for automotive purposes, contributing to a more environmentally friendly alternative compared to conventional plastics. These factors collectively contribute to the remarkable position and value of corn as a globally significant crop (Farm Progress).

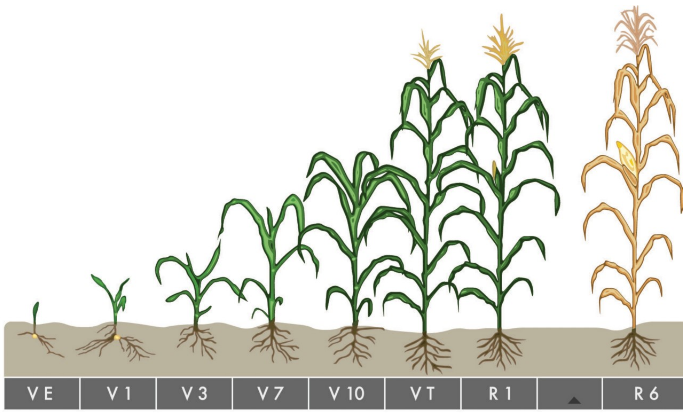

FIGURE 1.1: Corn growth stages from emergence to maturity.

1.1.1 Corn phenology

Based on the above, the need to study corn crops in order to achieve the best management of available resources and improve the production process should have become clear. Management strategies for improving corn yield, though, are most effective when one can identify the growth stage in which potential yield is affected. For example, the effects of fertilization, frost or hail, moisture stress, plant diseases, insect injury, and pesticide application on yield will be determined by the growth stage in which these events occur.

Corn growth stages are classified as vegetative (V) or reproductive (R). The leaf collar method is a method used to stage corn while it is in vegetative stages. With this method, the vegetative stages are based on the number of visible leaf collars. Staging begins at emergence (VE), and each new leaf with a fully developed leaf collar is called stage V(n). For instance, the stage in which the plant has 3 visible collar leaves is the V3 stage. Vegetative leaf staging for corn plants ends when the corn develops a tassel (VT); once the plant has silks visible outside of the husks it has reached the first reproductive stage (R1). In total, there exist 6 reproductive stages, namely silk (R1), blister (R2), milk (R3), dough (R4), dent (R5), and maturity (R6) (Abendroth et al., 2011). Figure 1.1 depicts the stages of corn development. These stages along with a brief description of them (as given in Nutrien Ekonomics) are given below.

Pre-Season

The Pre-Season stage is not an actual corn phenological stage, but in most applications is added artificially. It represents the time period from the beginning of data recordings until corn seed planting takes place. Hence, the introduction of this stage allows for the synchronization of data recordings from different growing seasons.

Planted

Planting time can vary depending on the climate and the weather, but generally in the United States will begin in the early Mid-April and will continue until mid to late May.

Emerged (VE)

Emergence happens when the first leaf or the spike leaf appears above the ground. The seed absorbs water and oxygen for fermentation. The radicle root emerges from the tip of the kernel. Soil temperature should be 50-55 degrees Fahrenheit with moisture present for optimal emergence. Planting depth is critical for this stage—too deep can delay emergence and too shallow can create a weak plant.

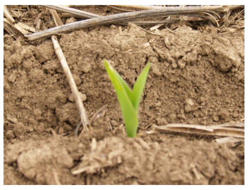
FIGURE 1.2: VE

Silking (R1)

During this stage, flowering begins and silk is visible on the outside of the husks. The first silks are attached to potential kernels near the base of the ear. Maximum plant height is achieved, pollination is occurring and the potential number of kernels is being determined. Nutrient needs are high for fertilization.

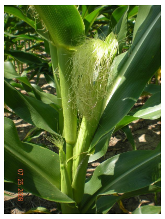
FIGURE 1.3: Silking Stage

Blister (R2)

At this stage, silks darken and are drying out. Kernels are about 85% moisture and will soon fill with starch. Environmental stress can reduce yield potential and final grain count.

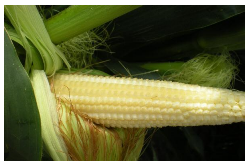
FIGURE 1.4: Blister stage

Milk (R3)

Silks are dried out. The kernels are turning yellow and a milk-like fluid can be squeezed out of the kernels. Any stress at this point will cause kernel abortion at the tip of the ear.

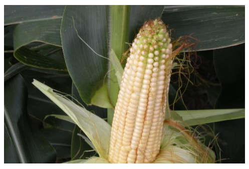
FIGURE 1.5: Milk stage

Dough (R4)

Starch materials inside the kernels have a dough-like consistency. Nutrients and starch are accumulating with kernels being at 70% moisture. Environmental stress at this point can cause reduced yield by reducing kernel size and cause kernels to be shallow or chaffy. The impact of frost on grain quality can be severe with a 25-40% yield.

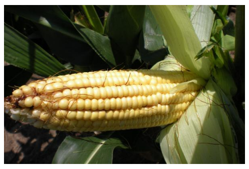
FIGURE 1.6: Dough stage

Dent (R5)

At this stage, most of the kernels are dented. Moisture is declining to about 55%. Stress will reduce kernel weight. Many growers will harvest for silage soon.

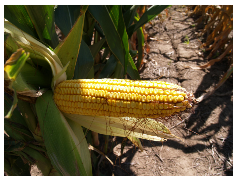
FIGURE 1.7: Dented stage

Maturity (R6)

At maturity, a black layer forms at the bottom of the kernel, blocking any movement of dry matter and nutrients from the plant to the kernel. Kernels will be around 30-35% moisture and are physiologically mature. At this stage, grain is not ready for storage, but maximum dry weight has been attained. Disease and pests can result in physical damage and grain loss. Scouting is suggested for ear drops from pest damage.

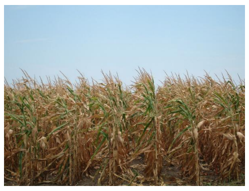
FIGURE 1.8: Maturity stage

Harvested

While not strictly a stage of grain development, harvest maturity is often defined as that grain moisture content where the harvest can occur with minimal kernel damage and mechanical harvest loss.

1.1.2 Crop Progress Reports

Our study concerns corn crops in America and specifically in the State of Nebraska. In order to monitor the condition of the crops, the National Agricultural Statistics Service (NASS) of the United States Department of Agriculture (USDA) conducts crop progress surveys throughout the growing season, namely from the beginning of April until late December. In particular, every week NASS assigns its staff to visit the cultivated areas and record what percentage of the plants are at or beyond each phenological stage. These reports are called Crop Progress Reports (CPR) and include the stages planted, emerged, silking, dough, dented, mature, and harvested.

Our data covers growing seasons 2002-2021. For each season the starting date ranges from week 12 to week 15 of the calendar year. Regardless of the starting date, for each season our data tracks CPR values for 38 consecutive weeks, hence the ending week of the study for each season varies from 49 to 52 (see Table 1.1). The CPR data is freely available through the Quick Stats Database. The

Weeks	Seasons
12-49	2012, 2015, 2021
13-50	2002, 2006, 2007, 2011, 2016, 2017, 2018, 2019
14-51	2003, 2004, 2005, 2008, 2010, 2013, 2014, 2020
15-52	2009

TABLE 1.1: Weeks of the calendar year studied for each growing season.

Quick Stats Database is the most comprehensive tool for accessing agricultural data published by NASS and contains officially published aggregate estimates related to U.S. agricultural production, available at the county, state, or country level.

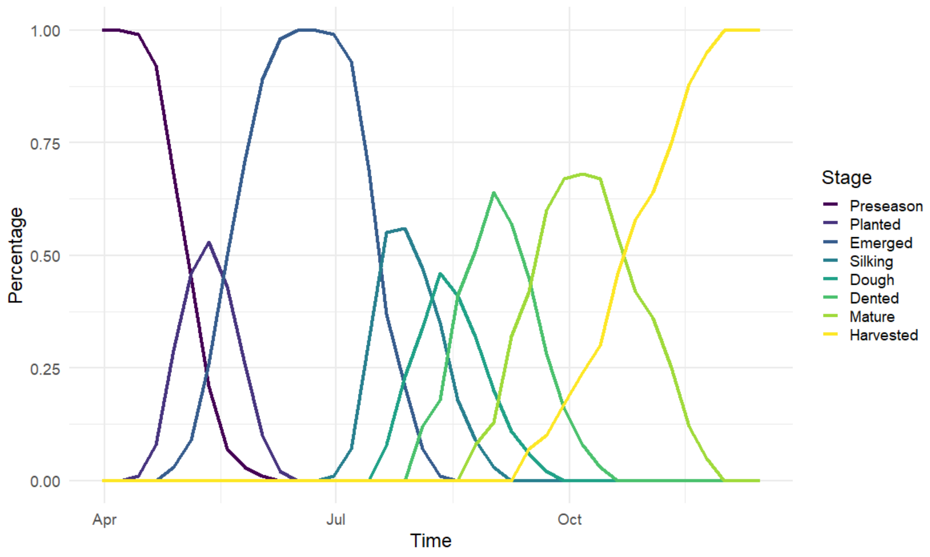

FIGURE 1.9: CPR data for the growing season 2002

In Figure 1.9 we can see the evolution of the CPR statistics for the growing season of 2002. For the first week of the study, planting has not yet started, thus we can see that 100% of the crops occupy the "Preseason" stage. The Preseason percentages decrease in time, giving their place to the "Planting" stage. As weeks pass by, we can see that each stage's percentages increase, as more and more plants reach that specified stage, and then decrease, as the plants move on to the next one. In the final weeks of the study, the crops are absorbed in the "Harvested" stage.

The CPR percentages are the core of the problem we have to solve since what we would like is to build a mathematical model that can predict for a specific future week what percentage of the plants of the cultivated area we are studying occupy each phenological stage of those included in the CPRs. To achieve this, we must take into account environmental and non-environmental factors prevailing at the given time in the area of interest. These factors are presented in the following sections.

1.2 Environmental factors

In this section, we present the environmental factors used in our study to predict the CPR percentages. Data for all of the following environmental factors are obtained through Daymet, a research product of the Environmental Sciences Division at Oak Ridge National Laboratory (ORLN). Daymet provides long-term, continuous, gridded estimates of daily weather and climatology variables by interpolating and extrapolating ground-based observations through statistical modeling techniques.

Daymet weather variables include daily minimum and maximum temperature, precipitation, vapor pressure, shortwave radiation, snow water equivalent, and day length produced on a 1 km x 1 km gridded surface over continental North America and Hawaii from 1980 and over Puerto Rico from 1950 through the end of the most recent full calendar year.

1.2.1 Thermal Time

Temperature is a primary factor affecting the rate of plant development as the latter is dependent upon the temperature of the air surrounding the plant. In particular, plants can only develop within a certain range of temperatures and, thus, each species has a specific range of temperatures represented by a base (T_b), a ceiling (T_c), and an optimum (T_o) temperature. Temperature extremes, that is temperatures above the ceiling or below the base temperatures, may harm plant growth. For instance, research has shown that the major impact of warmer temperatures was during the reproductive stages of development, and grain yield in corn was significantly reduced by as much as 80%-90% from a normal temperature regime (Hatfield and Prueger, 2015).

To take into account the dependence of plants on the ambient air temperature in applications, *Growing Degree Days* (GDD) are used. GDD is a measure of heat accumulation used in agriculture and biology to predict plant and animal development rates such as the date that a crop will reach maturity or an insect will emerge from dormancy. In agriculture, GDD in fact represents the amount of heat absorbed by the plant in a single day. If we sum the GDD for a time interval we get the *Accumulated Growing Degree Days* (AGDD), a measure that expresses the total amount of heat absorbed by the plant in that interval. A more formal definition of GDD and AGDD is the following:

Definition 1.1 (GDD - AGDD)**.** If T_b is the base temperature, T_c the ceiling temperature, $T_{min}(t)$ and $T_{max}(t)$ the minimum and the maximum temperature at day t, then

1. The **average truncated temperature** of day t is defined as

$$T^*_{av}(t) := \frac{T^*_{max}(t) + T^*_{min}(t)}{2} - T_b, \tag{1.1}$$

   where

$$T^*_{max}(t) = \max\{\min\{T_{max}(t), T_c\}, T_b\}$$

$$T^*_{min}(t) = \min\{\max\{T_{min}(t), T_c\}, T_c\}$$

2. The **Growing Degree Days** at day t, denoted by $GDD(t)$ are defined as

$$GDD(t) := T_o - |T^*_{av}(t) - T_o|, \tag{1.2}$$

3. The **Accumulated Growing Degree Days** until day t, denoted by $AGDD(t)$ are defined as

$$AGDD(t) := \sum_{i=1}^{t} GDD(i). \tag{1.3}$$

For corn, the base and ceiling temperatures used in practice are $T_b = 10°C$ and $T_c = 50°$, while the optimum temperature is $T_o = 30°C$ (Cross and Zuber, 1972).

1.2.2 Precipitation

In meteorology, precipitation refers to the amount, usually expressed in millimeters or inches of liquid water depth, of the water substance that has fallen at a given point over a specified period of time (American Meteorological Society). The main form of precipitation is rain but it also includes drizzle, sleet, snow, ice pellets, graupel, and hail.

Plant development heavily depends on the amount of precipitation as well as on its intraseasonal distribution. Limitations in water availability are frequently a restrictive factor in plant development, and water is essential for the maintenance of physiological and chemical processes within the plant, acting as an energy exchanger and carrier of nutrient food supply in solution (Geneti, 2019). On the other hand, excessive precipitation can severely hinder development or even destroy the crops (Kunkel et al., 1992), since high levels of soil moisture favor the growth of microorganisms such as fungi and bacteria which can be absorbed by the plant.

For corn, precipitation levels are crucial for crop development. In particular, the interaction effects of soil water, precipitation, and plant population on final yield are determined by the level of soil water at the beginning of the rapid growth period and by the amount and distribution of precipitation during this period (Holt et al., 1964; Holt and Timmons, 1968). Therefore, knowledge of the precipitation levels may be useful information for the CPSPP. Again, we accumulate the daily precipitation data obtained from Daymet getting the following definition:

Definition 1.2 (Accumulated Precipitation)**.** If $P(t)$ is the precipitation of day t, then the **Accumulated Precipitation** until day t, denoted by $AP(t)$, is defined as:

$$AP(t) := \sum_{i=1}^{t} AP(i). \tag{1.4}$$

1.2.3 Vapor pressure

Partial vapor pressure, which refers to the pressure exerted by the water vapor in the air, can significantly affect plant growth. Partial vapor pressure affects the rate at which plants lose water through transpiration. If the partial vapor pressure is low, the rate of transpiration decreases, and plants may struggle to take up enough water to support their growth. Conversely, if the partial vapor pressure is high, the rate of transpiration increases, and plants may lose water faster than they can take it up, leading to wilting and other symptoms of water stress. In addition, partial vapor pressure can also affect the ability of plants to take up nutrients from the soil. When the air is dry and the partial vapor pressure is low, the concentration of mineral nutrients in the soil solution may become more concentrated due to reduced water availability. This can lead to an increase in salinity and other problems that can limit nutrient uptake by the plants (Grossiord et al., 2020; Ding et al., 2022).

For corn, vapor pressure plays a crucial role in determining the amount of water that is available to the plant. High vapor pressure can cause high levels of transpiration, which in turn can lead to water stress and reduced yield. High vapor pressure can also contribute to heat stress in corn plants. At the same time, high humidity levels can prevent plants from cooling themselves through transpiration, leading to overheating and reduced growth. Vapor pressure can also impact the success of corn pollination. High vapor pressure can reduce the effectiveness of pollinators, such as bees, and interfere with pollen movement. This can lead to reduced seed set and yield (Ray et al., 2002; Hsiao et al., 2019).

Definition 1.3 (Accumulated Vapor Pressure)**.** If $VP(t)$ is the partial vapor pressure of day t, then the **Accumulated Vapor Pressure** until day t, denoted by $AVP(t)$, is defined as:

$$AVP(t) := \sum_{i=1}^{t} AVP(i). \tag{1.5}$$

1.2.4 Day length

Day length, also known as photoperiod, is an important environmental factor that affects plant growth and development. Plants have evolved to sense changes in day length, which helps them coordinate important physiological processes such as flowering, leaf production, and root growth. The effect of day length on plant growth depends on the specific plant species and its photoperiodic response. Plants can be classified as short-day, long-day, or day-neutral based on their response to changes in day length. Short-day plants require a certain amount of darkness each day to trigger flowering (typically less than 12 hours of light per day). Long-day plants require more than 12 hours (typically 14 to 16 hours) of light per day to flower. Day-neutral plants do not require a specific day length to initiate flowering. They will flower regardless of day length as long as they have reached a certain level of maturity.

For corn, experts debate whether it should be classified as a long-day, short-day, or day-neutral plant (Karl, 2000). Some specific corn varieties may be classified as long-day while others as short-day (or day-neutral) and growing conditions do affect this classification (Chen et al., 2015).

During the early vegetative stage, corn plants undergo rapid leaf and stem growth, and longer day lengths can help stimulate this growth (Chang, 1981). However, if the day length is too long, it can cause excessive vegetative growth and delay reproductive development, ultimately reducing yield potential. As corn plants enter the reproductive stage, day length becomes even more critical. This is because corn plants require a specific amount of daylight hours to initiate the reproductive process, which involves the formation of tassels and ears. In particular, if the day length is too short, it can delay or inhibit the formation of tassels and ears, resulting in reduced yield potential. On the other hand, if the day length is too long, it can cause tassels and ears to form prematurely, leading to poor pollination and reduced yields (Hunter et al., 1974; Birch et al., 1998). In addition to affecting the timing of reproductive development, day length can also affect the number of leaves and the size of the ear. Studies have shown that corn plants grown under longer day lengths tend to produce more leaves and larger ears, while plants grown under shorter day lengths tend to produce fewer leaves and smaller ears (Warrington and Kanemasu, 1983).

Definition 1.4 (Accumulated Day Length)**.** If $DL(t)$ is the day length of day t, then the **Accumulated Day Length** until day t, denoted by $AVP(t)$, is defined as:

$$ADL(t) := \sum_{i=1}^{t} DL(i). \tag{1.6}$$

1.3 Non-Environmental factors

1.3.1 Normalized Difference Vegetation Index

Scientists have been using satellite RS to monitor fluctuation in vegetation at the Earth's surface since the 1960s. Since then, various indices that quantify how healthy a plant or arable land is using data from satellite images have been introduced. These indices are called *Vegetation Indices* (VIs) and in fact, are single values calculated by transforming the observations from multiple spectral bands. They are used to enhance the presence of green, vegetation features and thus help to distinguish them from the other objects present in the image (Hi-Phen).

This is done by carefully measuring the wavelengths and intensity of visible and near-infrared light reflected by the land surface back up into space. More specifically, when sunlight strikes objects, certain wavelengths of this spectrum are absorbed and other wavelengths are reflected.

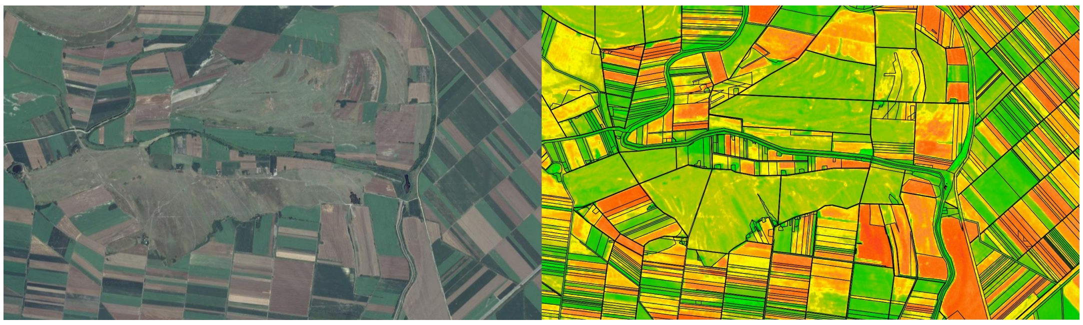
FIGURE 1.10: Raw corn field image (left) vs its NDVI counterpart (right) (Smart AKIS)

The pigment in plant leaves, chlorophyll, strongly absorbs visible light (from 0.4 to 0.7 µm) for use in photosynthesis. The cell structure of the leaves, on the other hand, strongly reflects near-infrared light (from 0.7 to 1.1 µm). If there is much more reflected radiation in near-infrared wavelengths than in visible wavelengths, then the vegetation in that pixel is likely to be dense and may contain some type of forest. If there is very little difference in the intensity of visible and near-infrared wavelengths reflected, then the vegetation is probably sparse and may consist of grassland, tundra, or desert. Nearly all satellite VIs employ this difference formula to quantify the density of plant growth on Earth — near-infrared radiation minus visible radiation divided by near-infrared radiation plus visible radiation (ESA). The most common one is the *Normalized Difference Vegetation Index* (NDVI) which is defined as follows:

Definition 1.5 (NDVI)**.** Let ρ_{RED} and ρ_{NIR} represent the surface reflectances averaged over ranges of wavelengths in the visible red $(0.7 - 0.9\mu m)$ and near-infrared $(0.5 - 0.7\mu m)$ regions of spectrum, respectively. Then, **the Normalized Difference Vegetation Index (NDVI)** is given by:

$$NDVI = \frac{\rho_{NIR} - \rho_{RED}}{\rho_{NIR} + \rho_{RED}} \tag{1.7}$$

As we can see from equation 1.7, NDVI takes values within the interval $[-1, 1]$. In view of the foregoing discussion, we see that healthy vegetation will have large ρ_{NIR} and small ρ_{RED} values, while the situation is reversed for the case of unhealthy vegetation (see Figure 1.11).

Although NDVI is quite interpretable by definition, its value calculation turns out to be sensitive to several perturbing factors. The most common ones are clouds. Deep (optically thick) clouds may be quite noticeable in satellite imagery and yield characteristic NDVI values that ease their screening. However, thin clouds (such as the ubiquitous cirrus), or small clouds with typical linear dimensions smaller than the diameter of the area actually sampled by the sensors, can significantly contaminate the measurements. Similarly, cloud shadows in areas that appear clear can affect NDVI values and lead to misinterpretations. Another influential factor is the actual composition of the atmosphere (in particular concerning water vapor and aerosols) as it can significantly affect the measurements made

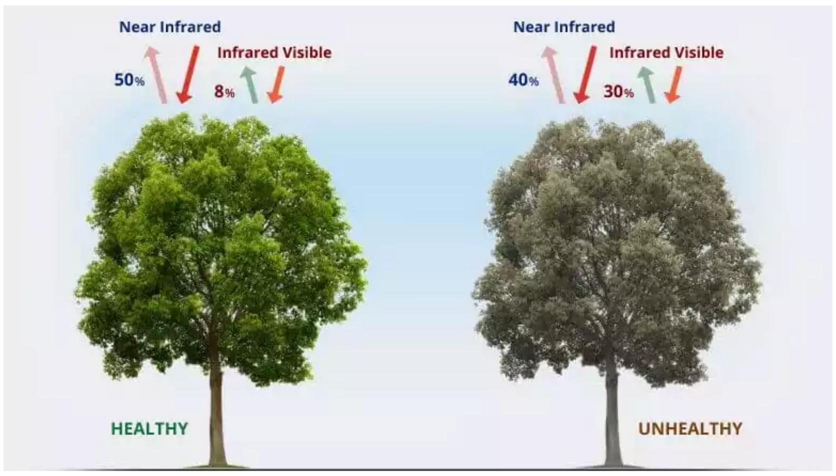

FIGURE 1.11: Healthy (left) vs Unhealthy (right) vegetation reflectances (EOS)

in space. Hence, the latter may be misinterpreted if these effects are not properly taken into account (as is the case when the NDVI is calculated directly based on raw measurements). Other factors that might affect the calculation of the NDVI value include soil, anisotropic and spectral effects (NDVI).

Several solutions have been proposed in the literature to address these limitations. Most of them belong to the pre-processing stage and help us make the most of NDVI as a predictive tool for the crop phenological stages estimation problem. The raw NDVI data used in this thesis is taken from NASA's Moderate Resolution Imaging Spectroradiometer (MODIS) and, unlike CPR percentages, are daily resulting in $38 \times 7 = 266$ days for each growing season. All the pre-processing steps applied to this data along with a detailed description of them can be found in Ioannis Oikonomidis' thesis (Oikonomidis, 2020).

1.3.2 Calendar time

Since past CPR percentages are available on a weekly basis, the week on which we want to make predictions may itself be quite informative for both the prevailing factors in the area of interest and the percentage of plants occupying each phenological stage at this time point of the growing season. For instance, suppose we want to predict the CPR percentages for week 48, which typically corresponds to late November/early December, that is, to the end of the growing season for corn. Therefore, what we would expect for the CPR percentages are high values for the harvested stage and low values for all other phenological stages.

Chapter 2

The Machine Learning Framework

This chapter introduces the basic principles of Machine Learning and establishes a framework upon which the algorithms and methodology used in this analysis can be presented in a comprehensive way in the following chapters. In addition, performance evaluation measures are introduced that allow for comparison between the various approaches to the problem at hand.

2.1 Learning from data

The problem presented in Chapter 1 belongs to the wide family of *supervised learning* tasks. In particular, we want to construct a model that predicts the value of one or more variables of interest when the values of some other variables are known. In Machine Learning the variables of interest are called **target**, **response**, or **output variables**, while the variables whose observations are used in order to make a prediction for the value(s) of the output variable(s) are called **input variables**, **predictors** or **features**.

2.1.1 Terminology

To make it more formal, suppose that we have S output and p input variables. Each one of the input variables is denoted by X_k, their observed values by x_k, and their domains by $\mathcal{X}_k$ (for $k = 1, 2, \ldots, p$). Together, the input variables constitute a p - dimensional variable $X = (X_1, X_2, \ldots, X_p)'$ taking values $\mathbf{x} = (x_1, x_2, \ldots, x_p)'$, also known as **input vectors**, from the **input space** $\mathcal{X} = \mathcal{X}_1 \times \mathcal{X}_2 \times \cdots \times \mathcal{X}_p$. Accordingly, each target variable is denoted by Y_s, the corresponding observed values by y_s, and their domains by $\mathcal{Y}_s$ (for $s = 1, 2, \ldots S$). The S - dimensional output variable $Y = (Y_1, Y_2, \ldots Y_S)'$ formed by them takes values denoted by $\mathbf{y} = (y_1, y_2, \ldots, y_S)'$ from the **output space** $\mathcal{Y} = \mathcal{Y}_1 \times \mathcal{Y}_2 \times \cdots \times \mathcal{Y}_S$. The input values $\mathbf{x}$ along with their corresponding output values $\mathbf{y}$ form vectors $(\mathbf{x}, \mathbf{y})$ which are usually referred to as **samples**, **instances** or **cases**. We assume that both the input and output spaces contain all possible values for the input and output variables respectively.

A basic distinction concerns the type of variables used in the analysis, which are two, namely *numerical* and *categorical*. These two types of variables are defined as follows:

Definition 2.1 (Numerical variable)**.** A variable X_k is **ordered** if $\mathcal{X}_k$ is a totally ordered set. Especially, if $\mathcal{X}_k = \mathbb{R}$, then X_k is called **numerical**.

Definition 2.2 (Categorical variable)**.** A variable X_k is **categorical** if $\mathcal{X}_k$ is a finite set of values, without any natural order.

In a supervised learning task, we summarize the past observations in our possession in a data set, called *learning set*. It consists of a number of observed input vectors along with their corresponding output vector. A more formal definition is the following:

Definition 2.3. A **learning set** $\mathcal{L}$ is a set of n pairs of input and output vectors $(\mathbf{x}_1, \mathbf{y}_1)$ $(\mathbf{x}_2, \mathbf{y}_2)$, $\dots (\mathbf{x}_n, \mathbf{y}_n)$, where $\mathbf{x}_i \in \mathcal{X}$ and $\mathbf{y}_i \in \mathcal{Y}$.

For the sake of brevity, we may also denote the learning set by

$$\mathcal{L} = (\mathbf{X}, \mathbf{Y}), \tag{2.1}$$

where $\mathbf{X}$ is a $n \times p$ matrix and $\mathbf{Y}$ is a $n \times S$ matrix.
Each row of these matrices refers to one of the n observed vectors and each column to one of the input or output variables. For $S = 1$, $\mathbf{Y}$ becomes a vector which we may denote by $\mathbf{y}$.

Under this formulation, the supervised learning task can be restated as an attempt to use a learning set $\mathcal{L}$ to construct a function $\phi : \mathcal{X} \to \mathcal{Y}$ whose predictions $\phi(\mathbf{x})$, also denoted by $\hat{Y}$, are as good as possible. In section 2.2 we quantify what "good" means by introducing performance evaluation criteria.

Most often, ϕ is built according to an algorithm $\mathcal{A}$, which describes the construction details of ϕ. $\mathcal{A}$ may be quite simple producing functions ϕ in a trivial manner or more complex involving many steps and calculations. Chapters 3 and 4 present the implementation details of the decision tree and random forest algorithms respectively.

We now distinguish between two types of supervised learning tasks according to the type of output variables. In particular, when each coordinate of the S - dimensional variable Y is categorical, then the supervised learning task is a *multi-output* or *multivariate classification* problem, when it consists entirely of numerical variables we have a *multi-output* or *multivariate regression* problem and when both numerical and categorical constitute Y, we have a *mixed multi-output* or *multivariate* problem. The corresponding models are defined as follows:

Definition 2.4. A **classifier** or **classification rule** is a function $\phi : \mathcal{X} \to \mathcal{Y}$, where $\mathcal{Y} = \mathcal{Y}_1 \times \mathcal{Y}_2 \times \cdots \times \mathcal{Y}_S$ and each $\mathcal{Y}_s$ is a finite set of labels, denoted by $\{c_1, c_2, \ldots c_{J_s}\}$.

Definition 2.5. A **regressor** is a function $\phi : \mathcal{X} \to \mathcal{Y}$, where $\mathcal{Y} = \mathbb{R}^S$.

As discussed in chapter 5, the problem presented in chapter 1 constitutes a regression problem, thus, we will not get into detail about the classification task in the rest of this thesis.

2.1.2 Repeated measurement data

In real-world applications the general form of a learning set presented in 2.1 may receive more particular forms. One such example is that of *repeated measurement data*. Repeated measurement data typically occur in studies in which the response variables are measured at multiple time points.

More formally, assume that there exist N *subjects* and for each of them S output and p input variables are measured over n_i time points t_{n_i} (for $i = 1, 2, \ldots, N$). This results in a total of $n = \sum_{i=1}^{N} n_i$ observations. The general form of a repeated measurement learning set is summarized in the following table. Here each row of the learning set is of the form $\mathbf{z}_{ij} = (t_{ij}, \mathbf{x}_{ij}, \mathbf{y}_{ij})'$ for $i = 1, 2, \ldots, N$

Subject	Time	Predictors			Responses		
1	t_{11}	x_{111}	$\cdots$	x_{11p}	y_{111}	$\cdots$	y_{11S}
1	t_{12}	x_{121}	$\cdots$	x_{12p}	y_{121}	$\cdots$	y_{12S}
.	.	.	$\ddots$	.	.	$\ddots$	.
1	t_{1n_1}	x_{1n_11}	$\cdots$	x_{1n_1p}	y_{1n_11}	$\cdots$	y_{1n_1S}
$\vdots$	$\vdots$	$\vdots$		$\vdots$	$\vdots$	$\vdots$	$\vdots$
N	t_{N1}	x_{N11}	$\cdots$	x_{N1p}	y_{N11}	$\cdots$	y_{N1S}
N	t_{N2}	x_{N21}	$\cdots$	x_{N2p}	y_{N21}	$\cdots$	y_{N2S}
.	.	.	$\ddots$	.	.	$\ddots$	.
N	t_{Nn_N}	x_{Nn_N1}	$\cdots$	x_{Nn_Np}	y_{Nn_N1}	$\cdots$	y_{Nn_NS}

TABLE 2.1: General structure of repeated measure data

and $j = 1, 2, \ldots, n_i$. Therefore, a repeated measurement learning set $\mathcal{L}$ may be summarized in the following form:

$$\mathcal{L} = \{\mathbf{z}_{ij} | \mathbf{z}_{ij} = (t_{ij}, \mathbf{x}_{ij}, \mathbf{y}_{ij})', i = 1, 2, \ldots, N, j = 1, 2, \ldots, n_i\}. \tag{2.2}$$

Vectors $\mathbf{x}_{ij}$, $\mathbf{y}_{ij}$ refer to the observed input and output vectors respectively for the i-th subject at the j-th observation time t_{ij}.

Remark 2.1. For the sake of clarity regarding the introduced notation, we note that:

– When dealing with a repeated measurement learning set $\mathcal{L}$, we will be referring to each row of $\mathcal{L}$ using subscripts i, j (for $i = 1, 2, \ldots, N$ and $j = 1, 2, \ldots, n_i$), to emphasize that each row is subject- and time-dependent.

– On the contrary, when we make no assumptions about the form of $\mathcal{L}$ (i.e. when $\mathcal{L}$ is of the general form 2.1), we may denote each row of it by using subscript i (for $i = 1, 2, \ldots, n$).

Note that a predictor may be constant or time-varying within the observation time interval in the context of repeated measurement data. In the latter case, the predictor may require special treatment in order to adequately model its behavior. What is more, observations in a repeated measurement learning set may be regularly or irregularly distributed within time. For instance, a researcher may track health outcomes of individuals with a chronic disease at the time of their initial diagnosis, and thereafter at irregular intervals based on when relapses occur or according to a fixed schedule (e.g. every month's first).

Analyzing repeated measurement data is not an easy task. Repeated measurements from the same subject are likely to be correlated, while inter-subject variability may induce clustering effects to the problem. A proper statistical analysis should take into account both inter-subject variability and inter-subject correlation. For a comprehensive study on statistical methods for repeated measurement data see Davis (2002).

2.2 Performance evaluation

In this section, we specify what creating a model $\phi_{\mathcal{L}}$ with accurate predictions means in the context of machine learning. In particular, we define the generalization error and we describe methods to estimate it.

2.2.1 Generalization Error

From a statistical perspective, $X = (X_1, X_2, \ldots, X_p)'$ and $Y = (Y_1, Y_2, \ldots, Y_S)'$ are random vectors taking their values $(\mathbf{x}, \mathbf{y})$ from $\mathcal{X} \times \mathcal{Y}$ according to a joint probability distribution $P(X, Y)$. Recall now that our goal in a supervised learning task is to use a learning set $\mathcal{L}$ to construct a model $\phi_{\mathcal{L}}$ which accurately predicts the output value for a specific input vector $\mathbf{x}$. In fact, what we really want for the constructed model $\phi_{\mathcal{L}}$ is not to make the most accurate predictions over the given learning set $\mathcal{L}$. In such a scenario, our model may capture the irrelevant information present in $\mathcal{L}$, leading to a phenomenon called *overfitting*. Rather, what we want is our model to be accurate over all possible data. In other words, we seek a model that *generalizes* well, that is a model which has small *generalization error*, which is defined as follows:

Definition 2.6 (Generalization Error, Geurts (2002))**.** Let $\phi_{\mathcal{L}}$ be a model built using a learning set $\mathcal{L}$. Then, the **generalization error** or **expected prediction error** of $\phi_{\mathcal{L}}$ is defined as

$$Err(\phi_{\mathcal{L}}) = \mathbb{E}_{X,Y}\left[L\left(Y, \phi_{\mathcal{L}}\left(X\right)\right)\right], \tag{2.3}$$

where L is a loss function measuring the discrepancy between its two arguments.

We note that for regression problems, the most common loss function is the *squared loss*.

Definition 2.7 (Squared Loss)**.** Let $\phi_{\mathcal{L}}(X)$ be the prediction of a regressor $\phi_{\mathcal{L}}$ built using a learning set $\mathcal{L}$ on an input variable X and Y a numerical output variable. The **squared loss** function of $\phi_{\mathcal{L}}(X)$ with respect to Y is defined as:

$$L\left(Y, \phi_{\mathcal{L}}\left(X\right)\right) = \left(Y - \phi_{\mathcal{L}}\left(X\right)\right)'\left(Y - \phi_{\mathcal{L}}\left(X\right)\right), \tag{2.4}$$

which for the univariate case ($S = 1$) simplifies to $L\left(Y, \phi_{\mathcal{L}}\left(X\right)\right) = \left(Y - \phi_{\mathcal{L}}\left(X\right)\right)^2$.

Note that squared loss function attains high values when the difference between the two arguments is large and small when the predicted value is close to the true one.

From equation 2.3 we see that in order to calculate the generalization error we need to know the joint probability distribution $P(X,Y)$. In practice, though, this is rarely the case. Actually, we cannot even estimate the $Err(\phi_{\mathcal{L}})$ empirically, since in most cases we are not able to draw additional data.

A similar, yet easier to be estimated in practical settings (Hastie et al., 2009b), quantity is the *expected generalization error* defined as follows:

Definition 2.8. The **expected generalization error**, for a model $\phi_{\mathcal{L}}$ built using a learning set $\mathcal{L}$ is defined as:

$$EGE\left(\phi_{\mathcal{L}}\right) := \mathbb{E}_{\mathcal{L}}\left[Err\left(\phi_{\mathcal{L}}\right)\right]. \tag{2.5}$$

Note that equation 2.5 averages over everything which is random, including the randomness in the learning set $\mathcal{L}$. As a result, this quantity measures the performance of the model $\phi_{\mathcal{L}}$ over all possible learning sets $\mathcal{L}$. Section 2.2.2 describes methods for estimating the Generalization Error.

2.2.2 Estimating the Generalization Error

In this section, we discuss different ways to estimate the Generalization Error for a given model $\phi_{\mathcal{L}}$. We first give the following definition:

Definition 2.9 (Average Prediction Error)**.** Let $\phi_{\mathcal{L}}$ be a model built using a learning set $\mathcal{L}$. The **average prediction error** of $\phi_{\mathcal{L}}$ over a learning set $\mathcal{L}'$ (which may be different from the learning set $\mathcal{L}$) is denoted by $\bar{E}\left(\phi_{\mathcal{L}}, \mathcal{L}'\right)$ and defined as:

$$\bar{E}\left(\phi_{\mathcal{L}}, \mathcal{L}'\right) = \frac{1}{n'} \sum_{(\mathbf{x}'_i, \mathbf{y}'_i) \in \mathcal{L}'} L\left(\mathbf{y}'_i, \phi_{\mathcal{L}}\left(\mathbf{x}'_i\right)\right), \tag{2.6}$$

where n' is the size of the learning set $\mathcal{L}'$.

A related quantity often used in univariate regression problems is the *Root Mean Square Error* or *Root Mean Square Prediction Error* (RMSPE) which is given by the following definition:

Definition 2.10 (RMSPE)**.** Let $\phi_{\mathcal{L}}$ be a model built using a learning set $\mathcal{L}$. The **root mean square prediction error** of $\phi_{\mathcal{L}}$ over a learning set $\mathcal{L}'$, denoted by $RMSPE\left(\phi_{\mathcal{L}}, \mathcal{L}'\right)$, is the square root of the corresponding average prediction error:

$$RMSPE\left(\phi_{\mathcal{L}}, \mathcal{L}'\right) = \sqrt{\bar{E}\left(\phi_{\mathcal{L}}, \mathcal{L}'\right)}, \tag{2.7}$$

where for the calculation of $\bar{E}\left(\phi_{\mathcal{L}}, \mathcal{L}'\right)$ the squared loss function is used.

A simple, yet poor, estimate of the generalization error is the *resubstitution estimate* or *training sample estimate* which consists in evaluating the performance of the model $\phi_{\mathcal{L}}$ on the same data used to be built and is defined as:

Definition 2.11. Let $\phi_{\mathcal{L}}$ be a model built using a learning set $\mathcal{L}$. The **resubstitution estimate** or **training sample estimate**, denoted by $\widehat{Err}^{train}\left(\phi_{\mathcal{L}}\right)$, is given by the following equation:

$$\widehat{Err}^{train}\left(\phi_{\mathcal{L}}\right) = \bar{E}\left(\phi_{\mathcal{L}}, \mathcal{L}\right) \tag{2.8}$$

Using learning set $\mathcal{L}$ for both learning $\phi_{\mathcal{L}}$ and estimating $Err(\phi_{\mathcal{L}})$, as equation 2.8 proposes, is undesirable since most machine learning algorithms during the learning process seek the minimization of $\widehat{Err}^{train}\left(\phi_{\mathcal{L}}\right)$. As a result, the latter quantity underestimates the generalization error and this fact stresses the need to find a more reliable method to estimate $Err(\phi_{\mathcal{L}})$. In this direction, we give the following definition:

Definition 2.12. Let $\{\mathcal{L}_{train}, \mathcal{L}_{test}\}$ be a partition of a given learning set $\mathcal{L}$ [1] and $\phi_{\mathcal{L}}$ be a model. The set $\mathcal{L}_{train}$ is referred to as the **training set**, used to learn the model $\phi_{\mathcal{L}}$ and the set $\mathcal{L}_{test}$ is referred to

[1]Non-empty disjoint sets covering $\mathcal{L}$

as the **test set**, used to estimate the generalization error of $\phi_{\mathcal{L}}$. The **test sample estimate** or **hold-out estimate** of the generalization error $Err(\phi_{\mathcal{L}})$, denoted by $\widehat{Err}^{test}(\phi_{\mathcal{L}})$, is given as the average prediction error of the model $\phi_{\mathcal{L}_{train}}$, built on $\mathcal{L}_{train}$, over the set $\mathcal{L}_{test}$:

$$\widehat{Err}^{test}(\phi_{\mathcal{L}}) = \bar{E}\left(\phi_{\mathcal{L}_{train}}, \mathcal{L}_{test}\right). \tag{2.9}$$

In practice, the size of $\mathcal{L}_{train}$ is taken to be approximately the 70% of the size of the original learning set $\mathcal{L}$ and the size of $\mathcal{L}_{test}$ the remaining 30%. Furthermore, we usually split $\mathcal{L}$ at random to guarantee that the samples in $\mathcal{L}_{train}$ are independent of those in $\mathcal{L}_{test}$ (Joseph, 2022).

Contrary to the resubstitution estimate, the test sample estimate uses different learning sets for the construction and the performance evaluation of $\phi_{\mathcal{L}}$. Yet, it also has some drawbacks. First and foremost, using only a percentage of the original data set $\mathcal{L}$ reduces the effective sample size used to learn the model $\phi_{\mathcal{L}_{train}}$. This results in a high variance among the test sample estimates and, thus, the true generalization error of the model might not be approximated well, especially when $\mathcal{L}$ is small. What is more, when we are interested in estimating the expected generalization error, this strategy might not be appropriate as it does not take into account the variability of the different training sets.

When the size of the original learning set $\mathcal{L}$ is small, we usually prefer the *K-fold cross-validation estimate*, which is defined as follows:

Definition 2.13. Let $\mathcal{L}$ be a learning set, $\{\mathcal{L}_1, \mathcal{L}_2, \ldots, \mathcal{L}_K\}$ a partition of $\mathcal{L}$ and $\phi_{\mathcal{L}}$ a model built using $\mathcal{L}$. The **K-fold cross validation estimate** of the generalization error $Err(\phi_{\mathcal{L}})$, denoted by $\widehat{Err}^{CV}(\phi_{\mathcal{L}})$, is defined as the average prediction error over the folds $\mathcal{L}_k$ of the models $\phi_{\mathcal{L}\setminus\mathcal{L}_k}$ learned from the remaining data:

$$\widehat{Err}^{CV}(\phi_{\mathcal{L}}) = \frac{1}{K}\sum_{k=1}^{K} \bar{E}(\phi_{\mathcal{L}\setminus\mathcal{L}_k}, \mathcal{L}_k). \tag{2.10}$$

The appealing feature of the K-fold cross-validation estimate is that the entire learning set $\mathcal{L}$ is used for estimating $Err(\phi_{\mathcal{L}})$. In particular, contrary to setting aside a relatively large portion of the original data set as test data, each model $\phi_{\mathcal{L}\setminus\mathcal{L}_k}$ is now built using almost the entire $\mathcal{L}$, resulting in estimates that are all close to $Err(\phi_{\mathcal{L}})$. At the same time, each sample is used both for training and testing. This, along with the final averaging step, reduce the variance introduced during the split into training and test sets.

Another way to improve the test sample estimate is to repeatedly split the given learning set into training and testing sets, calculate for each partition the test sample estimate and average to obtain a single estimate (Burman, 1989). This method is known as *Monte-Carlo Cross Validation* (MCCV) and is formally defined as follows:

Definition 2.14. Let $\{\mathcal{L}^m_{train}, \mathcal{L}^m_{test}\}_{m=1}^M$ be M partitions of a learning set $\mathcal{L}$ and $\phi_{\mathcal{L}}$ a model built using $\mathcal{L}$. The **monte-carlo cross validation estimate** of the generalization error $Err(\phi_{\mathcal{L}})$, denoted by $\widehat{Err}^{MC}(\phi_{\mathcal{L}})$, is given by the following equation:

$$\widehat{Err}^{MC}(\phi_{\mathcal{L}}) = \frac{1}{M}\sum_{m=1}^{M} \bar{E}\left(\phi_{\mathcal{L}^m_{train}}, \mathcal{L}^m_{test}\right). \tag{2.11}$$

Comparing the K-fold cross-validation estimate with the MCCV one, we see that the former is less biased since, as we have already mentioned, each model $\phi_{\mathcal{L}\setminus\mathcal{L}_k}$ is close to $\phi_{\mathcal{L}}$. However, the latter allows for a better exploration of the possible partitions of $\mathcal{L}$. More specifically, in K-fold cross-validation the number of partitions tested is exactly K, which, in practice, is usually taken to be 10 or 20 (Kohavi, 1995). In MCCV, though, we can test M out of the $\binom{n}{n_{train}}$ possible partitions, where n, n_{train} are the sizes of the learning sets $\mathcal{L}, \mathcal{L}_{train}$ respectively. It is clear that the number of possible partitions increases rapidly with n; hence, we can increase M as much as we want in order to test our model on a considerable number of different partitions and get a less variant estimate of the generalization error. Of course, increasing M comes with a computational cost, since a new model must be built for each partition. Nevertheless, Zhang (1993) has shown that using $M = n^2$ yields results similar to those taken when testing all the $\binom{n}{n_{train}}$ possible partitions. We should note, though, that even n^2 may be computationally infeasible, so in many cases values from 50 to 500 are chosen for the value of M.

We should note that in all 4 equations 2.8, 2.9, 2.10, 2.11 we can replace the average prediction error with the RMSPE. Yet, the resulting quantities estimate $\sqrt{Err(\phi_{\mathcal{L}})}$ rather than $Err(\phi_{\mathcal{L}})$.

2.2.3 Bayes Model

As foretold, in real-world applications the underlying probability distribution $P(X, Y)$ is in general unknown. When this probability distribution is known, as is the case with simulated data, we can find the best possible model with respect to the generalization error. This model is known as the *Bayes model* and its generalization error as *residual error*.

Definition 2.15. A model ϕ_B is a **Bayes model**, if for any model ϕ built from any learning set $\mathcal{L}$, we have that $Err(\phi_B) \leq Err(\phi_{\mathcal{L}})$. Its generalization error, $Err(\phi_B)$, is called **residual error**.

By definition, residual error represents a lower bound for the generalization error of any model and, as such, can serve as a reference point to see how well a model $\phi_{\mathcal{L}}$ performs, that is, how close $Err(\phi_{\mathcal{L}})$ is to $Err(\phi_B)$.

To illustrate how knowledge of $P(X, Y)$ allows for an explicit calculation of the Bayes model, we recall equation 2.3. In particular, by conditioning on X, we get the following expression for the generalization error of the Bayes model:

$$\mathbb{E}_{X,Y}\left[L\left(Y,\phi_B\left(X\right)\right)\right] = \mathbb{E}_X\left[\mathbb{E}_{Y|X}\left[L\left(Y,\phi_B\left(X\right)\right)\right]\right] \tag{2.12}$$

Minimizing the expression above is equivalent to minimizing $\mathbb{E}_{Y|X}\left[L\left(Y,\phi_B\left(X\right)\right)\right]$ point-wise, so, in general, the Bayes model takes the following form:

$$\phi_B(\mathbf{x}) = \arg\min_{y\in\mathcal{Y}} \mathbb{E}_{Y|X=\mathbf{x}}\left[L\left(Y,y\right)\right] \tag{2.13}$$

In a regression setting, where the squared error loss function is used, the Bayes model coincides with the regression function of Y on X:

$$\begin{aligned}\phi_B(\mathbf{x}) &= \arg\min_{y\in\mathcal{Y}} \mathbb{E}_{Y|X=\mathbf{x}}\left[\left(Y-y\right)'\left(Y-y\right)\right]\\ &= \mathbb{E}_{Y|X=\mathbf{x}}\left[Y\right]\end{aligned}$$

In other words, when the joint probability $P(X,Y)$ is known, the best possible model is the one that predicts the average value of Y given that $X = \mathbf{x}$.

Despite its use for measuring the effectiveness of a model on simulated data, Bayes model can also be used to assess the theoretical properties of an algorithm. In particular, when using an algorithm $\mathcal{A}$ to produce models $\phi_{\mathcal{L}}$, it is desirable that the generalization errors of these models get arbitrarily close to the lowest possible generalization error $Err(\phi_B)$, as the size of $\mathcal{L}$ gets large. This property, known as *consistency*, is formally defined as follows (Devroye et al., 1996):

Definition 2.16 (Weakly consistent algorithm)**.** A learning algorithm $\mathcal{A}$ is said to be **weakly consistent** for a certain probability distribution $P(X,Y)$ if

$$\mathbb{E}_{\mathcal{L}}\left[Err\left(\phi_{\mathcal{L}}\right)\right] \to Err(\phi_B),$$

as the size n of the learning set $\mathcal{L}$ used to build $\phi_{\mathcal{L}}$ using $\mathcal{A}$ tends to infinity.

Definition 2.17 (Strongly consistent algorithm)**.** A learning algorithm $\mathcal{A}$ is said to be **strongly consistent** for a certain probability distribution $P(X,Y)$ if

$$Err\left(\phi_{\mathcal{L}}\right) \xrightarrow{a.s.} Err(\phi_B),$$

as the size n of the learning set $\mathcal{L}$ used to build $\phi_{\mathcal{L}}$ using $\mathcal{A}$ tends to infinity.

It is clear from the definitions above that a learning algorithm $\mathcal{A}$ may be consistent for some distributions and not for others. If $\mathcal{A}$ is consistent for any distribution $P(X, Y)$, then it is said to be *universally (strongly) consistent*.

Chapter 3

Decision Trees

We have already stressed the need for creating accurate models and the role that machine learning has in meeting this need. The rapid development experienced by this field in recent decades has created fertile ground for the rise of a wide family of algorithms, known as *tree-based methods*. While the first attempts for the construction of tree-based models date back to the 1970s (Morgan and Sonquist, 1963), it took some years before decision trees receive the form we know today. Breiman et al. (1984) in their famous book *Classification and Regression Trees* (CART) proposed a unified framework for the induction of decision trees. Their work was later complemented by that of Quinlan (1986, 1993) who introduced two successful decision tree algorithms, namely ID3 and C4.5. Since their introduction, these algorithms have become a powerful tool in the machine learning field and several variants have been built upon these. The reason for this success is the fact that the resulting decision trees are:

- easily understandable even by non-experts, yielding interpretable results,
- non-parametric, making no assumptions about the relation between the inputs and the outputs,
- robust to outliers, irrelevant features, and data noise,
- able to deal well with missing data, and
- capable of handling both ordered and categorical data (or even a mix of them).

Despite the aforementioned advantages, decision trees suffer from high variability, that is, even small changes in the given learning set may result in large discrepancies in the final model. As will be demonstrated later, this variability is the main reason why decision trees serve as the building block of well-known ensemble methods, such as random forests (see chapter 4) and boosting (Freund and Schapire, 1997). Hence, being aware of the constructing details of decision trees is a prerequisite to better understanding these algorithms and making the most of their learning capabilities.

3.1 Tree-based framework

The theoretical background (and, thus, the algorithmic details) of decision trees can be explained geometrically. The principle idea behind decision trees is to recursively split the input space $\mathcal{X}$ into subspaces to obtain a close approximation of the Bayes model. To better illustrate this idea, we need to give the following definitions:

Definition 3.1. A **graph** is an ordered pair $G = (V, E)$ comprising:

- V, a set of vertices (also referred to as **nodes**) and
- $E \subseteq \{\{x, y\} \,|x, y \in V \text{ and } x \neq y\}$, a set of **edges** which are **unordered** pairs of vertices.

Definition 3.2. A **tree** is a graph $G = (V, E)$, which consists of a set of nodes V and edges E, in which any two branches or nodes are connected by exactly one path.

Definition 3.3. A **directed graph** is an ordered pair $G = (V, E)$ comprising:

- V, a set of nodes and
- $E \subseteq \{(x, y) \,|\, (x, y) \in V^2 \text{ and } x \neq y\}$, a set of edges which are **ordered** pairs of nodes.

Definition 3.4. Let $G = (V, E)$ be a directed graph.

– If $t_1, t_2 \in V$ are two nodes and there exists an edge $e = (t_1, t_2) \in E$, then node t_1 is said to be the **parent node** of t_2 and t_2 is said to be the **child node** of t_1.

– If for a node $t_0 \in V$ there exist no parent nodes, then t_0 is called **root** node and G is a **rooted tree**.

– If a node $t \in V$ has one or more children, then t is called **internal**

– If a node $t \in V$ has no children, then t is called **terminal** or **leaf**.

Given these definitions, we can formally define decision trees as follows:

Definition 3.5. A **decision tree** is a model $\phi : \mathcal{X} \to \mathcal{Y}$ represented by a rooted tree $G = (V, E)$, where every node $t \in V$ representes a subspace $\mathcal{X}_t \subseteq \mathcal{X}$ of the input space, with the root node t_0 corresponding to $\mathcal{X}$ itself.

Definition 3.6. Let ϕ be a decision tree and $G = (V, E)$ the corresponding rooted tree. Let also $t \in V$ be a node corresponding to a subset $\mathcal{X}_t$ of the input space $\mathcal{X}$. Then, a **split** s of node t is a rule taken from a set of questions (or tests) $\mathcal{Q}$, which divides t into m ($m \geq 2$) children t_l (for $l = 1, 2, \ldots m$). For $m = 2$, split s_t is further referred to as **binary**, whilst for $m > 2$, s_t is called **multiway**.

Remark 3.1. Note that given the equivalence between nodes t and subsets $\mathcal{X}_t$, a split s is also a rule partitioning $\mathcal{X}_t$ into m $(m \geq 2)$ subsets $\mathcal{X}_{t_l}$ (for $l = 1, 2, \dots m$).

Regarding the above formulation, we can now briefly describe the decision tree algorithm for the case where only binary splits are used. In particular, given a learning set $\mathcal{L}$, we start by partitioning the observations of the root node t_0 according to a split s taken from a set of questions $\mathcal{Q}$. Most often, these questions have the form "Does $\mathbf{x} \in \mathcal{X}_A$?", where $\mathcal{X}_A \subset \mathcal{X}$. As a result, we get a partition of t_0 consisting of two subsets of $\mathcal{L}$; one containing the observations that do belong to $\mathcal{X}_A$ and one containing the observations that do not. We then repeat this process for each of the resulting nodes. That is, for each node t we divide the corresponding subspace $\mathcal{X}_t$ into two subspaces $\mathcal{X}_{t_L} = \mathcal{X}_t \cap \mathcal{X}_A$ and $\mathcal{X}_{t_R} = \mathcal{X}_t \cap (\mathcal{X} \setminus \mathcal{X}_A)$. This loop is halted once a *stopping criterion* is met and, as an output, we get a set of terminal nodes, which corresponds to a partition of the input space $\mathcal{X}$. The last step of the construction procedure consists in assigning a prediction $\hat{y}_t$ at each terminal node.

Algorithm 3.1. Prediction for a new instance $\mathbf{x}$

```
1: function PREDICT($\phi$, $\mathbf{x}$)
2:     $t = t_0$
3:     while $t$ is not a terminal node do
4:         $t$ = the child node $t'$ of $t$ such that $\mathbf{x} \in \mathcal{X}_{t'}$
5:     end while
6:     return $\hat{y}_t$
7: end function
```

Once the decision tree is constructed, it is ready to be used to predict the output value of a new instance $\mathbf{x}$. To do this, we start from the root node, as Algorithm 3.1 indicates, and we propagate $\mathbf{x}$ down the tree until it reaches a terminal node. We then use as a prediction for the output value of $\mathbf{x}$, the value assigned to that node during the construction of the tree.

To better understand the context described so far, we will use a toy regression problem. Figure 3.1 presents the decision tree built over the artificially produced learning set $\mathcal{L}$ of Table 3.1, which consists of two numerical features X_1, X_2 taking values from the input space $\mathcal{X} = [0, 10] \times [0, 10]$ and one output variable Y taking values from $\mathcal{Y} = [20, 140]$. We see that the resulting tree consists of 5 nodes; the root node, an internal node and 3 terminal nodes. Terminal nodes are denoted by rectangles, while the root and the internal nodes are denoted by circles. As foretold, the construction starts from the root node, which in this case is split according to X_2. The splitting rule for that node is "Does $\mathbf{x} \in \mathcal{X}_A$?", where $\mathcal{X}_A = [0, 10] \times [0, 5.75]$. This results in two child nodes; t_1, for which the stopping criterion is met and t_2, which is further split to give the nodes t_3 and t_4. The corresponding partition produced by this tree, as well as the assigned values to each terminal node, can be shown in Figure 3.2. Suppose, now, that we want to predict the output value for a new instance $\mathbf{x} = (6, 6)'$.

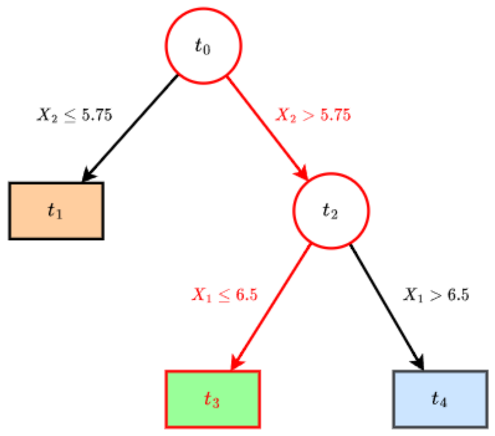

FIGURE 3.1: The binary decision tree ϕ built using the data of Table 3.1. The root followed by $\mathbf{x} = (6,6)'$ when propagated through ϕ, is marked with red color.

Implementing Algorithm 3.1, we see that $\mathbf{x}$ follows the root denoted by red in Figure 3.1 ending up in node t_3 and, hence, the prediction for it is 65.

The next sections of this chapter are devoted to answering the following questions that may have arisen so far:

- How do we choose which feature will be used to split each node?
- Once we have found that feature, how do we choose the cut-off point?
- What are the stopping criteria used to halt the construction process?
- How do we choose the value assigned to each terminal node?

i	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
X_{i1}	6.0	3.0	1.0	2.5	9.5	4.2	8.0	1.5	9.0	7.0	5.0	4.6	6.5	4.0	5.4	4.5
X_{i2}	4.0	4.0	3.0	2.0	1.0	2.9	3.0	1.5	3.5	4.5	4.5	3.8	2.0	4.5	2.7	8.0
Y_i	20	20	25	25	25	25	30	30	30	30	35	35	40	40	40	50
i	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31	32
X_{i1}	1.5	5.0	2.5	4.3	4.0	1.5	3.5	3.0	2.0	8.5	9.0	8.8	8.3	9.5	8.0	9.6
X_{i2}	9.5	9.0	9.0	9.3	7.0	8.5	8.5	8.2	8.0	7.0	9.0	8.3	9.1	9.5	8.0	7.5
Y_i	50	60	60	65	70	70	70	75	80	115	125	125	130	135	140	140

TABLE 3.1: The data used in the toy regression example

3.2 Induction of decision trees

Learning an ideal decision tree consists in finding a partition that best captures the true relationship between the inputs and the outputs. Since in most cases this is unknown, decision trees are content with finding a model that partitions the given learning set as well as possible. We should note,

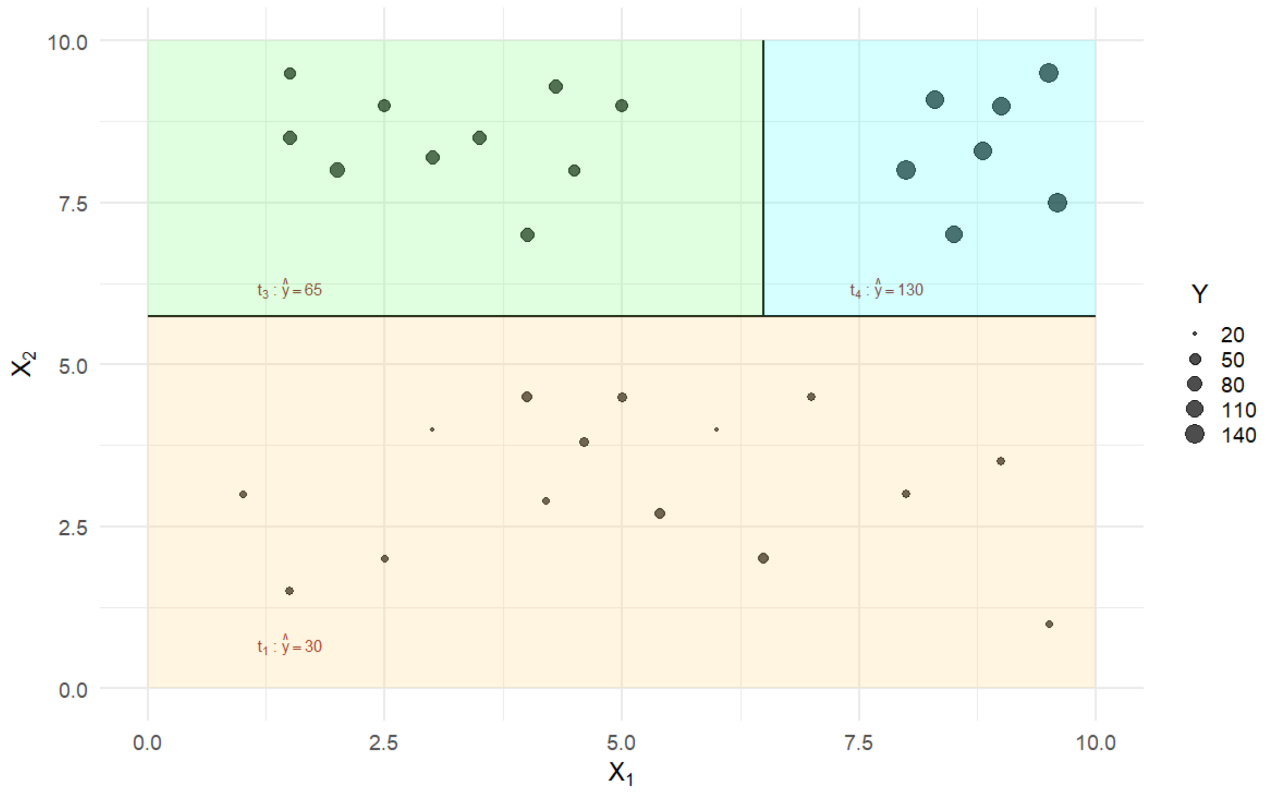

FIGURE 3.2: The partition of the input space $\mathcal{X} = [0, 10] \times [0, 10]$ induced by the tree of Figure 3.1. Points are plotted with respect to their output value, that is smaller points indicate small output values.

though, that for computational and interpretability reasons small trees are preferred over large ones. Therefore, among all trees that may explain the learning set equally best, we want our decision tree algorithm to find the smallest one.

Unfortunately, Hyafil and Rivest (1976) have proven that the task of finding an optimal tree is an NP-complete problem, and as such, it has no computationally efficient solution. That is why, most popular decision tree algorithms such as CART and C4.5 seek near-optimal decision trees using heuristics. We should point out that research towards finding globally optimal trees has been conducted as well. In particular, Bennett (1995) used an iterative linear programming algorithm to find globally optimal classification trees , while more recent studies formulated the objective of finding optimal trees as a mixed-integer programming problem which in many cases is computationally tractable (Bertsimas and Dunn, 2017; Bertsimas et al., 2021). Nevertheless, despite their high predictive performance and the rapid increase of computational power in recent years, scalability to large datasets is still the most salient obstacle with globally optimal trees.

This thesis follows the framework of the more traditional algorithms CART, C4.5, and ID3, according to which the heuristic used to reach near-optimal partitions consists in splitting each node in such a way that the offspring nodes are more homogeneous or *purer* with respect to the outputs. To quantify how pure a node is we give the following definition:

Definition 3.7 (Impurity measure)**.** Let $\phi : \mathcal{X} \to \mathcal{Y}$ be a decision tree and $G = (V, E)$ its associated rooted tree. An **impurity measure** is a function $i : V \to \mathbb{R}$, which measures the goodness of a node $t \in V$.

Under this definition, a pure node t is one with small values of $i(t)$ and, correspondingly, a good split is one that either guarantees that the *overall impurity* of the child nodes will be as small as possible or, equivalently, maximizes the *impurity decrease*.

Definition 3.8. Let $\phi : \mathcal{X} \to \mathcal{Y}$ be a decision tree, $G = (V, E)$ its associated rooted tree, and $s \in \mathcal{Q}$ a binary split dividing a node t into a left node t_L and a right node t_R. Let also p_L (respectively p_R) be the proportion $\frac{n_{t_L}}{n_t}$ (respectively $\frac{n_{t_R}}{n_t}$) of learning samples from $\mathcal{L}_t$ going to t_L (respectively from $\mathcal{L}_t$ to t_R) and n_t be the size of $\mathcal{L}_t$. Then, the **overall impurity** of the offspring nodes for s is:

$$\mathcal{D}(s, t) = p_L i(t_L) + p_R i(t_R). \tag{3.1}$$

Accordingly, the **impurity decrease** of s is:

$$\Delta i(s, t) = i(t) - \mathcal{D}(s, t). \tag{3.2}$$

Remark 3.2. Given a node t, it is obvious from equations 3.1,3.2 that finding a split s^* that minimizes the overall impurity is equivalent to finding the split that maximizes the impurity decrease:

$$s^* = \arg\min_{s \in \mathcal{Q}} \mathcal{D}(s, t) = \arg\max_{s \in \mathcal{Q}} \Delta i(s, t).$$

It is clear that a decision tree with terminal nodes that can not be made purer leads to accurate prediction over the learning set. It is also clear that by iteratively choosing the split that maximizes equation 3.2 at each node, the greedy assumption of making that decision tree as small as possible is satisfied. As we will see in section 3.5, forcing terminal nodes to be pure might lead to overfitting, and halting the construction earlier might be a better strategy.

In Algorithm 3.2 the greedy induction of a decision tree with binary splits is formally presented. Note, however, that this algorithm can be naturally expanded to the case of multiway splits. The rest of this chapter explores specific parts of this algorithm in more detail.

Algorithm 3.2. Greedy induction of a binary decision tree

```
 1: function BUILDDECISIONTREE($\mathcal{L}$)
 2:     Create a decision tree $\phi$ with root node $t_0$
 3:     Create an empty list $S$ of open nodes $(t, \mathcal{L})$
 4:     Append $(t_0, \mathcal{L})$ to $S$
 5:     while $S$ is not empty do
 6:         $(t, \mathcal{L})$ = the last appended element of $S$
 7:         if the stopping criterion is met for $t$ then
 8:             $\hat{\mathbf{y}}_t$ = some constant value
 9:         else
10:             Find the split on $\mathcal{L}_t$ that maximizes impurity decrease
                    $s^* = \arg\max_{s \in \mathcal{Q}} \Delta i(s, t)$
11:             Partition $\mathcal{L}_t$ into $\mathcal{L}_{t_L} \cup \mathcal{L}_{t_R}$ according to $s^*$
12:             Create the left child node $t_L$ of $t$
13:             Create the right child node $t_R$ of $t$
14:             Append $(t_R, \mathcal{L}_{t_R})$ to $S$
15:             Append $(t_L, \mathcal{L}_{t_L})$ to $S$
16:         end if
17:     end while
18:     return $\phi$
19: end function
```

3.3 Assignment rules

In this section, we discuss line 8 of Algorithm 3.2. More specifically, once a node t has been declared as terminal, we have to label it with a constant prediction $\hat{\mathbf{y}}_t$. As already mentioned, $\hat{\mathbf{y}}_t$ will be used as a prediction for the output value for every instance $\mathbf{x}$ falls in t after propagated down tree ϕ. In other words, each node t constitutes a naive model defined locally on $\mathcal{X}_t \times \mathcal{Y}$ with constant predictions $\hat{\mathbf{y}}_t$. This division of ϕ into simpler models facilitates the minimization of the generalization error since this task is mostly reduced to the minimization of the individual simpler models. To illustrate this point, let $\mathcal{T} \subseteq V$ denote the set of terminal nodes in ϕ. Then:

$$Err(\phi) = \mathbb{E}_{X,Y}\left[L\left(Y, \phi\left(X\right)\right)\right] = \sum_{t \in \mathcal{T}} P(X \in \mathcal{X}_t)\mathbb{E}_{X,Y|t}\left[L\left(Y, \hat{\mathbf{y}}_t\right)\right] \tag{3.3}$$

From equation 3.3 we see that minimizing the local generalization error $\mathbb{E}_{X,Y|t}\left[L\left(Y, \hat{\mathbf{y}}_t\right)\right]$ separately for each t, results in minimization of $Err(\phi)$ as well.

For regression, where the squared error loss function is used, we have:

$$\hat{\mathbf{y}}_t^* = \arg\min_{\hat{\mathbf{y}}_t \in \mathcal{Y}} \mathbb{E}_{X,Y|t}\left[\left(Y - \hat{\mathbf{y}}_t\right)'\left(Y - \hat{\mathbf{y}}_t\right)\right] = \mathbb{E}_{X,Y|t}\left[Y\right] \tag{3.4}$$

Without knowledge of the theoretical probability distribution $P(X, Y)$, solving equation 3.4 explicitly is infeasible, and, thus, we approximate its solution using estimates of the local generalization error:

$$\hat{\mathbf{y}}_t = \arg\min_{\hat{\mathbf{y}} \in \mathcal{Y}} \frac{1}{n_t} \sum_{(\mathbf{x},\mathbf{y}) \in \mathcal{L}_t} (\mathbf{y} - \hat{\mathbf{y}}_t)'(\mathbf{y} - \hat{\mathbf{y}}_t) = \frac{1}{n_t} \sum_{(\mathbf{x},\mathbf{y}) \in \mathcal{L}_t} \mathbf{y} \tag{3.5}$$

We should make clear that by approximating $P(X \in \mathcal{X}_t)$ and $\mathbb{E}_{X,Y|t}\left[L\left(Y, \hat{\mathbf{y}}_t\right)\right]$ with their empirical estimates $p(t) := \frac{n_t}{n}$ and $\frac{1}{n_t} \sum_{(\mathbf{x},\mathbf{y}) \in \mathcal{L}_t} (\mathbf{y} - \hat{\mathbf{y}}_t)'(\mathbf{y} - \hat{\mathbf{y}}_t)$ respectively, equation 3.3 coincides with $\widehat{Err}^{train}(\phi)$ and, hence, equation 3.5, which is also referred to as *assignment rule*, minimizes the training error rather than the generalization error.

Proposition 3.1. For any non-empty split of a terminal node $t \in \mathcal{T}$ into t_L and t_R, resulting in a new tree ϕ' with set of terminal nodes $\mathcal{T}'$ where $\hat{y}_{t_L}$ and $\hat{y}_{t_R}$ are assigned with rule 3.5, it holds that:

$$\widehat{Err}^{train}(\phi) \geq \widehat{Err}^{train}(\phi') \tag{3.6}$$

with equality if $\hat{y}_{t_L} = \hat{y}_{t_R} = \hat{y}_t$.

Proof. In order to prove 3.6 we will start from it and we will end up with something that necessarily holds.

$$\widehat{Err}^{train}(\phi) \geq \widehat{Err}^{train}(\phi')$$

$$\sum_{t \in \mathcal{T}} p(t) \left(\frac{1}{n_t} \sum_{(\mathbf{x},\mathbf{y}) \in \mathcal{L}_t} (\mathbf{y} - \hat{\mathbf{y}}_t)' (\mathbf{y} - \hat{\mathbf{y}}_t) \right) \geq \sum_{t \in \mathcal{T}'} p(t) \left(\frac{1}{n_t} \sum_{(\mathbf{x},\mathbf{y}) \in \mathcal{L}_t} (\mathbf{y} - \hat{\mathbf{y}}_t)' (\mathbf{y} - \hat{\mathbf{y}}_t) \right)$$

$$\sum_{(\mathbf{x},\mathbf{y}) \in \mathcal{L}_t} (\mathbf{y} - \hat{\mathbf{y}}_t)' (\mathbf{y} - \hat{\mathbf{y}}_t) \geq \sum_{(\mathbf{x},\mathbf{y}) \in \mathcal{L}_{t_L}} (\mathbf{y} - \hat{\mathbf{y}}_{t_L})' (\mathbf{y} - \hat{\mathbf{y}}_{t_L}) + \sum_{(\mathbf{x},\mathbf{y}) \in \mathcal{L}_{t_R}} (\mathbf{y} - \hat{\mathbf{y}}_{t_R})' (\mathbf{y} - \hat{\mathbf{y}}_{t_R})$$

Using that

$$\hat{\mathbf{y}}_t = \frac{1}{n_t} \sum_{(\mathbf{x},\mathbf{y}) \in \mathcal{L}_t} \mathbf{y}, \qquad \hat{\mathbf{y}}_{t_L} = \frac{1}{n_{t_L}} \sum_{(\mathbf{x},\mathbf{y}) \in \mathcal{L}_{t_L}} \mathbf{y}, \qquad \hat{\mathbf{y}}_{t_R} = \frac{1}{n_{t_R}} \sum_{(\mathbf{x},\mathbf{y}) \in \mathcal{L}_{t_R}} \mathbf{y},$$

the last inequality is equivalent to

$$\cancel{\sum_{(\mathbf{x},\mathbf{y}) \in \mathcal{L}_t} \mathbf{y}'\mathbf{y}} - n_t \hat{\mathbf{y}}_t' \hat{\mathbf{y}}_t \geq \cancel{\sum_{(\mathbf{x},\mathbf{y}) \in \mathcal{L}_{t_L}} \mathbf{y}'\mathbf{y}} - n_{t_L} \hat{\mathbf{y}}_{t_L}' \hat{\mathbf{y}}_{t_L} + \cancel{\sum_{(\mathbf{x},\mathbf{y}) \in \mathcal{L}_{t_R}} \mathbf{y}'\mathbf{y}} - n_{t_R} \hat{\mathbf{y}}_{t_R}' \hat{\mathbf{y}}_{t_R} \tag{3.7}$$

Let us now define for a node t the following vector:

$$\mathbf{s}_t := \sum_{(\mathbf{x},\mathbf{y}) \in \mathcal{L}_t} \mathbf{y} = n_t \hat{\mathbf{y}}_t$$

Then, using that by definition $\mathbf{s}_t = \mathbf{s}_{t_L} + \mathbf{s}_{t_R}$, we get the following:

$$\begin{aligned}\frac{\mathbf{s}_t'\mathbf{s}_t}{n_t} &\leq \frac{\mathbf{s}_{t_L}'\mathbf{s}_{t_L}}{n_{t_L}} + \frac{\mathbf{s}_{t_R}'\mathbf{s}_{t_R}}{n_{t_R}} \\ \frac{\left(\mathbf{s_{t_L}} + \mathbf{s_{t_R}}\right)'\left(\mathbf{s_{t_L}} + \mathbf{s_{t_R}}\right)}{n_{t_L} + n_{t_R}} &\leq \frac{\mathbf{s}_{t_L}'\mathbf{s}_{t_L}}{n_{t_L}} + \frac{\mathbf{s}_{t_R}'\mathbf{s}_{t_R}}{n_{t_R}}\end{aligned}$$

With some simple calculus, we get:

$$\begin{aligned}\frac{n_{t_R}\left(n_{t_L} + n_{t_R}\right)\mathbf{s}_{t_L}'\mathbf{s}_{t_L} + n_{t_L}\left(n_{t_L} + n_{t_R}\right)\mathbf{s}_{t_R}'\mathbf{s}_{t_R} - n_{t_L}n_{t_R}\left(\mathbf{s}_{t_L}'\mathbf{s}_{t_L} + \mathbf{s}_{t_R}'\mathbf{s}_{t_L} + \mathbf{s}_{t_L}'\mathbf{s}_{t_R} + \mathbf{s}_{t_R}'\mathbf{s}_{t_R}\right)}{n_{t_L}n_{t_R}\left(n_{t_L} + n_{t_R}\right)} &\geq 0 \\ n_{t_R}\left(n_{t_L} + n_{t_R}\right)\mathbf{s}_{t_L}'\mathbf{s}_{t_L} + n_{t_L}\left(n_{t_L} + n_{t_R}\right)\mathbf{s}_{t_R}'\mathbf{s}_{t_R} - n_{t_L}n_{t_R}\left(\mathbf{s}_{t_L}'\mathbf{s}_{t_L} + \mathbf{s}_{t_R}'\mathbf{s}_{t_L} + \mathbf{s}_{t_L}'\mathbf{s}_{t_R} + \mathbf{s}_{t_R}'\mathbf{s}_{t_R}\right) &\geq 0 \\ n_{t_L}n_{t_R}\mathbf{s}_{t_L}'\mathbf{s}_{t_R} - n_{t_R}^2\mathbf{s}_{t_L}'\mathbf{s}_{t_L} + n_{t_L}n_{t_R}\mathbf{s}_{t_R}'\mathbf{s}_{t_L} - n_{t_L}^2\mathbf{s}_{t_R}'\mathbf{s}_{t_R} &\geq 0 \\ \left(n_{t_R}\mathbf{s}_{t_L} - n_{t_L}\mathbf{s}_{t_R}\right)'\left(n_{t_R}\mathbf{s}_{t_L} - n_{t_L}\mathbf{s}_{t_R}\right) &\geq 0\end{aligned}$$

The latter inequality is true since the left-hand of the last inequality is a sum of squares. Equality holds if $n_{t_R}\mathbf{s}_{t_L} = n_{t_L}\mathbf{s}_{t_R}$, which is equivalent to $\hat{y}_{t_L} = \hat{y}_{t_R}$. This concludes the proof.

Proposition 3.1 indicates that growing a decision tree as much as possible is a good strategy for minimizing its training error. However, this does not mean that the generalization error will be small as well. In fact, minimizing the training error is more likely to lead to large generalization error values and, thus, it is necessary to find a good compromise between a decision tree that is neither too shallow nor too deep. In this direction, halting the constructing procedure earlier is an option and the most common ways to achieve this are presented in section 3.5.

3.4 Splitting rules

In this section, we discuss line 10 of Algorithm 3.2. In other words, assuming that the stopping criterion is not met for node t, we want to find the best split s^*. Notwithstanding that research has been conducted towards decision trees with multiway splits (Biggs et al., 1991; Fulton et al., 1995; Kim and Loh, 2001; Liu et al., 2020), binary splits are often preferred in practice. Indeed, although using multiway splits might yield decision trees that are smaller and more interpretable than the conventional ones built using binary splits, in many cases finding multiple cut-off points at once is computationally intractable. For this reason, the most common decision tree algorithms focus on binary splits and so will this thesis.

Hence, following the discussion of section 3.2, the best split s^* for a node t is the one that maximizes equation 3.2. Let us now denote by $\mathcal{S}$ the set of possible splits s. It is clear that as soon as $\mathcal{X}_t$ is infinitely large, then $\mathcal{S}$ is infinitely large as well, indicating that we have to choose the best split

s^* from an infinite set of values. To address this issue, we usually assume that s^* - or at least a good enough approximation - lives in a family $\mathcal{Q} \subset \mathcal{S}$ of candidate splits of restricted structure.

3.4.1 Ordered variables

For a set $\{X_1, \ldots, X_p\}$ of ordered variables, we distinguish between two different choices for $\mathcal{Q}$.

Axis-parallel split

The first and most common choice contains the class of *axis-parallel* splits like those constructed in the toy regression example of section 3.1 (see Figure 3.2). More formally, let us denote by $\mathcal{Q}(X_k)$ the set of non-crossing axis-parallel partitions on X_k:

$$\mathcal{Q}(X_k) = \left\{ s_k^v = \{\, \{\mathbf{x}|x_k \leq v\}\,, \{\mathbf{x}|x_k > v\} \,\}|v \in \mathcal{X}_k \right\}. \tag{3.8}$$

Then, for a node t the best split s^* is sought in $\mathcal{Q}$, where

$$\mathcal{Q} = \left\{ s|s \in \bigcup_{k=1}^{p} \mathcal{Q}(X_k), \mathcal{L}_{t_L} \neq \emptyset, \mathcal{L}_{t_R} \neq \emptyset \right\}.$$

In other words, for a node t we choose the best split s^* by finding the best split s_k^* for each variable X_k and selecting the best among them:

$$s^* = \underset{\substack{s_k^* \\ k=1,\ldots,p}}{\arg\max}\, \Delta i(s_k^*, t) \tag{3.9}$$

$$s_k^* = \underset{\substack{s \in \mathcal{Q}(X_k) \\ \mathcal{L}_{t_L}, \mathcal{L}_{t_R} \neq \emptyset}}{\arg\max}\, \Delta i(s, t) \tag{3.10}$$

Therefore, finding the best axis-parallel split simplifies finding the threshold v^* that results in a partition that maximizes the impurity decrease.

If we denote by $\mathcal{X}_{k|t} = \{x_k|(\mathbf{x}, \mathbf{y}) \in \mathcal{L}_t\}$ the unique values of X_k within the nodes samples in node t, then there exist $|\mathcal{X}_{k|t}| - 1$ possible binary partitions of $\mathcal{L}_t$ into two non-empty sets $\mathcal{L}_{t_L}, \mathcal{L}_{t_R}$, where:

$$\mathcal{L}_{t_L} = \{(\mathbf{x}, \mathbf{y})|(\mathbf{x}, \mathbf{y}) \in \mathcal{L}_t, x_k \leq v\}$$
$$\mathcal{L}_{t_R} = \{(\mathbf{x}, \mathbf{y})|(\mathbf{x}, \mathbf{y}) \in \mathcal{L}_t, x_k > v\}\,.$$

That is because for each $v \in \mathcal{X}_{k|t}$ we get a different partition of $\mathcal{L}_t$, except for the largest value of $\mathcal{X}_{k|t}$, which leads to an invalid partition, since in that case, it holds that $\mathcal{L}_{t_R} = \emptyset$.

We should note that if v_m, v_{m+1} are two immediately consecutive values in $\mathcal{X}_{k|t}$, then the partition of $\mathcal{L}_t$ produced by each value $v \in [v_m, v_{m+1})$ is the same in the sense that both $\mathcal{L}_{t_L}$ and $\mathcal{L}_{t_R}$ remain unchanged in that interval. Thus, all these values attain the same impurity decrease Δi. However, in terms of generalization error, all these thresholds are different, since they do not produce the same partition of $\mathcal{X}_t$.

In order to find which partition maximizes the impurity decrease among the $|\mathcal{X}_{k|t}| - 1$ possible ones, we arbitrarily choose a representative threshold v'_m for each interval $[v_m, v_{m+1})$. Two commonly used choices are the following:

- Breiman (2001) in the original random forest algorithm suggests choosing as the interval representative the mid-cut-point $v'_m = \frac{v_m + v_{m+1}}{2}$ (see also Figure 3.2). This choice is implemented in the `randomForest` package (Liaw et al., 2002) in `R` (R Core Team, 2023).
- An alternate consists in using as a representative the first value of the interval, that is, $v'_m = v_m$. This choice is implemented in the `randomForestSRC` package (Ishwaran et al., 2021b).

Regardless of which method is followed, the $|\mathcal{X}_{k|t}| - 1$ representatives are then compared in terms of impurity decrease, and the best one is chosen.

Oblique splits

An alternate to axis-parallel splits is the *oblique* or *hyperplane* splits. As their name indicates, contrary to axis-parallel splits, this approach results in partitions of the input space consisting of hyperplanes rather than hyperrectangles. To illustrate the latter, let us recall the regression problem of section 3.2. Figure 3.3 depicts the partition induced by an oblique tree. Mathematically, a hyperplane can be expressed by equations of the form: $\sum_{j=1}^{p} w_j X_j = v$. Therefore, the set of oblique splits from which the best split s^* is chosen is the following:

$$\mathcal{Q} = \left\{ s^{v,\mathbf{w}} = \left\{ \left\{ \mathbf{x} | \mathbf{w}'\mathbf{x} \leq v \right\}, \left\{ \mathbf{x} | \mathbf{w}'\mathbf{x} > v \right\} \right\} | \mathbf{w} \in \mathbb{R}^p, v \in \mathbb{R} \right\}. \tag{3.11}$$

Thus, finding s^* consists in specifying the weight vector $\mathbf{w}$ and constant v. Unfortunately, in most cases, the size of $\mathcal{Q}$ is extremely large and, hence, highly prohibitive to exhaustively explore. For this reason, existing approaches focus on finding a near-optimal oblique split rather than the best one.

The first one, called CART-LC, was introduced by Breiman et al. (1984) who suggested a deterministic hill-climbing approach that sequentially updates the coefficients of the split until a local optimum is reached. To escape local optima, Heath et al. (1993) propose a simulated annealing heuristic that perturbs the hyperplane parameters one at a time. In their algorithm called OC1, Murthy et al.

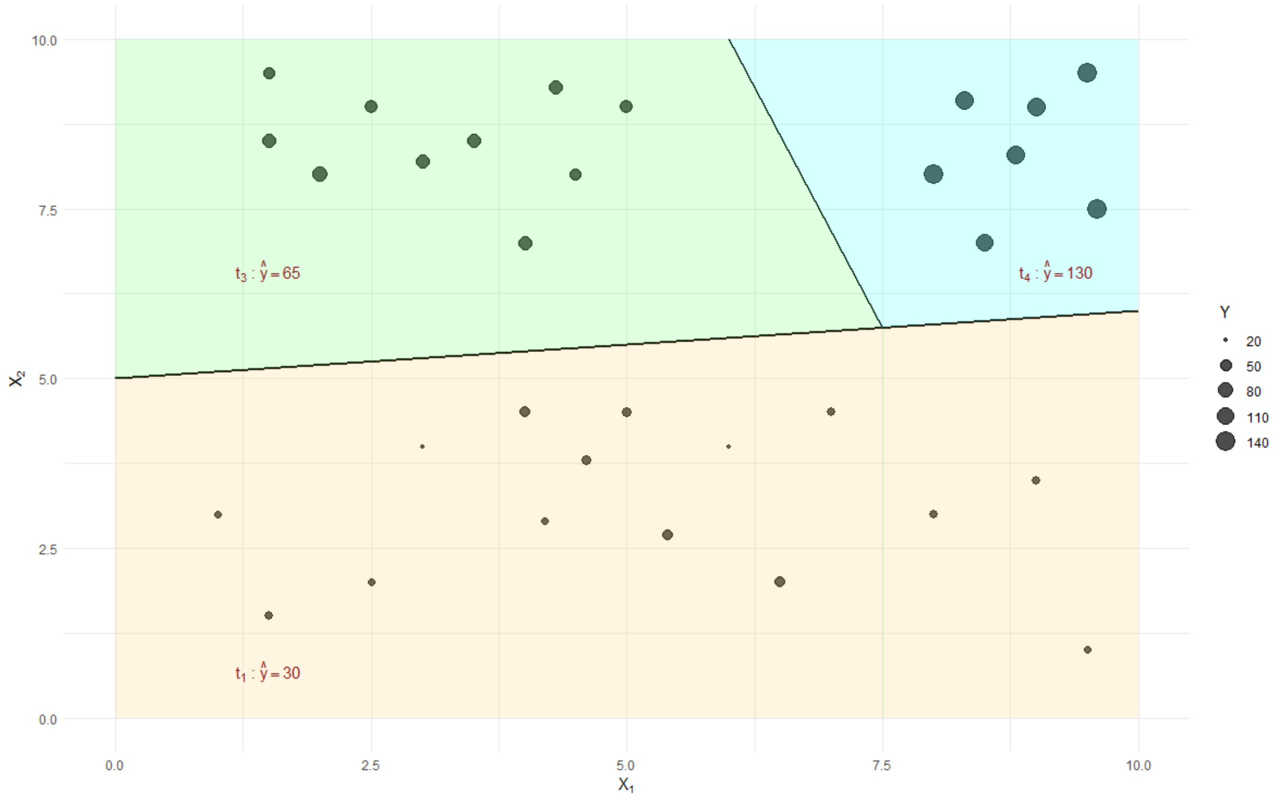

FIGURE 3.3: A partition derived from oblique splits for the data of Table 3.1.

(Murthy et al., 1993, 1994; Murthy, 1996) improve Breiman's hill climbing approach by introducing randomization techniques with the goal of avoiding premature convergence.

Other strategies for finding oblique splits are based on meta-heuristics such as simulated annealing, genetic algorithms or evolutionary algorithms (Cantu-Paz and Kamath, 2003) or algorithms based on logistic regression (Mola and Siciliano, 2002; Truong, 2009), linear discriminants (Loh and Shih, 1997; Li et al., 2003; Siciliano et al., 2008; López-Chau et al., 2013) or Householder transformations (Wickramarachchi et al., 2016). Recently, also mathematical optimization approaches have been proposed for inducing oblique decision trees that neglect the recursive partitioning scheme (Bertsimas and Dunn, 2017; Blanquero et al., 2020). More recently, Bollwein and Westphal (2022) proposed a cross-entropy optimization method for finding oblique splits.

We should note that although the idea of oblique splits is applicable in both classification and regression tasks, little research has been conducted for the latter case.

3.4.2 Categorical variables

For a set $\{X_1, \ldots, X_p\}$ of categorical variables where each variable X_j takes values from $\mathcal{X}_k = \{b_{k_1}, \ldots b_{k_L}\}$, then the set $\mathcal{Q}(X_k)$ of binary splits on X_k is the set of all binary non-empty partitions of $\mathcal{X}_k$:

$$\mathcal{Q}(X_k) = \{\{\{\mathbf{x}|x_k \in \mathcal{B}\}, \{\mathbf{x}|x_k \in \overline{\mathcal{B}}\}\} \,|\mathcal{B} \subset \{b_{k_1}, \ldots, b_{k_L}\}\},$$

where $\overline{\mathcal{B}} = \{b_{k_1}, \ldots, b_{k_L}\} \setminus \mathcal{B}$ is the complementary set of $\mathcal{B}$.

We see that if a categorical variable has L classes, then the number of possible partitions is $2^{L-1} - 1$ which even for small values of L may be infeasible to exhaustively explore. Fortunately, Breiman et al. (1984) showed that for a class of impurity measures the search of the optimal split for any categorical variable can be reduced to $L - 1$ possible partitions. In particular, categorical predictors are first converted into ordered variables, which are then used to form axis-parallel or oblique splits as described in section 3.4.1.

Axis-parallel splits

When we use axis-parallel splits to build a regression tree, categorical features are most often converted into ordered ones by replacing values b_l with the mean output value at b_l, say $\tilde{b}_l$:

$$\tilde{b}_l = \sum_{\substack{(\mathbf{x},\mathbf{y}) \in \mathcal{L}_t \\ \mathbf{x}_k = b_l}} \mathbf{y} \tag{3.12}$$

Once input variables are converted, we use equation 3.10 to find the best threshold $\tilde{b}^*$ which, under the equivalence between $\tilde{b}_l$ and b_l, corresponds to a particular partition of their input space.

To make things clearer, let us consider the following example. Suppose that X_j is a categorical variable with input space $\mathcal{X}_k = \{b_1, b_2, b_3, b_4, b_5\}$. Furthermore, assume that for the corresponding ordered values of $\mathcal{X}_k$ holds that:

$$\tilde{b}_5 \leq \tilde{b}_2 \leq \tilde{b}_4 \leq \tilde{b}_3 \leq \tilde{b}_1$$

and that $\tilde{b}_4 = \tilde{b}^*$. Then, if X_j is eventually the chosen variable to be used to split node t, the resulting child nodes are the following:

$$\mathcal{L}_{t_L} = \{(\mathbf{x},\mathbf{y}) | (\mathbf{x},\mathbf{y}) \in \mathcal{L}_t, x_k \in \mathcal{B}\}$$
$$\mathcal{L}_{t_R} = \{(\mathbf{x},\mathbf{y}) | (\mathbf{x},\mathbf{y}) \in \mathcal{L}_t, x_k \in \overline{\mathcal{B}}\},$$

where $\mathcal{B} = \{b_5, b_2, b_4\}$.

Oblique splits

For decision trees built using hyperplane splits, we note that incorporating categorical features in oblique splits has not been explored to any great extent. Nevertheless, we point out that the QUEST algorithm presented by Loh and Shih (1997) is capable of finding oblique splits using both ordered and categorical features. This algorithm was also used later by Wickramarachchi et al. (2016) and consists in using Pearson's chi-square test to calculate the association between the target and any of the categorical features.

To conclude, in order to find the best split on a set including both ordered and categorical variables, we first convert the latter into ordered ones and we proceed as in section 3.4.1 for the new set of ordered variables.

3.4.3 Regression impurity measures

In this subsection we present different impurity measures to calculate the impurity decrease of a candidate binary split (see equation 3.2).

Univariate regression

For univariate regression problems the local resubstitution error is used as an evaluation criterion:

Definition 3.9. In univariate regression, the impurity function $i_R(t)$ based on the local resubstitution estimate defined on the squared error loss is:

$$i_R(t) = \frac{1}{n_t} \sum_{i=1}^{n_t} (y_i - \hat{y}_t)^2 . \tag{3.13}$$

Considering the discussion of section 3.3 we see that equation 3.13 corresponds to the within node variance of node t, since $\hat{y}_t$ is replaced by $\bar{y}_t$. Accordingly, s^* is the split that maximizes the reduction of variance $\Delta i(s,t)$ in the child nodes.

Multivariate regression

In some cases, we may want to use an input $\mathbf{x}$ to simultaneously predict more than one output variables. In order to construct a decision tree that accurately predicts all output variables, we need to introduce an impurity criterion that somehow takes into account the total variability of the node with respect to all output variables

In this direction, a natural extension of criterion 3.13 is to use the sum of the within node variances for each of the output variables (De'Ath, 2002).

Definition 3.10. In multivariate regression, the impurity function $i_{MR}(t)$ based on the total variance of node t is:

$$i_{MR}(t) = \frac{1}{n_t} \sum_{s=1}^{S} \sum_{i=1}^{n_t} (y_{is} - \hat{y}_{t,s})^2 , \tag{3.14}$$

where $\hat{y}_{t,s}$ is the assignment rule used to predict the value of the s-th response coordinate in node t.

Again, we use the assignment rule 3.5, that is, $\hat{y}_{t,s} = \frac{1}{n_t}\sum_{i=1}^{n_t} y_{is}$, $s = 1, \dots, S$ and, thus, the right hand of equation 3.14 corresponds to the sum of the within node variances.

A more sophisticated approach is the one that uses the *Mahalanobis distance* to exploit any correlation between the output variables. This idea was first used by Segal (1992) for the case of longitudinal data and was later expanded to construct multivariate random forests (Segal and Xiao, 2011). We start by defining Mahalanobis distance (Mahalanobis, 2018):

Definition 3.11 (Mahalanobis distance)**.** Let Y be a S - dimensional variable with mean μ_Y and covariance Σ_Y. Then, the (squared) **Mahalanobis distance** of Y to its mean is defined as follows:

$$D_M(Y) = (Y - \mu_Y)'\Sigma_Y^{-1}(Y - \mu_Y). \tag{3.15}$$

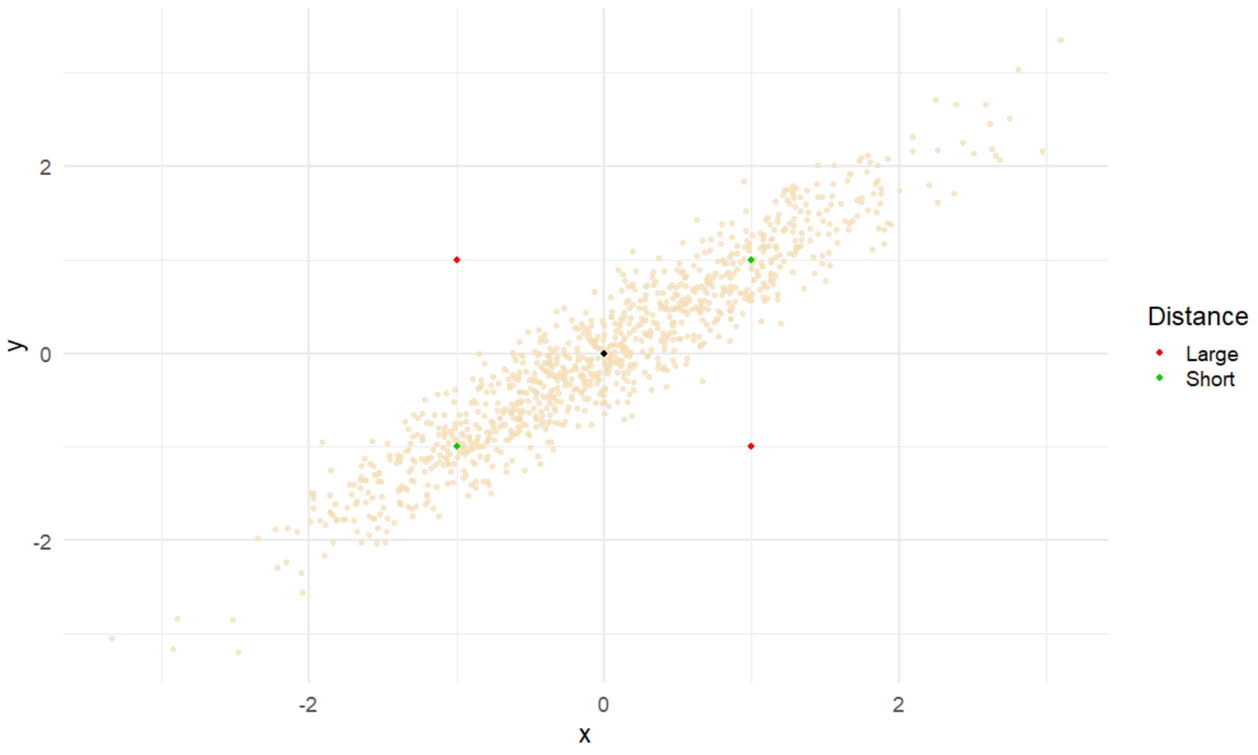

FIGURE 3.4: Illustration of Mahalanobis distance

Contrary to the commonly used Euclidean distance, the use of Σ_Y in equation 3.15, guarantees that the correlation between the coordinates of Y is also taken into account. To better illustrate this, Figure 3.4 depicts a simulated data set generated from a bivariate normal distribution. According to the Euclidean distance, the green and red points are equidistant from the black point in the middle. This is not the case, though, when Mahalanobis distance is used. In particular, the red points have larger Mahalanobis distance values than the green ones, since they do not coincide with the covariance structure of the rest points.

Remark 3.3. Given a learning set $\mathcal{L}_t$, the Mahalanobis distance of a particular observation $\mathbf{y}$ is calculated by replacing the theoretical mean μ and covariance matrix Σ with their sample estimates. Yet, in practice when the size n_t of $\mathcal{L}_t$ becomes small, as is usually the case when growing a decision tree, Σ may be singular. A way to overcome this obstacle is to replace Σ^{-1} with its *generalized inverse* (Penrose, 1955), which always exists. This is the approach followed by Ishwaran et al. (2021d) in the `randomForestSRC` package. The generalized inverse, also known as the Moore-Penrose inverse, is defined as follows:

Definition 3.12. For any matrix $\mathbf{A} \in \mathbb{R}^{n\times p}$ the **generalized inverse** of $\mathbf{A}$ is the unique matrix $\mathbf{A}^{+} \in \mathbb{R}^{p\times n}$ satisfying:

1. $\mathbf{A}\mathbf{A}^{+}\mathbf{A} = \mathbf{A}$
2. $\mathbf{A}^{+}\mathbf{A}\mathbf{A}^{+} = \mathbf{A}^{+}$
3. $(\mathbf{A}\mathbf{A}^{+})^{T} = \mathbf{A}\mathbf{A}^{+}$.

Note that if $\mathbf{A}$ is non-singular, then $\mathbf{A}^{+} = \mathbf{A}^{-1}$.

The following theorem shows that the derivation of the Moore-Penrose inverse is based on the *singular value decomposition* of $\mathbf{A}$ which is defined as follows:

Definition 3.13 (Singular value decomposition)**.** Let $\mathbf{A} \in \mathbb{R}^{n\times p}$ matrix with rank $r \leq \min(n, p)$ where $n \geq p$. The **singular value decomposition** of $\mathbf{A}$ is

$$\mathbf{A} = \mathbf{U}\mathbf{D}\mathbf{V}',$$

where $\mathbf{U} \in \mathbb{R}^{n\times p}$ is an orthonormal matrix ($\mathbf{U}'\mathbf{U} = \mathbf{I}_p$), $V \in \mathbb{R}^{p\times p}$ is an orthogonal matrix ($\mathbf{V}'\mathbf{V} = \mathbf{V}\mathbf{V}' = \mathbf{I}_p$) and $\mathbf{D} \in \mathbb{R}^{p\times p}$ is a diagonal matrix containing the singular values $\{d_1, \ldots, d_p\}$. Without loss of generality, we assume that these are ordered such that

$$d_1 \geq d_2 \geq \cdots \geq d_r > 0, \qquad d_{r+1} = \cdots = d_p = 0.$$

Notice that $d_j > 0$ for $j = 1, \ldots, p$ if $\mathbf{A}$ has full column rank ($r = p$).

Theorem 3.1. The generalized inverse for a matrix $\mathbf{A} \in \mathbb{R}^{n\times p}$ where $n \geq p$ with singular value decomposition $\mathbf{A} = \mathbf{U}\mathbf{D}\mathbf{V}^{\mathbf{T}}$ and $\text{rank}(\mathbf{A}) = r$ is:

$$\mathbf{A}^{+} = \mathbf{V}\mathbf{D}^{+}\mathbf{U}^{\mathbf{T}},$$

where $\mathbf{D}^+$ is the generalized inverse of $\mathbf{D}$ defined as the $p \times p$ diagonal matrix with entries $1/d_k$ for $k = 1, \ldots, r$ and 0 for $k = r+1, \ldots, p$.
If $n < p$ then transpose $\mathbf{A}$ so that the above result applies and transpose the resulting inverse, that is $\mathbf{A}^+ = \left((\mathbf{A}^T)^+ \right)^T$.

In practice, we are interested in symmetric square matrices of the form $\mathbf{Q} = \mathbf{L}'\mathbf{L}$. The generalized inverse for Σ is then easily obtained by Theorem 3.1 by setting $\mathbf{A} = \mathbf{Q}$. Since the number of rows equals the number of columns in $\mathbf{Q}$, the first part of the theorem applies in this case.

Definition 3.14 (Mahalanobis splitting rule)**.** Let t denote a node which we want to split, $\mathbf{L}_t^*$ the corresponding $n_t \times S$ matrix containing the centered outcome values[1], that is:

$$\mathbf{L}_t^* = \begin{pmatrix} (\mathbf{y}_1 - \bar{\mathbf{y}}_t)^T \\ \vdots \\ (\mathbf{y}_{n_t} - \bar{\mathbf{y}}_t)^T \end{pmatrix}$$

and s a binary split dividing t into two child nodes t_L and t_R. Then, the sample covariance matrix for $\mathcal{L}_t$ is $\mathbf{Q}_t^*/n_t$, where $\mathbf{Q}_t^* = (\mathbf{L}_t^*)' \, \mathbf{L}_t^*$, and the overall impurity of t_L and t_R is:

$$\mathcal{D}(s,t) = \frac{n_{t_L}}{n_t} \sum_{i=1}^{n_{t_L}} (\mathbf{y}_i - \hat{\mathbf{y}}_{t_L})' \, (\mathbf{Q}_t^*)^+ \, (\mathbf{y}_i - \hat{\mathbf{y}}_{t_L}) \tag{3.16}$$

$$+ \frac{n_{t_R}}{n_t} \sum_{i=1}^{n_{t_R}} (\mathbf{y}_i - \hat{\mathbf{y}}_{t_R})' \, (\mathbf{Q}_t^*)^+ \, (\mathbf{y}_i - \hat{\mathbf{y}}_{t_R}), \tag{3.17}$$

where, once again, the assignment rule 3.5 is used for $\hat{\mathbf{y}}_{t_L}, \hat{\mathbf{y}}_{t_R}$.

We should note that the incorporation of the correlations among the response variables in the splitting procedure comes with a toll, since the calculation of the generalized inverse as described above, significantly increases computational costs.

3.5 Stopping criteria

In this section, we discuss line 7 of Algorithm 3.2. As we have shown through proposition 3.6, splitting a node in any way reduces the training error of the tree it belongs to. However, growing deep trees with accurate predictions over the given learning set does not necessarily mean that the prediction over a new instance will be accurate too. To prevent this phenomenon, there have been proposed various strategies in the literature. One strategy, known as *pre-pruning*, consists in using

[1]We use * notation to emphasize that the matrix is centered

stopping criteria to halt the recursive partition early and limit the size of the tree, while an alternate referred to as *post-pruning*, is to first fully develop the tree and then prune the branches that degrade the generalization error. For a comprehensive study on pre-and post-pruning see also Shamrat et al. (2021).

While post-pruning, in general, yields better results than pre-pruning in the context of single decision trees, when building ensembles (see chapter 4) pre-pruning is most of the time adequate to produce accurate predictions. Therefore, we will focus our discussion on pre-pruning and, in particular, the stopping criteria used to stop the tree induction early. We distinguish between stopping criteria that are *data driven* and, thus, inherent to the iterative partition procedure, and stopping criteria that are *user defined* and induced to prevent overfitting:

- **Data driven**: A node t is inevitably set as terminal when $\mathcal{L}_t$ can no longer be split, which happens in the following cases:
  - When t is pure, that is, if $\mathbf{y} = \mathbf{y}'$ for all $(\mathbf{x}, \mathbf{y}), (\mathbf{x}', \mathbf{y}') \in \mathcal{L}_t$.
  - When each input variable X_k is locally constant in $\mathcal{L}_t$, that is, if $\mathbf{x}_k = \mathbf{x}'_k$ for all $(\mathbf{x}, \mathbf{y}), (\mathbf{x}', \mathbf{y}') \in \mathcal{L}_t$.
- **User defined**: To avoid overfitting, we usually force node t be terminal in the following cases:
  - When t contains less than N_{min} samples.
  - When the depth of t, d_t, is greater or equal to a threashold d_{max}.
  - When the best split s^* results in a total impurity decrease $p(t)\Delta(s^*, t)$ less than a threashold β.
  - When there is no split of t such that both t_L and t_R count at least N_{leaf} samples.

In practice, the values $N_{min}, d_{max}, \beta, N_{leaf}$ are hyper-parameters that have to be tuned in order to build a tree that adequately captures the underlying structure of the data and generalizes fairly well.

Chapter 4

Random Forests

4.1 Ensemble learning

This section introduces the ensemble learning framework upon which the Random Forest algorithm presented in section 4.3 is built. **Ensemble methods** is a machine learning technique that combines several base models to produce one optimal predictive model. More formally, suppose that we have a learning set $\mathcal{L}$ and an algorithm $\mathcal{A}$ according to which we want to construct a model ϕ. Most machine learning algorithms contain a hyper-parameter θ controlling the execution of $\mathcal{A}$. If $\mathcal{A}$ includes some stochastic steps, then we further assume that θ contains a parameter, usually referred to as **random seed**, determining the execution of these steps. As such, distinct random seeds may result in models that are more or less different from one another. From now on and without loss of generality, we assume that θ only controls the randomness of $\mathcal{A}$. Hence, when we refer to θ, we will actually be referring to the random seed parameter. The role that random seed has in the construction of a model ϕ will be made clearer in section 4.3.1.

Definition 4.1. Let $\mathcal{A}$ be an algorithm including a stochastic procedure, θ be a random seed controlling the execution of $\mathcal{A}$, and $\mathcal{L}$ be a learning set. Then, the model $\phi_{\mathcal{L},\theta}$ constructed using $\mathcal{L}$ and following the implementation details of $\mathcal{A}$ is called **randomized model**.

Now, let us assume that we have a set of M randomized models $\{\phi_{\mathcal{L},\theta_m}\}_{m=1}^{M}$ each built from a different random seed θ_m but all learned on the same data $\mathcal{L}$. From a statistical perspective, θ_m, $m = 1, \dots, M$ are thought to be iid random variables. Ensemble methods combine the predictions of these models into a new one, $\psi_{\mathcal{L},\theta_1,\dots,\theta_M}$ with the intention of reducing the expected generalization error of the individual models. In general, the way the predictions of the base models are aggregated is task and case-dependent.

A generic formulation for the prediction of the ensemble for a new instance $\mathbf{x}^*$ is the following:

$$\psi_{\mathcal{L},\theta_1,\dots,\theta_M}(\mathbf{x}^*) = \frac{\sum_{m=1}^{M} w_m(\mathbf{x}^*)\phi_{\mathcal{L},\theta_m}(\mathbf{x}^*)}{\sum_{m=1}^{M} w_m(\mathbf{x}^*)}, \tag{4.1}$$

where $w_m(\mathbf{x})$ corresponds to the weight assigned to the m-th randomized model for the prediction of $\mathbf{x}$. For regression the most common choice for the weights is setting equal weights to each model, that is, $w_m(\mathbf{x}) = 1$, $m = 1, \ldots, M$. In this case, equation 4.1 simplifies to:

$$\psi_{\mathcal{L},\theta_1,\ldots,\theta_M}(\mathbf{x}) = \frac{1}{M} \sum_{m=1}^{M} \phi_{\mathcal{L},\theta_m}(\mathbf{x}). \tag{4.2}$$

An alternate choice for $w_m(\mathbf{x})$ is presented in section 4.4.

The idea of combining models to improve accuracy has been explored since the 1970s when Tukey et al. (1977) presented an ensemble of two linear regression models. Two years later, Dasarathy and Sheela (1979) suggested the composition of a classifier system using two or more classifiers of different categories. Hansen and Salamon (1990) showed for the first time that the generalization error of neural networks could be reduced when invoking ensembles of neural networks. Around the same time, Schapire (1990) described how combining weak models may outperform one strong model. The next section will provide a more formal explanation of this result.

4.2 Decomposition of generalization error

In this section, we show why combining several weak models may significantly improve predictive accuracy. Let us first recall the expected prediction error of a model $\phi_{\mathcal{L}}$ with respect to a loss function L, that is equation 2.3:

$$Err(\phi_{\mathcal{L}}) = \mathbb{E}_{X,Y}\left[L\left(Y, \phi_{\mathcal{L}}\left(X\right)\right)\right].$$

Accordingly, the expected prediction error of $\phi_{\mathcal{L}}$ at a given point $X = \mathbf{x}$ is:

$$Err\left(\phi_{\mathcal{L}}\left(\mathbf{x}\right)\right) = \mathbb{E}_{Y|X=\mathbf{x}}\left[L\left(Y, \phi_{\mathcal{L}}\left(\mathbf{x}\right)\right)\right].$$

For the rest of this section and for the sake of simplicity, we restrict ourselves to the case of the univariate regression where we assign equal weights to each tree as in equation 4.2.

4.2.1 Bias-variance decomposition of a model

We start by presenting the bias-variance decomposition for an arbitrary model $\phi_{\mathcal{L}}$.

Theorem 4.1 (Geman et al. (1992))**.** For the squared error loss, the bias-variance decomposition of the expected generalization error $\mathbb{E}_{\mathcal{L}}\left[Err\left(\phi_{\mathcal{L}}\left(\mathbf{x}\right)\right)\right]$ at $X = \mathbf{x}$ is

$$\mathbb{E}_{\mathcal{L}}\left[Err\left(\phi_{\mathcal{L}}\left(\mathbf{x}\right)\right)\right] = noise(\mathbf{x}) + bias^2(\mathbf{x}) + var(\mathbf{x}), \tag{4.3}$$

where

$$\begin{aligned} noise(\mathbf{x}) &= Err\left(\varphi_B\left(\mathbf{x}\right)\right) \\ bias^2(\mathbf{x}) &= \left(\varphi_B\left(\mathbf{x}\right) - \mathbb{E}_{\mathcal{L}}\left[\varphi_{\mathcal{L}}\left(\mathbf{x}\right)\right]\right)^2 \\ var(\mathbf{x}) &= \mathbb{E}_{\mathcal{L}}\left[\left(\varphi_{\mathcal{L}}\left(\mathbf{x}\right) - \mathbb{E}_{\mathcal{L}}\left[\varphi_{\mathcal{L}}\left(\mathbf{x}\right)\right]\right)^2\right] \end{aligned}$$

Proof. Let us first show how the expected prediction error of a model $\varphi_{\mathcal{L}}$ at a given point $X = \mathbf{x}$ can be rewritten in terms of the Bayes error:

$$\begin{aligned} Err\left(\varphi_{\mathcal{L}}\left(\mathbf{x}\right)\right) &= \mathbb{E}_{Y|X=\mathbf{x}}\left[\left(Y - \varphi_{\mathcal{L}}\left(\mathbf{x}\right)\right)^2\right] \\ &= \mathbb{E}_{Y|X=\mathbf{x}}\left[\left(Y - \varphi_B(\mathbf{x}) + \varphi_B(\mathbf{x}) + \varphi_{\mathcal{L}}\left(\mathbf{x}\right)\right)^2\right] \\ &= \mathbb{E}_{Y|X=\mathbf{x}}\left[\left(Y - \varphi_B(\mathbf{x})\right)^2\right] + \mathbb{E}_{Y|X=\mathbf{x}}\left[\left(\varphi_B(\mathbf{x}) - \varphi_{\mathcal{L}}(\mathbf{x})\right)^2\right] \\ &+ \mathbb{E}_{Y|X=\mathbf{x}}\left[2\left(Y - \varphi_B(\mathbf{x})\right)\left(\varphi_B(\mathbf{x}) - \varphi_{\mathcal{L}}(\mathbf{x})\right)\right] \\ &= \mathbb{E}_{Y|X=\mathbf{x}}\left[\left(Y - \varphi_B(\mathbf{x})\right)^2\right] + \mathbb{E}_{Y|X=\mathbf{x}}\left[\left(\varphi_B(\mathbf{x}) - \varphi_{\mathcal{L}}(\mathbf{x})\right)^2\right] \\ &= Err\left(\varphi_B\left(\mathbf{x}\right)\right) + \left(\varphi_B(\mathbf{x}) - \varphi_{\mathcal{L}}(\mathbf{x})\right)^2, \end{aligned} \tag{4.4}$$

since $\mathbb{E}_{Y|X=\mathbf{x}}\left[Y - \varphi_B(\mathbf{x})\right] = \mathbb{E}_{Y|X=\mathbf{x}}\left[Y\right] - \varphi_B(\mathbf{x}) = 0$ by definition of the Bayes model. Notice that the first term of equation 4.4 corresponds to the Bayes error at $X = \mathbf{x}$, while the second term represents the discrepancy of $\varphi_{\mathcal{L}}$ from the Bayes model.

Now assuming that the leaning set $\mathcal{L}$ is itself a random variable, we get from equation 4.4 that:

$$\mathbb{E}_{\mathcal{L}}\left[Err\left(\varphi_{\mathcal{L}}(\mathbf{x})\right)\right] = \underbrace{Err\left(\varphi_B\left(\mathbf{x}\right)\right)}_{noise(\mathbf{x})} + \mathbb{E}_{\mathcal{L}}\left[\left(\varphi_B(\mathbf{x}) - \varphi_{\mathcal{L}}(\mathbf{x})\right)^2\right] \tag{4.5}$$

The second term can be decomposed as follows:

$$\begin{aligned} \mathbb{E}_{\mathcal{L}}\left[\left(\varphi_B(\mathbf{x}) - \varphi_{\mathcal{L}}(\mathbf{x})\right)^2\right] &= \mathbb{E}_{\mathcal{L}}\left[\left(\varphi_B(\mathbf{x}) - \mathbb{E}_{\mathcal{L}}\left[\varphi_{\mathcal{L}}\left(\mathbf{x}\right)\right] + \mathbb{E}_{\mathcal{L}}\left[\varphi_{\mathcal{L}}\left(\mathbf{x}\right)\right] - \varphi_{\mathcal{L}}(\mathbf{x})\right)^2\right] \\ &= \mathbb{E}_{\mathcal{L}}\left[\left(\varphi_B(\mathbf{x}) - \mathbb{E}_{\mathcal{L}}\left[\varphi_{\mathcal{L}}\left(\mathbf{x}\right)\right]\right)^2\right] + \mathbb{E}_{\mathcal{L}}\left[\left(\mathbb{E}_{\mathcal{L}}\left[\varphi_{\mathcal{L}}\left(\mathbf{x}\right)\right] - \varphi_{\mathcal{L}}(\mathbf{x})\right)^2\right] \\ &+ \mathbb{E}_{\mathcal{L}}\left[2\left(\varphi_B(\mathbf{x}) - \mathbb{E}_{\mathcal{L}}\left[\varphi_{\mathcal{L}}\left(\mathbf{x}\right)\right]\right)\left(\mathbb{E}_{\mathcal{L}}\left[\varphi_{\mathcal{L}}\left(\mathbf{x}\right)\right] - \varphi_{\mathcal{L}}(\mathbf{x})\right)\right] \\ &= \mathbb{E}_{\mathcal{L}}\left[\left(\varphi_B(\mathbf{x}) - \mathbb{E}_{\mathcal{L}}\left[\varphi_{\mathcal{L}}\left(\mathbf{x}\right)\right]\right)^2\right] + \mathbb{E}_{\mathcal{L}}\left[\left(\mathbb{E}_{\mathcal{L}}\left[\varphi_{\mathcal{L}}\left(\mathbf{x}\right)\right] - \varphi_{\mathcal{L}}(\mathbf{x})\right)^2\right] \\ &= \underbrace{\left(\varphi_B(\mathbf{x}) - \mathbb{E}_{\mathcal{L}}\left[\varphi_{\mathcal{L}}\left(\mathbf{x}\right)\right]\right)^2}_{bias^2(\mathbf{x})} + \underbrace{\mathbb{E}_{\mathcal{L}}\left[\left(\mathbb{E}_{\mathcal{L}}\left[\varphi_{\mathcal{L}}\left(\mathbf{x}\right)\right] - \varphi_{\mathcal{L}}(\mathbf{x})\right)^2\right]}_{var(\mathbf{x})}, \end{aligned}$$

since $\mathbb{E}_{\mathcal{L}}\left[\mathbb{E}_{\mathcal{L}}\left[\varphi_{\mathcal{L}}\left(\mathbf{x}\right)\right] - \varphi_{\mathcal{L}}(\mathbf{x})\right] = \mathbb{E}_{\mathcal{L}}\left[\varphi_{\mathcal{L}}\left(\mathbf{x}\right)\right] - \mathbb{E}_{\mathcal{L}}\left[\varphi_{\mathcal{L}}\left(\mathbf{x}\right)\right] = 0$.

Remark 4.1. Let us now give an interpretation for each term in theorem 4.1.

- $noise(\mathbf{x})$: the residual error. This term is independent of both the learning algorithm and the learning set and provides for any model a theoretical lower bound on its generalization error.
- $bias^2(\mathbf{x})$: a measure of discrepancy between the average prediction and the prediction of the Bayes model.
- $var(\mathbf{x})$: a measure of the variability of the predictions at $X = \mathbf{x}$ due to the variability of the models built using different learning sets.

4.2.2 Bias-variance decomposition of an ensemble

From theorem 4.1 we see that the generalization error of a model decomposes into three terms. Therefore, reducing one of them while keeping the others stable, will result in an improvement in terms of predictive accuracy. That is exactly what ensemble methods do. In particular, as shown below, they combine several randomized models to reduce the variance term in a way that the bias term remains (almost) the same. The noise term remains the same as well, since it is independent of both the learning set and the learning algorithm.

For the sake of simplicity, we denote by $\mu_{\mathcal{L},\theta_m}(\mathbf{x})$ the mean prediction at $X = \mathbf{x}$ of an individual randomized model $\phi_{\mathcal{L},\theta_m}$ and by $\sigma^2_{\mathcal{L},\theta_m}(\mathbf{x})$ its respective variance, that is:

$$\mu_{\mathcal{L},\theta_m}(\mathbf{x}) = \mathbb{E}_{\mathcal{L},\theta_m}\left[\phi_{\mathcal{L},\theta_m}(\mathbf{x})\right], \qquad \sigma^2_{\mathcal{L},\theta_m}(\mathbf{x}) = \mathbb{V}_{\mathcal{L},\theta_m}\left[\phi_{\mathcal{L},\theta_m}(\mathbf{x})\right].$$

Notice that now the expectations are taken with respect to both $\mathcal{L}$ and θ_m, as random seeds are seen as random variables. In this context, theorem 4.1 can be naturally extended to the expected generalization error $\mathbb{E}_{\mathcal{L},\theta}\left[Err\left(\phi_{\mathcal{L},\theta}(\mathbf{x})\right)\right]$ by replacing the expectations $\mathbb{E}_{\mathcal{L}}\left[\cdot\right]$ with $\mathbb{E}_{\mathcal{L},\theta}\left[\cdot\right]$. Therefore, the bias-variance decomposition, in that case, is as follows:

$$\mathbb{E}_{\mathcal{L},\theta}\left[Err\left(\phi_{\mathcal{L}}\left(\mathbf{x}\right)\right)\right] = noise(\mathbf{x}) + bias^2(\mathbf{x}) + var(\mathbf{x}),$$

where

$$\begin{aligned} noise(\mathbf{x}) &= Err\left(\phi_B\left(\mathbf{x}\right)\right) \\ bias^2(\mathbf{x}) &= \left(\phi_B\left(\mathbf{x}\right) - \mathbb{E}_{\mathcal{L},\theta}\left[\phi_{\mathcal{L},\theta}\left(\mathbf{x}\right)\right]\right)^2 \\ var(\mathbf{x}) &= \mathbb{E}_{\mathcal{L},\theta}\left[\left(\phi_{\mathcal{L}}\left(\mathbf{x}\right) - \mathbb{E}_{\mathcal{L},\theta}\left[\phi_{\mathcal{L},\theta}\left(\mathbf{x}\right)\right]\right)^2\right] \end{aligned}$$

Lemma 4.1. Let $\rho(\mathbf{x})$ be the correlation coefficient between the prediction of two randomized models $\varphi_{\mathcal{L},\theta_1}$, $\varphi_{\mathcal{L},\theta_2}$ built on the same learning set $\mathcal{L}$, but by using different random seeds, θ_1 and θ_2. Then, it holds that:

$$\mathbb{E}_{\mathcal{L},\theta_1,\theta_2}\left[\varphi_{\mathcal{L},\theta_1}(\mathbf{x})\,\varphi_{\mathcal{L},\theta_2}(\mathbf{x})\right] = \rho(\mathbf{x})\sigma^2_{\mathcal{L},\theta}(\mathbf{x}) + \mu^2_{\mathcal{L},\theta}(\mathbf{x}). \tag{4.6}$$

Proof. By definition of Pearson's correlation coefficient, we get:

$$\begin{aligned}
\rho(\mathbf{x}) &= \frac{\mathbb{E}_{\mathcal{L},\theta_1,\theta_2}\left[(\varphi_{\mathcal{L},\theta_1}(\mathbf{x}) - \mu_{\mathcal{L},\theta_1}(\mathbf{x}))\,(\varphi_{\mathcal{L},\theta_2}(\mathbf{x}) - \mu_{\mathcal{L},\theta_2}(\mathbf{x}))\right]}{\sigma_{\mathcal{L},\theta_1}(\mathbf{x})\sigma_{\mathcal{L},\theta_2}(\mathbf{x})} \\
&= \frac{\mathbb{E}_{\mathcal{L},\theta_1,\theta_2}\left[\varphi_{\mathcal{L},\theta_1}(\mathbf{x})\,\varphi_{\mathcal{L},\theta_2}(\mathbf{x}) - \varphi_{\mathcal{L},\theta_1}(\mathbf{x})\,\mu_{\mathcal{L},\theta_2}(\mathbf{x}) - \varphi_{\mathcal{L},\theta_2}(\mathbf{x})\,\mu_{\mathcal{L},\theta_1}(\mathbf{x}) + \mu_{\mathcal{L},\theta_1}(\mathbf{x})\mu_{\mathcal{L},\theta_2}(\mathbf{x})\right]}{\sigma^2_{\mathcal{L},\theta}(\mathbf{x})} \\
&= \frac{\mathbb{E}_{\mathcal{L},\theta_1,\theta_2}\left[\varphi_{\mathcal{L},\theta_1}(\mathbf{x})\,\varphi_{\mathcal{L},\theta_2}(\mathbf{x})\right] - \mu^2_{\mathcal{L},\theta}(\mathbf{x})}{\sigma^2_{\mathcal{L},\theta}(\mathbf{x})},
\end{aligned}$$

and the result is then straightforward. Lemma 4.1 will be used to prove the following theorem, which explains why combining several randomized may result in a degradation of the generalization error.

Theorem 4.2. For the squared error loss, the bias-variance decomposition of the expected generalization error $\mathbb{E}_{\mathcal{L},\theta_1,\dots,\theta_M}\left[Err\left(\psi_{\mathcal{L},\theta_1,\dots,\theta_M}(\mathbf{x})\right)\right]$ at $X = \mathbf{x}$ of an ensemble of M randomized models $\varphi_{\mathcal{L},\theta_m}$ is

$$\mathbb{E}_{\mathcal{L},\theta_1,\dots,\theta_M}\left[Err\left(\psi_{\mathcal{L},\theta_1,\dots,\theta_M}(\mathbf{x})\right)\right] = noise(\mathbf{x}) + bias^2(\mathbf{x}) + var(\mathbf{x}), \tag{4.7}$$

where

$$\begin{aligned}
noise(\mathbf{x}) &= Err\left(\varphi_B(\mathbf{x})\right) \\
bias^2(\mathbf{x}) &= \left(\varphi_B(\mathbf{x}) - \mathbb{E}_{\mathcal{L},\theta}\left[\varphi_{\mathcal{L},\theta}(\mathbf{x})\right]\right)^2 \\
var(\mathbf{x}) &= \rho(\mathbf{x})\sigma^2_{\mathcal{L},\theta}(\mathbf{x}) + \frac{1-\rho(\mathbf{x})}{M}\sigma^2_{\mathcal{L},\theta}(\mathbf{x}).
\end{aligned}$$

Proof. We first calculate the mean value of the ensemble's prediction:

$$\begin{aligned}
\mathbb{E}_{\mathcal{L},\theta_1,\dots,\theta_M}\left[\psi_{\mathcal{L},\theta_1,\dots,\theta_M}(\mathbf{x})\right] &= \mathbb{E}_{\mathcal{L},\theta_1,\dots,\theta_M}\left[\frac{1}{M}\sum_{m=1}^{M}\varphi_{\mathcal{L},\theta_m}(\mathbf{x})\right] \\
&= \frac{1}{M}\sum_{m=1}^{M}\mathbb{E}_{\mathcal{L},\theta_m}\left[\varphi_{\mathcal{L},\theta_m}(\mathbf{x})\right] \\
&= \mu_{\mathcal{L},\theta}(\mathbf{x})
\end{aligned}$$

Therefore, the bias of an ensemble is equal to the one obtained for any of the individual models:

$$bias^2(\mathbf{x}) = \left(\varphi_B(\mathbf{x}) - \mu_{\mathcal{L},\theta}(\mathbf{x})\right)^2 \tag{4.8}$$

For the variance, we have that

$$\begin{aligned}
var(\mathbf{x}) &= \mathbb{V}_{\mathcal{L},\theta_1,\dots,\theta_M}\left[\psi_{\mathcal{L},\theta_1,\dots,\theta_M}(\mathbf{x})\right] = \mathbb{V}_{\mathcal{L},\theta_1,\dots,\theta_M}\left[\frac{1}{M}\sum_{m=1}^{M}\phi_{\mathcal{L},\theta_m}(\mathbf{x})\right] \\
&= \frac{1}{M^2}\left\{\mathbb{E}_{\mathcal{L},\theta_1,\dots,\theta_M}\left[\left(\sum_{m=1}^{M}\phi_{\mathcal{L},\theta_m}(\mathbf{x})\right)^2\right] - \mathbb{E}_{\mathcal{L},\theta_1,\dots,\theta_M}\left[\sum_{m=1}^{M}\phi_{\mathcal{L},\theta_m}(\mathbf{x})\right]^2\right\} \\
&= \frac{1}{M^2}\left\{\mathbb{E}_{\mathcal{L},\theta_i,\theta_j}\left[\sum_{i,j}\phi_{\mathcal{L},\theta_i}(\mathbf{x})\phi_{\mathcal{L},\theta_j}(\mathbf{x})\right] - \left(M\mu_{\mathcal{L},\theta}(\mathbf{x})\right)^2\right\} \\
&= \frac{1}{M^2}\left\{\sum_{i,j}\mathbb{E}_{\mathcal{L},\theta_i,\theta_j}\left[\phi_{\mathcal{L},\theta_i}(\mathbf{x})\phi_{\mathcal{L},\theta_j}(\mathbf{x}) - M^2\mu^2_{\mathcal{L},\theta}(\mathbf{x})\right]\right\} \\
&= \frac{1}{M^2}\left\{M\mathbb{E}_{\mathcal{L},\theta}\left[\phi_{\mathcal{L},\theta}(\mathbf{x})^2\right] + \left(M^2 - M\right)\mathbb{E}_{\mathcal{L},\theta_1,\theta_2}\left[\phi_{\mathcal{L},\theta_1}(\mathbf{x})\,\phi_{\mathcal{L},\theta_2}(\mathbf{x})\right] - M^2\mu_{\mathcal{L},\theta}(\mathbf{x})^2\right\} \\
&= \frac{1}{M^2}\left\{M\left(\sigma^2_{\mathcal{L},\theta}(\mathbf{x}) + \mu_{\mathcal{L},\theta}(\mathbf{x})^2\right) + \left(M^2 - M\right)\left(\rho(\mathbf{x})\sigma^2_{\mathcal{L},\theta}(\mathbf{x}) + \mu_{\mathcal{L},\theta}(\mathbf{x})^2\right) - M^2\mu_{\mathcal{L},\theta}(\mathbf{x})^2\right\} \\
&= \frac{\sigma^2_{\mathcal{L},\theta}(\mathbf{x})}{M} + \rho(\mathbf{x})\sigma^2_{\mathcal{L},\theta}(\mathbf{x}) - \rho(\mathbf{x})\frac{\sigma^2_{\mathcal{L},\theta}(\mathbf{x})}{M} \\
&= \rho(\mathbf{x})\sigma^2_{\mathcal{L},\theta}(\mathbf{x}) + \frac{1-\rho(\mathbf{x})}{M}\sigma^2_{\mathcal{L},\theta}(\mathbf{x}).
\end{aligned}$$

Remark 4.2. Theorem 4.2 shows exactly how the reduction of the expected generalization error is achieved. To begin with, let us first note that $\rho(\mathbf{x})$, in fact, represents the effect that the random perturbations introduced in the learning algorithm have on the predictions. In particular, when $\rho(\mathbf{x}) \to 1$, this means that the predictions of the models are correlated and, in that case, $var(\mathbf{x}) \to \sigma^2_{\mathcal{L},\theta}(\mathbf{x})$, indicating that building an ensemble brings no benefit. On the other hand, when $\rho(\mathbf{x}) \to 0$, the random perturbations are strong enough and $var(\mathbf{x}) \to \frac{\sigma^2_{\mathcal{L},\theta}(\mathbf{x})}{M}$. The last term can be further driven to 0 by increasing the size M of the ensemble. Therefore, in order to get a model with a small generalization error, we combine a large number of trees each built according to a randomized procedure capable to produce trees that are sufficiently different one from another. Such randomized procedures are discussed in the next section.

4.3 Random forest algorithms

In this section we present the standard Random Forest algorithm proposed by Breiman (2001), the Extremely Randomized Trees algorithm proposed by Geurts et al. (2006), the Historical Random Forest algorithm proposed by Sexton and Laake (2018), as well as some implementation issues. In brief, random forests are ensembles of randomized trees, where each tree is built upon a randomly drawn replicate of the given learning set. More specifically, we start by giving the following definition:

Definition 4.2 (Bootstrap sample)**.** Let $\mathcal{L}$ be a learning set of size n. Then, the **bootstrap sample** of $\mathcal{L}$, denoted by $\widetilde{\mathcal{L}}$, is a sample of size b drawn with replacement from $\mathcal{L}$.

Although the definition above suggests that the sample of the bootstrap replicate may differ from that of the ininial learning set, for the rest we assume that, unless otherwise stated, $b = n$. Note also that when $b = n$, approximately 37% of the cases $(\mathbf{x}, \mathbf{y})$ present in $\mathcal{L}$ will be missing from the bootstrap sample $\widetilde{\mathcal{L}}$. Indeed, for each case $(\mathbf{x}, \mathbf{y})$ the probability of not have been selected after n draws is:

$$(1 - \frac{1}{n})^n \simeq \frac{1}{e} \simeq 0.368, \text{when } n \text{ is large.}$$

Remark 4.3. Note that when the learning set $\mathcal{L}$ contains repeated measurements for different subjects it may be undesirable to expose the construction of each individual tree to all subjects of $\mathcal{L}$, since this may contaminate predictions, especially when the intra-subject correlation and inter-subject variability are high. To address this problem, Karpievitch et al. (2009) proposes bootstrapping at the *subject-level*.

Definition 4.3 (Subject-level bootstrap)**.** Let $\mathcal{L}$ be a learning set consisting of repeated measurements over N subjects, that is, $\mathcal{L}$ is of the form

$$\mathcal{L} = \left\{ \mathbf{z}_{ij} | \mathbf{z}_{ij} = (t_{ij}, \mathbf{x}_{ij}, \mathbf{y}_{ij})', \, i = 1, 2, \ldots, N, \, j = 1, 2, \ldots, n_i \right\}.$$

If $\mathcal{N} = \{1, 2, \ldots, N\}$ is the set of subject indices and $\widetilde{\mathcal{N}}$ a bootstrap replicate of $\mathcal{N}$, then the **subject-level bootstrap sample** of $\mathcal{L}$, denoted by $\widetilde{\mathcal{L}}$, is

$$\widetilde{\mathcal{L}} = \left\{ \mathbf{z}_{ij} | \mathbf{z}_{ij} = (t_{ij}, \mathbf{x}_{ij}, \mathbf{y}_{ij})', \, i \in \widetilde{\mathcal{N}}, \, j = 1, 2, \ldots, n_i \right\}. \tag{4.9}$$

As shown by definition 4.3, subject-level bootstrapping consists in bootstrapping subjects instead of observations, that is, if subject i is included in the bootstrap replicate $\widetilde{\mathcal{N}}$, then all of its n_i associated observations are included in $\widetilde{\mathcal{L}}$ (see also Figure 4.1).

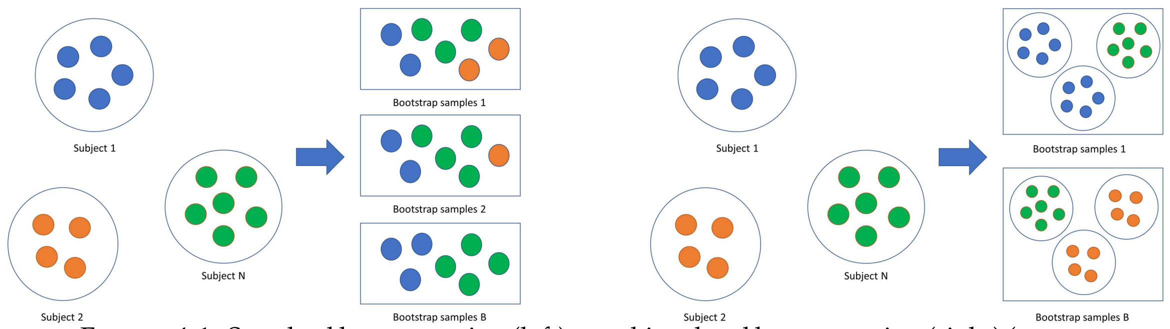

FIGURE 4.1: Standard bootstrapping (left) vs subject-level bootstrapping (right) (source: Hu and Szymczak, 2023)

4.3.1 Random Forest algorithm

Algorithm 4.1. Induction of a Random Forest

```
1: function BUILDRANDOMFOREST($\mathcal{L}$, $K$, $N_{min}$)
2:     for $m = 1, 2, \ldots, M$ do
3:         Create a bootstrap replicate $\widetilde{\mathcal{L}}^m$ of $\mathcal{L}$.
4:         Learn a tree $\phi_{\widetilde{\mathcal{L}}^m}$ using a random subset of $K$ out of the $p$ total features at each node.
5:     end for
6:     return $\{\phi_{\widetilde{\mathcal{L}}^m}\}_{m=1}^{M}$
7: end function
```

As already mentioned in chapter 3, decision trees suffer from high variability in the sense that even small changes in the given learning set may result in totally different decision trees. Now following the discussion of remark 4.2, it becomes clear why decision trees constitute an ideal option to serve as base models of an ensemble; their predictions can be decorrelated easily. At the same time, they are unbiased which is important considering that the bias of the ensemble equals the bias of the individual models.

Throughout the last decades, there have been proposed several ways to decorrelate the predictions of the individual trees in an ensemble through randomization. Breiman (1994) proposed building M decision trees, each on a different bootstrap sample $\widetilde{\mathcal{L}}^m$ of $\mathcal{L}$, and aggregating their predictions. This approach, known as *Bagging*[1], not only achieves good predictive performance, but can also be extended to any type of models. Dietterich and Kong (1995) proposed building an ensemble of trees, but for the construction of each tree we do not seek the best split. Rather, we randomly choose among the best 20 splits for each node t. The approach of using sub-optimal splits in each tree was also adopted by Amit et al. (1997) who suggested at each node t choosing at random K out of the p ($K \leq p$) features and selecting the best split among these features.

[1]Bagging stands for **B**ootstrap **agg**regat**ing**

Algorithm 4.2. Splitting a node t in RF

```
1: function SPLITRF($\mathcal{L}_t$, $K$)
2:     Randomly select $K$ of the $p$ features $\{X_1, X_2, \dots X_K\}$.
3:     for $k = 1, 2, \dots, K$ do
4:         Solve equation 3.10:
               $$s_k^* = \underset{\substack{s \in \mathcal{Q}(X_k) \\ \mathcal{L}_{t_L}, \mathcal{L}_{t_R} \neq \emptyset}}{\arg\max} \; \Delta i(s, t),$$
5:     end for
6:     Solve equation 3.9:
               $$s^* = \underset{\substack{s_k^* \\ k=1,\dots,K}}{\arg\max} \, \Delta i(s_k^*, t)$$
7:     return $s^*$
8: end function
```

The Random Forest algorithm[2] proposed by Breiman (2001) combines Bagging with the randomized variable selection approach of Amit et al. (1997). More specifically, in order to build a random forest we first create M bootstrap samples from $\mathcal{L}$, as in Bagging, and then, we use each of them to build a decision tree. The stopping criterion used for each decision tree is setting a node t as terminal if it contains less than N_{min} samples. To split each node in the forest, we first sample K features and the best of them is chosen to be used as a split variable. These steps are summarized in Algorithms 4.1, 4.2.

Remark 4.4. As we have already presented the most common ways to induce randomness in the construction of a decision tree, we can now better explain the role of the random seed parameter θ. For instance, in the context of Bagging, different random seeds θ_m, θ_n will result in different bootstrap samples $\widetilde{\mathcal{L}}^m, \widetilde{\mathcal{L}}^n$ and, consequently, in different decision trees. Therefore, in this context and for the sake of simplicity in the notation, we may replace $\phi_{\mathcal{L},\theta_m}$ with $\phi_{\widetilde{\mathcal{L}}^m}$. In the context of Random Forest, though, θ is in addition responsible for the random sub-sampling of the K features in each node. Since this study focuses on the original Random Forest algorithm, we can once again replace for the rest of this thesis $\phi_{\mathcal{L},\theta_m}$ with $\phi_{\widetilde{\mathcal{L}}^m}$, keeping in mind the additional level of randomness.

Today, Random Forest is considered one of the most successful machine learning algorithms capable of achieving state-of-the-art performance both in terms of exploration and prediction in various tasks and settings. That is why several variants have been built upon it since its introduction. One of these variants, namely the *Extremely Randomized Trees* (ETs) proposed by Geurts et al. (2006), has been widely used in applications and it has been well-known for its computational efficiency. In addition

[2]In general the term *random forest*, without capitals, refers to any ensemble of randomized trees. To distinguish the standard random forest algorithm from the other tree-based ensembles we will be using capitals, that is, henceforth, Random Forest will be referred to the original algorithm proposed by Breiman (2001).

to reducing computational time, in many cases, ETs achieve better predictive performance compared to the standard random forest algorithm.

4.3.2 Extremely Randomized Trees

The Extremely Randomized Trees algorithm is based on an empirical investigation of Geurts (2002) who showed that the cut-point variance appears to be responsible for a significant part of the generalization error of decision trees. As a way to smoothen the decision boundary, Geurts et al. (2006) induce a further level of randomness in the split selection procedure. In particular, just as in the Random Forest algorithm, each decision tree in the ETs algorithm is built on a bootstrap replicate of $\mathcal{L}$ and for each node t the best split is found on a randomly chosen subsample of K features. The difference between RF and ETs lies in that in ETs we do not seek the best threshold v, that is, we do not solve equation 3.10. Rather, for each candidate feature X_k we randomly choose a single threshold v from $\mathcal{X}_{k|t}$, we calculate the impurity decrease $\Delta i(s_k^v, t)$ of that threshold and we solve equation 3.9.

Algorithm 4.3. Splitting a node t in ETs

```
1: function SPLITETS($\mathcal{L}_t$, $K$, $N_{split}$)
2:     Randomly select $K$ of the $p$ features $\{X_1, X_2, \dots X_K\}$.
3:     for $k = 1, 2, \dots, K$ do
4:         Draw uniformly at random $N_{split}$ thresholds $\mathcal{V}^k = \left\{v_1^k, \dots, v_{N_{split}}^k\right\}$ from $\mathcal{X}_{k|t}$.
5:         Solve equation 3.10:
               $$s_k^* = \underset{\substack{s \in \mathcal{Q}(X_k) \\ \mathcal{L}_{t_L}, \mathcal{L}_{t_R} \neq \emptyset}}{\arg\max} \ \Delta i(s, t),$$
           using
               $$\mathcal{Q}(X_k) = \left\{s_k^v = \{\{\mathbf{x}_i | x_{ik} \leq v\}, \{\mathbf{x}_i | x_{ik} > v\}\} | v \in \mathcal{V}^k\right\}.$$
6:     end for
7:     Solve equation 3.9:
               $$s^* = \underset{\substack{s_k^* \\ k=1,\dots,K}}{\arg\max} \Delta i(s_k^*, t)$$
8:     return $s^*$
9: end function
```

It is clear that this approach is computationally faster than the original Random Forest algorithm since it does not require explicitly solving equation 3.10, that is, calculating $|\mathcal{X}_{k|t}| - 1$ impurity decreases in each node of a decision tree and finding the best among them. Of course, this results in individual decision trees that have a larger prediction bias than the ones produced by the RF algorithm. A compromise is to extend ETs by randomly picking a subset of N_{split} candidate thresholds v, instead of picking a single one. In that case, equation 3.10 is solved for each feature over the candidate subset, thus producing individual trees with a smaller bias. Again, computational time decreases but to a smaller extent compared to the standard ETs ($N_{split} = 1$). The splitting procedure of ETs is summarized in algorithm 4.3.

4.3.3 Historical Random Forests

All previous random forests assume that observations in the available data set are independent. As already discussed, though, when handling repeated measurement data, this is rarely the case. In most cases, within-subject observations are serially correlated and incorporating this correlation in the model construction may improve predictive performance.

One way to do so is proposed by Sexton and Laake (2018). To account for the inter-subject variability, they propose constructing each tree in a subsample of the initial learning set $\mathcal{L}$, where subsampling refers to subjects and all associated observations are included in the resulting subsample. At the same time, to account for the within-subject correlation, they modify the splitting procedure used for time-varying features by augmenting the set over which the best split is sought in every node. On the contrary, the split for time-invariant features is performed as in the standard Random Forest. The resulting algorithm is referred to as *Historical Random Forest (HRF)* and is described in this section.

In particular, let $\mathcal{L}$ be a learning set of repeated measurements as in equation 2.2 and $\mathcal{N} = \{1, 2, \ldots, N\}$ the set of subject indices in $\mathcal{L}$. If $\mathcal{N}^m$ a randomly drawn (without replacement) subset of $\mathcal{N}$, then the m-th decision tree is built using

$$\mathcal{L}^m = \{\mathbf{z}_{ij} | \mathbf{z}_{ij} = (t_{ij}, \mathbf{x}_{ij}, \mathbf{y}_{ij})', \, i \in \mathcal{N}^m, \, j = 1, 2, \ldots, n_i\} \tag{4.10}$$

Concerning the splitting procedure, for each feature X_k, apart from the splits of the form 3.8, we also consider splits that take into account preceding values of X_k as well. This is done, by using a *summary function* g summarizing the covariate history of X_k within a specific time interval of δ time units, where δ belongs to a set of candidate lags $\mathcal{D}$. Therefore, in order to split a node t in an HRF, apart from the axis-parallel splits used in the standard Random Forest algorithm:

$$\mathcal{Q}(X_k) = \{s_k^v = \{\{\mathbf{x}_{ij} | x_{ijk} \leq v\}, \{\mathbf{x}_{ij} | x_{ijk} > v\}\} \, | v \in \mathcal{X}_k\},$$

we also seek the best split in the following set:

$$\mathcal{H}(X_k; \mathcal{D}, g) = \left\{ s_k^v = \Big\{ \{\mathbf{x}_{ij} | g(\bar{x}_{ijk}; \delta) \leq v\}, \{\mathbf{x}_{ij} | g(\bar{x}_{ijk}; \delta) > v\} \Big\} \Big| v \in g(\mathcal{X}_k; \delta), \delta \in \mathcal{D} \right\}, \tag{4.11}$$

where $\bar{x}_{ijk} = \{x_{ihk} | t_{ih} < t_{ij}\}$ is the covariate history of X_k up to (but not including) observation time t_{ij} and $g(\mathcal{X}_k;\delta) = \{g(\bar{x}_{ijk};\delta) | x_{ijk} \in \mathcal{X}_k\}$. Overall, searching in the union of $\mathcal{Q}(X_k)$ and $\mathcal{H}(X_k;\mathcal{D},g)$ guarantees that both historical information (if available) and current values of X_k are used for modeling.

The splitting procedure in the HRF algorithm is summarized in the following algorithm:

Algorithm 4.4. Splitting a node t in HRF

```
1: function SPLITHRF($\mathcal{L}_t$, $K$, $g$, $\mathcal{D}$)
2:     Randomly select $K$ of the $p$ features $\{X_1, X_2, \dots X_K\}$.
3:     Get the sets $\mathcal{I}, \mathcal{V}$ of time-invariant and time-varying predictors respectively.
4:     for $k = 1, 2, \dots, K$ do
5:         if $X_k \in \mathcal{V}$ then
               $s_k^* = \underset{\substack{s \in \mathcal{Q}(X_k) \cup \mathcal{H}(X_k;\mathcal{D},g) \\ \mathcal{L}_{t_L}, \mathcal{L}_{t_R} \neq \emptyset}}{\arg\max} \Delta i(s,t),$
6:         else
               $s_k^* = \underset{\substack{s \in \mathcal{Q}(X_k) \\ \mathcal{L}_{t_L}, \mathcal{L}_{t_R} \neq \emptyset}}{\arg\max} \Delta i(s,t).$
7:         end if
8:     end for
9:     Solve equation 3.9:
           $s^* = \underset{\substack{s_k^* \\ k=1,\dots,K}}{\arg\max} \Delta i(s_k^*, t)$
10:    return $s^*$
11: end function
```

One exemplary summary function is the *mean summary function*, which is defined as follows:

Definition 4.4 (Mean summary function)**.** Let $\mathcal{L}$ be a repeated measurement learning set, $\delta \in \mathbb{R}$, t_{ij} the j-th observation time for subject i and $\bar{x}_{ijk} = \{x_{ihk} | t_{ih} < t_{ij}\}$ the set of values on the k-th feature for i that were observed before t_{ij}. Then, the **mean summary function** for i at t_{ij} averages the observed values of $\bar{x}_{ijk}$ within the time interval $[t_{ij} - \delta, t_{ij})$:

$$g(\bar{x}_{ijk};\delta) = \sum_{h: t_{ij} - \delta \leq t_{ih} < t_{ij}} \frac{x_{ijk}}{n_{ij}(\delta)}, \tag{4.12}$$

where $n_{ij}(\delta)$ denotes the size of $\bar{x}_{ijk}$[3]. In the special case where $n_{ij}(\delta) = 0$, we set $g(\bar{x}_{ijk};\delta) = 0$.

Remark 4.5.

[3]We assume that for each subject i and each time moment t_{ij}, we have observations on every feature, that is no missing data is present on $\mathcal{L}$. Therefore, $n_{ij}(\delta)$ is independent of k.

– The mean summary function presented above is not the only summary function that can be used to summarize the history of a feature X_k. Other summary functions such as

$$g(\bar{x}_{ijk};\delta,c) = \sum_{h:t_{ij}-\delta\le t_{ih}<t_{ij}} I(x_{ihk}<c), \qquad g(\bar{x}_{ijk};\delta,c) = \sum_{h:t_{ij}-\delta\le t_{ih}<t_{ij}} \frac{I(x_{ihk}<c)}{n_{ij}(\delta)} \tag{4.13}$$

may be used as well. What is more, in many real-world applications it might be desirable to consider a specific interval of the covariate history, in order to split a node. Therefore, windowed versions of the summary functions- where instead of $[t_{ij}-\delta, t_{ij})$, sums in equations 4.12, 4.13 are taken over $[t_{ij}-\delta_1, t_{ij}-\delta_2)$- are also available. Note, however, that adding parameters in the summary function, further increases the computational expenses of the algorithm. In any case, the summary function used should be *invertible* in the sense that knowledge of $g(\bar{x}_{ijk};\cdot)$ is equivalent to knowledge of the covariate history $\bar{x}_{ijk}$ (Sexton, 2018).

– Note that in an HRF it is also feasible to incorporate the response history in the model construction. This can be done by treating response variables Y_s ($s = 1,2,\dots,S$) as additional time-invariant predictors. In that case, however, the best split on Y_s is found by maximizing $\Delta i(s,t)$ over $\mathcal{H}(Y_s;\mathcal{D},g)$ rather than $\mathcal{Q}(Y_s)\cup\mathcal{H}(Y_s;\mathcal{D},g)$, since we assume that $\mathbf{y}_{ij}$ is unknown at observation time t_{ij}.

4.3.4 Implementation considerations

In algorithm 4.1 there exist 3 parameters affecting the performance of the Random Forest algorithm, namely the size M of the ensemble, the number K of randomly drawn candidate variables in each node and the minimum number N_{min} allowed in a node to be further split. In `R` programming language (R Core Team, 2023), in almost every random forest package, these parameters are referred to as `ntree`, `mtry` and `nodesize` respectively. While different Random Forest variants may introduce more parameters (e.g. the N_{split} parameter in ETs), we restrict the parameter discussion to these 3 parameters, which are present in almost every tree-based ensemble.

The decomposition of theorem 4.2 implies that ensemble learners achieve better performance when the individual predictors are accurate, yet adequately different from one another. To find a good compromise tuning of `mtry` and `nodesize` is crucial. In general, the choice of hyperparameters should be guided by the purposes of the statistical analysis. This means that the choice of optimal parameters may be different when Random Forest is used for exploration than when it is used for prediction. Below, we discuss each of these 3 parameters separately.

ntree

In remark 4.2 we argued that increasing `ntree` results in reducing the variance term of theorem 4.2. At the same time, Random Forest products, such as OOB-error estimates and proximity measures (see sections 4.5.1 and 4.5.2 respectively), are more accurate, since asymptotic properties are valid for large `ntree` values. However, as pointed out by Hastie et al. (2009a) increasing `ntree` too much may overfit the data. Probst and Boulesteix (2017) investigated under which circumstances increasing `ntree` does not come with a decrease in error. They showed that these non-monotonous behaviors are related to the choice of the impurity measure and concluded that for regression tasks when the squared error is used, larger values of `ntree` bring better results. Therefore, when this is the case, `ntree` is not a parameter that has to be tuned and fixing it to the largest computationally affordable value is a good strategy.

mtry

According to Probst et al. (2019b), among all the parameters that may be present in a random forest algorithm, `mtry` is the most influential. Lower values of `mtry` increase the variance among the individual trees, hence producing better results when aggregating. What is more, forests constructed using low `mtry` values tend to better exploit variables with moderate effect on the response variable that would be *masked* by variables with strong effect if those had been candidates for splitting. However, decreasing `mtry` too much may result in an increase in bias, since in this case sub-optimal splits are more frequent.

Apart from the bias-variance trade-off above, more factors should be taken into consideration before choosing `mtry`. The first one concerns the fraction of relevant variables. When the total number of variables p is large, but this fraction is small, then random forests tend to perform poorly, because the chance of splitting on a relevant variable is small. The second factor concerns the available computational power, as testing more variables requires more computing time. In particular, Wright and Ziegler (2017) argue that computation time decreases approximately linearly with lower `mtry` values. The default choice for regression tasks in most statistical software is setting `mtry` to $p/3$.

nodesize

Since `nodesize` specifies the minimum number of observations in a terminal node, small values will lead to trees of larger depth. In the context of single decision trees, small values of `nodesize` may lead to overfitting. As with `mtry`, the presence of noisy variables and computational power may affect the choice of this parameter. In particular, Segal (2004) showed that a larger number of irrelevant variables results in a larger optimal `nodesize`. What is more, Probst et al. (2019b) argue that computation time decreases approximately exponentially with increasing `nodesize`. While the default value for `nodesize` for regression is 5, some authors (e.g. Hastie et al., 2009a) suggest fully growing decision trees (i.e. setting `nodesize` $= 1$), so that this parameter need not be tuned. Probst et al. (2019a) show experimentally that tuning `nodesize` brings minor improvements in terms of prediction error.

4.4 Aggregation schemes

Regardless of the method followed to construct the decision trees in a random forest algorithm, once the ensemble has been constructed, we need to aggregate the predictions of the individual decision trees to obtain a single prediction for our random forest algorithm. However, the aggregation can be done in different ways. This thesis concerns two different aggregation methods; a *scaled* and an *unscaled* approach. Let us recall equation 4.1:

$$\psi_{\mathcal{L}}(\mathbf{x}^*) = \frac{\sum_{m=1}^{M} w_m(\mathbf{x}^*)\phi_{\widetilde{\mathcal{L}}^m}(\mathbf{x}^*)}{\sum_{m=1}^{M} w_m(\mathbf{x}^*)}$$

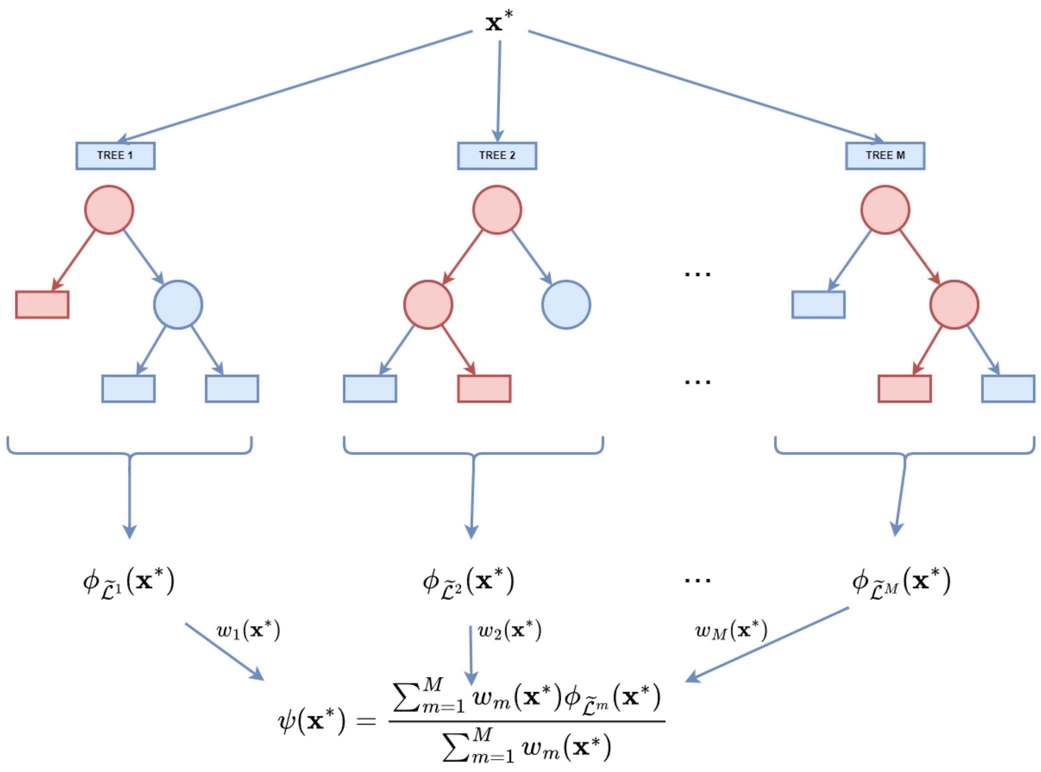
$\mathbf{x}^*$
TREE 1
TREE 2
TREE M
...
$\phi_{\widetilde{\mathcal{L}}^1}(\mathbf{x}^*)$
$\phi_{\widetilde{\mathcal{L}}^2}(\mathbf{x}^*)$
$\phi_{\widetilde{\mathcal{L}}^M}(\mathbf{x}^*)$
$w_1(\mathbf{x}^*)$
$w_2(\mathbf{x}^*)$
$w_M(\mathbf{x}^*)$
$\psi(\mathbf{x}^*) = \frac{\sum_{m=1}^{M} w_m(\mathbf{x}^*)\phi_{\widetilde{\mathcal{L}}^m}(\mathbf{x}^*)}{\sum_{m=1}^{M} w_m(\mathbf{x}^*)}$

FIGURE 4.2: Random Forest prediction for a new instance $\mathbf{x}^*$

The difference between the two methods actually lies in the way the weights $w_m(\mathbf{x}^*)$ are chosen. The random forest prediction for an observation $\mathbf{x}^*$ is summarized as shown in Figure 4.2.

4.4.1 Scaled approach

This is the commonly used approach proposed by Breiman (2001) and puts equal weights to the predictions of each individual tree, that is $w_m(\mathbf{x}^*) = 1$, for $m = 1, \ldots, M$. The prediction of the ensemble in this case is simply the mean value of the predictions of the trees:

$$\psi_{\mathcal{L}}(\mathbf{x}^*) = \frac{\sum_{m=1}^{M} \phi_{\widetilde{\mathcal{L}}^m}(\mathbf{x}^*)}{M}.$$

4.4.2 Unscaled approach

Contrary to the scaled approach, where each individual tree is weighted equally during the aggregation step, the unscaled approach weighs each tree according to the size of the terminal node in which instance $\mathbf{x}^*$ lands once propagated down the forest. More formally, let $\mathcal{L} = \{(\mathbf{x}_i, \mathbf{y}_i), i = 1, 2, \ldots, n\}$ be a learning set of size n which is used to construct a random forest $\psi_{\mathcal{L}}$ and $\{\widetilde{\mathcal{L}}^m\}_{m=1}^{M}$ the bootstrap samples of $\mathcal{L}$ used to construct the M decision trees of $\psi_{\mathcal{L}}$. We now introduce the following notation:

- $\mathcal{N}_m(\mathbf{x}^*)$: the indices of the samples in $\mathcal{L}$ sharing the same terminal node with $\mathbf{x}^\star$ in the m - th tree, that is:

$$\mathcal{N}_m(\mathbf{x}^*) = \{i : (\mathbf{x}_i, \mathbf{y}_i) \text{ is in the same terminal node as } \mathbf{x}^*, i = 1, 2 \ldots, n\},$$

- $b_m(i)$: the number of times case $(\mathbf{x}_i, \mathbf{y}_i)$ of $\mathcal{L}$ appears in the m-th bootstrap sample $\widetilde{\mathcal{L}}^m$.

Following this notation, the weights used in the unscaled approach are:

$$w_m(\mathbf{x}^*) = \sum_{i \in \mathcal{N}_m(\mathbf{x}^*)} b_m(i), \qquad m = 1, 2, \ldots, M. \tag{4.14}$$

Note that equation 4.14 in fact represents the size of the terminal node of m-th decision tree in which $\mathbf{x}^*$ lands. Hence, decision trees with larger terminal nodes have a greater impact on the prediction of the random forest.

Remark 4.6. The unscaled aggregation approach is equivalent to *pooling*, which consists in combining the terminal nodes containing $\mathbf{x}^*$ in a larger one called *pool* and using as a prediction for $\mathbf{x}^*$ the mean response value of the pool. To illustrate this point, let us first note that when propagating $\mathbf{x}^*$ down

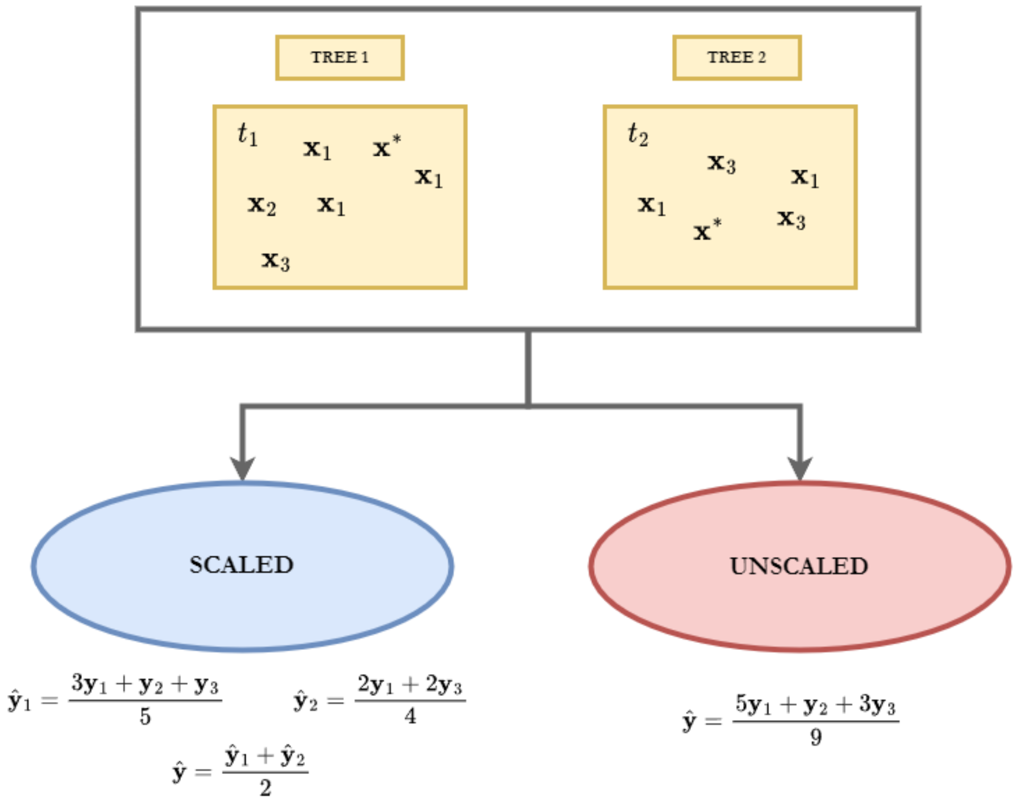
TREE 1
TREE 2
t_1
$\mathbf{x}_1$
$\mathbf{x}^*$
$\mathbf{x}_1$
$\mathbf{x}_2$
$\mathbf{x}_1$
$\mathbf{x}_3$
t_2
$\mathbf{x}_3$
$\mathbf{x}_1$
$\mathbf{x}_1$
$\mathbf{x}^*$
$\mathbf{x}_3$
SCALED
UNSCALED
$\hat{\mathbf{y}}_1 = \frac{3\mathbf{y}_1 + \mathbf{y}_2 + \mathbf{y}_3}{5}$
$\hat{\mathbf{y}}_2 = \frac{2\mathbf{y}_1 + 2\mathbf{y}_3}{4}$
$\hat{\mathbf{y}} = \frac{\hat{\mathbf{y}}_1 + \hat{\mathbf{y}}_2}{2}$
$\hat{\mathbf{y}} = \frac{5\mathbf{y}_1 + \mathbf{y}_2 + 3\mathbf{y}_3}{9}$

FIGURE 4.3: Scaled vs Unscaled aggregation approach. t_1 and t_2 are the terminal nodes at which $\mathbf{x}^*$ lands once propagated down the forest.

the m-th tree, the assignment rule 3.5 can be reexpressed in terms of the notation introduced above as follows:

$$\phi_{\mathcal{L},\theta_m}(\mathbf{x}^*) = \frac{\sum_{i\in\mathcal{N}_m(\mathbf{x}^*)} b_m(i)\mathbf{y}_i}{\sum_{i\in\mathcal{N}_m(\mathbf{x}^*)} b_m(i)}. \tag{4.15}$$

Using equations 4.14 and 4.15, we get:

$$\begin{aligned}
\psi_{\mathcal{L}}(\mathbf{x}^*) &= \frac{\sum_{m=1}^{M} w_m(\mathbf{x}^*)\phi_{\tilde{\mathcal{L}}^m}(\mathbf{x}^*)}{\sum_{m=1}^{M} w_m(\mathbf{x}^*)} \\
&= \frac{\sum_{m=1}^{M} w_m(\mathbf{x}^*)\frac{\sum_{i\in\mathcal{N}_m(\mathbf{x}^*)} b_m(i)\mathbf{y}_i}{\sum_{i\in\mathcal{N}_m(\mathbf{x}^*)} b_m(i)}}{\sum_{m=1}^{M} w_m(\mathbf{x}^*)} \\
&= \frac{\sum_{m=1}^{M} \cancel{\sum_{i\in\mathcal{N}_m(\mathbf{x}^*)} b_m(i)}\frac{\sum_{i\in\mathcal{N}_m(\mathbf{x}^*)} b_m(i)\mathbf{y}_i}{\cancel{\sum_{i\in\mathcal{N}_m(\mathbf{x}^*)} b_m(i)}}}{\sum_{m=1}^{M} \sum_{i\in\mathcal{N}_m(\mathbf{x}^*)} b_m(i)} \\
&= \frac{\sum_{m=1}^{M} \sum_{i\in\mathcal{N}_m(\mathbf{x}^*)} b_m(i)\mathbf{y}_i}{\sum_{m=1}^{M} \sum_{i\in\mathcal{N}_m(\mathbf{x}^*)} b_m(i)}.
\end{aligned} \tag{4.16}$$

Equation 4.16 represents the mean output value of the instances in the pool. In Figure 4.3 we illustrate pooling through a simple example of a random forest of 2 decision trees. In this example t_1, t_2 denote the terminal nodes of each tree at which the new observation $\mathbf{x}^*$ lands.

Another way to see this approach is that instead of assigning different weights to each individual tree, we assign different weights to the instances of $\mathcal{L}$ (Sage, 2018). Indeed, continuing equation 4.16, we get:

$$\begin{aligned}
\psi_{\mathcal{L}}(\mathbf{x}^*) &= \frac{\sum_{m=1}^{M} \sum_{i \in \mathcal{N}_m(\mathbf{x}^*)} b_m(i)\mathbf{y}_i}{\sum_{m=1}^{M} \sum_{i \in \mathcal{N}_m(\mathbf{x}^*)} b_m(i)} \\
&= \frac{\sum_{m=1}^{M} \sum_{i=1}^{n} b_m(i)\mathbb{1}_{\{i \in \mathcal{N}_m(\mathbf{x}^*)\}}\mathbf{y}_i}{\sum_{m=1}^{M} \sum_{i=1}^{n} b_m(i)\mathbb{1}_{\{i \in \mathcal{N}_m(\mathbf{x}^*)\}}} \\
&= \frac{\sum_{i=1}^{n} \sum_{m=1}^{M} b_m(i)\mathbb{1}_{\{i \in \mathcal{N}_m(\mathbf{x}^*)\}}\mathbf{y}_i}{\sum_{i=1}^{n} \sum_{m=1}^{M} b_m(i)\mathbb{1}_{\{i \in \mathcal{N}_m(\mathbf{x}^*)\}}} \\
&= \sum_{i=1}^{n} p_i(\mathbf{x}^*)\mathbf{y}_i, \qquad (4.17)
\end{aligned}$$

where

$$p_i(\mathbf{x}^*) = \frac{\sum_{m=1}^{M} b_m(i)\mathbb{1}_{\{i \in \mathcal{N}_m(\mathbf{x}^*)\}}}{\sum_{i=1}^{n} \sum_{m=1}^{M} b_m(i)\mathbb{1}_{\{i \in \mathcal{N}_m(\mathbf{x}^*)\}}} \qquad (4.18)$$

represents the percentage of the observations that $(\mathbf{x}_i, \mathbf{y}_i)$ occupies in the pool of $\mathbf{x}^*$. That is, observations of $\mathcal{L}$ that share the same terminal node with $\mathbf{x}^*$ are thought to be similar to that observation and, hence assigning larger weights to them is reasonable. The similarity between observation in the context of random forests is further discussed in section 4.5.2.

4.5 Properties and features

4.5.1 Out-of-bag estimate

In section 2.2.2 we highlighted that using different learning sets for constructing and evaluating a model is essential to produce unbiased estimates of the generalization error. An appealing feature of ensemble methods that construct models on bootstrap samples, such as random forests, is the built-in possibility of using the left-out observations to estimate the generalization error. Let us first give the following definition:

Definition 4.5. Let $\mathcal{L}$ be a learning set and $\widetilde{\mathcal{L}}$ a bootstrap replicate of $\mathcal{L}$. Then, the **out-of-bag set** $\mathcal{L}_{OOB}$ of $\widetilde{\mathcal{L}}$ is the set of samples of $\mathcal{L}$ that do not appear in $\widetilde{\mathcal{L}}$, that is:

$$\mathcal{L}_{OOB} = \{(\mathbf{x}_i, \mathbf{y}_i) \in \mathcal{L} : b(i) = 0\}.$$

The observations belonging to the out-of-bag set are called **out-of-bag observations** and, correspondingly, the observations that are included in the bootstrap sample $\widetilde{\mathcal{L}}$ are called **in-bag observations**.

The out-of-bag set of an individual tree can serve as a test set to calculate the average prediction error of that tree (see definition 2.9) and averaging over all trees of the forest one can estimate the generalization error of the ensemble. More formally:

Definition 4.6 (Out-of-bag estimate)**.** Let $\mathcal{L}$ be a learning set, $\left\{\widetilde{\mathcal{L}}^m\right\}_{m=1}^M$ a collection of M bootstrap replicates of $\mathcal{L}$ and $\{\mathcal{L}_{OOB}^m\}_{m=1}^M$ the respective out-of-bag sets. If $\psi_{\mathcal{L}}$ is an ensemble consisting of the individual models $\{\phi_{\widetilde{\mathcal{L}}^m}\}_{m=1}^M$, then the **out-of-bag estimate** of $Err(\psi_{\mathcal{L}})$ is defined as follows:

$$\widehat{Err}^{OOB}(\psi_{\mathcal{L}}) = \frac{1}{M}\sum_{m=1}^{M}\widehat{Err}_m^{OOB}, \tag{4.19}$$

where $\widehat{Err}_m^{OOB} = \bar{E}\left(\phi_{\widetilde{\mathcal{L}}^m}, \mathcal{L}_{OOB}^m\right)$ denotes the out-of-bag error of tree m.

The out-of-bag estimate constitutes a beautifully simple way of estimating the generalization error of an ensemble. At the same time, it is computationally cheaper compared to the K-fold cross-validation or the MCCV methods presented in section 2.2.2, since it demands the construction of a single model to produce an estimate for the generalization error.

Despite its aforementioned advantages, the out-of-bag estimate is sometimes biased. More specifically, as pointed out by Louppe and Geurts (2012), the randomness induced by the bootstrapping process may result in a decrease in the model's accuracy, especially in cases where the sample size is small relatively to the number of the input variables (Matthew et al., 2011; Janitza and Hornung, 2018).

4.5.2 Proximity measures

Another interesting built-in feature of tree-based ensemble methods is the *proximity measures*. These measures are, in fact, measures of similarity between any two observations of a learning set. The random forest proximity was first introduced by Breiman (2002). Formally, we have the following definition:

Definition 4.7 (Original random forest proximity)**.** In a random forest, the **proximity** between two cases $(\mathbf{x}_i, \mathbf{y}_i)$ and $(\mathbf{x}_j, \mathbf{y}_j)$ is defined as the proportion of trees in which these two cases reside in the same terminal node, that is:

$$p_{Or}(\mathbf{x}_i, \mathbf{x}_j) = \frac{1}{M}\sum_{m=1}^{M}\mathbb{1}_{\{j\in\mathcal{N}_m(\mathbf{x}_i)\}} \tag{4.20}$$

Note that definition 4.7 does not take into account an observation's bootstrap status (whether or not the observation was used during the construction of any particular tree) in the proximity calculation: both in-bag and out-of-bag samples are used.

An alternate is to calculate the proximity between two observations assuming that both of them are out-of-bag samples in the m-th tree (Hastie et al., 2009a). In particular:

Definition 4.8 (OOB proximity)**.** The OOB proximity between two observations $(\mathbf{x}_i, \mathbf{y}_i)$ and $(\mathbf{x}_j, \mathbf{y}_j)$ is defined as the proportion of trees in which these observations reside in the same terminal node, both being out-of-bag, that is:

$$p_{OOB}(\mathbf{x}_i, \mathbf{x}_j) = \frac{\sum_{m=1}^{M} \mathbb{1}_{\{b_m(i)+b_m(j)=0\}} \mathbb{1}_{\{j \in \mathcal{N}_m(\mathbf{x}_i)\}}}{\sum_{m=1}^{M} \mathbb{1}_{\{b_m(i)+b_m(j)=0\}}}. \tag{4.21}$$

In definition 4.8, the condition $b_m(i) + b_m(j) = 0$ preserves that both $(\mathbf{x}_i, \mathbf{y}_i)$ and $(\mathbf{x}_j, \mathbf{y}_j)$ are out-of-bag for the m-th tree, since in general $b_m(k) \geq 0$.

Rhodes et al. (2022) argue that both the original and the OOB definitions do not preserve the geometry learned from the data. In contrast, they propose a new random forest proximity definition that exactly characterizes the random forest performance on both in-bag and out-of-bag samples. In particular, according to their definition, training examples (in-bag observations) are used to construct the proximity values to unseen (out-of-bag) observations, thus mimicking the typical scenario of constructing a model on a learning set and subsequently testing it on a validation set of previously unseen observations.

Definition 4.9 (Random forest-geometry and accuracy-preserving proximity)**.** Let $\mathcal{B}_m = \left\{ i : (\mathbf{x}_i, \mathbf{y}_i) \in \widetilde{\mathcal{L}}^m \right\}$ be the multiset of (potentially repeated) indices of in-bag observations in the m-th tree of a random forest. If $(\mathbf{x}_i, \mathbf{y}_i)$, $(\mathbf{x}_j, \mathbf{y}_j)$ $(i \neq j)$ are two cases, then we can define the following:

- $\mathcal{J}_m(\mathbf{x}_i) = \mathcal{B}_m \cap \mathcal{N}_m(\mathbf{x}_i)$ is the set of indices of in-bag observations which share the terminal node with $(\mathbf{x}_i, \mathbf{y}_i)$ in tree m
- $\mathcal{S}(\mathbf{x}_i) = \{m : b_m(i) = 0\}$ is the set of tree indices in which observation $(\mathbf{x}_i, \mathbf{y}_i)$ is out-of-bag.

Following the notation introduced above, the **random forest-geometry and accuracy-preserving proximity** between $(\mathbf{x}_i, \mathbf{y}_i)$ and $(\mathbf{x}_j, \mathbf{y}_j)$ is the average proportion of in-bag observations in the shared terminal node of $(\mathbf{x}_i, \mathbf{y}_i)$ and $(\mathbf{x}_j, \mathbf{y}_j)$ over all trees where $(\mathbf{x}_i, \mathbf{y}_i)$ is out of bag and $(\mathbf{x}_j, \mathbf{y}_j)$ is in bag, that is:

$$p_{GAP}(\mathbf{x}_i, \mathbf{x}_j) = \frac{1}{|\mathcal{S}(\mathbf{x}_i)|} \sum_{m \in \mathcal{S}(\mathbf{x}_i)} \frac{\mathbb{1}_{\{j \in \mathcal{J}_m(\mathbf{x}_i)\}}}{|\mathcal{J}_m(\mathbf{x}_i)|}. \tag{4.22}$$

For the sake of simplicity, we often formulate proximity using a $n \times n$ matrix $P = (p_{ij})_{1 \leq i,j \leq n}$, where p_{ij} is the proximity between observations $\mathbf{x}_i$, $\mathbf{x}_j$ calculated using any of the three definitions introduced so far.

A common aspect within all three proximity measures introduced so far is that they are defined as proportions of trees in which the two observations reside in the same terminal node. The idea is that these proportions give an indication of how close the observations are in the eyes of the random forest. In particular, proximity values close to 1 indicate that the observations reach the same terminal node and thus are similar according to the forest. On the contrary, when proximity is close to 0, observations reach different terminal nodes suggesting that the samples are structurally different from each other.

This interpretation of proximity measures has made them a powerful tool that can be used in a variety of tasks such as clustering, data visualization, imputation of missing values (see also section 4.5.4), and outlier detection.

4.5.3 Variable importance

Variable selection is a crucial issue in many applied classification and regression problems. It is of interest for statistical analysis as well as for modelization or prediction purposes to remove irrelevant variables, select all important ones or determine a sufficient subset for prediction. These main different objectives from a statistical learning perspective involve variable selection to simplify statistical problems, to help with diagnosis and interpretation, and to speed up data processing. An appealing feature of tree-based ensemble methods is that they offer the possibility to assess the variable importance of the included predictors.

Breiman (2002) proposed to evaluate the importance of an input variable X_k for predicting Y by adding up the weighted impurity decreases for all nodes where X_k is used as a split variable, averaged over all trees. Formally, we have the following definition:

Definition 4.10 (Mean decrease impurity importance). Let $\psi_{\mathcal{L}}$ be an ensemble of M trees $\{\phi_{\tilde{\mathcal{L}}_m}\}_{m=1}^{M}$. Then, the **mean decrease impurity (MDI) importance** of an input variable X_k is defined as follows:

$$Imp(X_k) = \frac{1}{M} \sum_{m=1}^{M} \sum_{t \in \phi_{\tilde{\mathcal{L}}_m}} \mathbb{1}_{\{k_t = j\}} \left[p(t) \Delta i(s_t, t) \right], \tag{4.23}$$

where $p(t)$ is the proportion $\frac{n_t}{n}$ of samples reaching node t and k_t is an identifier of the variable used to split node t.

Since its introduction, MDI importance has been widely used in many applications and, during the last years, its theoretical properties have been studied as well (Louppe et al., 2013; Sutera et al., 2016; Scornet, 2023).

In addition to MDI importance, Breiman and Cutler (2003) proposed another way to assess the importance. In particular, in order to calculate the importance of a single variable, they proposed shuffling its entries in the OOB set and computing the difference between the out-of-bag error on the permuted and the original OOB set. This importance measure known as *mean decrease accuracy (MDA)* or *permutation importance* is defined as follows:

Definition 4.11 (Permutation importance)**.** Let $\psi_{\mathcal{L}}$ be an ensemble of M trees $\{\phi_{\tilde{\mathcal{L}}_m}\}_{m=1}^{M}$ with respective out-of-bag sets $\{\mathcal{L}^m_{OOB}\}_{m=1}^{M}$. Let, also $\tilde{\mathbf{x}}_i^{(k)}$ denote the i-th input vector of $\mathcal{L}$, where the k-th coordinate is permuted. Then, the **permutation importance** of an input variable X_k is:

$$Imp(X_k) = \frac{1}{M}\sum_{m=1}^{M}\left(\widetilde{Err}_m^{OOB} - \widehat{Err}_m^{OOB}\right), \tag{4.24}$$

where $\widetilde{Err}_m^{OOB}$ denotes the OOB error of tree m calculated using a permuted OOB set rather than $\mathcal{L}_m^{OOB}$, that is, a set containing the input vectors $\tilde{\mathbf{x}}_i^{(k)}$ rather than the original ones $\mathbf{x}_i$.

The rationale of permutation importance is that if X_k is important, that is, associated with the output variable Y, then permuting its values should come with a substantial increase of error. In other words, if permuting the values does not harm predictive performance, then X_k is considered unimportant. On the contrary, if prediction accuracy decreases substantially when values of X_k are permuted, this indicates that X_k is an important predictor.

4.5.4 Missing values

Missing data is a problem often encountered in real-world applications. This can happen due to practical limitations, physical constraints or privacy reasons. Yet, data that is missing is problematic, since most machine learning algorithms assume an ideal scenario in which given data are known for all input variables. This forces researchers to choose between imputing data or discarding missing values. However, simply discarding missing data is not a reasonable practice, as valuable information may be lost. Therefore, imputing missing data in such settings is a more reasonable and practical way to proceed. Random forests offer several ways to deal with the task of imputing missing values (Tang and Ishwaran, 2017).

Rough imputation

We first begin by presenting a naive imputation method that is not based on the random forest algorithm. Nevertheless, this algorithm is computationally fast, and that is why it is often used to initialize some of the RF-based imputation algorithms presented later. When performing a *rough imputation*, the missing values of a random variable X_k are imputed as follows:

- If X_k is numerical, then its missing values are imputed using the median (or mean) value of the non-missing ones.
- If X_k is categorical, then its missing values are imputed by taking the mode over the non-missing ones (ties are broken at random).

Proximity imputation

In section 4.5.2 we saw that proximity values indicate how close two observations are in the eyes of a random forest. Therefore, it is reasonable to use these values to fill the missing data. This algorithm starts by imputing the missing values using a rough imputation. Then, a random forest is constructed using this imputed data and the resulting proximity matrix is used to re-impute the original missing values. In particular:

- If X_k is numerical, then its missing values are imputed using the proximity-weighted average of non-missing data.
- If X_k is categorical, then a proximity-weighted vote is taken using the responses of all observed cases and the category receiving a plurality is taken as the imputed value.

The updated data is used to construct a new random forest and the process is iterated. Breiman and Cutler (2003) suggest performing at most six iterations.

On-the-fly imputation

A drawback of the weighted proximity imputation method is that estimates for OOB error obtained by this imputation method are biased (Breiman and Cutler, 2003). As such, measures based on OOB error, such as permutation importance, may be biased as well. What is more, this approach is unable to handle test data containing missing values on coordinates that are used for splitting. To deal with these issues, Ishwaran et al. (2008) propose imputing missing values as the forest is grown.

In particular, to calculate the impurity of each node only non-missing data is used. When assigning an instance to child nodes, if the variable used to split the node contains missing values, these values are imputed by drawing a random value from the in-bag non-missing data of that node. Once the split is performed, imputed data are reset to missing and the process is repeated until terminal nodes are reached. Again, after the terminal node assignment, missing data are reset back to missing. In other words, these random imputations are performed only to assign cases to child nodes, since the resulting forest still contains missing data up to this point. To impute the missing terminal node data, we use the OOB terminal node data of all trees in the forest; for numerical variables, we use their mean value, while their mode is used for categorical variables.

Ishwaran et al. (2008) argue that performing this process repeatedly improves the accuracy of the imputed values. Note that in the second and subsequent iterations, missing values are imputed by randomly drawing from observed values of cases in the same terminal node as the case for which imputation is performed, using the forest grown in the previous iteration. A random draw is made from each tree in the forest and new values are imputed by averaging or using a weighted vote.

As already mentioned, this method works even when the task is making predictions for test cases containing missing values. More specifically, the missing values of such instances are again imputed by randomly drawing from non-missing in-bag data and then reset to missing until these cases reach a terminal node.

missForest

Stekhoven and Bühlmann (2012) propose imputing missing values of X_k by predicting them using a random forest grown on the other variables. First, a rough imputation is performed on all variables. Then, a random forest is constructed in which X_k serves as the output variable. Cases with non-missing X_k values are used for constructing the random forest. Cases with X_k missing are then predicted and the prediction is used as the imputed value. This is done for each predictor variable sequentially and the process is iterated until the change in the imputed values becomes sufficiently small.

Note that the former approach requires the construction of p random forests at each iteration. Therefore, in situations where p is large, it might be computationally expensive. Tang and Ishwaran (2017) suggest a computationally faster version of missForest, referred to as mForest, which consists in randomly dividing the p input variables into mutually exclusive groups of approximate size αp, where $0 < \alpha < 1$ (these groups are approximately $1/\alpha$ in total). Then, each of these groups serves as a multivariate response variable to be predicted from the remaining variables. Again, complete multivariate response cases are used to construct a forest using a multivariate splitting rule and the cases containing missing values are predicted using the grown forest. This process is cycled over the approximately $1/\alpha$ groups and this completes one iteration.

Unsupervised imputation

The unsupervised imputation method is actually similar to the on-the-fly method except that now we assume that no response variable is available and, hence, a multivariate unsupervised splitting (Ishwaran et al., 2021c) is used. More specifically, in the original random forest algorithm, a set of `mtry` input variables are selected as potential splitting variables. However, since there is no output variable, for each of the `mtry` variables, we randomly choose `ytry` of the remaining input variables to serve as the multivariate response. Again, a multivariate splitting rule calculated over non-missing data only is used and, among the `mtry` candidate variables, the one leading to the best split is chosen. As with the on-the-fly method, if the split variable contains missing data, these values are imputed using the non-missing in-bag data, and after the split are reset back to missing. Once again, the missing data in terminal nodes are imputed using the OOB data.

Chapter 5

Results

In this chapter, we illustrate how the methodology presented in previous chapters can be applied in the CPSPP presented in chapter 1. Let us first note that the Random Forest algorithm, as well as many of its variants, have already been applied in the agriculture field to estimate crop yield using environmental and remote sensing data, as well as for exploratory purposes, such as the identification of the most informative variables (to cite a few Adam et al., 2014; Wang et al., 2019; Zhang et al., 2019; Sakamoto, 2020). In this section, we aim to deal with the two-fold objective of identifying the most informative factors in predicting the CPR percentages and building a model that can accurately generalize over incoming growing seasons.

5.1 Evaluation setting

Before we present the results of our analysis, it is necessary to pose the CPSPP within the machine learning framework presented in chapter 2. To begin with, we mention that CPSPP constitutes a multivariate regression with $S = 8$ output variables (one for each phenological stage) and 6 input features, the environmental and non-environmental factors presented in sections 1.2, 1.3. Each output variable tracks the percentage of the plants in the area of interest occupying the corresponding phenological stage at each observation time. Let us note that since the Pre-Season stage is added artificially, our models do not predict the percentages of this particular phenological stage. Instead, we predict the percentages for the remaining 7 phenological stages and we then normalize the percentages so that they sum up to 1.

As stated in chapter 1, our data covers 20 growing seasons, from 2002 to 2021. Each growing season may start at a different week of the calendar year. In any case, once the growing season has started, data are, thereafter, recorded every 7 days for $W = 38$ consecutive weeks. As a result, the set of weeks $\mathcal{W}_i$ for which we have observations may be different for each growing season i. For instance, based on the Table 1.1, for season 2002 we have that $\mathcal{W}_1 = \{13, 14, \ldots, 50\}$, while for growing season 2009 we have $\mathcal{W}_8 = \{15, 16, \ldots, 52\}$.

Based on the above, our data form a repeated measurement data set as shown in Table 2.2. Here different growing seasons are treated as subjects, hence the set of subjects is $\mathcal{N} = \{1, 2, \ldots, 20\}$ [1] and the total number of subjects is $N = 20$. For each season, we have the same number of observations, which means that $n_i = 38 = W$ for all $i \in \mathcal{N}$. Observation t_{ij} refers to the week of the calendar year on which the j-th observation for season i corresponds, that is $\mathcal{W}_i = \{t_{i1}, \ldots, t_{iW}\}$. What is more, observations are regularly spaced within time, since $t_{i,j+1} - t_{i,j} = 1$ for all $i \in \mathcal{N}$ and $j = 1, 2, \ldots, W - 1$.

5.1.1 Generalization error estimation method

We can now describe the method followed to compare the different approaches applied in the CPSPP. Since our target is to predict the CPR percentages for an unobserved growing season based on the available CPR percentages from past seasons and the concurrent values of input variables, we estimate the RMSPE for each approach using Monte-Carlo cross-validation, where splitting is performed at the season level. In particular, in each iteration of the MCCV, we randomly choose the training seasons $\mathcal{N}_{train} \subset \mathcal{N}$ and we include in the learning set $\mathcal{L}_{train}$ used to train our model all the observations from $\mathcal{L}$ associated with these seasons. The observations of $\mathcal{L}$ associated with the remaining seasons $\mathcal{N} \setminus \mathcal{N}_{train}$ serve as the learning set used to evaluate the model performance[2]. The model $\phi_{\mathcal{L}_{train}}$ is then used to predict the instances in $\mathcal{L}_{test}$, and the RMSPE for each stage and each week is calculated. We repeat this process for a large number of times M as MCCV procedure suggests (see also section 2.2.2) and we average the errors for each stage and week. The preceding procedure is summarized in the following algorithm:

[1]subject 1 refers to growing season 2002, subject 2 to 2003 etc.

[2]Note that this splitting procedure is similar to the subject-level bootstrap discussed in section 4.3 but this time sampling from the initial learning set $\mathcal{L}$ is done *without replacement*.

Algorithm 5.1. Monte-Carlo Cross Validation

```
1: function CROSSVAL($N_{train}$, $M$)
2:     for $m = 1, 2, \dots, M$ do
3:         Split $\mathcal{N}$ into $\mathcal{N}^m_{train}$ and $\mathcal{N}^m_{test}$ with respective sizes $N_{train}$ and $N_{test} = N - N_{train}$.
4:         Get the corresponding partition $\{\mathcal{L}^m_{train}, \mathcal{L}^m_{test}\}$.
5:         Learn $\psi_{\mathcal{L}^m_{train}}$ and obtain predictions $\hat{\mathbf{y}}_{ij}$ for $i \in \mathcal{N}^m_{test}$.
6:         Calculate the RMSPE for each stage $s = 1, 2, \dots, S$ and each week $w \in \bigcup_{i \in \mathcal{N}^m_{test}} \mathcal{W}_i$:
               $$RMSPE^m_{ws} = \sqrt{\frac{\sum_{i \in \mathcal{N}^m_{test}} (y_{ijs} - \hat{y}_{ijs})^2}{N_{test}}}.$$
7:     end for
8:     Calculate the average RMSPE for each stage and each week and the average cumulative RMSPE:
           $$RMSPE_{ws} = \frac{1}{M} \sum_{m=1}^{M} RMSPE^m_{ws}, \qquad ARMSPE = \frac{\sum_{w=1}^{W} \sum_{s=1}^{S} RMSPE_{ws}}{\left| \bigcup_{m=1}^{M} \left( \bigcup_{i \in \mathcal{N}^m_{test}} \mathcal{W}_i \right) \right|}.$$
9:     return $\{RMSPE_{ws}\}$ for $s = 1, 2, \dots, S$ and $w \in \bigcup_{m=1}^{M} \left( \bigcup_{i \in \mathcal{N}^m_{test}} \mathcal{W}_i \right)$ and the $ARMSPE$
10: end function
```

Remark 5.1. In line 6 of the preceding algorithm $\bigcup_{i \in \mathcal{N}^m_{test}} \mathcal{W}_i$ denotes the set of weeks at which the model is tested in the m-th iteration of the MCCV. For instance, if $\mathcal{N}_{test}$ includes seasons 2003, 2004, 2005, 2008 and 2010, then according to Table 1.1, the RMSPE is calculated only for weeks $\{14, 15, \dots, 51\}$. Correspondingly, $\bigcup_{m=1}^{M} \left(\bigcup_{i \in \mathcal{N}^m_{test}} \mathcal{W}_i \right)$ denotes the set of weeks for which RMSPE is calculated once the MCCV algorithm is implemented. It is clear that a sufficiently large M guarantees that the RMSPE is calculated for all weeks of the study, that is $\bigcup_{m=1}^{M} \left(\bigcup_{i \in \mathcal{N}^m_{test}} \mathcal{W}_i \right) = \{12, 13, \dots, 52\}$.

In our case, we choose $M = 500$ for the number of different splits for the MCCV algorithm. In addition, we invest $N_{train} = 15$ out of the total 20 growing seasons on training and the remaining 5 in testing, yielding a 75-25% split ratio. Finally, in line 5 of algorithm 5.1, we build 500 decision trees for each ensemble $\psi_{\mathcal{L}^m_{train}}$.

5.1.2 Optimizing hyperparameters

As already discussed in section 4.3.4, tuning `mtry` and `nodesize` parameters in a random forest algorithm, may be crucial for its predictive performance. In our analysis, we conducted a grid search over the sets $\{1, 2, \dots, p\}$ (where p denotes the variables included in the model construction) and $\{1, 2, \dots, 10\}$ for choosing the `mtry` and `nodesize` parameters respectively, thus resulting in a total of $10 \cdot p$ possible pairs. What is more, since performance is evaluated with algorithm 5.1, hyperparameter selection should be guided by a similar procedure. Ideally, in order to choose the optimal (in terms of MCCV error) hyperparameter combination for a single approach, one should implement algorithm 5.1 for each possible pair of `mtry` and `nodesize` parameters and pick the one with the best performance in terms of RMSPE. However, this idea is computationally infeasible for our problem.

To reduce computational time, we seek near-optimal hyperparameters by fitting a single forest of 500 trees for each combination of hyperparameters and selecting the combination with the smallest forest OOB error. To guarantee that the OOB error of each forest is close to the ARMSPE obtained from the MCCV, we grow each forest using *subject-level subsampling*. Each subsample used to grow the individual trees consists of data from 15 growing seasons while data from the remaining seasons are out-of-bag, again resulting in a 75-25% split ratio.

For the CPSPP, we implemented both multivariate and univariate random forests. Their performance is compared in the following sections.

5.2 Multivariate Random Forest

In this section we discuss two multivariate approaches applied to the CPSPP. Both of them are implemented using the `randomForestSRC` package of `R` (Ishwaran et al., 2021b). Another common aspect of them is that they assume all output variables are related with the features according to a function f, which they seek to estimate:

$$\mathbf{Y} = f(\mathbf{X}) + \epsilon. \tag{5.1}$$

The main discrimination between the two approaches followed concerns the impurity criterion used to split each node.

5.2.1 Indepedent MRF

The first approach presented is the independent Multivariate Random Forest (iMRF). This approach trains a single multivariate random forest for all phenological stages simultaneously and assumes that the response variables are independent one from another. Therefore the impurity criterion used for each split is the one given by equation 3.14. Therefore, this approach assumes no dependence among the response variables.

Feature extraction

In the first stage of our analysis, we identified the most important features for predicting the CPR statistics. We did that by building an iMRF using the standard bootstrap scheme and the scaled aggregation approach. In Table 5.1 we see the permutation variable importance of each variable

	Planted	Emerged	Silking	Dough	Dented	Mature	Harvested
TIME	8.097	8.916	13.753	11.033	9.660	7.270	4.117
AGDD	0.341	0.385	0.662	0.356	0.257	0.238	1.428
ADL	0.325	0.318	0.567	0.768	0.885	0.757	0.761
AP	0.018	0.008	-0.002	-0.002	-0.002	0.002	0.0004
AVP	0.021	0.014	0.018	0.006	0.003	0.003	0.015
NDVI	0.039	0.181	0.043	0.062	0.754	3.767	5.347

TABLE 5.1: Permutation importance of variables for each phenological stage

for each phenological stage. Note that since the Pre-Season stage is added artificially, we are not interested in identifying the most informative variables for predicting the CPR statistics of this stage.

The results indicate that calendar time is by far the most important variable. Thermal time and photoperiod are important variables for all phenological stages, while NDVI is mostly important for Emerged, Dented, Mature and Harvested stages. This is reasonable because NDVI is sensitive to changes in the chlorophyll levels of the vegetation, so we expect it to be highly influential for the last phenological stages. Furthermore, Emerged is the first phenological stage at which chlorophyll is visible from the satellite sensors and this explains why NDVI's importance for this stage is slightly larger. On the other hand, precipitation and vapor pressure do not appear to be significant factors in predicting the CPR statistics for any of the phenological stages, so we excluded these input variables from the rest of our analysis.

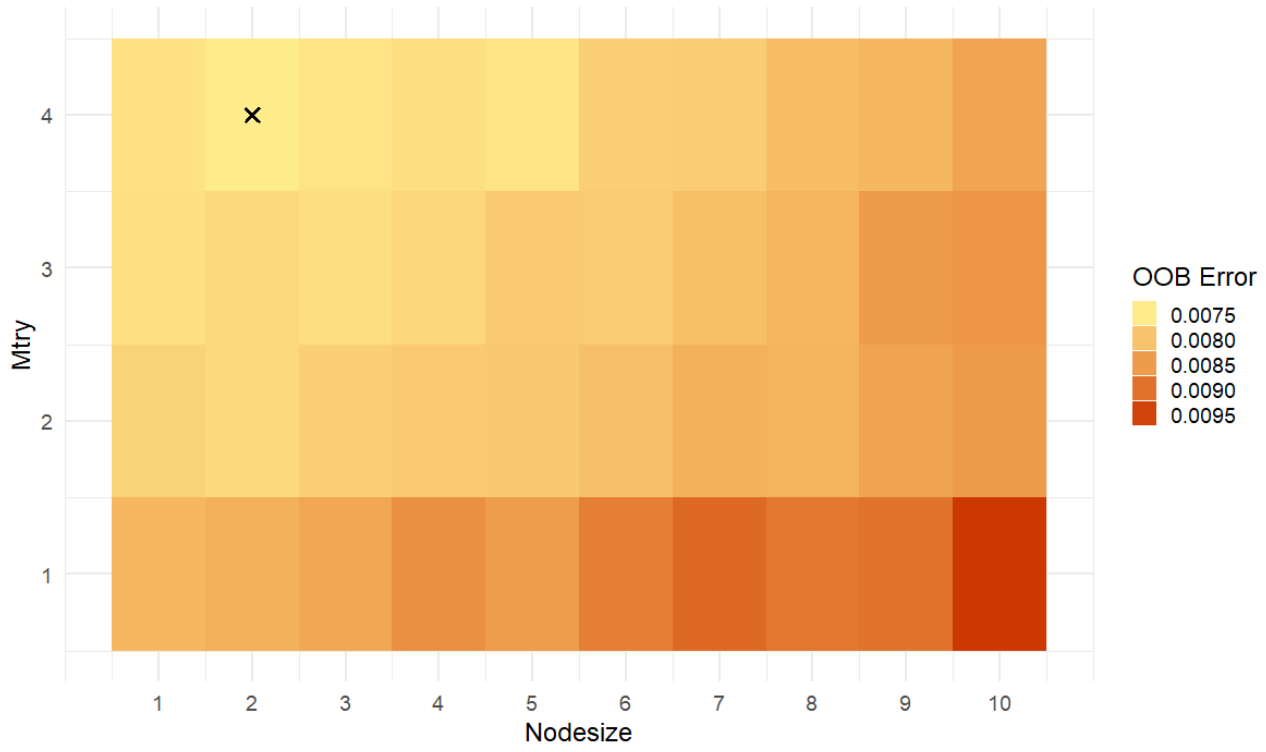

FIGURE 5.1: Hypertuning for iMRF. The optimal combination of `mtry` and `nodesize` parameters is denoted by X.

Independent MRF results

Implementing algorithm 5.1 for iMRF using the remaining 4 covariates, the standard bootstrapping scheme and the scaled aggregation approach, gives the results shown in Figures 5.1, 5.2. More specifically, in Figure 5.1 the results from the hypertuning procedure are presented. In general, large values for `mtry` and small values for `nodesize` parameters yielded smaller OOB errors, suggesting that learning the training data adequately is a good strategy for fine generalization performance. The optimal combination for this approach is `mtry` $=4$ and `nodesize` $=2$.

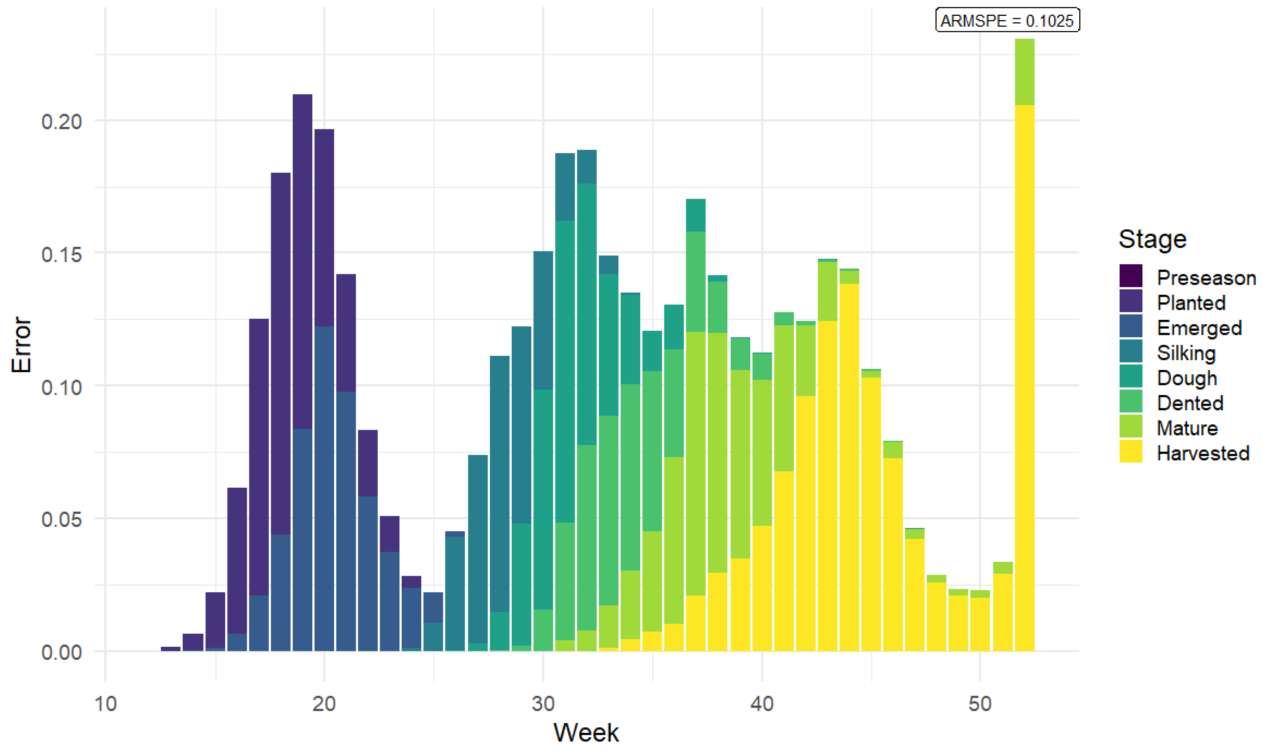

FIGURE 5.2: Cumulative RMSPE across phenological stages for each week of the study.

In Figure 5.2 we see the cumulative RMSPE across the stages for each week, that is $\sum_{s=1}^{S} RMSPE_{ws}$. For instance, for week 20 the total RMSPE across all stages is approximately 0.2 ($\simeq$ 0.12 for Planted, 0.08 for Preseason and 0 for all other stages). We also see that there exist some periods with low RMSPEs and others with large ones. This is because, for some weeks of almost every growing season, all plants occupy the same phenological stage. For instance, for all growing seasons of our study, the majority of plants occupy the Emerged phenological stage at week 25, which typically corresponds to late June (see also Figure 1.9). Hence, for these time periods, the inter-season variance of the CPR statistics is small and that explains the low prediction errors. On the contrary, at periods when plants occupy many different phenological stages, there is ambiguity regarding the percentage of plants occupying each phenological stage and that is why prediction error at these periods is larger.

What is more, the large prediction errors during the last week of the growing season can be explained by the lack of available data for this week. As already shown in Table 1.1, data for week 52 come exclusively from season 2009. Therefore, if season 2009 is not included in $\mathcal{N}_{train}^{m}$, then the random forest is not trained in this week at all and the corresponding prediction is poor. A similar argument holds for weeks 50 and 51 for which we have available data from 9 and 17 seasons respectively, but to a smaller extent than week 52. At this point, we should make clear that such pathogenic situations should be accounted for and caution is needed especially when performing extrapolation.

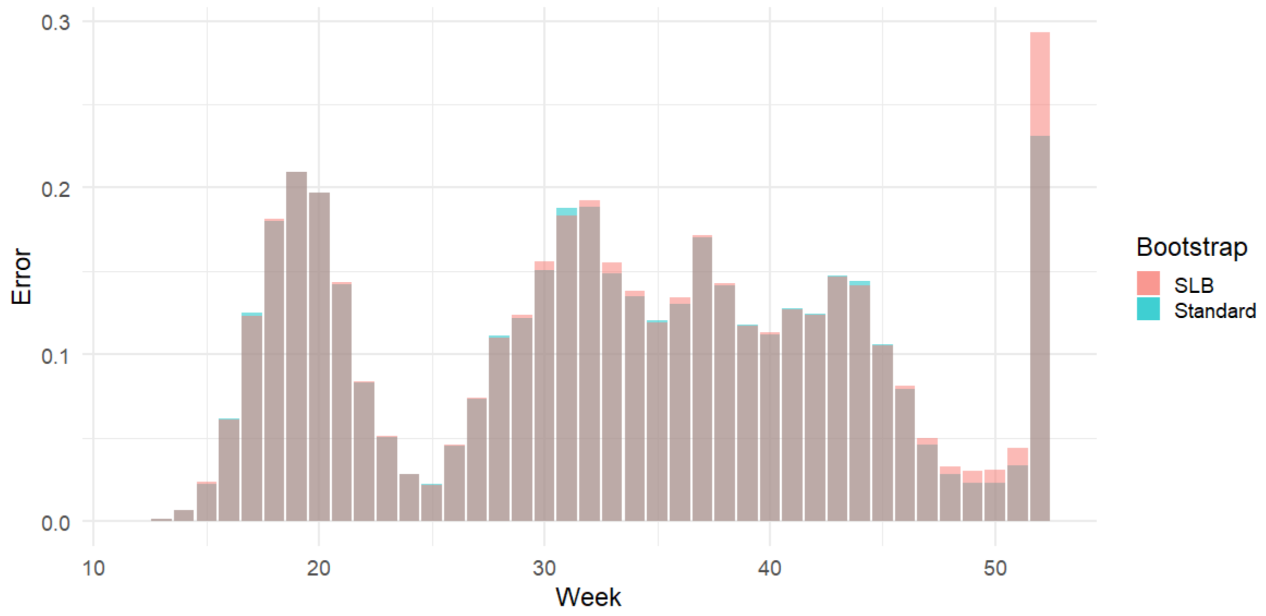

FIGURE 5.3: Standard vs Subject-Level-Bootstrap for iMRF. The ARMSPEs are 10.25% and 10.53% respectively.

Bootstrap selection

In Figure 5.3 the two different bootstrap schemes are compared. We see that the use of subject-level bootstrap brings no improvement for most weeks of the study. On the contrary, for the last weeks of the study, the standard bootstrap scheme performs better in terms of prediction error. These results can be explained by the fact that the subject-level bootstrap exploits the inter-subject variability to induce a further level of randomness. In other words, building each tree without using observations from all subjects, preserves the variability between the subjects within each bootstrap replicate and guarantees that the decorrelation effect of theorem 4.2 will be substantial.

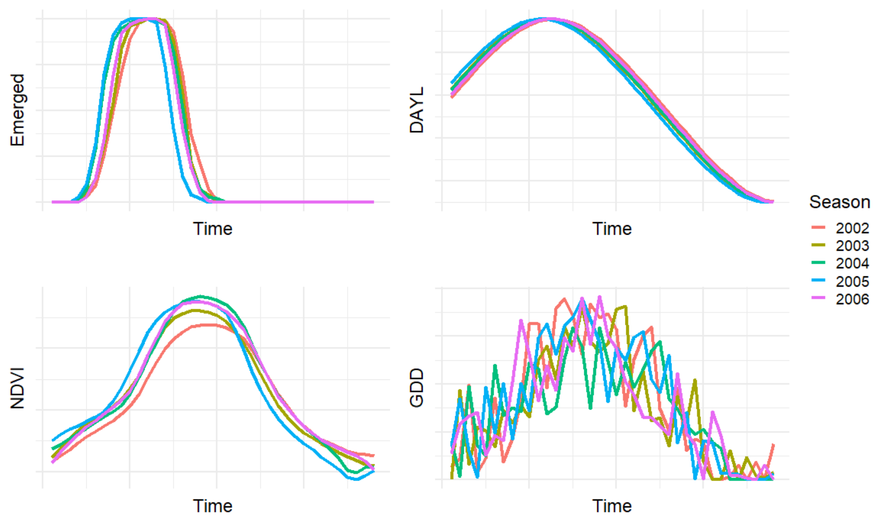

FIGURE 5.4: Evolution of the CPR statistics and covariates for the first 5 growing seasons.

In our case, however, inter-growing season variability is small. The evolution of the CPR statistics shown in Figure 1.9 is similar for all growing seasons and the same holds for the evolution of the covariates. This is illustrated in Figure 5.4. For the sake of readability, only the Emerged stage and the first 5 growing seasons are considered.

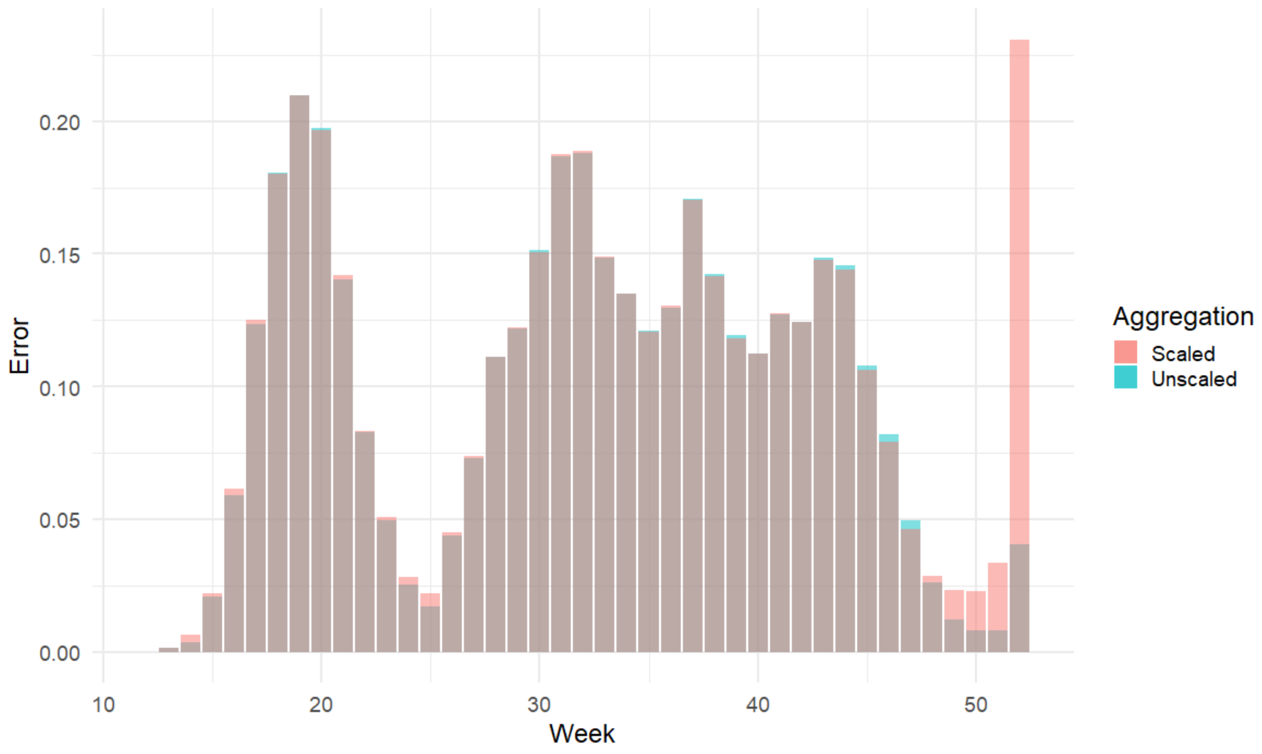

FIGURE 5.5: Scaled vs Unscaled aggregation for iMRF. The corresponding ARMSPEs are 10.25% and 9.62%

Aggregation scheme selection

In Figure 5.5, we compare the aggregation approaches for the iMRF. For the sake of comparison, we use the same hyperparameters and the same bootstrap protocol, namely the standard bootstrap scheme. In general, the two aggregation approaches gave similar results for most weeks of the study. This result is in line with the findings of Sage et al. (2020), who pointed out that for regression problems, the two approaches have similar performance. However, for the last weeks of the study, the unscaled approach clearly outperforms the scaled one, suggesting that even if a season is not included in many of the decision trees in a random forest, information for predicting it is extracted from decision trees that include that season.

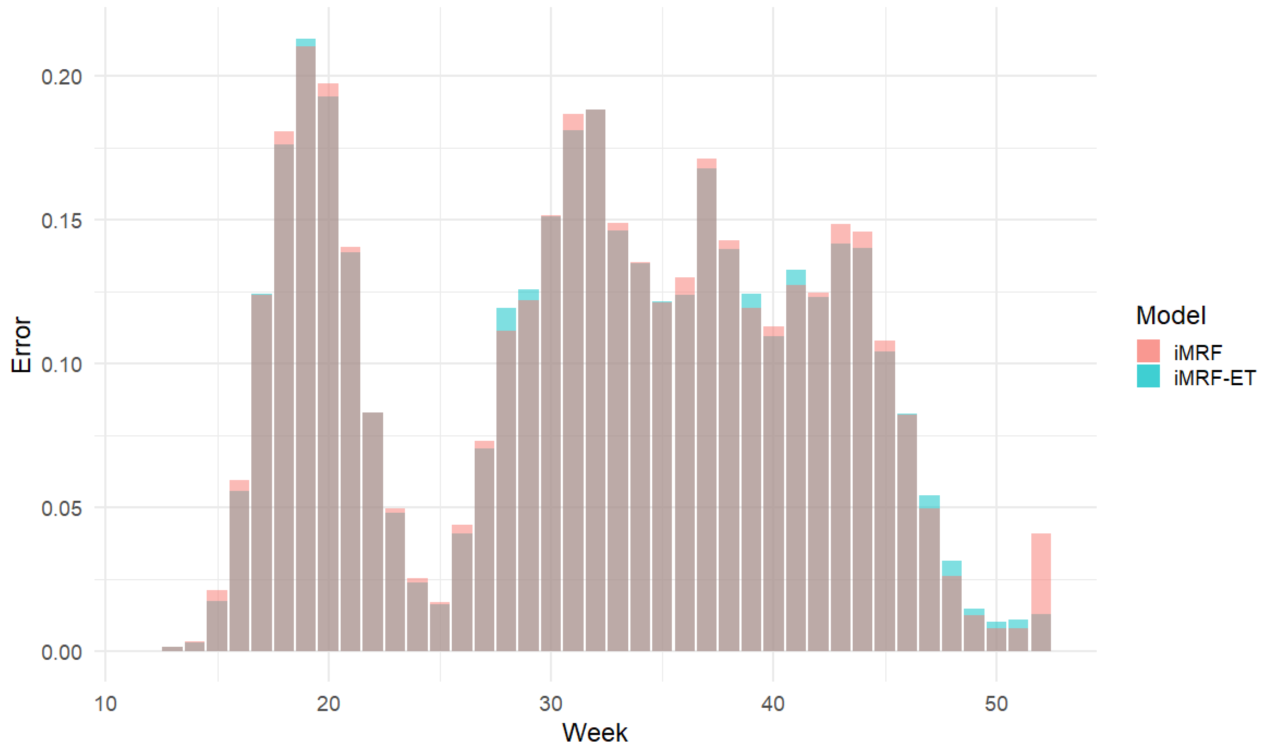

FIGURE 5.6: iMRF vs iMRF - ET. The ARMSPEs are 9.62% and 9.49% respectively.

Randomization effect

We also considered implementing the Extremely Randomized Trees for the case of iMRF. The resulting model is denoted by iMRF-ET. We used the default option of `randomForestSRC` package in `R` for choosing the value of N_{split} which corresponds to choosing 10 random candidate thresholds (Ishwaran et al., 2021a). The results are shown in Figure 5.6. We notice that introducing randomization in the splitting procedure not only reduces computational costs but also improves predictive performance for several weeks.

5.2.2 Correlated MRF

As already mentioned, the iMRF approach uses equation 3.14 to split each node. However, incorporating the correlations among the output variables into the splitting procedure, may yield random forests with better predictive performance, especially when these correlations are substantial. Therefore, another approach we consider is replacing impurity criterion 3.14 with 3.16. Once again, we use standard bootstrap and the scaled aggregation approach. The resulting approach is denoted by cMRF. For our data, the correlation matrix is shown in Table 5.2. As expected, the CPR percentages for each stage are highly correlated with the CPR percentages of the immediately preceding and upcoming stages.

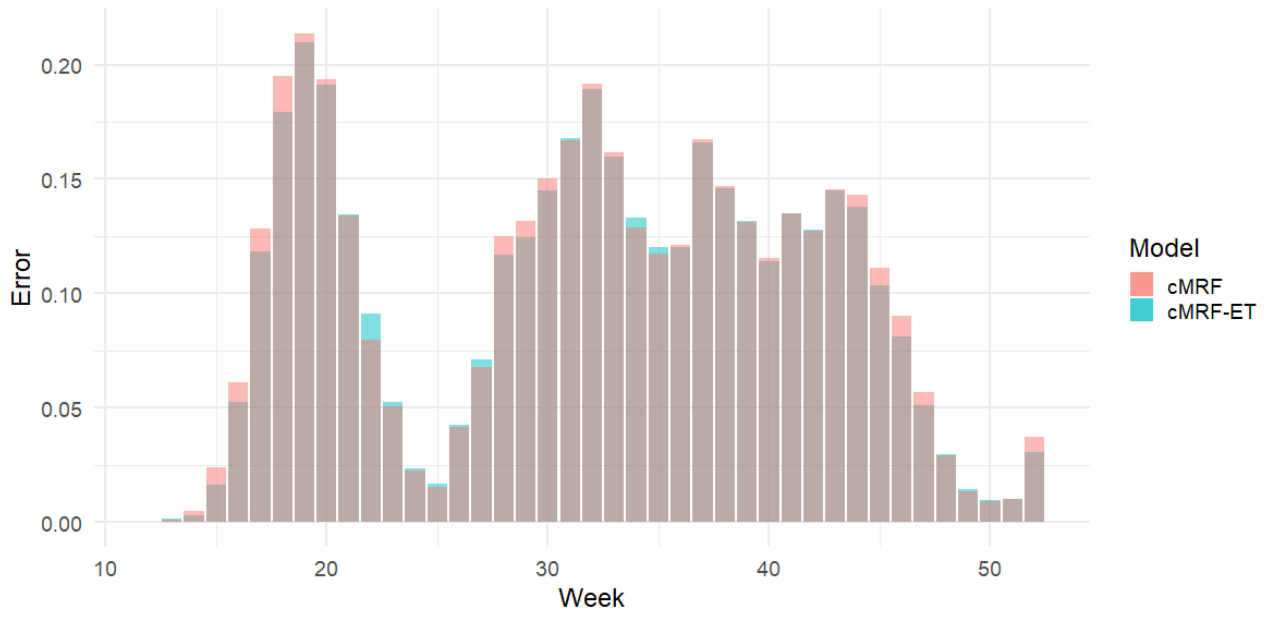

FIGURE 5.7: cMRF (9.75%) vs cMRF-ET (9.55%).

	Planted	Emerged	Silking	Dough	Dented	Mature	Harvested
Planted	1.000	0.937	0.547	0.470	0.414	0.339	0.266
Emerged	0.937	1.000	0.647	0.555	0.489	0.401	0.314
Silking	0.547	0.647	1.000	0.922	0.819	0.671	0.526
Dough	0.470	0.555	0.922	1.000	0.954	0.800	0.628
Dented	0.414	0.489	0.819	0.954	1.000	0.907	0.720
Mature	0.339	0.401	0.671	0.800	0.907	1.000	0.876
Harvested	0.266	0.314	0.526	0.628	0.720	0.876	1.000

TABLE 5.2: The correlation matrix for the response variables.

Randomization effect

In Figure 5.7 we examine the impact of randomizing the splitting procedure for the case of cMRF. As with iMRF, we notice that Extremely Randomized Trees seem to work better for most weeks of the study. Note that for both cMRF and cMRF-ET we used standard bootstrap and the unscaled aggregation approach.

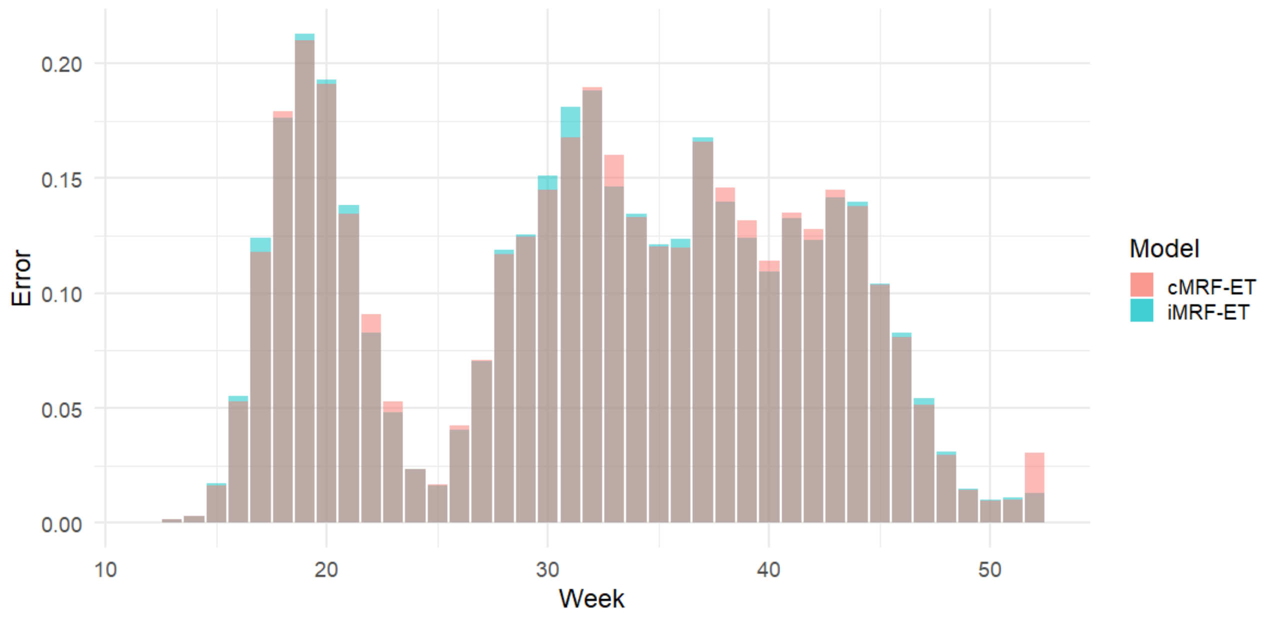

FIGURE 5.8: iMRF-ET (9.49%) vs cMRF-ET (9.55%).

Correlation effect

In Figure 5.8 we compare iMRF-ET with cMRF-ET. We see that incorporating the correlations of the response variables into the splitting procedure does not systematically produce lower errors.

In conclusion, regarding the multivariate approaches, we saw that using Extremely Randomized Trees yields better results in terms of both generalization and computational performance. What is more, concerning the correlated approaches (cMRF, cMRF-ET), we would say that their predictive performance does not compensate for the additional computational cost introduced by the use of the generalized inverse matrix at each split. Therefore, iMRF-ET seems to be the best choice overall, so this will be the multivariate approach to be compared with the stage random forests in the following section.

5.3 Stage Random Forest

All approaches presented so far simultaneously construct a single random forest for all phenological stages. Therefore, for all stages, all 4 covariates (calendar time, thermal time, day length and NDVI) are used to predict their CPR percentages. In this section, we present two additional approaches to the CPSPP. Both of them construct a univariate random forest for each phenological stage separately. In other words, here we assume that the percentages of each phenological stage are related with the features with a function f_s, which may differ from one stage to another.

Doing so provides flexibility in the choice of both the covariates used for predicting the CPR percentages for each stage and the hyperparameters used for each of the random forests. This is desirable, since, as already shown in Table 5.1, the impact of each predictive feature in predicting the CPR percentages changes throughout the growing season. Therefore, in a stage random forest we may replace $\mathbf{X}$ with $\mathbf{X}_s$ in equation 5.1:

$$Y_s = f_s(\mathbf{X}_s) + \epsilon_s, \quad s = 2, 3, \ldots, 8, \tag{5.2}$$

with the hope of improving predicting performance. What is more, hypertuning can now be performed for each random forest separately, allowing for more personalization.

5.3.1 Standard Random Forest

The first stage approach trains a standard Random Forest for each phenological stage. We use standard bootstrap and the scaled aggregation scheme. The resulting approach is denoted by SRF. Like the methods presented in the previous section, this approach is also implemented using the `randomForestSRC` package. To exploit the advantage of using a distinct set of covariates for each stage, we use NDVI only for the prediction of the phenological stages that seems to be associated with (as shown in Table 5.1, namely the Emerged, Dented, Mature and Harvested stages.

Randomization effect

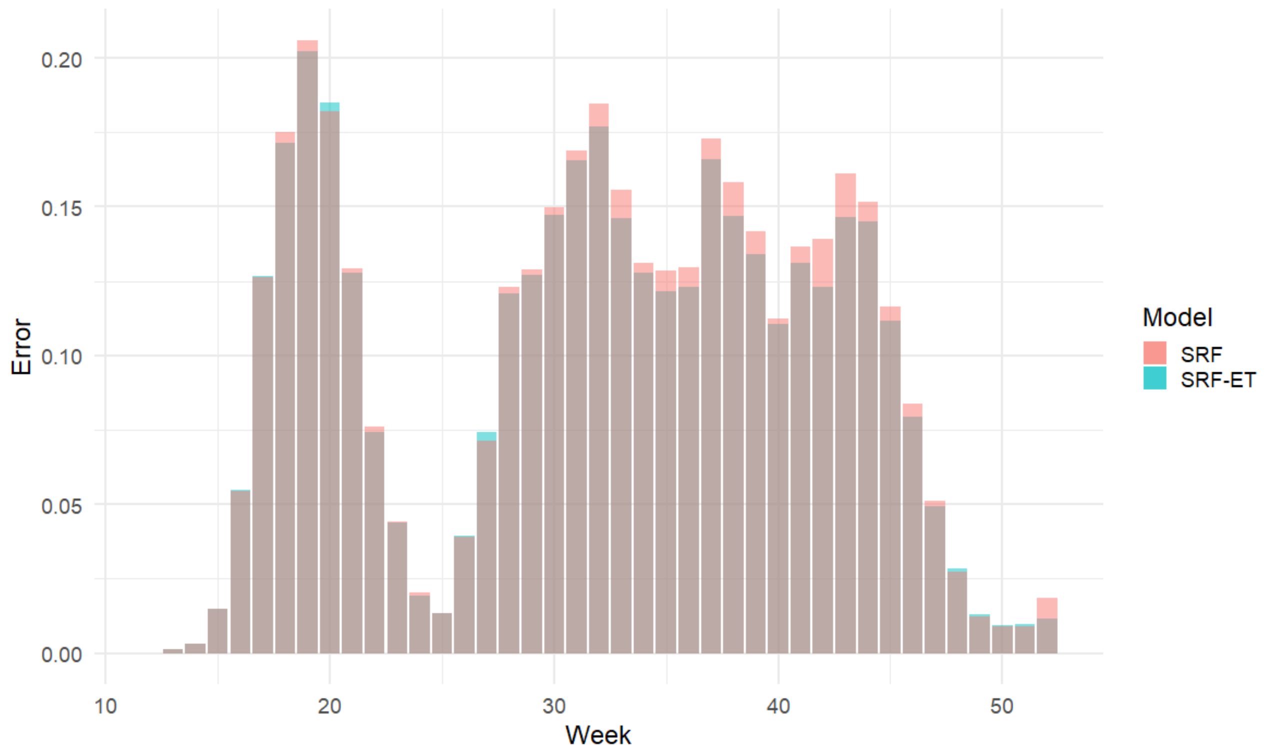

FIGURE 5.9: SRF (9.66%) vs SRF-ET (9.33%).

In Figure 5.9, we investigate whether using Extremely Randomized Trees improve results. Once again, we notice that ETs do improve results for most weeks of the growing season, yielding a lower ARMSPE by 0.33%.

Stage vs Multivariate Random Forest

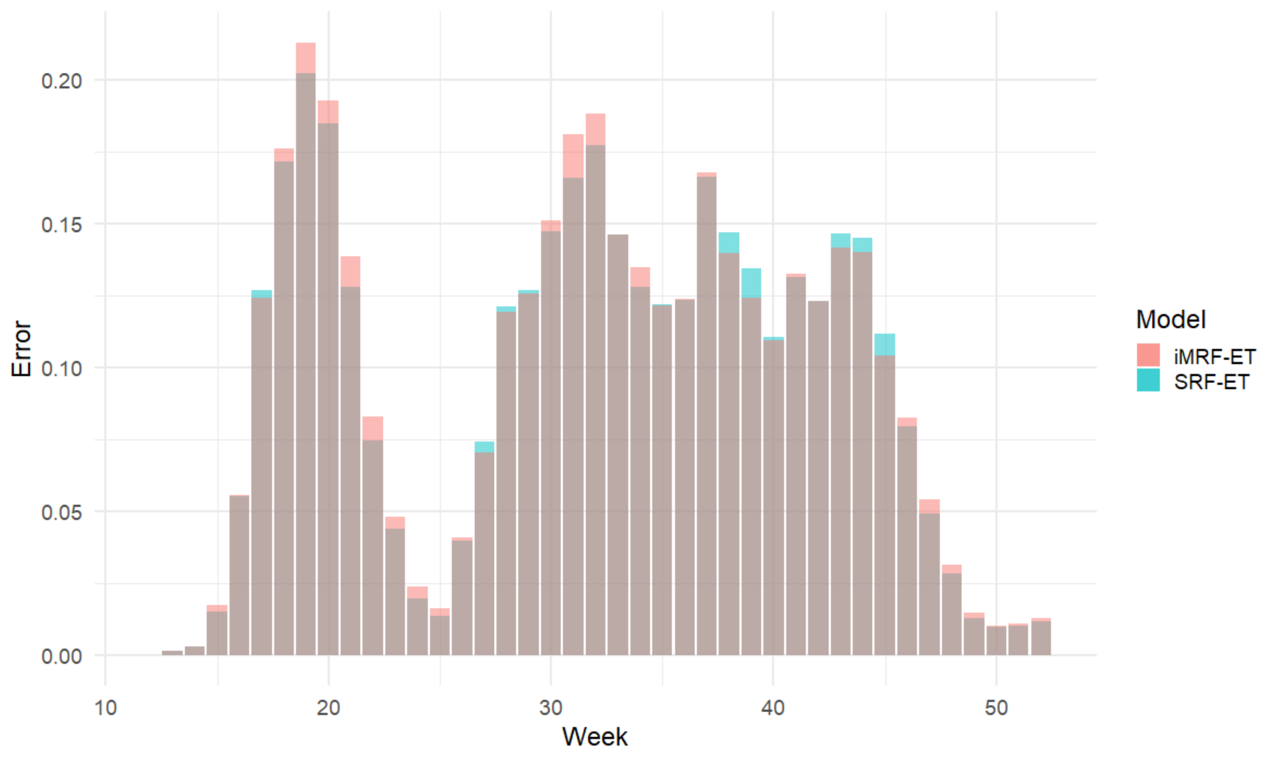

FIGURE 5.10: SRF-ET (9.33%) vs iMRF-ET (9.49%).

In Figure 5.10 we compare the SRF-ET with iMRF-ET, that is, the best-performing approaches of stage and multivariate random forests respectively. We observe that, indeed calibrating the covariates and the hyperparameters used for each phenological stage separately, improves predictions. Yet, there is still room for improvement before we say that stage random forests clearly outperform the multivariate ones.

5.3.2 Historical Random Forest

As a last approach to the CPSPP we considered constructing a historical random forest for each phenological stage. We denote this approach by HRF. The splitting procedure followed for this approach is the one described in section 4.3.3. HRF is implemented using the `htree` package of `R` (Sexton, 2018). Note that, as opposed to the `randomForestSRC` package, `htree` does not provide any options regarding the choice of the bootstrap scheme and aggregation approach. Concerning the former, `htree` package uses *subject-level subsampling*, so each tree is built on a random subsample of size α drawn *without replacement* from the initial learning set. As for the latter, the scaled aggregation approach is used.

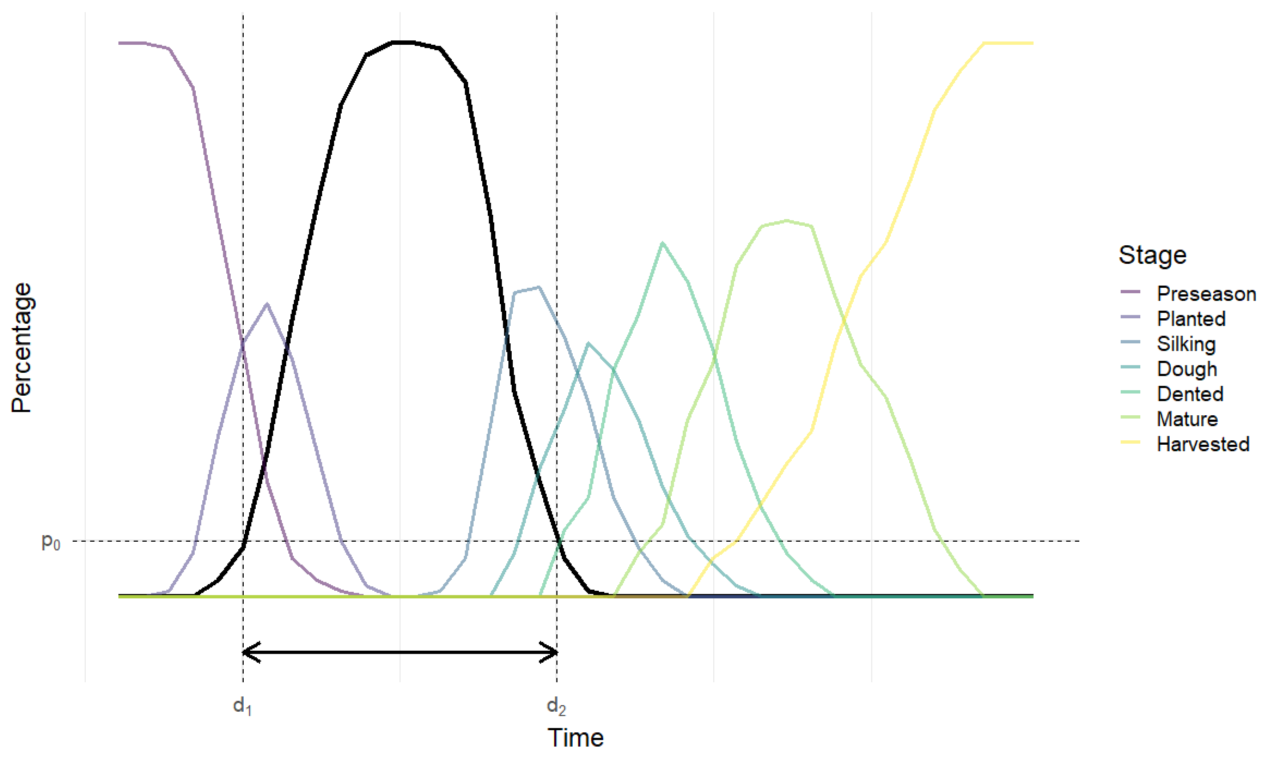

FIGURE 5.11: Time interval during which the average CPR percentages for Emerged stage exceed a threshold p_0.

Choice of lags

For the choice of lags used in equation 4.11, we use 3 different options. The first is to look at the immediately preceding value of each covariate. The second one is to look back as many days as the number of days that the average percentage of each phenological stage exceeds a pre-specified threshold p_0 (see also Figure 5.11) and the third is to look at the entire history from the beginning of the growing season. In other words, we use a different set of lags $\mathcal{D}_s$ for the different random forests applied for each phenological stage. In particular, for a given time moment t_{ij} we use:

$$\mathcal{D}_s = \{\delta_1, \delta_2^s, \delta_3\},$$

where

$$\delta_1 = 1, \qquad \delta_2^s = \left| \left\{ j : \sum_{i \in \mathcal{N}_{train}} \frac{y_{ijs}}{N_{train}} > p_0 \right\} \right|, \qquad \delta_3 = t_{ij} - t_{i1}$$

In our study, we set $p_0 = 0.1$. However, any value in the interval $[0, 0.15]$ yielded similar results in our analysis.

Standard vs Historical Random Forest

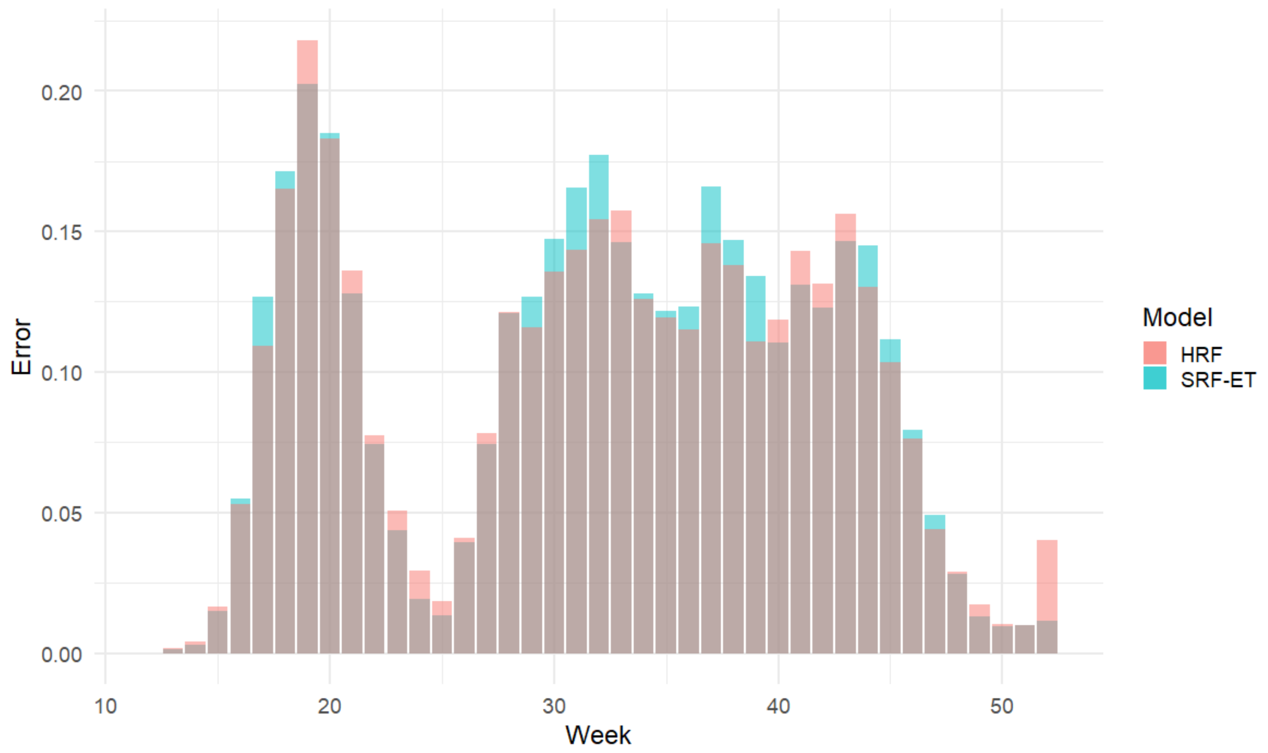

FIGURE 5.12: SRF-ET (9,33%) vs HRF (9.21%)

Figure 5.12 presents a comparative analysis between the historical stage random forest (HRF) and the standard random forest (SRF-ET). The observed performance of HRF surpasses that of SRF-ET in numerous weeks. Although there are instances where SRF-ET outperforms HRF, the ARMSPE of HRF is 0.11% lower. It is strongly hypothesized that this reduction would have been more significant if the restrictions pertaining to bootstrap selection and aggregation schemes were not in place.

Chapter 6

Conclusions

This thesis aimed to provide an efficient solution to the CPSPP along with a comprehensive presentation of the methodology used. To this end, we posed our problem within the machine learning framework, we studied the induction of decision trees and random forests and discussed different approaches regarding bootstrapping, splitting and aggregation. The different options were compared on their performance on the CPSPP using both stage and multivariate random forests.

Concerning the sampling strategy, we saw that subject-level bootstrap did not outperform the standard one, due to the small variability between the growing seasons. Regarding aggregation, we observed minor differences between the scaled and the unscaled approaches for most weeks of the growing season. However, the latter performed clearly better in weeks when there was an absence of data. Contrary to the bootstrap scheme and aggregation approach, randomizing the choice of the cut-point during splitting, turned out to be more influential, improving both computational and predictive performance in all cases.

Comparing the multivariate random forest with the stage ones, we saw that the flexibility invoked by using a distinct set of covariates and hyperparameters improved to some extent the results of the MRF. Implementing the historical splitting, further improved results, achieving the best predictive performance among all the RF-based approaches discussed so far. It is our belief though, that a software implementation that allows users to choose the sampling and aggregation schemes would improve the results of HRF. Yet, creating one such statistical software is beyond the scope of this thesis.

In the near future, we are going to implement more RF-based approaches to the CPSPP that are appropriate to deal with repeated measurements. One such approach is the Repeated Measurements Random Forest (RMRF) proposed by Calhoun et al. (2021), which uses a hypothesis test to split each node following the idea of Hothorn et al. (2006) for more statistically oriented random forests. Other approaches are the Generalized Mixed-Effects Random Forest (GMERF) proposed by Pellagatti et al. (2021) or the so-called DynForest (DRF) approach recently proposed by Devaux et al. (2023). A common aspect within all these approaches is that they involve the training of a GLM or a GLMEM either to split each node (RMRF, DRF) or to estimate the residuals in equations 5.1, 5.2 (GMERF).

Bibliography

L. J. Abendroth, R. W. Elmore, M. J. Boyer, S. K. Marlay, et al. *Corn growth and development*. Iowa State University Ames, 2011.

E. Adam, O. Mutanga, E. M. Abdel-Rahman, and R. Ismail. Estimating standing biomass in papyrus (cyperus papyrus l.) swamp: exploratory of in situ hyperspectral indices and random forest regression. *International Journal of Remote Sensing*, 35(2):693–714, 2014. doi: 10.1080/01431161.2013.870676. URL https://doi.org/10.1080/01431161.2013.870676.

American Meteorological Society, 2023. URL https://web.archive.org/web/20081009142439/http://amsglossary.allenpress.com/glossary/search?id=precipitation1.

Y. Amit, D. Geman, and K. Wilder. Joint induction of shape features and tree classifiers. *IEEE transactions on pattern analysis and machine intelligence*, 19(11):1300–1305, 1997.

K. Bennett. Global tree optimization: A non-greedy decision tree algorithm. *Computing Sciences and Statistics*, 26, 04 1995.

D. Bertsimas and J. Dunn. Optimal classification trees. *Machine Learning*, 106(7):1039–1082, 2017.

D. Bertsimas, J. Dunn, and Y. Wang. Near-optimal nonlinear regression trees. *Operations Research Letters*, 49(2):201–206, 2021.

D. Biggs, B. De Ville, and E. Suen. A method of choosing multiway partitions for classification and decision trees. *Journal of applied statistics*, 18(1):49–62, 1991.

C. Birch, G. Hammer, and K. Rickert. Temperature and photoperiod sensitivity of development in five cultivars of maize (zea mays l.) from emergence to tassel initiation. *Field Crops Research*, 55(1):93–107, 1998. ISSN 0378-4290. doi: https://doi.org/10.1016/S0378-4290(97)00062-2. URL https://www.sciencedirect.com/science/article/pii/S0378429097000622.

R. Blanquero, E. Carrizosa, C. Molero-Río, and D. R. Morales. Sparsity in optimal randomized classification trees. *European Journal of Operational Research*, 284(1):255–272, 2020.

F. Bollwein and S. Westphal. Oblique decision tree induction by cross-entropy optimization based on the von mises–fisher distribution. *Computational Statistics volume*, page 2203–2229, 2022.

L. Breiman. Bagging predictors, 1994.

L. Breiman. Random forests. *Machine learning*, 45(1):5–32, 2001.

L. Breiman. Manual on setting up, using, and understanding random forests v3. 1. *Statistics Department University of California Berkeley, CA, USA*, 1(58):3–42, 2002.

L. Breiman and A. Cutler. Manual–setting up, using, and understanding random forests v4. 0. 2003. *URL https://www. stat. berkeley. edu/˜ breiman/Using_random_forests_v4. 0. pdf*, 2003.

L. Breiman, J. H. Friedman, R. A. Olshen, and C. J. Stone. *Classification and regression trees*. Routledge, 1984.

P. Burman. A comparative study of ordinary cross-validation, v-fold cross-validation and the repeated learning-testing methods. *Biometrika*, 76(3):503–514, 1989. ISSN 00063444. URL `http://www.jstor.org/stable/2336116`.

P. Calhoun, R. A. Levine, and J. Fan. Repeated measures random forests (rmrf): Identifying factors associated with nocturnal hypoglycemia. *Biometrics*, 77(1):343–351, 2021. doi: https://doi.org/10.1111/biom.13284.

E. Cantu-Paz and C. Kamath. Inducing oblique decision trees with evolutionary algorithms. *IEEE Transactions on Evolutionary Computation*, 7(1):54–68, 2003. doi: 10.1109/TEVC.2002.806857.

J.-H. Chang. Corn yield in relation to photoperiod, night temperature, and solar radiation. *Agricultural Meteorology*, 24:253–262, 1981. ISSN 0002-1571. doi: https://doi.org/10.1016/0002-1571(81)90049-2. URL `https://www.sciencedirect.com/science/article/pii/0002157181900492`.

Q. Chen, H. Zhong, X.-W. FAN, and Y.-Z. LI. An insight into the sensitivity of maize to photoperiod changes under controlled conditions. *Plant, Cell & Environment*, 38(8):1479–1489, 2015.

H. Z. Cross and M. S. Zuber. Prediction of flowering dates in maize based on different methods of estimating thermal units1. *Agronomy Journal*, 64(3):351–355, 1972. doi: https://doi.org/10.2134/agronj1972.00021962006400030029x. URL `https://acsess.onlinelibrary.wiley.com/doi/abs/10.2134/agronj1972.00021962006400030029x`.

B. V. Dasarathy and B. V. Sheela. A composite classifier system design: Concepts and methodology. *Proceedings of the IEEE*, 67(5):708–713, 1979.

C. S. Davis. Statistical methods for the analysis of repeated measurements. Technical report, Springer, 2002.

Daymet, 2023. URL `https://daymet.ornl.gov/`.

G. De'Ath. Multivariate regression trees: a new technique for modeling species–environment relationships. *Ecology*, 83(4):1105–1117, 2002.

A. Devaux, C. Proust-Lima, and R. Genuer. Random forests for time-fixed and time-dependent predictors: The dynforest r package. *arXiv preprint arXiv:2302.02670*, 2023.

L. Devroye, L. Györfi, and G. Lugosi. *A Probabilistic Theory of Pattern Recognition*. Springer New York, NY, 1996.

T. G. Dietterich and E. B. Kong. Machine learning bias, statistical bias, and statistical variance of decision tree algorithms. Technical report, Citeseer, 1995.

J. Ding, X. Jiao, P. Bai, Y. Hu, J. Zhang, and J. Li. Effect of vapor pressure deficit on the photosynthesis, growth, and nutrient absorption of tomato seedlings. *Scientia Horticulturae*, 293: 110736, 2022. ISSN 0304-4238. doi: https://doi.org/10.1016/j.scienta.2021.110736. URL `https://www.sciencedirect.com/science/article/pii/S0304423821008438`.

EOS, 2023. URL `https://eos.com/make-an-analysis/ndvi/`.

ESA, 2023. URL `https://www.esa.int/SPECIALS/Eduspace_Global_EN/SEMC6FNW91H_0.html`.

Farm Progress, 2023. URL `https://www.farmprogress.com/vegetables/13-ways-corn-used-our-everyday-lives`.

Y. Freund and R. E. Schapire. A decision-theoretic generalization of on-line learning and an application to boosting. *Journal of computer and system sciences*, 55(1):119–139, 1997.

T. Fulton, S. Kasif, and S. Salzberg. Efficient algorithms for finding multi-way splits for decision trees. In A. Prieditis and S. Russell, editors, *Machine Learning Proceedings 1995*, pages 244–251. Morgan Kaufmann, San Francisco (CA), 1995. ISBN 978-1-55860-377-6. doi: https://doi.org/10.1016/B978-1-55860-377-6.50038-4. URL `https://www.sciencedirect.com/science/article/pii/B9781558603776500384`.

S. Geman, E. Bienenstock, and R. Doursat. Neural networks and the bias/variance dilemma. *Neural computation*, 4(1):1–58, 1992.

T. Z. Geneti. Review on the effect of moisture or rain fall on crop production. *Civ. Environ. Res*, 11(2), 2019.

P. Geurts. *Contributions to decision tree induction: bias/variance tradeoff and time series classification*. PhD thesis, ULiège - Université de Liège, 2002. URL `http://www.montefiore.ulg.ac.be/services/stochastic/pubs/2002/Geu02`.

P. Geurts, D. Ernst, and L. Wehenkel. Extremely randomized trees. *Machine learning*, 63:3–42, 2006.

M. Ghamghami, N. Ghahreman, P. Irannejad, and H. Pezeshk. A parametric empirical bayes (peb) approach for estimating maize progress percentage at field scale. *Agricultural and Forest Meteorology*, 281:107829, 2020. ISSN 0168-1923. doi: https://doi.org/10.1016/j.agrformet.2019.107829. URL `https://www.sciencedirect.com/science/article/pii/S0168192319304459`.

Grain Market Report, 2023. URL `http://www.igc.int/downloads/gmrsummary/gmrsumme.pdf`.

C. Grossiord, T. N. Buckley, L. A. Cernusak, K. A. Novick, B. Poulter, R. T. W. Siegwolf, J. S. Sperry, and N. G. McDowell. Plant responses to rising vapor pressure deficit. *New Phytologist*, 226(6): 1550–1566, 2020. doi: https://doi.org/10.1111/nph.16485.

L. K. Hansen and P. Salamon. Neural network ensembles. *IEEE transactions on pattern analysis and machine intelligence*, 12(10):993–1001, 1990.

T. Hastie, R. Tibshirani, J. Friedman, T. Hastie, R. Tibshirani, and J. Friedman. Random forests. *The elements of statistical learning: Data mining, inference, and prediction*, pages 587–604, 2009a.

T. Hastie, R. Tibshirani, J. H. Friedman, and J. H. Friedman. *The elements of statistical learning: data mining, inference, and prediction*, volume 2. Springer, 2009b.

J. L. Hatfield and J. H. Prueger. Temperature extremes: Effect on plant growth and development. *Weather and Climate Extremes*, 10:4–10, 2015. ISSN 2212-0947. doi: https://doi.org/10.1016/j.wace.2015.08.001. URL `https://www.sciencedirect.com/science/article/pii/S2212094715300116`. USDA Research and Programs on Extreme Events.

D. Heath, S. Kasif, and S. Salzberg. Induction of oblique decision trees. In *IJCAI*, volume 1993, pages 1002–1007. Citeseer, 1993.

Hi-Phen, 2023. URL `https://www.hiphen-plant.com/vegetation-index/3582/#:~:text=A%20Vegetation%20Index%20is%20a,objects%20present%20in%20the%20image`.

R. F. Holt and D. R. Timmons. Influence of precipitation, soil water, and plant population interactions on corn grain yields1. *Agronomy Journal*, 60(4):379–381, 1968. doi: https://doi.org/10.2134/agronj1968.00021962006000040014x.

R. F. Holt, D. R. Timmons, W. B. Voorhees, and C. A. Van Doren. Importance of stored soil moisture to the growth of corn in the dry to moist subhumid climatic zone1. *Agronomy Journal*, 56(1):82–85, 1964. doi: https://doi.org/10.2134/agronj1964.00021962005600010026x.

T. Hothorn, K. Hornik, and A. Zeileis. Unbiased recursive partitioning: A conditional inference framework. *Journal of Computational and Graphical Statistics*, 15(3):651–674, 2006. doi: 10.1198/106186006X133933.

J. Hsiao, A. L. Swann, and S.-H. Kim. Maize yield under a changing climate: The hidden role of vapor pressure deficit. *Agricultural and Forest Meteorology*, 279:107692, 2019.

J. Hu and S. Szymczak. A review on longitudinal data analysis with random forest. *Briefings in Bioinformatics*, 2023.

R. B. Hunter, L. A. Hunt, and L. W. Kannenberg. Photoperiod and temperature effects on corn. *Canadian Journal of Plant Science*, 54(1):71–78, 1974. doi: 10.4141/cjps74-012. URL `https://doi.org/10.4141/cjps74-012`.

L. Hyafil and R. L. Rivest. Constructing optimal binary decision trees is np-complete. *Inf. Process. Lett.*, 5:15–17, 1976.

H. Ishwaran, U. B. Kogalur, E. H. Blackstone, and M. S. Lauer. Random survival forests. *The Annals of Applied Statistics*, 2(3), sep 2008. doi: 10.1214/08-aoas169. URL `https://doi.org/10.1214%2F08-aoas169`.

H. Ishwaran, M. Lu, and U. B. Kogalur. randomForestSRC: speedup random forest analyses vignette, 2021a. URL `http://randomforestsrc.org/articles/speedup.html`.

H. Ishwaran, M. Lu, and U. B. Kogalur. randomForestSRC: getting started with randomForestSRC vignette, 2021b. URL `http://randomforestsrc.org/articles/getstarted.html`.

H. Ishwaran, F. Tang, M. Lu, and U. B. Kogalur. randomForestSRC: multivariate splitting rule vignette, 2021c. URL `http://randomforestsrc.org/articles/mvsplit.html`.

H. Ishwaran, F. Tang, M. Lu, and U. B. Kogalur. randomForestSRC: multivariate splitting rule vignette, 2021d. URL `http://randomforestsrc.org/articles/mvsplit.html`.

S. Janitza and R. Hornung. On the overestimation of random forest's out-of-bag error. *PloS one*, 13 (8):e0201904, 2018.

V. R. Joseph. Optimal ratio for data splitting. *Statistical Analysis and Data Mining: The ASA Data Science Journal*, 15(4):531–538, 2022. doi: https://doi.org/10.1002/sam.11583. URL `https://onlinelibrary.wiley.com/doi/abs/10.1002/sam.11583`.

J. Karl. Maize is not day neutral; day length / photoperiod, 06 2000.

Y. V. Karpievitch, E. G. Hill, A. P. Leclerc, A. R. Dabney, and J. S. Almeida. An introspective comparison of random forest-based classifiers for the analysis of cluster-correlated data by way of rf++. *PloS one*, 4(9):e7087, 2009.

H. Kim and W.-Y. Loh. Classification trees with unbiased multiway splits. *Journal of the American Statistical Association*, 96(454):589–604, 2001. doi: 10.1198/016214501753168271. URL `https://doi.org/10.1198/016214501753168271`.

R. Kohavi. 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, page 1137–1143, San Francisco, CA, USA, 1995. Morgan Kaufmann Publishers Inc. ISBN 1558603638.

K. E. Kunkel, S. A. Changnon, and R. T. Shealy. *Applications of statistical methods to the study of climate and flooding fluctuations in the central United States*, volume 523. Illinois State Water Survey, 1992.

X.-B. Li, J. R. Sweigart, J. T. Teng, J. M. Donohue, L. A. Thombs, and S. M. Wang. Multivariate decision trees using linear discriminants and tabu search. *IEEE transactions on systems, man, and cybernetics-part a: systems and humans*, 33(2):194–205, 2003.

A. Liaw, M. Wiener, et al. Classification and regression by randomforest. *R news*, 2(3):18–22, 2002.

Z. Liu, T. Wen, W. Sun, and Q. Zhang. A novel multiway splits decision tree for multiple types of data. *Mathematical Problems in Engineering*, 2020, 2020.

W.-Y. Loh and Y.-S. Shih. Split selection methods for classification trees. *Statistica sinica*, pages 815–840, 1997.

A. López-Chau, J. Cervantes, L. López-García, and F. G. Lamont. Fisher's decision tree. *Expert Systems with Applications*, 40(16):6283–6291, 2013.

G. Louppe and P. Geurts. Ensembles on random patches. In P. A. Flach, T. De Bie, and N. Cristianini, editors, *Machine Learning and Knowledge Discovery in Databases*, pages 346–361, Berlin, Heidelberg, 2012. Springer Berlin Heidelberg. ISBN 978-3-642-33460-3.

G. Louppe, L. Wehenkel, A. Sutera, and P. Geurts. Understanding variable importances in forests of randomized trees. *Advances in neural information processing systems*, 26, 2013.

P. C. Mahalanobis. On the generalized distance in statistics. *Sankhyā: The Indian Journal of Statistics, Series A (2008-)*, 80:S1–S7, 2018.

Maize. Maize, 2023. URL `https://en.wikipedia.org/wiki/Maize`.

W. Matthew et al. Bias of the random forest out-of-bag (oob) error for certain input parameters. *Open Journal of Statistics*, 2011, 2011.

MODIS, 2023. URL `https://modis.gsfc.nasa.gov/about/`.

F. Mola and R. Siciliano. Discriminant analysis and factorial multiple splits in recursive partitioning for data mining. In *International workshop on multiple classifier systems*, pages 118–126. Springer, 2002.

J. N. Morgan and J. A. Sonquist. Problems in the analysis of survey data, and a proposal. *Journal of the American Statistical Association*, 58(302):415–434, 1963. doi: 10.1080/01621459.1963.10500855. URL `https://www.tandfonline.com/doi/abs/10.1080/01621459.1963.10500855`.

S. K. Murthy. *On growing better decision trees from data*. The Johns Hopkins University, 1996.

S. K. Murthy, S. Kasif, S. Salzberg, and R. Beigel. Oc1: A randomized algorithm for building oblique decision trees. In *Proceedings of AAAI*, volume 93, pages 322–327. Citeseer, 1993.

S. K. Murthy, S. Kasif, and S. Salzberg. A system for induction of oblique decision trees. *Journal of artificial intelligence research*, 2:1–32, 1994.

NDVI, 2023. URL `https://en.wikipedia.org/wiki/Normalized_difference_vegetation_index`.

Nutrien Ekonomics, 2023. URL `https://nutrien-ekonomics.com/latest-fertilizer-research/tissue-and-growth-model-corn/`.

I. Oikonomidis. Regression techniques for aggregate predictions of crop progress stages with remote sensing: An application to usa croplands. Master's thesis, MSc Mathematics, National and Kapodistrian University of Athens, 2020.

I. Oikonomidis and S. Trevezas. Cumulative link mixed-effects models in the service of remote sensing crop progress monitoring, 2023.

ORLN, 2023. URL `https://www.ornl.gov/`.

M. Pellagatti, C. Masci, F. Ieva, and A. M. Paganoni. Generalized mixed-effects random forest: A flexible approach to predict university student dropout. *Statistical Analysis and Data Mining: The ASA Data Science Journal*, 14(3):241–257, 2021. doi: https://doi.org/10.1002/sam.11505.

R. Penrose. A generalized inverse for matrices. *Mathematical Proceedings of the Cambridge Philosophical Society*, 51(3):406–413, 1955. doi: 10.1017/S0305004100030401.

P. Probst and A.-L. Boulesteix. To tune or not to tune the number of trees in random forest. *The Journal of Machine Learning Research*, 18(1):6673–6690, 2017.

P. Probst, A.-L. Boulesteix, and B. Bischl. Tunability: Importance of hyperparameters of machine learning algorithms. *The Journal of Machine Learning Research*, 20(1):1934–1965, 2019a.

P. Probst, M. N. Wright, and A.-L. Boulesteix. Hyperparameters and tuning strategies for random forest. *Wiley Interdisciplinary Reviews: data mining and knowledge discovery*, 9(3):e1301, 2019b.

Quick Stats, 2023. URL `https://www.nass.usda.gov/Quick_Stats/`.

J. R. Quinlan. Induction of decision trees. *Machine learning*, 1(1):81–106, 1986.

J. R. Quinlan. Program for machine learning. *C4.5*, 1993. URL `https://cir.nii.ac.jp/crid/1573387449987904000`.

R Core Team. *R: A Language and Environment for Statistical Computing*. R Foundation for Statistical Computing, Vienna, Austria, 2023. URL `https://www.R-project.org/`.

J. D. Ray, R. W. Gesch, T. R. Sinclair, and L. H. Allen. The effect of vapor pressure deficit on maize transpiration response to a drying soil. *Plant and Soil*, 239, 2002. ISSN 0032079X. doi: 10.1023/A:1014947422468.

Remote Sensing, 2023. URL `https://en.wikipedia.org/wiki/Remote_sensing`.

J. S. Rhodes, A. Cutler, and K. R. Moon. Geometry-and accuracy-preserving random forest proximities. *arXiv preprint arXiv:2201.12682*, 2022.

A. Sage. Random forest robustness, variable importance, and tree aggregation. *Iowa State University Capstones, Theses, and Dissertations*, 2018.

A. J. Sage, U. Genschel, and D. Nettleton. Tree aggregation for random forest class probability estimation. *Statistical Analysis and Data Mining: The ASA Data Science Journal*, 13(2):134–150, 2020.

T. Sakamoto. Incorporating environmental variables into a modis-based crop yield estimation method for united states corn and soybeans through the use of a random forest regression algorithm. *ISPRS Journal of Photogrammetry and Remote Sensing*, 160:208–228, 2020.

R. E. Schapire. The strength of weak learnability. *Machine learning*, 5(2):197–227, 1990.

E. Scornet. Trees, forests, and impurity-based variable importance in regression. *Annales de l'Institut Henri Poincaré, Probabilités et Statistiques*, 59(1):21 – 52, 2023. doi: 10.1214/21-AIHP1240. URL `https://doi.org/10.1214/21-AIHP1240`.

M. Segal and Y. Xiao. Multivariate random forests. *Wiley interdisciplinary reviews: Data mining and knowledge discovery*, 1(1):80–87, 2011.

M. R. Segal. Tree-structured methods for longitudinal data. *Journal of the American Statistical Association*, 87(418):407–418, 1992.

M. R. Segal. Machine learning benchmarks and random forest regression. Technical report, UCSF: Center for Bioinformatics and Molecular Biostatistics, 2004. Retrieved from `https://escholarship.org/uc/item/35x3v9t4`.

J. Sexton. *htree: Historical Tree Ensembles for Longitudinal Data*, 2018. URL `https://CRAN.R-project.org/package=htree`. R package version 2.0.0.

J. Sexton and P. Laake. Historical random forests. Unpublished manuscript, 2018.

F. J. M. Shamrat, S. Chakraborty, M. M. Billah, P. Das, J. N. Muna, and R. Ranjan. A comprehensive study on pre-pruning and post-pruning methods of decision tree classification algorithm. In *2021 5th International conference on trends in electronics and informatics (ICOEI)*, pages 1339–1345. IEEE, 2021.

Y. Shen, W. L., D. L., Y. G., T. H., Y. G., and S. Y. Hidden markov models for real-time estimation of corn progress stages using modis and meteorological data. *Remote Sens*, 5:1734–1753, 4 2013. URL `https://doi.org/10.3390/rs5041734`.

R. Siciliano, M. Aria, and A. D'Ambrosio. Posterior prediction modelling of optimal trees. In *COMPSTAT 2008*, pages 323–334. Springer, 2008.

Smart AKIS, 2023. URL `https://www.smart-akis.com/wp-content/uploads/techhtmpdf/858.pdf`.

D. J. Stekhoven and P. Bühlmann. Missforest—non-parametric missing value imputation for mixed-type data. *Bioinformatics*, 28(1):112–118, 2012.

A. Sutera, G. Louppe, V. A. Huynh-Thu, L. Wehenkel, and P. Geurts. Context-dependent feature analysis with random forests. *arXiv preprint arXiv:1605.03848*, 2016.

F. Tang and H. Ishwaran. Random forest missing data algorithms. *Statistical Analysis and Data Mining: The ASA Data Science Journal*, 10(6):363–377, 2017.

A. Truong. *Fast growing and interpretable oblique trees via logistic regression models*. PhD thesis, Oxford University, UK, 2009.

J. W. Tukey et al. *Exploratory data analysis*, volume 2. Reading, MA, 1977.

S. Wang, G. Azzari, and D. B. Lobell. Crop type mapping without field-level labels: Random forest transfer and unsupervised clustering techniques. *Remote Sensing of Environment*, 222:303–317, 2019. ISSN 0034-4257. doi: https://doi.org/10.1016/j.rse.2018.12.026. URL `https://www.sciencedirect.com/science/article/pii/S0034425718305790`.

I. J. Warrington and E. T. Kanemasu. Corn growth response to temperature and photoperiod. iii. leaf number1. *Agronomy Journal*, 75(5):762–766, 1983. doi: https://doi.org/10.2134/agronj1983.00021962007500050010x. URL `https://acsess.onlinelibrary.wiley.com/doi/abs/10.2134/agronj1983.00021962007500050010x`.

D. C. Wickramarachchi, B. L. Robertson, M. Reale, C. J. Price, and J. Brown. Hhcart: an oblique decision tree. *Computational Statistics & Data Analysis*, 96:12–23, 2016.

World Population Review, 2023. URL `https://worldpopulationreview.com/state-rankings/corn-production-by-state`.

M. N. Wright and A. Ziegler. ranger: A fast implementation of random forests for high dimensional data in c++ and r. *Journal of Statistical Software*, 77(1):1–17, 2017. doi: 10.18637/jss.v077.i01. URL `https://www.jstatsoft.org/index.php/jss/article/view/v077i01`.

H. Zhang, D. Nettleton, and Z. Zhu. Regression-enhanced random forests. *arXiv preprint arXiv:1904.10416*, 2019.

P. Zhang. Model Selection Via Multifold Cross Validation. *The Annals of Statistics*, 21(1):299 – 313, 1993. doi: 10.1214/aos/1176349027. URL `https://doi.org/10.1214/aos/1176349027`.