PEDRO JORGE MIGUEIS VAZ MARTINS

SMART IRRIGATION PLATFORM BASED ON MACHINE AND DEEP LEARNING

UNIVERSIDADE DO ALGARVE
Instituto Superior de Engenharia
2022

PEDRO JORGE MIGUEIS VAZ MARTINS

SMART IRRIGATION PLATFORM BASED ON MACHINE AND DEEP LEARNING

Master in Electrical and Computer Engineering
Specialization in Information Technologies and Telecommunications

Work done under the supervision of:
Maria Gabriela Figueiredo de Castro Schütz, Ph.D.
Pedro Jorge Sequeira Cardoso, Ph.D.

UNIVERSIDADE DO ALGARVE
Instituto Superior de Engenharia
2022

SMART IRRIGATION PLATFORM BASED ON MACHINE AND DEEP LEARNING

Declaração de autoria de trabalho

Declaro ser o autor deste trabalho, que é original e inédito. Autores e trabalhos consultados estão devidamente citados no texto e constam da listagem de referências incluída.

I hereby declare to be the author of this work, which is original and unpublished. Authors and works consulted are properly cited in the text and included in the reference list.

(Pedro Jorge Migueis Vaz Martins)

Abstract

Due to climate change, the hydrological drought is assuming a structural character with a tendency to worsen in many countries. In fact, the frequency and intensity of droughts is predicted to increase, particularly in the Mediterranean region and Southern Africa. Since a fraction of the fresh water that is consumed is used to irrigate urban fabric green spaces, which are typically made up of gardens, lanes and roundabouts, it is urgent to implement water waste prevention policies. Reference evapotranspiration (ET_o) is a measurement that can be used to estimate the amount of water being taken up or used by plants, allowing a better management of the watering volumes but, the exact computation of the evapotranspiration is not possible without using complex and expensive sensor systems.

As an alternative to the devoted sensor solutions, weather parameters can be used to estimate ET_o. However that also raises some problems, such as the fact that it requires the use of a dedicated weather station or of data available on the internet. In both contexts, solar radiation (SR) is not commonly available, and since evapotranspiration is dependent on solar radiation, the need to develop ET_o prediction models that do not require it as an input parameter arises.

This thesis presents some high accuracy reference evapotranspiration and solar radiation prediction models, both having as input a set of limited meteorological features, namely, temperature, humidity and wind, which exclude the need for solar radiation

as a parameter. Two approaches were explored for deriving such models: (i) the use of machine learning algorithms like linear regression (OLS, Ridge, Lasso), k-Nearest Neighbors (kNN), Support Vector Machine (SVM), Decision Tree and Random Forest, and (ii) the use of neural networks such as Artificial Neural Networks (ANN), Long Short Term Memory Networks (LSTM), Gated Recurrent Unit Networks (GRU), Recursive Neural Networks (RNN), and the development of hybrid neural network models such as LSTM-ANN, RNN-ANN, and GRU-ANN.

Using experimental data collected from a weather station located in Vale do Lobo, south Portugal, and using the machine learning approach mentioned in the previous paragraph, (i), the best performing ET_o model gave a coefficient of determination (R^2) of 0.975, and the best SR model gave an R^2 of 0.831, over the test dataset. When using the neural networks approach, (ii), the best performing ET_o model gave an R^2 of 0.977, and the best SR model gave an R^2 of 0.833.

As a final notice, the limited meteorological input parameters were carefully selected so that they are compatible with online freely available weather forecast services.

Keywords: Artificial Neural Networks, Machine Learning, Evapotranspiration, Solar Radiation, Smart Irrigation, Public Garden

Resumo

Devido às mudanças climáticas, a seca hidrológica está a assumir um carácter estrutural com tendência a se agravar em muitos países. Prevê-se que a frequência e a intensidade das secas aumentem, particularmente na região do Mediterrâneo e na África Austral. Uma vez que uma fração da água doce consumida é utilizada para regar os espaços verdes do tecido urbano, que são tipicamente jardins, arruamentos e rotundas, urge implementar políticas de prevenção do desperdício de água. A evapotranspiração de referência (ET_o) é uma medida que pode ser utilizada para estimar a quantidade de água absorvida ou utilizada pelas plantas, permitindo uma melhor gestão dos volumes de irrigação, mas o cálculo exato da evapotranspiração não é possível sem o uso de sistemas de sensores complexos e caros.

Como alternativa à utilização de sensores dedicados, também se pode utilizar parâmetros meteorológicos para estimar a ET_o, no entanto, tal levanta alguns problemas, como a necessidade de uma estação meteorológica dedicada ou a utilização de dados disponíveis na Internet. Neste contexto, a radiação solar geralmente não está disponível, e como a evapotranspiração depende da radiação solar (SR), surge a necessidade de desenvolvimento de modelos de previsão de ET_o que não a requeiram como parâmetro de entrada.

Esta tese estuda vários modelos de previsão de evapotranspiração de referência e radiação solar, ambos tendo como entrada apenas três parâmetros meteorológicos, a saber,

temperatura, humidade e vento, o que exclui a necessidade da radiação solar como elemento de entrada. Duas abordagens foram exploradas para derivar tais modelos: (i) o uso de algoritmos de aprendizagem de máquina: regressão linear (OLS, Ridge, Lasso), *k-Nearest Neighbors* (kNN), *Support Vector Machine* (SVM), *Decision Tree* e *Random Forest*, e (ii) o uso de redes neuronais: *Artificial Neural Networks* (ANN), *Long Short Term Memory Networks* (LSTM), *Gated Recurrent Unit Networks* (GRU), *Recursive Neural Networks* (RNN), e o desenvolvimento de modelos de redes neuronais híbridas: LSTM-ANN, RNN-ANN, e GRU-ANN.

Utilizando dados experimentais recolhidos de uma estação meteorológica localizada em Vale do Lobo, sul de Portugal, e seguindo os métodos de aprendizagem de máquina, referidos no parágrafo anterior, o modelo de ET_o com o melhor desempenho obteve um coeficiente de determinação (R^2) de 0,975, e o melhor modelo de SR obteve um R^2 de 0,831, sobre o conjunto de dados de teste. Ao utilizar as abordagens suportadas em redes neurais, os modelos de ET_o e SR com melhor desempenho obtiveram um (R^2) de 0,977 e 0,833, respectivamente.

Por fim, é de notar que os parâmetros meteorológicos limitados de entrada foram cuidadosamente selecionados para que sejam compatíveis com os serviços de previsão meteorológica disponíveis online de forma gratuita.

Palavras chave: Redes Neuronais Artificiais, Aprendizagem de Máquina, Evapotranspiração, Radiação Solar, Rega Inteligente, Jardim Público

Dedicated to my grandfather,

Eduardo J. Vaz

Acknowledgements

I would like to express my sincere gratitude to my advisors, Professors Gabriela Schütz and Pedro Cardoso for all the support and guidance during research and writing of this thesis. Their support, insightful comments and motivation has always been invaluable for the conduction of this work. Besides my advisors, I would also like to thank Professor Carlos Guerrero for the support and knowledge shared in agronomy and irrigation fields. Thank you all for allowing me to participate on project GSSIC – Green Spaces SMART Irrigation Control, grant number ALG-01-0247-FEDER047030, supported by CRESC Algarve 2020, Portugal 2020, FEDER. Also, particular thanks to GSSIC project's companies Visualforma - Tecnologias de Informação, S.A. and Itelmatis, Lda.

Thanks to Instituto Superior de Engenharia and all it's teaching body. Having access to your specialized knowledge in so many diverse engineering fields was key and determinant for both my professional and academic development, for that I am eternally grateful. I would also like to thank my fellow Computer Vision lab colleagues for the good and positive work environment and knowledge sharing, namely: Professor João Rodrigues, Ricardo Veiga, and Tomás Mendes.

A very special thanks goes to Carmen Zeiß for all the support and always being there. Last but not least, I would like to thank my parents, family, friends and colleagues for the support given.

Contents

List of Tables

List of Figures

List of Abbreviations

ANN	Artificial Neural Network.
API	Application Programming Interface.
CPU	Central Processing Unit.
DB	Database.
ET_c	Crop Evapotranspiration.
ET_o	Reference Evapotranspiration.
FAO	Food and Agriculture Organization of the United Nations.
FAO56-PM	FAO-56 Penman-Monteith.
FC	Field Capacity.
GRU	Gated Recurrent Unit.
GSSIC	Green Spaces SMART Irrigation Control.
ICCS	International Conference on Computational Science.
IPCC	Intergovernmental Panel on Climate Change.
IPMA	Instituto Potuguês do Mar e da Atmosfera.
JSON	JavaScript Object Notation.
K_{cb}	Basal Crop Coefficient.
K_e	Soil Surface Evaporation Component.
KNN	K-Nearest Neighbors.
LASSO	Lasso Regression.
LSTM	Long Short-Term Memory.
MAD	Management Allowable Depletion.
MAE	Mean Absolute Error.

MAPE	Mean Absolute Percentage Error.
MEIOR	Intelligent Scheduling and Optimized Watering Module.
ML	Machine Learning.
OLS	Ordinary Least Squares.
OWM	OpenWeatherMap.
R^2	Coefficient of Determination.
RESTFul	Representational State Transfer.
RF	Random Forest.
RIDGE	Ridge Regression.
RNN	Recursive Neural Network.
SR	Solar Radiation.
SVM	Support Vector Machine.
WP	Wilting Point.
WS	Weather Station.

— Logic is the beginning of wisdom, not the end.

Mr. Spock

1

Introduction

The hydrological drought is becoming a structural character with a tendency to worsen. According to the climate report "Climate Change and Land", from August 2019, by the Intergovernmental Panel on Climate Change, due to climate change, the frequency and intensity of droughts will increase, particularly in the Mediterranean region and Southern Africa (Shukla et al., 2019). A fraction of the fresh water that is consumed is used to irrigate green spaces in the urban fabric, which typically consist of gardens, lanes and roundabouts, as well as green spaces in hotel and resort chains. The irrigation methodology of these green spaces makes use of basic stand-alone irrigation controllers that have no connectivity, and are configured, in most situations, according to the experience of the responsible for the green space maintenance. This is the typical profile of the irrigation methodology found in an overwhelming number of

green spaces.

A known strategy to make a more water efficient irrigation system, is to rely on soil humidity sensors and keep the humidity levels between the field capacity (FC) and the management allowable depletion (MAD), which is a percentage of the available water holding capacity (Zhang et al., 2021). Figure 1.1 shows the various levels of soil moisture from saturation to permanent wilting point. Apart from cost, the problem of implementing such a system on a public green space is that soil sensors may be exposed to vandalism or theft, since normally there is no security or surveillance in such locations. Further, the available water holding capacity changes significantly with soil type (JONG and Shields, 1988), requiring that for each specific soil, samples would need to be sent for laboratory analyses, in order to determine the soil humidity values corresponding to the field capacity and wilting point.

As an alternative to the use of humidity sensors, the crop evapotranspiration (ET_c), also known as the crop water use, which is the water that is used by a crop (Al-Kaisi and Broner, 2009) can be used to estimate crop's watering requirements. Although many studies were made, resulting in a high number of methods for reference evapotranspiration (ET_o) computation (Thornthwaite, 1948; Blaney and Criddle, 1962; Hargreaves and Samani, 1982; Priestley and Taylor, 1972; Makkink, 1957; Penman, 1948; Doorenbos, 1977), the *Food and Agriculture Organization of the United Nations* (FAO)

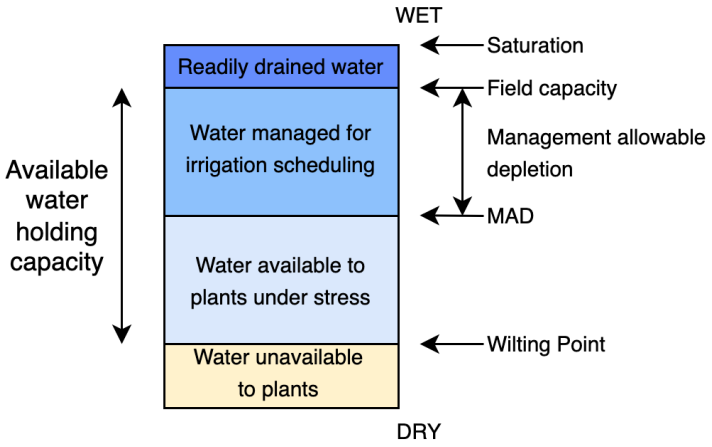

Figure 1.1: Soil water reservoir components. (Adapted from: Sharma (2022))

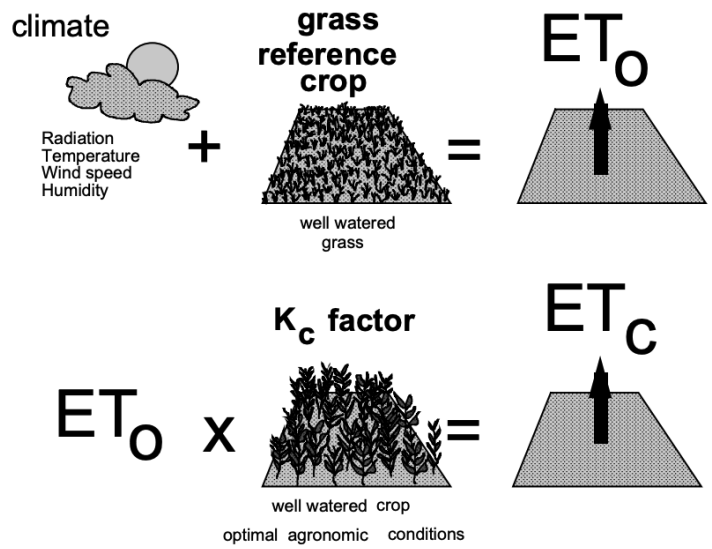

Figure 1.2: Graphic representation of ET_o computation. (Source: Allen et al. (1998))

recommends using the FAO-56 Penman-Monteith (FAO56-PM) formula as a reference method for ET_o computation (Allen et al., 1998). ET_c and ET_o are related by the crop coefficient, K_c, as graphically depicted on Fig. 1.2, where $ET_c = K_c ET_o$, with ET_o being the reference evapotranspiration, as defined by Allen et al. (1998). Figure 1.2 also suggests that computation of ET_o, as per the FAO56-PM formula, requires four main meteorological parameters: temperature, humidity, wind and solar radiation (SR). In fact, it is well known that SR is the main driver of ET_o (Allen et al., 1998). However, its measurement requires sensors like pyranometers, which are typically associated with expensive weather stations that need to be properly maintained and calibrated. Also, solar radiation forecast application programming interfaces (APIs) are not common and generally present an extra system cost.

Hence, the approach presented on this thesis for designing a sensorless smart irrigation system is to predict current and future ET_o needs, by developing ET_o prediction models supported on machine and deep learning, that use as input limited meteorological parameters, namely, temperature, humidity and wind. The model input parameters, as well as precipitation information, can be easily obtained from weather forecast APIs, making possible the calculation of current and future crop's water needs. This allows the definition of the required watering volumes in such a way that plants keep their

health, while water and all other associated costs (e.g., energy) are being saved.

1.1 Scope of this thesis

The work presented on this thesis is part of the author's contribution to the Intelligent Scheduling and Optimized Watering Module (MEIOR) which is part of project GSSIC – Green Spaces SMART Irrigation Control, grant number ALG-01-0247-FEDER-047030, supported by CRESC Algarve 2020, Portugal 2020, FEDER. The goal of the MEIOR module is the computation of crop water requirements, derived from reference evapotranspiration prediction models, providing an optimal irrigation schedule in terms of start(s) and duration(s), optimizing the water and energy expenditure, as well as the well-being of the crop. The development of the MEIOR module started on July 2021 and is ongoing. The MEIOR module culminates in the delivery of a Smart Irrigation Control Framework, which will be supported in the use of machine and deep learning, meteorological data from weather stations on the field, as well as data and meteorological forecasts from APIs available on the internet. Through the use of the ET_o models developed in this thesis, the MEIOR module will be able to predict current and future irrigation needs, contributing to reducing water consumption, and have a proper water management in terms of future needs.

The main contribution of this thesis is the study and development of several ET_o prediction models, that use as input limited weather parameters, namely, temperature, humidity and wind, excluding the need of solar radiation as an input feature.

1.2 Objectives

A smart irrigation platform can be developed using data from various sources such as weather forecast services, field probes, weather stations, etc. In this thesis, as main objectives, the author proposes the research and development of a framework consisting

of:

- **Automatically collect, analyse and process data from various sources such as sensors, probes, weather stations and weather services:** meteorological data from Vale do Lobo weather station was collected, analysed and further processed into a dataset, that was used in Chaps. 2 and 3. Further, as presented in Sec. 4.1, a Python OpenWeatherMap (OWM) (OpenWeatherMap, 2022) data scraper module was developed, that is storing weather forecasts and nowcasts since September 2021 into a mongoDB database (MongoDB, 2022). Finally, in Sec. 4.2 a detailed API proposal is presented for interfacing with the MEIOR module, although the final development and integration of this module is out of the scope of this thesis.

- **Use APIs from weather forecasting services to estimate relevant agricultural parameters for irrigation scheduling:** ET_o is a key parameter for irrigation scheduling, and the studies presented in Chaps. 2 and 3 use limited weather parameters as input features to the ET_o and SR models that were developed, being compatible with weather forecasting services like OWM or Instituto Português do Mar e da Atmosfera (IPMA) (IPMA, 2022).

- **Apply machine and deep learning algorithms to improve the estimation of agricultural parameters such as reference evapotranspiration:** the studies presented in Chaps. 2 and 3 are the main contribution of this thesis, and present high accuracy reference evapotranspiration and solar radiation prediction models that use as input limited weather features.

- **Provide an API to access the framework, which allows a user to transparently consult the future (forecasts/schedule), present and past of irrigation scheduling, weather forecasts, measurements of agricultural and atmospheric parameters:** on Chap. 4 a detailed API proposal specification is provided, however the final integration with the GSSIC project's irrigation platform is out of the scope of this thesis, being future work.

- **Publish the results obtained in at least one scientific journal or conference:** the

study presented in Chap. 2 was published on the International Conference on Computational Science (Vaz et al., 2022b), and the study presented in Chap. 3 was submitted for publication on the IEEE Access Journal (Vaz et al., 2022a).

1.3 Overview of the thesis

The current chapter presented the theme of this thesis, objectives, contributions and scope. The next two chapters correspond to publications made by the author on the subject of evapotranspiration and solar radiation prediction. The first publication was published on the International Conference on Computational Science (ICCS) (Vaz et al., 2022b) and the second was submitted for publication on the IEEE Access Journal (Vaz et al., 2022a). Since both publications present a continuous work on the same subject and objectives, information between publications may overlap on those chapters, mainly in the introduction and state-of-the-art sections.

In more detail, Chap. 2 presents the results obtained by the use of ML algorithms for the prediction of evapotranspiration and solar radiation over limited weather parameters. The ML algorithms used are linear regression (OLS, Ridge, Lasso), k-Nearest Neighbors (kNN), Support Vector Machine (SVM), Decision Tree and Random Forest. All model input features are compatible with weather parameters provided by common weather forecast online services. Since solar radiation is not provided by common online weather forecast services, or present a high-cost penalty to acquire it, different approaches were taken: (i) directly estimate ET_o using limited input features, namely, maximum and minimum temperature, average humidity and average wind, (ii) firstly estimate solar radiation using the limited features, and then inject the estimated solar radiation as a feature into another ML model, and (iii) a hybrid approach that uses the estimated solar radiation as input for ET_o computation using the FAO56-PM method. The hybrid approach presented the best result, with an R^2 equal to 0.975, mean absolute error (MAE) of 0.18 mm/day and a mean absolute percentage error (MAPE) of 5.51 %. As mentioned, these results were accepted for publication on the International

Conference on Computational Science (Vaz et al., 2022b).

Chapter 3 continuous the previous work and has the same overall strategy as described in the last paragraph, but uses neural network algorithms, instead of ML algorithms. The following neural network algorithms were used: Artificial Neural Networks (ANN), Long Short Term Memory Networks (LSTM), Gated Recurrent Unit Networks (GRU), Recursive Neural Networks (RNN), and hybrid models like, LSTM-ANN, GRU-ANN, and RNN-ANN. As we will see, the advantage of the recursive hybrid models is that they require less training and inference time, since the number of trainable parameters is highly reduced due to the smaller networks that are used, while giving similar performance to the recursive non-hybrid models. Similarly to Chap. 2, the hybrid approach (estimate solar radiation, and use it as an input to FAO56-PM method) yielded the best result, with an R^2 equal to 0.977, MAE of 0.16 mm/day and a MAPE of 5.05 %. Compared with the results in Chap. 2, all studied neural network algorithm based models gave better performance than the ML based algorithms for both ET_o and SR estimation. At the time of writing, the journal article corresponding to this part of the study was already submitted to IEEE Access.

Chapter 4 presents a Python module that was developed for OWM meteorological data scraping and storage, and a RESTful API specification proposal for integration of the smart irrigation MEIOR module on the main GSSIC project.

Finally, Chap. 5 presents the discussion of the work done in this thesis, conclusions and future work suggestions.

— One can begin to reshape the landscape with a single flower.

Mr. Spock

2

A study on the prediction of evapotranspiration using freely available meteorological data

Summary

Due to climate change, the hydrological drought is assuming a structural character with a tendency to worsen in many countries. The frequency and intensity of droughts is predicted to increase, particularly in the Mediterranean region and in Southern Africa. Since a fraction of the fresh water that is consumed is used to irrigate urban

fabric green spaces, which are typically made up of gardens, lanes and roundabouts, it is urgent to implement water waste prevention policies. Evapotranspiration (ET_o) is a measurement that can be used to estimate the amount of water being taken up or used by plants, allowing a better management of the watering volumes but, the exact computation of the evapotranspiration volume is not possible without using complex and expensive sensor systems.

In this study, several machine learning models were developed to estimate reference evapotranspiration and solar radiation from a reduced-feature dataset, such has temperature, humidity, and wind. Two main approaches were taken: (i) directly estimate ET_o, or (ii) previously estimate solar radiation and then inject it into a function or method that computes ET_o. For the later case, two variants were implemented, namely the use of the estimated solar radiation as (ii.1) a feature of the machine learning regressors and (ii.2) the use of FAO-56PM method to compute ET_o, which has solar radiation as one of the input parameters. Using experimental data collected from a weather station located in Vale do Lobo, south Portugal, the later approach achieved the best result with a coefficient of determination (R^2) of 0.975 over the test dataset. As a final notice, the reduced-set features were carefully selected so that they are compatible with online freely available weather forecast services.

2.1 Introduction

The hydrological drought is assuming a structural character with a tendency to worsen in regions such as Algarve, Portugal. The problem is not particular to the region occurring, e.g., in most countries of the Mediterranean basin. The climate report, "Climate Change and Land", from August 2019, by the Intergovernmental Panel on Climate Change (IPCC) (Shukla et al., 2019), predicts that, due to climate change, the frequency and intensity of droughts will increase, particularly in the Mediterranean region and in Southern Africa.

A fraction of the fresh water that is consumed by humans is used to irrigate green spaces in the urban fabric, which are typically made up of gardens, lanes and roundabouts, as well as green spaces in hotel and resort chains. The irrigation methodology of these green spaces is commonly done using basic irrigation controllers that are configured according to the experience of those responsible for maintaining the green spaces, without the use of information regarding climate, plants, or soils, as well as data from sensors, nearby weather stations, and from a weather forecast application programming interface (API) that can provide real-time and predicted information. Furthermore, common irrigation controllers have no connectivity, are stand-alone solutions where irrigation schedules are pre-programmed, and only in more complete versions allow irrigation inhibition by means of a rain detection sensor. This is the typical profile that can be found in the overwhelming majority of green space irrigation control systems.

Evapotranspiration (ET_o) is a measurement that can be used to estimate the amount of water being taken up or used by plants, allowing a better management of the watering volumes. However, its exact computation is not possible without using complex and expensive sensor systems being many times estimated by formulas or other methods. The use of one over the other depends many times on the available weather parameters.

This chapter presents part of a framework to estimate ET_o supported by the use of machine learning, acquired intelligence, meteorological data from weather stations on the field, as well as meteorological data and forecasts from APIs available on the internet. The framework will include the computation of crop water requirements derived from the ET_o prediction methods, providing an optimal irrigation schedule in terms of start(s) and duration(s), in order to optimize water expenditure, energy expenditure, and the well-being of the crop. This allows the development of an intelligent irrigation solution, technologically differentiated from other platforms on the market, using innovative communications technology, hardware and software, aggregating devices such as probes, field controllers, meteorological stations, among others. The devel-

opment of the full framework will be done in project GSSIC – Green Spaces SMART Irrigation Control which is developing an innovative intelligent irrigation solution, in terms of reducing water consumption, reducing reaction time in solving problems, increasing efficiency in detecting anomalies, and maintaining the quality of green spaces.

The main contribution of this chapter is the study and proposal of a set of methods to estimate ET_o and solar radiation using features commonly available in most open weather forecast APIs.

The remaining chapter is structured as follows. The next section presents the problem's background and the methodologies used by others to tackle the problem in study. Section 2.3 explores the dataset and explains the computational setup. The fourth section presents the proposed methods and the associated performance analysis. The final section presents some conclusion and establishes some future work.

2.2 Problem's background

Water requirements depend on the reference evapotranspiration (ET_o) which is one of the fundamental parameters for irrigation scheduling as well as improving the management and use of water resources. Prediction of reference evapotranspiration for the following days plays a vital role in the design of intelligent irrigation scheduling, as it is proportional to the amount of water that will have to be restored during the irrigation period (Ferreira et al., 2019).

Some of the main characteristics that distinguish crop evapotranspiration (ET_c) from ET_o are (i) the crop cover density and total leaf area, (ii) the resistance of foliage epidermis and soil surface to the flow of water vapor, (iii) the aerodynamic roughness of the crop canopy, and (iv) the reflectance of the crop and soil surface to short wave radiation (Allen, 2003). In this context, known the crop coefficient (K_c), the crop evap-

otranspiration (ET_c) value for a specific time period can be estimated by

$$\mathrm{ET}_c = K_c ET_o.$$

The crop coefficient can be simple or have two components, one representing the basal crop coefficient (K_{cb}) and another representing the soil surface evaporation component (K_e), being computed by

$$K_c = K_s K_{cb} + K_e,$$

where $K_s \in [0,1]$ is used to introduce a K_c reduction in cases of environmental stresses such as lack of soil water or soil salinity (Allen, 2003).

It is thus clear that to make a prediction of a crop's water requirements (ET_c), it is necessary to accurately estimate the reference evapotranspiration (ET_o), which is the evapotranspiration of a reference surface, defined as hypothetical grass with a uniform height of 0.12 m, a fixed surface resistance of 70 sm^{-1}, and an albedo (reflection coefficient) of 0.23 (Allen et al., 1998).

Historically several deterministic methods have been developed to estimate evapotranspiration using single or limited weather parameters and are generally categorized as: temperature, radiation or combination based. Temperature based methods include Thornthwaite (1948), Blaney and Criddle (1962) and Hargreaves and Samani (1982); Radiation methods include Priestley and Taylor (1972) and Makkink (1957); Combination methods include Penman (1948), modified Penman (Doorenbos, 1977) and FAO56-PM (Allen et al., 1998).

Shahidian et al. (2012) give an overview of several methods and compare their performance under different climate conditions. For most of the methods, the authors concluded that when applied to climates different from those on which they were developed and tested they can yield a poor performance and may require the adjustment of empirical coefficients to accommodate local climate conditions, which is not ideal.

The *Food and Agriculture Organization of the United Nations* (FAO) recommends

using the FAO-56 Penman-Monteith (FAO-56PM) formula as a reference method for estimating ET_o (Allen et al., 1998). To give a deeper idea of the involved parameters, the formula is given by

$$ET_o = \frac{0.408\Delta(R_n - G) + \gamma \frac{900}{T+273} u_2 (e_s - e_a)}{\Delta + \gamma(1 + 0.34u_2)}, \tag{2.1}$$

where R_n is the net radiation at crop surface $[MJm^{-2}day^{-1}]$, G is the soil heat flux density $[MJm^{-2}day^{-1}]$, T is the air temperature at 2 m height $[^oC]$, u_2 is the wind speed at 2 m height $[ms^{-1}]$, e_s is the saturation vapor pressure $[kPa]$, e_a is the actual vapor pressure $[kPa]$, $e_s - e_a$ is the saturation vapor pressure deficit $[kPa]$, Δ is the slope vapor pressure curve $[kPa^oC^{-1}]$, and γ is the psychrometric constant $[kPa^oC^{-1}]$. Being based on physical principles, the formula has become widely adopted as a standard for ET_o computation since it performs well under different climate types. However, to compute ET_o using FAO56-PM the following main meteorological parameters are required: temperature, solar radiation, relative humidity, and wind speed.

All parameters can be easily obtained from weather forecast APIs except for solar radiation. Solar radiation forecasting APIs are, at the moment, not common and present a high-cost penalty. So, apart from the water availability in the topsoil, being the evaporation from a cropped soil mainly determined by the fraction of the solar radiation reaching the soil surface (Allen et al., 1998), there is the need to (i) develop alternative methods for ET_o estimation using limited meteorological parameters, that do not require solar radiation and are compatible with the weather parameters obtained by freely available weather forecast and historical weather data APIs, or (ii) to estimate the solar radiation itself and use it as an approximation on the solar radiation dependent methods. This is also important since in most situations a proper functioning, maintained, and calibrated weather station, with solar radiation measurement capability is not close to the area of interest.

Recently, as an alternative, several authors have used machine and deep learning to estimate ET_o. For instance, Granata (2019) compared three different evapotranspiration

models, which differ in the input variables, using data collected in Central Florida, a humid subtropical climate. For each of this models four variants of machine learning algorithms were applied: M5P Regression Tree, Bagging, Random Forest, and Support Vector Machine (SVM). However, all three models included as input variable the net solar radiation. Wu and Fan (2019) evaluated eight machine learning algorithms divided in four classes: neuron based (MLP – Multilayer Perceptron, GRNN – General Regression Neural Network, and ANFIS – Adaptive Network-based Fuzzy Inference System), kernel-based (SVM, KNEA – Kernel-based Non Linear Extension of Arps decline model), tree-based (M5Tree – M5 model tree, Extreme Gradient Boosting – XGBoost), and curve based (MARS – Multivariate Adaptive Regression Spline). The methods were applied to data collected from 14 weather stations in various climatic regions of China and used only temperature or temperature and precipitation as input to the models. Daily ET_o estimates were satisfactory, but can be possibly improved by including further weather parameters and using different machine learning algorithms. Ferreira et al. (2019) used six alternative empirical reduced-set equations, such as Hargreaves and Samani (1982), and compared the estimated values with the ones from an Artificial Neural Network (ANN) and SVM model. Data was collected from 203 weather stations and used for daily ET_o estimation for the entirety of Brazil. Temperature or temperature and humidity was used as input features. They concluded that in general ANN was the best performing model when including as input features data from up to four previous days. Results were good considering that only temperature or temperature and humidity were used as inputs.

In this study, we explore and develop machine learning based ET_o prediction models supported on data from a weather station placed in the Algarve region, in south Portugal, as it will be described in Sec. 2.3.

2.3 Dataset and experimental setup

Data from February 2019 up to September 2021 was collected from a weather station that uses sensors from Davis Instruments, located in Vale do Lobo, in south Portugal. The following weather parameters were periodically measured throughout the day and stored with a daily resolution: temperature (minimum, maximum, and average), dew point (minimum, maximum, and average), relative humidity (minimum, maximum, and average), solar radiation (maximum and average), wind speed (minimum, maximum, and average), wind direction, atmospheric pressure (minimum, maximum, and average), rain intensity, and precipitation.

The time series was split using a ratio of 75 % for training and 25 % for testing, naturally without shuffling. This results in test data starting from February 4, 2021 onward. Furthermore, train data was divided into 10 folders using time series cross-validation (Hyndman and Athanasopoulos, 2021) and a grid search was used to tune the hyperparameters for each model that was used. As foreseeable, all presented model evaluation metric values are obtained using the test data that the models never saw while training. In this context, for model statistical evaluation and performance comparison the coefficient of determination (R^2), mean absolute error (MAE) and mean absolute percentage error (MAPE) were used. Considering y_t the actual value and $\hat{y}_t$ the estimated value at instants $t = 1, 2, ..., n$, and $\overline{y}$ the mean value of the actual samples, they are defined as

$$R^2 = 1 - \frac{\sum_{t=1}^{n}(y_t - \hat{y}_t)^2}{\sum_{t=1}^{n}(y_t - \overline{y})^2}, \tag{2.2}$$

$$MAE = \frac{1}{n}\sum_{t=1}^{n}|y_t - \hat{y}_t|, \tag{2.3}$$

and

$$MAPE = \frac{1}{n}\sum_{t=1}^{n}\left|\frac{y_t - \hat{y}_t}{y_t}\right| \times 100\%. \tag{2.4}$$

The FAO-56PM equation, see Eq. (2.1), was used to compute the target ET_o from the data collected from the weather station. When using solar radiation as target, the average solar radiation from the weather station was used.

During the conduction of this study the following widely known machine learning regression models were used: Ordinary Least Squares (OLS), Ridge Regression (Ridge), Lasso Regression (Lasso), k-Nearest Neighbors (kNN), Support Vector Machine (SVM), Decision Tree (Tree), and Random Forest (Forest) (Kishore Ayyadevara, 2018; Skiena, 2017). Table A.1 (Appendix A) summarizes the sets of hyperparameters used in the grid search procedure, being the final configurations presented in the corresponding sections.

Finally, to carry out the study, Python v3.8.2, Numpy v1.20.3 (Harris et al., 2020), Pandas v1.3.2 (McKinney, 2010; pandas development team, 2020), Scikit-learn v0.24.2 (Pedregosa et al., 2011), and PyET v1.0.1 (Vremec and Collenteur, 2021) were used. Pandas library was used for data analysis and manipulation, PyET to compute the reference evapotranspiration using the FAO56-PM method, and Sklearn is a widely used Python machine learning framework that includes regressors, data preprocessing, and model metrics evaluation tools.

2.4 Models for ET_o estimation: tuning and feature selection

This section divides in the following way. Firstly, in Sec. 2.4.1, a baseline method for ET_o estimation using the referred ML algorithms and having as input features all weather parameters that are provided by the weather station is presented. Then, and while still using the measured solar radiation, a first attempt is made at reducing the number of input features that are used, while maintaining similar model performance metrics. In Sec. 2.4.2 ML ET_o estimation models that do not use solar radiation as a feature are explored. In general, solar radiation either as a measurement or as a forecast is not available, hence the need to develop models that do not use it as an input feature.

Finally, since solar radiation seems to be one of the main ET_o drivers, in Sec. 2.4.3 ML solar radiation estimation models that use a reduced feature set are explored. Then, two different approaches are taken: (i) inject the solar radiation estimation into another ML model to predict ET_o or (ii) use FAO56-PM formula to compute ET_o having as input the estimated solar radiation.

2.4.1 An ET_o baseline using ML methods and measured solar radiation

To establish a baseline, the ML regression models were trained using all features available in the data collected from Vale do Lobo weather station, including the measured solar radiation. Furthermore, using the set of parameters described in Tab. A.1, the conducted grid search established the parameters values outlined in Tab. A.2 (Appendix A) as the best configurations. The set of features used are $Month \in \{1, 2, \ldots, 12\}$, $Day \in \{1, 2, \ldots, 31\}$, maximum, average and minimum temperature ($TempMax$, $TempAvg$, and $TempMin$), maximum, average and minimum humidity ($HumididtyMax$, $HumididtyAvg$ and $HumididtyMin$), maximum, average and minimum dewpoint ($DewpointMax$, $DewpointAvg$ and $DewpointMin$), maximum, average and minimum pressure ($PressureMax$, $PressureAvg$ and $PressureMin$), maximum and average wind speed ($WindMax$, $WindAvg$), wind gust ($WindGust$), rain intensity and precipitation ($RainIntensity$ and $Precipitation$), and maximum and average solar radiation ($SolarRadiationMax$, $SolarRadiationAVG$). Table 2.1 summarizes the attained metrics results. It can be seen that the best performing methods are OLS, Ridge, and Random Forest regressors with the best R^2 equal to 0.981, which corresponds to a MAE of 0.18 mm/day and a MAPE of 5.51 %.

Table 2.1: Comparison of several regression methods for ET_o estimation using all available features, including measured solar radiation.

	OLS	Ridge	Lasso	kNN	SVM	Tree	Forest
R^2	**0.981**	0.979	0.967	0.910	0.967	0.938	0.972
MAE (mm/day)	**0.18**	0.19	0.22	0.40	0.21	0.34	0.20
MAPE (%)	**5.51**	5.60	6.41	10.67	6.60	9.18	5.54

Table 2.2: Comparison of several regression methods for ET_o estimation using limited features, but including measured solar radiation.

	OLS	Ridge	Lasso	kNN	SVM	Tree	Forest
R^2	0.969	0.962	0.967	0.934	0.967	0.933	**0.971**
MAE (mm/day)	**0.21**	0.23	0.22	0.31	0.22	0.32	**0.21**
MAPE (%)	6.60	6.52	6.70	7.70	6.72	8.91	**5.89**

In a second phase, with the objective of reducing the feature set served as input to the algorithms (recall that, besides algorithms constraints, this reduction is important since many weather stations and weather APIs do not provide data for all relevant weather parameters), analyses of the Lasso coefficients and of the Random Forest feature importance, was conducted resulting in a new model with a reduced-set of features. As it can be seen on Tab. 2.2, it was observed that when using maximum and minimum temperature, average humidity, average wind, and average solar radiation as features, the models had similar performance to the previous results, and some even improved their metrics values. In this case, Random Forest gives the best R^2 score (0.971) being closely followed by OLS. This is an important result since, except for solar radiation, these features are easily obtained through weather forecast APIs such as OpenWeatherMap (OpenWeatherMap, 2022) or Instituto Português do Mar e da Atmosfera (IPMA, 2022).

2.4.2 ET_o estimation using ML methods with limited set of features (excluding solar radiation)

In a first attempt to use ML algorithms to directly estimate ET_o without using solar radiation as a feature, and using a tuning strategy similar to the one described in Sec. 2.4.1, it was found that $Month \in \{1, 2, \ldots, 12\}$ was an important feature. I.e., when comparing with the feature-set used to obtain the results in Tab. 2.2, the used features are similar except for adding *Month* and dropping the average solar radiation. Table 2.3 shows the results obtained with this set of features (month, maximum and minimum temperature, average humidity and average wind speed), and it can be clearly seen that Random Forest is the best performing model with an R^2 of 0.936, a MAE of

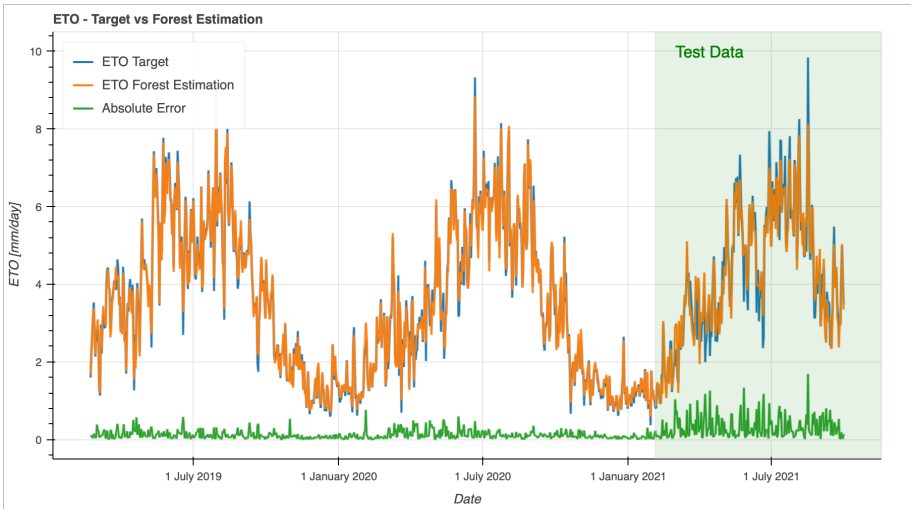

Figure 2.1: Target ET_o vs Random Forest estimation where solar radiation, actual or estimated, was not used as feature.

0.32 mm/day, and a MAPE of 9.11 %. Figure 2.1 sketches the ET_o target, the ET_o estimated using the Random Forest model, and the absolute error. The plot includes the predictions for the full dataset but, the shadowed region corresponds to the test data, the one used to compute the metrics, being visible the increase of the absolute error for those dates.

Table 2.3: Comparison of several regression methods for ET_o estimation using limited features.

	OLS	Ridge	Lasso	kNN	SVM	Tree	Forest
R^2	0.856	0.855	0.859	0.893	0.855	0.814	**0.936**
MAE (mm/day)	0.53	0.53	0.53	0.43	0.53	0.56	**0.32**
MAPE (%)	15.15	15.20	14.78	11.93	15.16	16.24	**9.11**

Maintaining the hyperparameters tuning strategy, attempts were made to improve the models' performance by doing some feature engineering, namely with new features constructed by: (i) computing the inverse of the features values (justified by the fact that some features appear in the denominator of reference FAO56-PM equation, Eq. (2.1)), (ii) polynomial features, and (iii) adding time lags. However, the success was minor and not noticeable to be presented here but, the idea was not abandoned as will be seen in the next sections.

2.4.3 ET_o estimation using approximated solar radiation values

In order to try to improve the limitation and results obtained in the previous sections, a different approach was tried. The idea was to use a reduced-set of features to previously estimate solar radiation and then either inject that solar radiation prediction into another ML regressor (with the same reduced-set features) or use FAO56-PM formula, Eq. (2.1), to simply approximate the ET_o values. Both approaches are presented next.

2.4.3.1 Estimating solar radiation using ML methods

In this section the solar radiation measured in the weather station was used as the target, i.e., the value to be estimated. Following the same tuning procedures as before (namely, the analyses of Lasso coefficients and Random Forest feature importance), the conclusion was that the best configuration for solar radiation estimation was attained for the Random Forest method with the following features: month, day, maximum and minimum temperature, average humidity, average wind speed, and average dew point. More precisely, the results presented in Tab. 2.4 show that the Random Forest model is the one with more satisfactory performance, with an R^2 of 0.814, a MAE of 21.31 $W/m^2/day$, and a MAPE of 11.29 %.

Again, further attempts were made to improve the models' performance by doing feature engineering such as polynomial features, inverse of features, and adding time lags. Of these, only polynomial features were helpful in improving models' performance. After individually analyzing the features' relevance for the models, it was found that by adding the following reduced-set polynomial feature $Month^2 \times Day$, the performance metrics were improved for all models, except Ridge and Lasso. As it occurs many times in ML (being considered out of the scope of this thesis) the justification for such is not obvious. Detailed in Tab. 2.5, Random Forest is still the best performing model, now with an R^2 of 0.822, a MAE of 20.63 $W/m^2/day$, and a MAPE of 10.99 %. Figure 2.2 plots the target solar radiation, the approximated solar radiation obtained

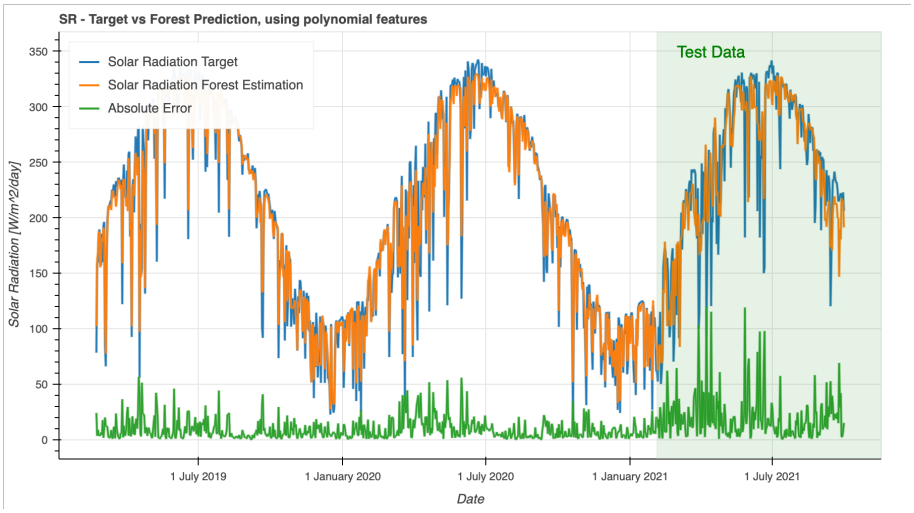

Figure 2.2: Target solar radiation vs Random Forest estimation with polynomial features.

with the Random Forest method, and absolute error curves (shadowed is the test set). This solar radiation estimation will be used next to predict the ET_o values.

2.4.3.2 ET_o estimation using ML and the approximated solar radiation

The predicted values from the best performing solar radiation estimation model in the previous section, which was the Random Forest model with polynomial restricted features (see Tab. 2.5), were injected as a feature together with maximum temperature, average humidity and average wind into the early studied methods to estimate the ET_o, being the obtained results summarized in Tab. 2.6. It can be seen that the Random Forest is the best performing model, with an R^2 of 0.951, a MAE of 0.26 mm/day and a MAPE of 7.44 %. This result is close to the before presented ET_o baseline that used ML and limited features, but included the measured solar radiation (see Tab. 2.2). As a reference, in that case, the MAPE was equal to 5.89 %. As before, some feature engineering was tested but brought no further improvement. Figure 2.3 (top) plots the

Table 2.4: Comparison of several regression methods for average solar radiation estimation using limited features.

	OLS	Ridge	Lasso	kNN	SVM	Tree	Forest
R^2	0.532	0.505	0.382	0.580	0.312	0.594	**0.814**
MAE ($W/m^2/day$)	39.05	40.48	45.59	36.30	48.09	30.67	**21.31**
MAPE (%)	19.61	20.55	22.32	19.33	23.26	16.34	**11.29**

Table 2.5: Comparison of several regression methods for average solar radiation estimation using polynomial features.

	OLS	Ridge	Lasso	kNN	SVM	Tree	Forest
R^2	0.553	0.313	0.375	0.590	0.400	0.605	**0.822**
MAE ($W/m^2/day$)	38.04	49.08	46.53	32.48	43.87	30.36	**20.63**
MAPE (%)	18.99	24.74	23.4	16.69	22.21	16.33	**10.99**

target ET_o, the estimated ET_o, and corresponding error curves for the train and test (shadowed) dataset.

2.4.3.3 ET_o estimation using FAO56-PM equation and the approximated solar radiation

To finalize our study, an hybrid approach was tested. In this case, the predicted solar radiation is used with the FAO56-PM equation to estimate target ET_o, being the results shown on Tab. 2.7. With an R^2 of 0.975, MAE of 0.18 mm/day and MAPE of 5.51 % over the unseen test data, this result is better than any of the previously obtained ones, even better than the ML reduced-set baseline (see Sec. 2.4.1) that used the weather station measured solar radiation as a feature. Figure 2.3 (bottom) plots ET_o target, estimated ET_o, and error curves, being evident the improvement in the error when compared with the top plot.

In short, Tab. 2.8 presents an overview of the best ET_o estimators that where previ-

Table 2.6: Comparison of several regression methods for ET_o estimation using limited features and the previously estimated solar radiation.

	OLS	Ridge	Lasso	kNN	SVM	Tree	Forest
R^2	0.944	0.944	0.893	0.934	0.937	0.921	**0.951**
MAE (mm/day)	0.30	0.30	0.38	0.32	0.31	0.37	**0.26**
MAPE (%)	9.36	8.99	9.64	8.58	9.60	10.54	**7.44**

Table 2.7: Result obtained when computing ET_o using FAO56-PM equation and using as solar radiation the previously calculated prediction from the best performing Random Forest model.

	FAO56-PM + SR_pred
R^2	0.975
MAE (mm/day)	0.18
MAPE (%)	5.51

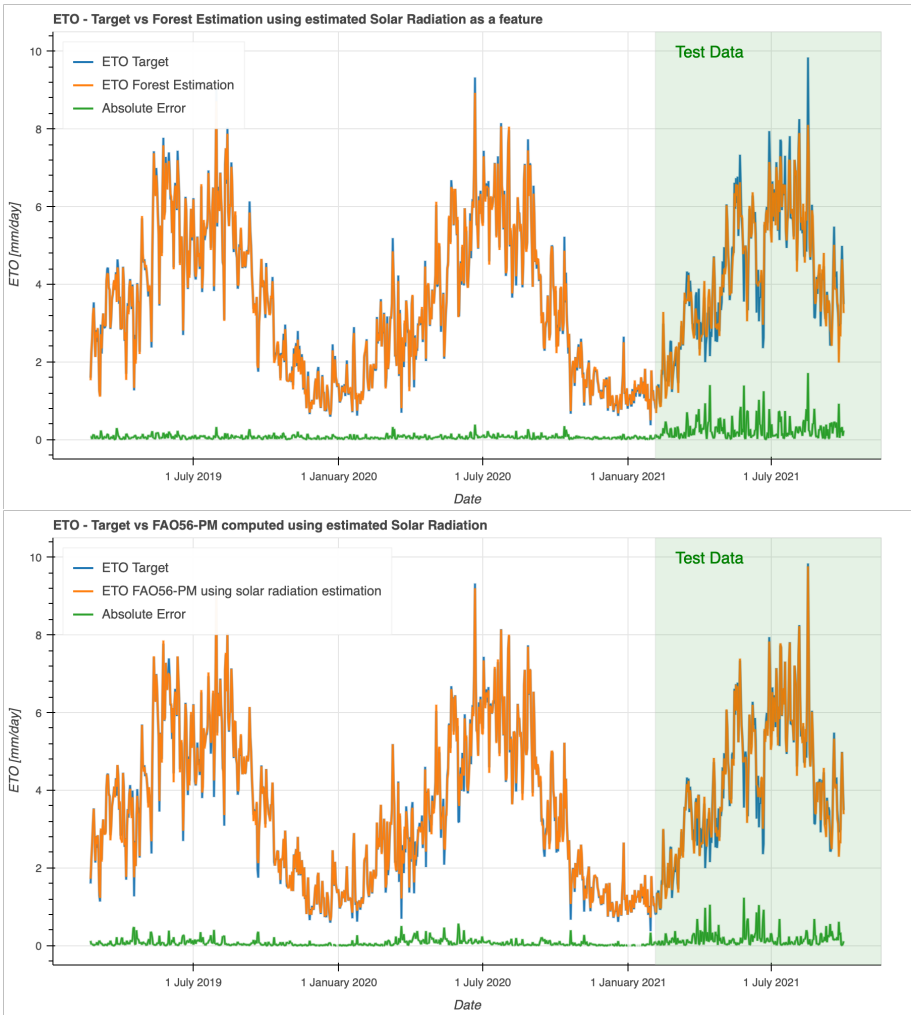

Figure 2.3: Target ET_o vs Random Forest estimation (top) and FAO56-PM equation (bottom) using estimated solar radiation.

ously presented. Comparing the first two columns it can be concluded that when the measured solar radiation is available, the use of a reduced-set has low impact on model performance. Further, the use of previously estimated solar radiation (last two columns) improves results when solar radiation measurement is not available. The hybrid method (last column) gives similar results to those of the ML baseline when using all weather parameters provided by the weather station, and gives better performance than the reduced-set ML baseline that used the actual measured solar radiation.

Table 2.8: Overview of the best ET_o estimators for each method that was presented.

	Measured solar radiation		No solar radiation	Estimated solar radiation	
	Table 2.1	Table 2.2	Table 2.3	Table 2.6	Table 2.7
R^2	**0.981**	0.971	0.936	0.951	0.975
MAE (mm/day)	**0.18**	0.21	0.32	0.26	**0.18**
MAPE (%)	**5.51**	5.89	9.11	7.44	**5.51**
Best Estimator	OLS	Random Forest	Random Forest	Random Forest	Random Forest

2.5 Conclusion and forthcoming work

In this chapter, several ML models and an hybrid approach for the ET_o estimation were tested with different degrees of success. Since solar radiation is the main ET_o driver, as stated by several authors and also concluded by us, models were also developed for estimating solar radiation using features usually available in the common weather forecast APIs. This allowed both the injection of the previously estimated solar radiation in ML regressors to estimate ET_o, but also the possibility to use the hybrid approach where solar radiation is previously estimated and then FAO56-PM algorithm is used to finally compute ET_o. The latter yielded the best results, with an R^2 of 0.975, a MAE of 0.18 mm/day and an MAPE of 5.51 %, which when compared with other authors works, is a good result considering the limited weather parameter features that were used.

Forthcoming work will include the use of other prediction methods (such as, recurrent neural network models) and a more extensive dataset, by using the existing weather station infrastructure that is installed in the Algarve region, in south Portugal. The objective will be to develop local and pooled models of ET_o predictors for the Algarve region. Also, since all limited feature models here presented are compatible with freely available weather forecast APIs a study needs to be made to assess the impact of using such APIs as input data to the ML models here developed.

— Our brains have been designed to blur the line between self and other. It is an ancient neural circuitry that marks every mammal, from mouse to elephant.

Frans de Waal

3

Applications of neural networks for evapotranspiration prediction over limited weather parameters

Summary

Evapotranspiration can be used to estimate the amount of water required by agriculture projects and green spaces, playing a key role in water management policies that combat the hydrological drought, which assumes a structural character in many countries. This work presents a study on reference evapotranspiration (ET_o) estimation

models, having as input limited meteorological parameters, namely: temperature, humidity, and wind. Since solar radiation (SR) is an important parameter in the determination of ET_o, SR estimation models are also developed. ET_o and SR estimation models compare the use of Artificial Neural Networks (ANN), Long Short Term Memory (LSTM), Gated Recurrent Unit (GRU), Recurrent Neural Network (RNN), and hybrid neural network models such as LSTM-ANN, RNN-ANN and GRU-ANN. Two main approaches were taken for ET_o estimation: (i) directly use those algorithms to estimate ET_o, and (ii) estimate solar radiation first and then use that estimation together with other meteorological parameters in a method that predicts ET_o. For the later case, two variants were implemented: the use of the estimated solar radiation as (ii.1) a feature of the neural network regressors, and (ii.2) the use of FAO-56PM method to compute ET_o, which has solar radiation as one of the input parameters. Using experimental data collected from a weather station (WS) located in Vale do Lobo, south Portugal, the later approach achieved the best result with a coefficient of determination (R^2) of 0.977. As a final notice, the reduced-set features were carefully selected so that they are compatible with free online weather forecast services.

3.1 Introduction

A known strategy, to make a water efficient irrigation system, is to rely on soil humidity sensors and keep the humidity levels between the field capacity (FC) and the management allowable depletion (MAD), which is a percentage of the soil available water holding capacity (Zhang et al., 2021). Apart from components, installation, and management costs associated to the implementation of such systems on public green space, other problems are common such as vandalism or theft, since normally there is no comprehensive security or surveillance in such locations. Further, the available water holding capacity changes significantly with soil type (JONG and Shields, 1988), requiring that for each specific soil, in order to determine the soil humidity values corresponding to the field capacity and wilting point (WP), samples would need to be sent

to a laboratory for analyses.

Crop evapotranspiration (ET_c), also known as the crop water use, is the water that is used by a crop (Al-Kaisi and Broner, 2009). The *Food and Agriculture Organization of the United Nations* (FAO) recommends using the FAO-56 Penman-Monteith (FAO-56PM) formula as a reference method for computing reference evapotranspiration (ET_o) (Allen et al., 1998), being ET_c and ET_o related by a crop coefficient (K_c). FAO56-PM formula uses four main meteorological parameters: temperature, humidity, wind, and solar radiation (SR). Several computational studies show that SR is the main driver of ET_o, however, its measurement requires sensors like pyranometers, which are typically associated with expensive WSs that need to be properly maintained and calibrated (Shahidian et al., 2012). Also, solar radiation forecast application programming interfaces (APIs) are not common (at least freely) and present a high system cost penalty.

As part of a framework for the computation of optimal crop water irrigation scheduling requirements (with special emphasis to green spaces), this chapter presents the computational models being prepared to estimate the ET_o using machine learning, deep learning, acquired intelligence, meteorological data from WSs on the field, as well as meteorological data and forecasts from APIs available on the internet. This will optimize water and energy expenditure, improve the well-being of the crop, reduce reaction time in solving problems, improve anomalies detection methods, and maintain the quality of green spaces. In short, the framework being developed will be an intelligent irrigation solution, technologically differentiated from other platforms on the market. The development of the full framework is being done under project GSSIC – Green Spaces SMART Irrigation Control. In this context, this study compares several ET_o and SR estimation models that use Artificial Neural Networks (ANN), Long Short Term Memory (LSTM), Gated Recurrent Unit (GRU), Recurrent Neural Network (RNN), and hybrid neural network models such as LSTM-ANN, RNN-ANN and GRU-ANN. Two approaches were taken for ET_o's estimation: (i) directly use those machine learning models to estimate it and (ii) firstly estimate solar radiation, and then use the

obtained value and other meteorological parameters in a method that predicts ET_o. Furthermore, for the later case, two variants were addressed, namely: the use of the estimated SR as (ii.1) a feature of a neural network regressor, and (ii.2) the use of FAO-56PM method to compute ET_o, which has SR as one of the input parameters. Using experimental data collected from a WS located in Vale do Lobo, south Portugal, the later approach achieved the best result with a coefficient of determination (R^2) of 0.977, improving results previously achieved.

The chapter is structured as follows. The next section presents the problem's background and the methodologies used by others to tackle the problem in study. Section 3.3.1 starts by detailing the computational setup and exploring the dataset, followed by Sec. 3.3.2 where the proposed neural network architectures, hyperparameters, and overall training approach is presented. Then, on Sec. 3.3.3 the proposed models and associated performance analysis is weighed. Finally, the last section presents the conclusion and establishes some future work.

3.2 Reference evapotranspiration problematic and known solutions

Reference evapotranspiration, ET_o, is the evapotranspiration of a reference surface, defined as hypothetical grass with a uniform height of 0.12 m, a fixed surface resistance of 70 sm^{-1}, and an albedo (reflection coefficient) of 0.23 (Allen et al., 1998). On other side, crop evapotranspiration, ET_c, represents the crop's water requirements and is proportional to reference evapotranspiration by means of the crop coefficient, K_c (Allen et al., 1998). Therefore, ET_o prediction plays an important role, making it one of the fundamental parameters for smart irrigation scheduling, since it is proportional to the amount of water that needs to be restored during the irrigation period (Ferreira et al., 2019).

Some of the main characteristics that distinguish ET_c from ET_o are (i) the crop cover

density and total leaf area, (ii) the resistance of foliage epidermis and soil surface to the flow of water vapor, (iii) the aerodynamic roughness of the crop canopy, and (iv) the reflectance of the crop and soil surface to short wave radiation (Allen, 2003). In this context, known the value of K_c, the ET_c value for a specific time period can be estimated by

$$ET_c = K_c ET_o. \tag{3.1}$$

The crop coefficient can be simple or have two components, one representing the basal crop coefficient (K_{cb}) and another representing the soil surface evaporation component (K_e), being computed by

$$K_c = K_s K_{cb} + K_e, \tag{3.2}$$

where $K_s \in [0, 1]$ is used to introduce a K_c reduction in cases of environmental stresses, such as lack of soil water or soil salinity (Allen, 2003).

So, it becomes clear that in order to make crop water requirement predictions, accurate estimation of ET_o is required. Historically, several deterministic methods have been developed to estimate reference evapotranspiration using single or limited weather parameters and being generally categorized as: temperature, radiation, or combination based. For example, temperature based methods include Thornthwaite (1948), Blaney and Criddle (1962), and Hargreaves and Samani (1982) formulas; radiation methods include Priestley and Taylor (1972) and Makkink (1957) formulas; and combination methods, requiring both temperature and radiation, include Penman (1948), modified Penman (Doorenbos, 1977), and FAO-56 Penman-Monteith (FAO56-PM)(Allen et al., 1998) formulas. Shahidian et al. (2012) give an overview of several these methods and compare their performance under different climate conditions. The authors concluded that, when applied to climates different from those on which the methods were developed and tested, most of them yield a poor performance and may require the adjustment of empirical coefficients to accommodate to local climate conditions, which is not ideal.

FAO recommends the use of the FAO-56PM formula as a reference method for estimat-

ing ET_o (Allen et al., 1998). To give a deeper idea of the involved parameters, measured in millimeters per day (mm/day), the formula to estimate ET_o is given by

$$ET_o = \frac{0.408\Delta(R_n - G) + \gamma \frac{900}{T+273} u_2 (e_s - e_a)}{\Delta + \gamma(1 + 0.34u_2)}, \tag{3.3}$$

where R_n is the net radiation at crop surface $[MJm^{-2}day^{-1}]$, G is the soil heat flux density $[MJm^{-2}day^{-1}]$, T is the air temperature at 2 m height $[^oC]$, u_2 is the wind speed at 2 m height $[ms^{-1}]$, e_s is the saturation vapor pressure $[kPa]$, e_a is the actual vapor pressure $[kPa]$, $e_s - e_a$ is the saturation vapor pressure deficit $[kPa]$, Δ is the slope vapor pressure curve $[kPa^oC^{-1}]$, and γ is the psychrometric constant $[kPa^oC^{-1}]$. Being based on physical principles, the formula has become widely adopted as a standard for ET_o computation since it performs well under different climate types (Shahidian et al., 2012). However, to compute ET_o using FAO56-PM the following main meteorological parameters are required: temperature, solar radiation, relative humidity, and wind speed. The remaining meteorological parameters are constants being derived from latitude and elevation above sea level.

Except for solar radiation, all parameters (real or estimated) required by the FAO56-PM formula for ET_o's computation, can be freely obtained from common weather forecast APIs. Solar radiation forecasting APIs are, at the moment, not common and the ones available present a high-cost penalty. Therefore, a major asset would be to (i) develop alternative methods for ET_o estimation using limited meteorological parameters, that do not require solar radiation and are compatible with the weather parameters obtained by freely available weather forecast and historical weather data APIs, and/or (ii) to estimate the solar radiation itself and use it as an approximation on the solar radiation dependent methods. This is also important since in most situations a proper functioning, maintained, and calibrated WS, with solar radiation measurement capability, is not close to the area of interest.

Recently, as an alternative, several authors have used machine and deep learning to estimate ET_o. For instance, using data collected in Central Florida, a humid subtropical

climate, Granata (2019) compared three different evapotranspiration models which differ in the input variables. In their work, four variants of machine learning algorithms were applied to each model, namely: M5P Regression Tree, Bagging, Random Forest, and Support Vector Machine (SVM). Their best results are achieved using M5P Regression Tree and Bagging with a coefficient of determination of 0.987 and a mean absolute error of 0.14 mm/day (see the metrics definition in Sec. 3.3.1). However, among other features, all models included as input variable the net solar radiation. Wu and Fan (2019) evaluated eight machine learning algorithms divided in four classes: neuron based (MLP – Multilayer Perceptron, GRNN – General Regression Neural Network, and ANFIS – Adaptive Network-based Fuzzy Inference System), kernel-based (SVM, KNEA – Kernel-based Non Linear Extension of Arps decline model), tree-based (M5Tree – M5 model tree, Extreme Gradient Boosting – XGBoost), and curve based (MARS – Multivariate Adaptive Regression Spline). The methods were applied to data collected from 14 WSs in various climatic regions of China and used only temperature or temperature and precipitation as input to the models. Ferreira et al. (2019) used six alternative empirical reduced-set equations, such as Hargreaves and Samani (1982), and compared the estimated values with the ones from an Artificial Neural Network and a SVM model. Data was collected from 203 WSs and used for daily ET_o estimation for the entirety of Brazil. Temperature or temperature and humidity were used as input features. They concluded that, in general, ANN was the best performing model when including, as input features, data from up to four previous days. With the best algorithms reaching an R^2 median value around 0.80 considering all stations, results were weighed good given that only temperature or temperature and humidity were used as input. Vaz et al. (2022b) used data from a Vale do Lobo WS, in south Portugal, and explored the use of machine learning for ET_o estimation. They concluded that instead of directly estimating ET_o, the best result was obtained by using machine learning for SR estimation, having as input a limited set of meteorological features, and then use that result together with temperature, humidity and wind speed as input to the FAO56-PM equation, achieving an R^2 of 0.975, a MAE of 0.18 mm/day, and a MAPE of 5.51 %. This

work explores and develops deep learning based ET_o prediction models, supported on the same data, improving the previous results, as it will be detailed in the next section.

3.3 Proposed models

3.3.1 Experimental setup and dataset

As a general introduction to the computational environment, this work was conducted using Python v3.9.7, Numpy v1.21.4 (Harris et al., 2020), Pandas v1.3.4 (McKinney, 2010; pandas development team, 2020), Tensor Flow v2.6.0 (Abadi et al., 2015), Keras 2.6.0 (Chollet, 2022), Scikit-learn v1.0.1 (Pedregosa et al., 2011), and PyET v1.1.0 (Vremec and Collenteur, 2021). The Pandas library was used for data analysis and manipulation, PyET to compute the reference evapotranspiration using the FAO56-PM method, Scikit-learn is a Python machine learning framework that includes data preprocessing, model selection and model metrics evaluation tools. Keras runs on top of Tensor Flow, and all neural network models here presented were developed using it. Finally, all computation was done on a 2020 MacBook Air with an Apple M1 SoC chip and 16GB of RAM, running MacOS Big Sur v11.6.4.

Data from Vale do Lobo WS, in south Portugal, was collected starting from February 2019 up to and including September 2021. The WS is composed of sensors from Davis Instruments, where the following weather parameters are measured periodically throughout the day and stored with a daily resolution: temperature (minimum, maximum, and average), dew point (minimum, maximum, and average), relative humidity (minimum, maximum, and average), solar radiation (maximum and average), wind speed (minimum, maximum, and average), wind direction, atmospheric pressure (minimum, maximum, and average), rain intensity, and precipitation. This is the same dataset previously used by Vaz et al. (2022b), where the use of machine learning algorithms was explored to create ET_o and SR estimation models. A ratio of 75 % to 25 % of train and test data was used, respectively, resulting in train data starting from

February 1st, 2019 up to February 3rd, 2021, and test data from February 4th, 2021 up to September 30th, 2021. Being a time series, no shuffling was made to the train and test data, and the train data was further divided into 10 folders, used to implement time series cross validation (Hyndman and Athanasopoulos, 2021). Furthermore, a hyper-parameter grid search strategy was used to tune the proposed machine/deep learning methods, as will be presented in the corresponding sections.

For model statistical evaluation and performance comparison the coefficient of determination (R^2), mean absolute error (MAE), and mean absolute percentage error (MAPE) were used. Just to recall, considering y_t the actual value and $\hat{y}_t$ the estimated value at instants $t = 1, 2, ..., n$, and $\overline{y}$ the mean value of the actual samples, the evaluation metrics are defined as

$$R^2 = 1 - \frac{\sum_{t=1}^{n}(y_t - \hat{y}_t)^2}{\sum_{t=1}^{n}(y_t - \overline{y})^2}, \tag{3.4}$$

$$MAE = \frac{1}{n}\sum_{t=1}^{n}|y_t - \hat{y}_t|, \tag{3.5}$$

and

$$MAPE = \frac{1}{n}\sum_{t=1}^{n}\left|\frac{y_t - \hat{y}_t}{y_t}\right| \times 100\%. \tag{3.6}$$

Depending on the regression problem in study (ET_o vs. SR), two targets were considered: (i) ET_o was computed using the FAO56-PM formula, as per Eq. (3.3), and using as input the data measured by the already referred WS, namely temperature, humidity, wind speed and solar radiation, and (ii) the average solar radiation measured by the WS was used as SR target.

3.3.2 Neural network models' architecture

The following neural network types were used for the conduction of this work: ANN, LSTM (Hochreiter and Schmidhuber, 1997), GRU (Cho et al., 2014), and Artificial RNN (Rumelhart et al., 1985), as well as hybrid models like LSTM-ANN, RNN-ANN, and GRU-ANN.

Artificial neural networks typically have an input layer, one or more hidden layers, and one output layer. As depicted in Fig. 3.1, the input layer dimension is defined by the number of inputs to the model, i.e., the number of features that are used on a particular model. Hidden layers are internal layers that can vary in quantity and number of neurons. The output layer can have one or more neurons, depending on the number of outputs a model has. Artificial neurons are activated using activation functions, which have an important role on the performance of neural networks, since they provide the non-linearity that is needed to learn a complex problem (Rasamoelina et al., 2020). During the conduction of this work Rectified Linear Unit (ReLU), Tangent Hyperbolic Function (Tanh) and Sigmoid activation functions were tested, however ReLU always yielded better results, being therefore fixed (Pomerat et al., 2019).

LSTM, GRU and RNN algorithms are deep learning methods, that unlike feed forward neural networks, implement feedback connections. This feedback connections provide the ability to add memory to the models, making its use justified through the fact that the dataset (and target) is a time series, where some patterns might be cyclic or information from the previous days might play an important role (Hochreiter and Schmidhuber, 1997; Cho et al., 2014; Rumelhart et al., 1985).

Furthermore, in the work presented by Vaz et al. (2022b) (see also Chap. 2), for the same problem and dataset in study here (using limited weather parameters as input features), several machine learning regression models were compared in their performance, namely: Ordinary Least Squares, Ridge, Lasso, *k*-Nearest Neighbors, Support Vector Machine, Decision Tree, and Random Forest (RF). Since Random For-

est (Kishore Ayyadevara, 2018; Skiena, 2017) gave the best results for ET_o and SR estimation models, RF will also be evaluated, so that the proposed neural network based models can be directly compared with other top performing methods.

Systematic testing (varying the number of neurons in powers of 2) allowed us to set the non-hybrid models to have two hidden layers: the first one consisting of 512 neurons, followed by a second layer that has 32 neurons, represented as having [512, 32] neurons. This configuration consistently gave the best results, being observed that augmenting the number of layers and/or neurons of the neural network would not increase model performance, while reducing would impact on model's performance. Following the same arrangement, tested configuration allowed us to decide the hybrid models' architecture (namely, LSTM-ANN, RNN-ANN, and GRU-ANN). In this case, balancing the training computational requirements and models' performance, it was decided to use four hidden layers. E.g, the LSTM-ANN model has the first two layers consisting of [32, 64] neurons of LSTM type, followed by two more layers of [64, 32] neurons of a fully connected ANN. The RNN-ANN and GRU-ANN models are similar, being all represented as having [32, 64; 64, 32] neurons.

Solver, loss function and hyperparameters like learning rate, number of epochs, and batch size play an important role on model training and performance (Smith, 2018). For all models, the Adam optimizer (Kingma and Ba, 2014) was selected as a solver, mean squared error (MSE) was used as the loss function, kernel initialization was done using the Glorot normal initialization (Glorot and Bengio, 2010), and the number of epochs was set to a high value of 1000, however, early stopping was used, with a patience value set to 150. Dropout regularization (Srivastava et al., 2014) was applied during training and its value was selected using grid search with values ranging from 0.0 up to 0.6, in 0.1 steps. During initial model development and training, the batch size was also selected using a grid search approach, with a range from 64 up to 256, in multiples of 32. However, a batch size of 128 always gave a good and consistent result, so later it was fixed to 128 for all models, reducing models training time. Input features were normalized to values between 0 and 1. Finally, max-norm weight constraint (Sri-

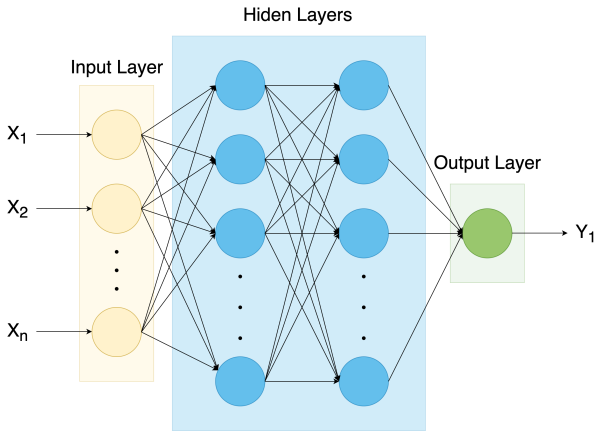

Figure 3.1: Neural network typical architecture.

vastava et al., 2014; Garbin et al., 2020) was applied, and its value was grid searched between 1 and 4, in steps of one.

It should also be noticed that time lag was introduced on all models, i.e., data from the previous days was introduced as an input feature together with current day data. In this case, time lags from 1 up to 10 days were tested, however it brought no improvement on models' performance. In this experimental context, attempts of introducing new features through the use of polynomial features and features inverse were also made, but it did also not present any further improvement in models metrics, as such it is not presented here.

In appendix, Tables B.1 and B.2 summarize the tested and tuned hyperparameters, respectively, for each of the proposed models.

3.3.3 ET_o estimation using ANN methods with a limited set of features

Solar radiation is not normally available either as a measurement or as a forecast, therefore the need to develop models to estimate ET_o that do not require it as an input feature. In this context, ET_o estimation models that have as input a limited set of parameters are explored in Sec. 3.3.3.1. Furthermore, since solar radiation is the main factor for the determination of ET_o (Allen et al., 1998), in Sec. 3.3.3.2 solar radiation estimation models that use a limited set of features are explored. Finally, two approaches

are taken: (i) use the previously estimated solar radiation as an input to another neural network model (Sec. 3.3.3.3) or (ii) use the FAO56-PM formula to compute ET_o, using as an input feature the estimated solar radiation (Sec. 3.3.3.4).

3.3.3.1 ET_o estimation using ANN methods (excluding solar radiation)

In this section neural network based models are used to directly estimate ET_o, using as inputs limited weather parameters. The set of features used are $Month \in \{1, 2, \ldots, 12\}$, maximum and minimum temperature ($TempMax$ and $TempMin$), average humidity ($HumididtyAvg$), and average wind speed ($WindAvg$).

Table 3.1 shows the results obtained by the non-hybrid models and RF (included for comparison purposes), being clear that neural network based models outperform, in all metrics, the RF model. ANN, LSTM, GRU and RNN give similar results, the best one being the GRU based model with an R^2 of 0.962, a MAE of 0.24 mm/day, and a MAPE of 7.25 %. The plot of the target ET_o (blue), of the ET_o estimated using the ANN model (orange), and of the corresponding absolute error (green) in Fig. 3.2 shows that the estimator follows relatively well the ET_o target value. In the same figure, the shadowed region corresponds to the test data, being visible the expectable slight increase of the absolute error, when comparing to the train region (as stated earlier, all presented metrics are calculated using only the test data). Table 3.2 also presents the results obtained using the hybrid models, namely, LSTM-ANN, RNN-ANN, GRU-ANN. The results are slightly worse than the ones obtained using the GRU, with the LSTM-ANN model attaining an R^2 of 0.959 against the 0.962 of GRU (MAE and MAPE are also very similar).

Table 3.1: Comparison of several regression methods for ET_o estimation using a limited set of features.

	ANN	LSTM	GRU	RNN	RF	LSTM-ANN	RNN-ANN	GRU-ANN
R^2	0.959	0.958	**0.962**	0.960	0.936	**0.959**	0.953	0.955
MAE (mm/day)	0.25	0.25	**0.24**	0.25	0.32	**0.25**	0.27	**0.25**
MAPE (%)	7.54	7.40	**7.25**	7.33	9.11	**7.45**	7.97	7.81

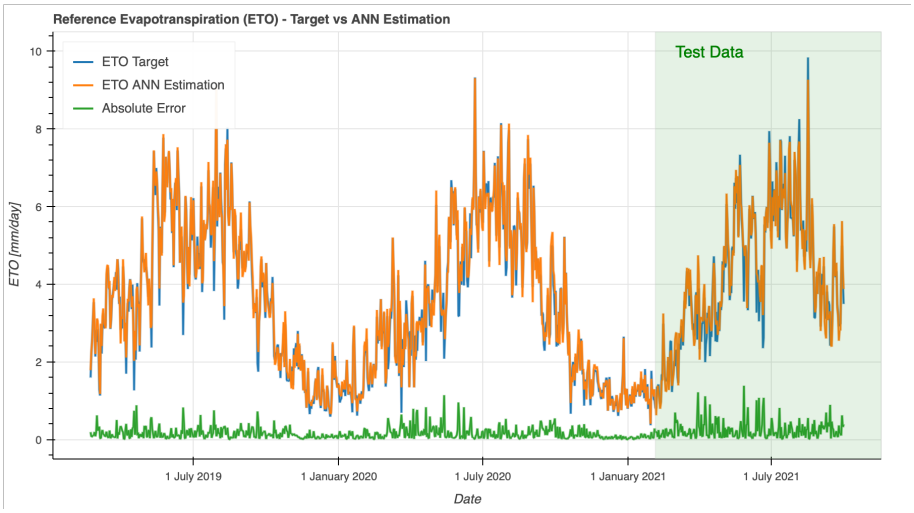

Figure 3.2: Target ET_o vs. ANN estimation, where solar radiation was not used as feature.

3.3.3.2 Solar radiation estimation using ANN methods with a limited set of features

As an alternative to the presented in the previous section, and since solar radiation is the main driver of evapotranspiration (Allen et al., 1998), this section studies solar radiation estimation models that have as input a limited feature set, compatible with the limited number of features returned by the common weather forecast APIs. The estimated result will be later injected as a feature in other machine learning model or used in FAO56-PM formula (Eq. 3.3).

So, in this section, the solar radiation measured by the WS was used as target. Initially, the features used in the solar radiation models were the same as the ones proposed by Vaz et al. (2022b), namely: $Month$, Day, maximum and minimum temperature ($TempMax$ and $TempMin$), average humidity ($HumididtyAvg$), average wind speed ($WindAvg$), dewpoint, and the polynomial feature $Month^2 \times Day$. However, it was found that the model metrics were improved by dropping the dewpoint and the polynomial feature $Month^2 \times Day$ and, instead, adding the sunset hour angle (ω_s) and daylight hours (N). As a note, the sunset hour angle (ω_s) was defined as (Allen et al., 1998)

Table 3.2: Comparison of several regression methods for average solar radiation estimation using a limited set of features.

	ANN	LSTM	GRU	RNN	RF	LSTM-ANN	RNN-ANN	GRU-ANN
R^2	0.831	0.827	0.825	**0.833**	0.808	**0.825**	0.807	0.819
MAE ($W/m^2/day$)	18.55	20.55	19.13	**18.46**	21.51	**18.69**	20.70	19.44
MAPE (%)	**10.22**	11.06	10.75	10.30	12.27	**10.37**	11.72	11.10

$$\omega_s = \arccos\left[-\tan(\varphi) \times \tan(\delta)\right], \tag{3.7}$$

where φ is the latitude of the WS (in radians) and

$$\delta = 0.409 \sin\left(\frac{2\pi}{365} J - 1.39\right) \tag{3.8}$$

is the solar declination (also in radians), and J is the number of the day in the year. Furthermore, the daylight hours (N) is given by

$$N = \frac{24}{\pi}\omega_s. \tag{3.9}$$

The results obtained are summarized in Table 3.2, where it can be seen that RNN and ANN are the best performing methods, RNN being slightly better with an R^2 of 0.833, a MAE of 18.46 $W/m^2/day$, and a MAPE of 10.30 %. This result is better than what was obtained by Vaz et al. (2022b), using the same dataset, on which Random Forest gave the best results, with an R^2 of 0.814, a MAE of 21.31 $W/m^2/day$, and a MAPE of 11.29 %. The inclusion of daylight hours and sunset angle improved performance for all neural network based models but, as can be seen on Table 3.2, it worsened Random Forest performance.

Figure 3.3 depicts the target solar radiation that was measured by the WS (blue), the approximated solar radiation obtained with ANN method (orange), and the absolute error curve (green). Shadowed is the test set. This solar radiation estimation will be used next to predict the ET_o values.

As in Sec. 3.3.3.1, the recursive hybrid LSTM-ANN, RNN-ANN and GRU-ANN mod-

els' results, summarized also in Tab. 3.2, give similar performance to the recursive non-hybrid methods, with the advantage of requiring less computation power, since the number of trainable parameters is highly reduced due to the smaller network that is used. E.g., the LSTM model has 1,128,609 trainable parameters, while the LSTM-ANN model only has 35,841, requiring approximately 21 % of the training time, and 58 % of the inference time.

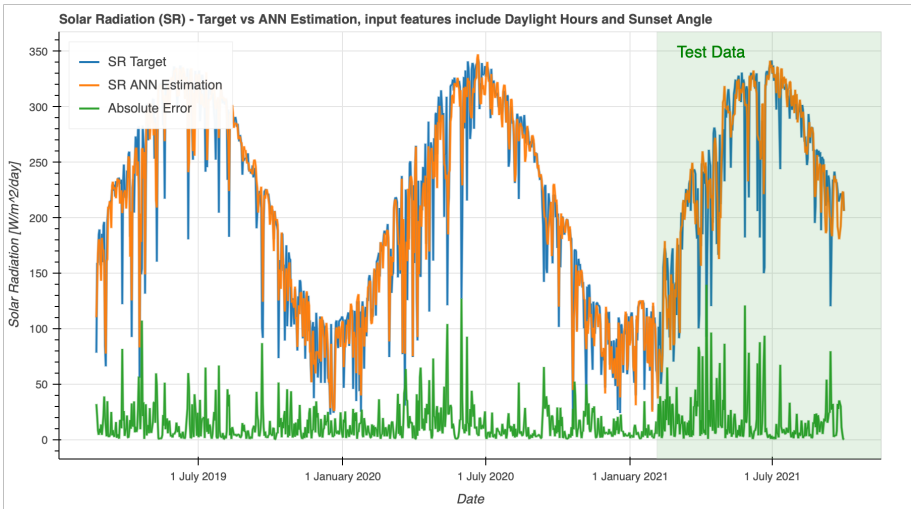

Figure 3.3: Target solar radiation vs ANN estimation, input features include daylight hours and sunset angle as derived features.

3.3.3.3 ET_o estimation using the approximated solar radiation

This section describes the implementation of the ET_o estimation using as features solar radiation (approximated using the ANN presented in Sec. 3.3.3.2) together with maximum temperature, average humidity, and average wind speed. To be more clear, the difference between the present models and the ones in Sec. 3.3.3.1 is the use of the (estimated) SR, which was not previously used.

The obtained results are summarized in Tab. 3.3. The ANN model was the best one with an R^2 of 0.968, a MAE of 0.22 mm/day and a MAPE of 6.66 % (against the R^2 of 0.962, a MAE of 0.24 mm/day, and a MAPE of 7.25% previously obtained with GRU model). Very similar, with the advantages previously stated, the LSTM-ANN model achieved a R^2 of 0.967, just one thousandth worse than the ANN model, but improving

Table 3.3: Comparison of several regression methods for ET_o estimation using a limited set of features, and the previously estimated solar radiation.

	ANN	LSTM	GRU	RNN	RF	LSTM-ANN	RNN-ANN	GRU-ANN	$SR_{pred} \rightarrow$ FAO56-PM
R^2	**0.968**	0.965	0.967	0.967	0.950	**0.967**	0.965	0.966	**0.977**
MAE (mm/day)	**0.22**	0.23	**0.22**	**0.22**	0.27	**0.22**	0.22	0.22	**0.16**
MAPE (%)	6.66	6.91	6.71	**6.58**	7.68	**6.50**	6.55	6.73	**5.05**

the MAPE to 6.5%. Also, the obtained result is better than the ones presented in Vaz et al. (2022b) (see also Chap. 2), using a similar technique but based on ML algorithms, where the best performing model was Random Forest with an R^2 of 0.951, a MAE of 0.26 mm/day and a MAPE of 7.44 %. Feature engineering as well as the use of other weather limited features was attempted, but no further improvements could be made. Figure 3.4 (top) depicts the target ET_o (blue), estimated ET_o (orange), and absolute error (green) curves for the train and test (shadowed) dataset, allowing to observe the model closely follows the reference evapotranspiration target.

3.3.3.4 ET_o estimation using FAO56-PM equation and the approximated solar radiation

In this section, another type of hybrid approach was tested, which uses the previously estimated solar radiation as an input, together with temperature, humidity and wind speed to compute ET_o using the FAO56-PM formula ($SR_{pred} \rightarrow$ FAO56-PM). The result obtained is presented in Tab. 3.3, with an R^2 of 0.977, MAE of 0.16 mm/day and MAPE of 5.05 %. This result is better than the previously obtained ones, and once again is also better than what was presented by Vaz et al. (2022b) (see also Chap. 2), which consisted of an R^2 of 0.975, MAE of 0.18 mm/day and MAPE of 5.51 %. Fig. 3.4 (bottom) plots ET_o target, estimated ET_o, and absolute error, where in the bottom plot an improvement of the absolute error can be seen when compared with the previous plots.

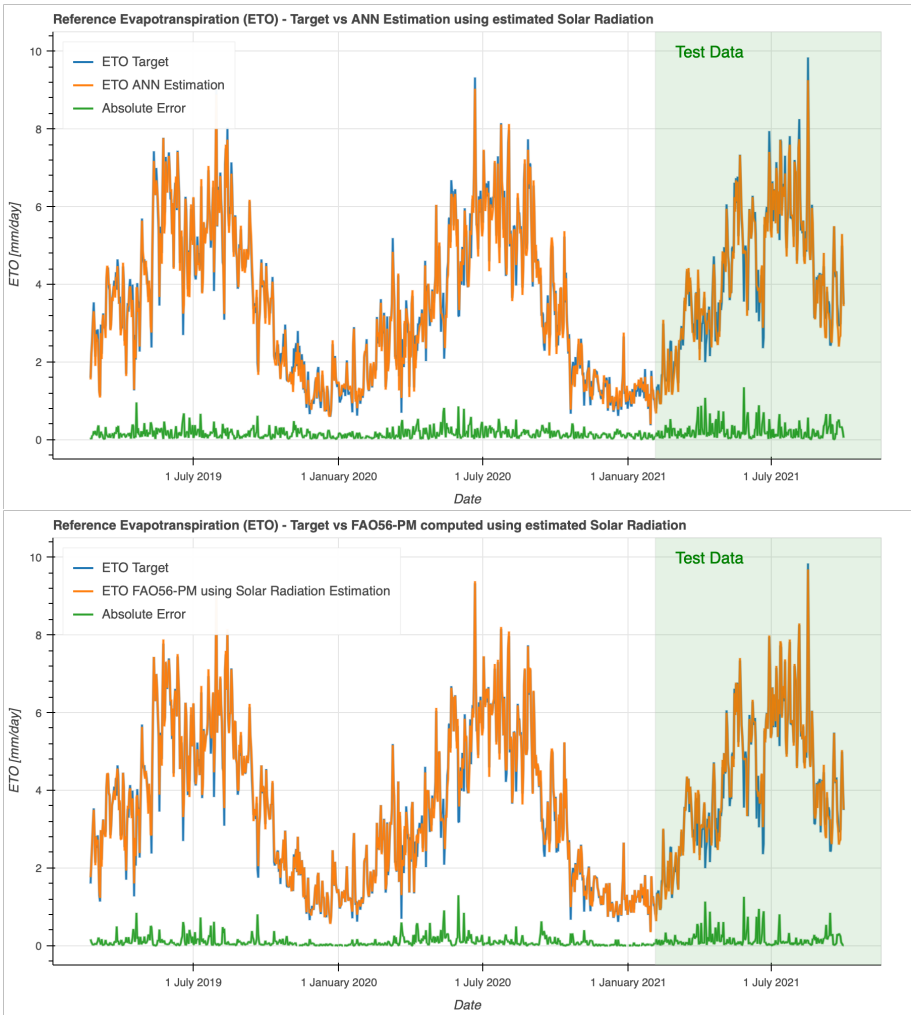

Figure 3.4: Target ET_o vs ANN estimation (top) and Target ET_o vs FAO56-PM values using estimated solar radiation (bottom).

3.4 Conclusion and future work

Evapotranspiration can be used to estimate the amount of water required by agriculture projects and green spaces, playing a key role in water management policies that combat the hydrological drought. Some equipments can be used to measure ET_o, but they are expensive to buy and maintain. As an alternative, many use meteorological data to estimate the ET_o values, but again, in the most well accepted formula (FAO56-PM), solar radiation is required, which is not common to have in general meteorological APIs.

In this chapter, several neural network based regression models (namely, ANN, LST,

GRU, and RNN) and hybrid approaches (namely, LSTM-ANN, RNN-ANN, GRU-ANN) for ET_o estimation were developed with different degrees of success. Since solar radiation is the main ET_o driver, as stated by several authors, models were also developed for estimating solar radiation using input features that are readily available in common weather forecast APIs. This allowed both the use of the previously estimated solar radiation in neural network regressors, to estimate ET_o, but also the possibility to use the hybrid approach where solar radiation is previously estimated and then the FAO56-PM algorithm is used to finally compute ET_o. The latter yielded the best results, with an R^2 of 0.977, a MAE of 0.16 mm/day, and an MAPE of 5.05 %. The attained results were also compared against the ones in a previous work from the authors, were several other machine learning methods were used (namely, Ordinary Least Squares, Ridge, Lasso, k-Nearest Neighbors, Support Vector Machine, Decision Tree, and Random Forest), demonstrating the overall good result, considering the limited weather parameter features that were used. Being the conditions different, and impossible to replicate, we can also see that the proposed methods generically improve the metric values attained by other authors such as the works from Ferreira et al. (2019).

Future work (out of scope of this thesis) will include a more extensive dataset, by using the existing WS infrastructure that is installed in the Algarve region. The objective will be to develop local and pooled models of ET_o predictors for the Algarve region. Also, since all limited feature models here presented are compatible with freely available weather forecast APIs, the impact of using such APIs as input data to the ML models here developed needs to be further assessed.

— It is not in our power to anticipate our destiny.

Winston Churchill

4

Weather data acquisition and RESTful API specification

Data, and in particular weather data, is the main driver of the methods proposed in this thesis. Also, communications present a fundamental role in a system such as the one being developed in the GSSIC project. In this context, this chapter will present in Sec. 4.1 the OpenWeatherMaps (OWM) data scrapper that was developed to collect and store in a database the meteorological data for Vale do Lobo, in south of Portugal. In Sec. 4.2 a RESTful API for the interconnection of the MEIOR module to the main GSSIC project is proposed, though the final development and integration of the MEIOR module is out of the scope of this thesis.

4.1 Data acquisition from OpenWeatherMaps

Meteorological forecasts and nowcasts (current weather) for Vale do Lobo are being collected from OWM and stored in a database since Semptember 2021. In this sense, a Python module was developed that collects the data through the OWM API (OpenWeatherMap, 2022), and is available in the following private GitHub link: https://github.com/pjmartins-ualg/OWM-scraper.git.

The collected data is stored in the non-relational MongoDB database, in two separate collections, one for nowcasts, and the other for forecasts. The documents are in JavaScript Object Notation (JSON) format. Figure 4.1 shows an example of a nowcast document, and Fig. 4.2 a partial example of a 48h, hourly, forecast (consisting of 48 total entries). The meteorological data gathered is to be used in future work that is out of the scope of this thesis.


```
{
    "_id":1630577700,
    "reference_time":1630577700
    "sunset_time":1630609214,
    "sunrise_time":1630562644,
    "clouds":0,
    "rain":{},
    "snow":{},
    "wind":{"speed":3.09,"deg":250},
    "humidity":73,
    "pressure":{"press":1017,"sea_level":null},
    "temperature":{"temp":22.96,"feels_like":23.22},
    "status":"Clear",
    "detailed_status":"clear sky",
    "weather_code":800,
    "weather_icon_name":"01d",
    "visibility_distance":10000,
    "dewpoint":17.86,
    "humidex":null,
    "heat_index":null,
    "utc_offset":null,
    "uvi":4.6,
    "precipitation_probability":null,
    "datetime": {"$date":"2021-09-02T10:15:00.000Z"}
}
```


Figure 4.1: Example of nowcast (current weather) document stored on the MongoDB database, with data retrieved from the OWM API.

```
{
    "_id":1631282400,
    "datetime":{"$date":"2021-09-10T14:00:00.000Z"},
    "forecast_data":[
        {
            "reference_time":1631282400,"sunset_time":null,"sunrise_time":
               null,"clouds":26,"rain":{},"snow":{},"wind":{"speed":6.11,"
               deg":231,"gust":6.05},"humidity":67,"pressure":{"press":1019
               ,"sea_level":null},"temperature":{"temp":24.2,"feels_like":2
               4.43},"status":"Clouds","detailed_status":"scattered clouds"
               ,"weather_code":802,"weather_icon_name":"03d","
               visibility_distance":10000,"dewpoint":17.69,"humidex":null,"
               heat_index":null,"utc_offset":null,"uvi":6.23,"
               precipitation_probability":0,"datetime":{"$date":"2021-09-10
               T14:00:00.000Z"}
        },
        {
            "reference_time":1631286000,"sunset_time":null,"sunrise_time":
               null,"clouds":20,"rain":{},"snow":{},"wind":{"speed":6.6,"
               deg":237,"gust":6.84},"humidity":69,"pressure":{"press":1019
               ,"sea_level":null},"temperature":{"temp":23.96,"feels_like":
               24.21},"status":"Clouds","detailed_status":"few clouds","
               weather_code":801,"weather_icon_name":"02d","
               visibility_distance":10000,"dewpoint":17.93,"humidex":null,"
               heat_index":null,"utc_offset":null,"uvi":4.41,"
               precipitation_probability":0,"datetime":{"$date":"2021-09-10
               T15:00:00.000Z"}
        },

        {...},

        {
            "reference_time":1631451600,"sunset_time":null,"sunrise_time":
               null,"clouds":0,"rain":{},"snow":{},"wind":{"speed":3.81,"
               deg":173,"gust":3.87},"humidity":57,"pressure":{"press":1014
               ,"sea_level":null},"temperature":{"temp":24.93,"feels_like":
               24.97},"status":"Clear","detailed_status":"clear sky","
               weather_code":800,"weather_icon_name":"01d","
               visibility_distance":10000,"dewpoint":14.73,"humidex":null,"
               heat_index":null,"utc_offset":null,"uvi":6.27,"
               precipitation_probability":0,"datetime":{"$date":"2021-09-12
               T13:00:00.000Z"}
        }
    ]
}
```


Figure 4.2: Partial example of next 48h, hourly, forecast document stored on the MongoDB database, with data retrieved from the OWM API.

4.2 RESTful API specification proposal for smart irrigation module integration

This section briefly describes the interconnection of the MEIOR module to the back-end and front-end of GSSIC. The communication will be implemented through a RESTful web service (Pautasso et al., 2013; REST, 2022), using Flask (Flask-Framework, 2022), which is a Python micro framework that allows the creation of a RESTful API. Internally, the non-relational MongoDB database (MongoDB, 2022) will be used for data management and storage. Thus, the interconnection of the MEIOR module will be done transparently through URIs/endpoints (Berners-Lee, 2022; Relan, 2019) implemented over standard HTTP methods such as GET, POST, PUT, DELETE and PATCH (W3C, 2022). Both the payload and API response will be in JSON format (JSON, 2022). All dates are represented in Unix Time, that is, using an integer representing the number of seconds that have passed since the beginning of the Unix Epoch, excluding leap seconds. The Unix Epoch time is defined as starting in 00:00:00 UTC on January 1, 1970 (Hauser, 2018).

The standard GET method is used to obtain information from the MEIOR module, and a JSON will be returned with the information requested from the API. For example, to add a new green space or a new sensor to the platform, the POST method is used, along with the required payload in JSON format. The POST method will also be used to add a new reading, for example, to a soil moisture sensor that was previously registered on the platform. The PATCH method allows modifying the registered data of a green space or a sensor previously registered on the platform. The DELETE method will be used to delete a sensor or a green space, however, for the operation to be performed, it is mandatory to provide a security token.

Table 4.1 summarizes the proposed endpoints. Due to space constraints, the full URI is not included, so the prefix `http://[hostname]/meior/api/v1.0/` should always be added to the addresses specified in the table. For example, to get the ET_o prediction

for a certain green space, the URI to use is: `http://[hostname]/meior/api/v1.0/gspace/<gspace_id>/ETO/FAO-56PM/predict/<num_days>`.

Table 4.1: Proposed API URIs/endpoints.

HTTP Method	URI / endpoint
	Green space management
POST	/gspace
PATCH	/gspace/<gspace_id>
GET	/gspace/<gspace_id>
GET	/gspace/
DELETE	/gspace/<gspace_id>/<security_token>
	Green space sensor management
POST	/gspace/<gspace_id>/sensors
PATCH	/gspace/<gspace_id>/sensors/<sensor_id>
POST	/gspace/<gspace_id>/sensors/<sensor_id>
GET	/gspace/<gspace_id>/sensors/<sensor_id>
GET	/gspace/<gspace_id>/sensors
DELETE	/gspace/<gspace_id>/sensors/<sensor_id>/<security_token>
	Green space reference evapotranspiration (real and predicted)
GET	/gspace/<gspace_id>/ETO/FAO-56PM/<start_date>/<end_date>
GET	/gspace/<gspace_id>/ETO/predict/<num_days>
	Green space irrigation (real and predicted)
GET	/gspace/<gspace_id>/irrigation/<start_date>/<end_date>
GET	/gspace/<gspace_id>/irrigation/predict/<num_days>
	Green space solar radiation (real and predicted)
GET	/gspace/<gspace_id>/solar_radiation/<start_date>/<end_date>
GET	/gspace/<gspace_id>/solar_radiation/predict/<num_days>
	Weather forecasts collected daily at 7pm
GET	/gspace/<gspace_id>/meteo_api/forecast/<start_date>/<end_date>
GET	/gspace/<gspace_id>/meteo_api/forecast/<num_days>
	Current weather data
GET	/gspace/<gspace_id>/meteo_api/<start_date>/<end_date>
	Data from weather station - if available
GET	/gspace/<gspace_id>/weather_station/<start_date>/<end_date>

The specification of this API, regarding endpoints and services is not final, and may change throughout the development and final integration of the MEIOR module (which is out of the scope of this thesis), whenever new needs arise, by adding, modifying or removing endpoints to it. A detailed specification of all the endpoints presented in Tab. 4.1 can be found on Annex C.

— Live long and prosper.

Mr. Spock.

5

Conclusion and future work

Evapotranspiration is a key parameter for irrigation water management for both agriculture and green spaces. In this thesis, high accuracy machine and deep learning models were developed to estimate evapotranspiration, as well as solar radiation, since as stated by several authors and verified by us, solar radiation is the main driver of evapotranspiration. FAO56-PM is the most well accepted formula for ET_0 computation, but it precisely requires solar radiation, which is not commonly available in general meteorological APIs. In this context, the models developed on this thesis use limited weather parameters as input features, like temperature, humidity, and wind speed, that are compatible with most weather forecast APIs, such as OpenWeatherMap (OpenWeatherMap, 2022) or IPMA (IPMA, 2022).

Chapter 2 presented the results obtained by the use of ML algorithms for the prediction of evapotranspiration and solar radiation over limited weather parameters. The ML algorithms used are OLS, Ridge, Lasso, kNN, SVM, Decision Tree and Random Forest. Firstly, using data from Vale do Lobo weather station, a baseline was established where all weather parameters recorded by the weather station were used as input features, namely temperature, dew point, relative humidity, solar radiation, wind speed, wind direction, atmospheric pressure, rain intensity, and precipitation. The FAO56-PM method (Allen et al., 1998) was used to compute the target ET_o. Having as reference the model performance of the established baseline, and while watching Lasso coefficients and Random Forest feature importance, input features from the established baseline where selectively dropped while maintaining similar model performance as the baseline, establishing a limited weather parameter features baseline that consists of maximum and minimum temperature, average humidity, average wind, and average solar radiation. Except for solar radiation, all input features are compatible with weather parameters provided by weather forecast online services. Since solar radiation is not provided by common online weather forecast services, or present a high-cost penalty, different approaches were taken: (i) directly estimate ET_o using as input features month, maximum and minimum temperature, average humidity and average wind speed, (ii) firstly estimate solar radiation using limited features, and then inject the estimated solar radiation as a feature into another ML model, and (iii) a hybrid approach that uses the estimated solar radiation as input for ET_o computation using the FAO56-PM method. The hybrid approach presented the best result, with an R^2 equal to 0.975, MAE of 0.18 mm/day and a MAPE of 5.51 %, which is similar to what was obtained in the initial baseline using all weather station provided parameters, and surpassing the results obtained on the limited weather features ML baseline, that used as input feature the actual solar radiation measured by the weather station. These results were accepted for publication on the International Conference on Computational Science (Vaz et al., 2022b).

In sequence, the next part of the study (Chap. 3) followed the same strategy that was

used previously, but using neural network algorithms, namely: Artificial Neural Networks (ANN), Long Short Term Memory (LSTM), Gated Recurrent Unit (GRU), Recursive Neural Network (RNN), and hybrid models like, LSTM-ANN, GRU-ANN, and RNN-ANN. The following approaches were taken: (i) directly estimate ET_o using as input features month, maximum and minimum temperature, average humidity and average wind speed, (ii) estimate solar radiation using limited features (plus the derived features sunset hour angle and daylight hours), and then inject the estimated solar radiation as a feature into another ML model, and (iii) a hybrid approach that uses the estimated solar radiation as input for ET_o computation using the FAO56-PM method. Like on Chapter 2, the hybrid approach (estimate solar radiation, and use it as input to FAO56-PM method) yielded the best result, with an R^2 equal to 0.977, MAE of 0.16 mm/day and MAPE of 5.05 %, being the more accurate result obtained on this thesis. However, good results were also obtained when directly estimating ET_o using limited features (no solar radiation actual or estimated) with and R^2 of 0.962, MAE of 0.24 mm/day and MAPE of 7.25 %, whereas for the work presented in Chap. 2 the best metrics obtained were an R^2 of 0.936, MAE of 0.32 mm/day and MAPE of 9.11 %. Further, all neural network algorithm based models that were developed always gave better performance than the ML algorithms based models for both ET_o and SR estimations. The recursive hybrid models (LSTM-ANN, GRU-ANN, and RNN-ANN) that were developed have the advantage of requiring less training and inference time, while maintaining comparable performance to the recursive non-hybrid models. This is a key factor when computationally limited hardware needs to be used on the field. Due to the unavailability of the authors' datasets and algorithms, it was not possible to directly compare the attained results with the ones from this thesis. Nevertheless, reported metrics, such as R^2, where improved by the methods here proposed.

Chapter 4 was divided in two sections. In the first section a data acquisition and storage module was presented, that is collecting OWM meteorological data since September 2021. In the second section, a RESTful API proposal was specified, that enables the use of the smart irrigation MEIOR module as a webservice, to facilitate the com-

munication of the MEIOR module with the remaining modules (WS, weather APIs, irrigation platform, etc.).

From all the contributions of this thesis, the most noteworthy to the advance of the state-of-the-art of ET_o estimators was the development of the hybrid method, where the solar radiation is estimated from limited weather features (using machine or deep learning) and then injected in the FAO56-PM, yielding a better result than directly trying to estimate ET_o from the limited weather parameters.

Future work will include the use of the weather station network installed in the Algarve region, south Portugal, and develop local and pooled ET_o prediction models. The final integration with the GSSIC project's irrigation platform was out of the scope of this thesis, being future work. Finally, the impact of using freely available weather forecast APIs as input data to the models needs to be assessed.

Bibliography

Abadi, M., Agarwal, A., Barham, P., Brevdo, E., Chen, Z., Citro, C., Corrado, G. S., Davis, A., Dean, J., Devin, M., Ghemawat, S., Goodfellow, I., Harp, A., Irving, G., Isard, M., Jia, Y., Jozefowicz, R., Kaiser, L., Kudlur, M., Levenberg, J., Mané, D., Monga, R., Moore, S., Murray, D., Olah, C., Schuster, M., Shlens, J., Steiner, B., Sutskever, I., Talwar, K., Tucker, P., Vanhoucke, V., Vasudevan, V., Viégas, F., Vinyals, O., Warden, P., Wattenberg, M., Wicke, M., Yu, Y., and Zheng, X. (2015). TensorFlow: Large-scale machine learning on heterogeneous systems. Software available from tensorflow.org.

Al-Kaisi, M. M. and Broner, I. (2009). *Crop water use and growth stages*. Colorado State University Extension Fort Collins, CO, USA.

Allen, R. (2003). Crop coefficients. Encyclopaedia of Water Science.

Allen, R. G., Pereira, L. S., Raes, D., and Smith, M. (1998). Crop evapotranspiration-guidelines for computing crop water requirements-FAO irrigation and drainage paper 56. *FAO, Rome*, 300(9):D05109.

Berners-Lee, e. a. (2022). URI generic syntax. https://datatracker.ietf.org/doc/html/rfc3986. Accessed: 2022-06-01.

Blaney, H. F. and Criddle, W. D. (1962). *Determining consumptive use and irrigation water requirements*. Number 1275. US Department of Agriculture.

Cho, K., Van Merriënboer, B., Gulcehre, C., Bahdanau, D., Bougares, F., Schwenk, H., and Bengio, Y. (2014). Learning phrase representations using RNN encoder-decoder for statistical machine translation. *arXiv preprint arXiv:1406.1078*.

Chollet, F. (2022). Keras. https://github.com/fchollet/keras. Accessed: 2022-06-01.

Doorenbos, J. (1977). Guidelines for predicting crop water requirements. *FAO irrigation and drainage paper*, 24:1–179.

Ferreira, L. B., da Cunha, F. F., de Oliveira, R. A., and Filho, E. I. F. (2019). Estimation of reference evapotranspiration in Brazil with limited meteorological data using ANN and SVM: A new approach. *Journal of Hydrology*, 572:556–570.

Flask-Framework (2022). Flask framework documentation. https://flask.palletsprojects.com/en/2.0.x/. Accessed: 2022-06-01.

Garbin, C., Zhu, X., and Marques, O. (2020). Dropout vs. batch normalization: an empirical study of their impact to deep learning. *Multimedia Tools and Applications*, 79(19):12777–12815.

Glorot, X. and Bengio, Y. (2010). Understanding the difficulty of training deep feedforward neural networks. In *Proceedings of the thirteenth international conference on artificial intelligence and statistics*, pages 249–256. JMLR Workshop and Conference Proceedings.

Granata, F. (2019). Evapotranspiration evaluation models based on machine learning algorithms. *Agricultural Water Management*, 217:303–315.

Hargreaves, G. H. and Samani, Z. A. (1982). Estimating potential evapotranspiration. *Journal of the Irrigation and Drainage Division*, 108(3):225–230.

Harris, C. R., Millman, K. J., van der Walt, S. J., Gommers, R., Virtanen, P., Cournapeau, D., Wieser, E., Taylor, J., Berg, S., Smith, N. J., Kern, R., Picus, M., Hoyer, S., van Kerkwijk, M. H., Brett, M., Haldane, A., del Río, J. F., Wiebe, M., Peterson, P., Gérard-Marchant, P., Sheppard, K., Reddy, T., Weckesser, W., Abbasi, H., Gohlke, C., and Oliphant, T. E. (2020). Array programming with NumPy. *Nature*, 585(7825):357–362.

Hauser, E. (2018). UNIX time, UTC, and datetime: Jussivity, prolepsis, and incorrigibility in modern timekeeping. *Proceedings of the Association for Information Science and Technology*, 55(1):161–170.

Hochreiter, S. and Schmidhuber, J. (1997). Long short-term memory. *Neural Computation*, 9(8):1735–1780.

Hyndman, R. J. and Athanasopoulos, G. (2021). *Forecasting: Principles and Practice*. OTexts, Australia, 3rd edition.

IPMA (2022). IPMA - Instituto Português do Mar e da Atmosfera. https://www.ipma.pt/pt/index.html. Accessed: 2022-06-01.

JONG, R. D. and Shields, J. (1988). Available water-holding capacity maps of Alberta, Saskatchewan and Manitoba. *Canadian journal of soil science*, 68(1):157–163.

JSON (2022). JSON data interchange standard. https://www.json.org/json-en.html. Accessed: 2022-06-01.

Kingma, D. P. and Ba, J. (2014). ADAM: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*.

Kishore Ayyadevara, V. (2018). *Pro machine learning algorithms*. APRESS, New York, NY, 1st edition.

Makkink, G. F. (1957). Testing the Penman formula by means of lysimeters. *Journal of the Institution of Water Engineers*, 11:277–288.

McKinney, W. (2010). Data Structures for Statistical Computing in Python. In Stéfan van der Walt and Jarrod Millman, editors, *Proceedings of the 9th Python in Science Conference*, pages 56 – 61.

MongoDB (2022). The application data platform. https://www.mongodb.com. Accessed: 2022-06-01.

OpenWeatherMap (2022). Openweathermap - current weather and forecast. https://openweathermap.org/. Accessed: 2022-06-01.

pandas development team, T. (2020). pandas-dev/pandas: Pandas.

Pautasso, C., Wilde, E., and Alarcon, R. (2013). *REST: Advanced Research Topics and Practical Applications*. Springer-Verlag GmbH.

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

Penman, H. L. (1948). Natural evaporation from open water, bare soil and grass. *Proceedings of the Royal Society of London. Series A. Mathematical and Physical Sciences*, 193(1032):120–145.

Pomerat, J., Segev, A., and Datta, R. (2019). On neural network activation functions and optimizers in relation to polynomial regression. In *2019 IEEE International Conference on Big Data (Big Data)*, pages 6183–6185.

Priestley, C. H. B. and Taylor, R. J. (1972). On the assessment of surface heat flux and evaporation using large-scale parameters. *Monthly weather rev.*, 100(2):81–92.

Rasamoelina, A. D., Adjailia, F., and Sinčák, P. (2020). A review of activation function for artificial neural network. In *2020 IEEE 18th World Symposium on Applied Machine Intelligence and Informatics (SAMI)*, pages 281–286. IEEE.

Relan, K. (2019). *Building REST APIs with Flask*. Apress.

REST (2022). REST API Tutorial. https://restfulapi.net/. Accessed: 2022-06-01.

Rumelhart, D. E., Hinton, G. E., and Williams, R. J. (1985). Learning internal representations by error propagation. Technical report, California Univ San Diego La Jolla Inst for Cognitive Science.

Shahidian, S., Serralheiro, R., Serrano, J., Teixeira, J., Haie, N., and Santos, F. (2012). *Hargreaves and other reduced-set methods for calculating evapotranspiration*. IntechOpen.

Sharma, V. (2022). Basics of irrigation scheduling. https://extension.umn.edu/irrigation/basics-irrigation-scheduling. Accessed: 2022-06-01.

Shukla, P. R., Skeg, J., Buendia, E. C., Masson-Delmotte, V., Pörtner, H., Roberts, D. C., Zhai, P., Slade, R., Connors, S., van Diemen, S., Ferrat, M., Haughey, E., Luz, S., Pathak, M., Petzold, J., Pereira, J. P., Vyas, P., Huntley, E., Kissick, K., Belkacemi, M., and Malley, J. (2019). *Climate Change and Land: An IPCC Special Report on Climate Change, Desertification, Land Degradation, Sustainable Land Management, Food Security, and Greenhouse Gas Fluxes in Terrestrial Ecosystems*.

Skiena, S. S. (2017). *The Data Science Design Manual*. Springer International Publishing.

Smith, L. N. (2018). A disciplined approach to neural network hyper-parameters: Part 1–learning rate, batch size, momentum, and weight decay. *arXiv preprint arXiv:1803.09820*.

Srivastava, N., Hinton, G., Krizhevsky, A., Sutskever, I., and Salakhutdinov, R. (2014). Dropout: a simple way to prevent neural networks from overfitting. *The journal of machine learning research*, 15(1):1929–1958.

Thornthwaite, C. W. (1948). An approach toward a rational classification of climate. *Geographical review*, 38(1):55–94.

Vaz, P. J., Schütz, G., Guerrero, C., and Cardoso, P. J. S. (2022a). Applications of neural networks for evapotranspiration prediction over limited weather parameters. *IEEE Access (submited for publication)*.

Vaz, P. J., Schütz, G., Guerrero, C., and Cardoso, P. J. S. (2022b). A study on the prediction of evapotranspiration using freely available meteorological data. In Groen, D., de Mulatier, C., Paszynski, M., Krzhizhanovskaya, V. V., Dongarra, J. J., and Sloot, P. M. A., editors, *Computational Science – ICCS 2022*, pages 436–450, Cham. Springer International Publishing.

Vremec, M. and Collenteur, R., X. Z. (2021). PyEt – open source Python package for calculating reference and potential evaporation (v1.0.1). Zenodo.

W3C (2022). HTTP method definitions. https://www.w3.org/Protocols/rfc2616/rfc2616-sec9.html. Accessed: 2022-06-01.

Wu, L. and Fan, J. (2019). Comparison of neuron-based, kernel-based, tree-based and curve-based machine learning models for predicting daily reference evapotranspiration. *PloS one*, 14(5):e0217520.

Zhang, J., Guan, K., Peng, B., Pan, M., Zhou, W., Jiang, C., Kimm, H., Franz, T. E., Grant, R. F., Yang, Y., et al. (2021). Sustainable irrigation based on co-regulation of soil water supply and atmospheric evaporative demand. *Nature communications*, 12(1):1–10.

A

Machine learning algorithms parameters

Table A.1: Sets of hyperparameters used in the grid search procedure (according to the available parameters in the Scikit-learn suite (Pedregosa et al., 2011))

Model	Hyperparameters	Range explored
OLS	fit_intercept	{False; True}
	normalize	{False; True
Ridge	alpha	$\{10^{-4}, 10^{-3}, \ldots, 10^{2}\}$
	fit_intercept	{False; True}
	normalize	{False; True}
Lasso	alpha	$\{10^{-4}, 10^{-3}, \ldots, 10^{2}\}$
	fit_intercept	{False; True}
	normalize	{False; True}
KNN	n_neighbours	{1, 2, . . . , 7}
	weights	{uniform; distance}
	leaf_size	{1, 3, 5, 10, 20, 30, 40}
SVM	C	{0.01, 0.1, 0.5, 1.0, 10.0, 100}
	max_iter	10000
	fit_intercept	{False; True}
DT	splitter	{best; random}
	criterion	{mse; friedman_mse; mae; poisson}
Forest	n_estimators	{10, 100, 250, 500, 750, 1000}
	min_samples_leaf	{1, 2, 3, 5, 10}
	max_depth	{3, 5, 10}
	criterion	{mse}
	max_features	{None; sqrt; log2}

Table A.2: Tunned parameters

Model	Hyperparameter	Table 2.1	Table 2.2	Table 2.3	Table 2.4	Table 2.5	Table 2.6
OLS	fit_intercept	True	False	False	True	True	False
	normalize	True	False	False	False	True	False
Ridge	alpha	100	0.1	100	100	1	0.1
	fit_intercept	True	True	False	True	True	True
	normalize	False	True	False	False	True	True
Lasso	alpha	0.1	0.1	0.1	20	100	0.01
	fit_intercept	True	False	False	False	False	True
	normalize	False	False	False	False	False	True
KNN	n_neighbours	4	2	2	4	2	2
	weights	distance	distance	distance	distance	distance	distance
	leaf_size	1	1	1	1	1	1
SVM	C	0,01	0,01	0,01	0,1	0,1	0.01
	max_iter	10000	10000	10000	10000	10000	10000
	fit_intercept	True	False	False	False	True	False
DT	splitter	Random	best	Random	best	best	Random
	criterion	friedman_mse	mae	friedman_mse	friedman_mse	friedman_mse	friedman_mse
Forest	n_estimators	100	1000	500	100	100	1000
	min_samples_leaf	1	1	1	1	1	1
	max_depth	10	10	10	10	10	10
	criterion	mse	mse	mse	mse	mse	mse
	max_features	None	None	None	None	None	None

B

Deep learning algorithms parameters

Table B.1: Sets of hyperparameters used in the grid search procedure.

Model	Hyperparameters	Range explored
ANN, LSTM, GRU, RNN, LSTM-ANN, RNN-ANN, GRU-ANN	activation	{ReLU; Tanh; Sigmoid}
	weight_constraint	{2, 3, 4, 5}
	dropout_rate	{0.0, 0.1, 0.2, 0.3, 0.4, 0.5, 0.6}
	use_bias	{False; True}
RF	n_estimators	{10, 100, 250, 500, 750, 1000}
	min_samples_leaf	{1, 2, 3, 5, 10}
	max_depth	{3, 5, 10}
	criterion	{squared_error}
	max_features	{None; sqrt; log2}

Table B.2: Tuned hyperparameters.

Model	Hyperparameter	Table 3.1	Table 3.2	Table 3.3
ANN	activation	ReLU	ReLU	ReLU
	weight_constraint	2	2	2
	dropout_rate	0.0	0.3	0.4
	use_bias	True	True	True
LSTM	activation	ReLU	ReLU	ReLU
	weight_constraint	3	3	3
	dropout_rate	0.0	0.3	0.3
	use_bias	True	True	True
GRU	activation	ReLU	ReLU	ReLU
	weight_constraint	2	2	2
	dropout_rate	0.0	0.1	0.1
	use_bias	True	True	True
RNN	activation	ReLU	ReLU	ReLU
	weight_constraint	3	3	3
	dropout_rate	0.1	0.2	0.1
	use_bias	True	True	True
LSTM-ANN	activation	ReLU	ReLU	ReLU
	lstm_weight_constraint	3	3	3
	ann_weight_constraint	3	3	3
	lstm_dropout_rate	0.0	0.0	0.0
	ann_dropout_rate	0.0	0.2	0.1
	lstm_use_bias	True	True	True
	ann_use_bias	True	True	True
RNN-ANN	activation	ReLU	ReLU	ReLU
	rnn_weight_constraint	2	2	2
	ann_weight_constraint	3	3	3
	rnn_dropout_rate	0.0	0.0	0.0
	ann_dropout_rate	0.2	0.3	0.1
	rnn_use_bias	True	True	True
	ann_use_bias	True	True	True
GRU-ANN	activation	ReLU	ReLU	ReLU
	gru_weight_constraint	3	3	2
	ann_weight_constraint	3	3	2
	gru_dropout_rate	0.0	0.0	0.0
	ann_dropout_rate	0.0	0.4	0.1
	gru_use_bias	True	True	True
	ann_use_bias	True	True	True
RF	n_estimators	500	10	1000
	min_samples_leaf	1	5	1
	max_depth	10	5	10
	criterion	mse	mse	mse
	max_features	None	None	None

C

Detailed API specification for smart irrigation module integration

C.1 app package

C.1.1 Routing table

POST /meior/api/v1.0/gspace

POST /meior/api/v1.0/gspace

Creates a new Green Space.

Request format:

form Payload

```
{
    "gspace_name" : <str:mandatory>,
    "latitude" : <str:mandatory>,
    "longitude" : <str:mandatory>,
    "address" : <str:optional>,
    "description" : <str:optional>,
    "zones" : [
        { "zone_id" : <int:mandatory>,
                "area" : <float:optional>,
                "Kc" : <float:mandatory>,
                "description" : <str:optional>
        },
        {...}
    ]
}
```

Example response:

```
{
    gspace_id" : <int>
}
```

Status Codes

- 201 Created – no error
- 404 Not Found – not found

GET /meior/api/v1.0/gspace/

GET /meior/api/v1.0/gspace/

Returns a JSON with a list of Green Spaces' infos.

Example response:

```
[
    {
        "gspace_id" : <int>,
        "gspace_name" : <str>,
        "latitude" : <float>,
        "longitude" : <float>,
        "address" : <str>,
        "description" : <str>,
        "zones" : [
            {...},
            {...}
        ]
    },
    {...}
]
```

Status Codes

- 200 OK – no error
- 404 Not Found – not found

GET /meior/api/v1.0/about

This endpoint returns more detailed information about the API version.

GET /meior/api/v1.0/gspace/(int: *gspace_id*)**/ETO/FAO-56PM/predict/**

int: *num_days*

GET /meior/api/v1.0/gspace/<int:gspace_id>/ETO/FAO-56PM/predict/<int:num_days>

Returns ETO predictions in mm for the number of days requested, up to a maximum of 7 days. Day zero represents current day prediction.

Example response:

```
{
    <int:day_number> : <float:ETO_prediction>,
    <int:day_number> : <float:ETO_prediction>,
    ...
}
```

Status Codes

- 200 OK – no error
- 404 Not Found – not found

GET /meior/api/v1.0/gspace/(int: *gspace_id*)**/solar_radiation/predict/int:** *num_days*

GET /meior/api/v1.0/gspace/<int:gspace_id>/solar_radiation/predict/<int:num_days>

Returns a JSON with predicted solar radiation data in W/m^2 for the number of days requested, up to a maximum of 7 days. Day zero represents current day prediction.

Example response:

```
{
    <int:day_number> : <float>,
    <int:day_number> : <float>,
    ...
}
```

Status Codes

- 200 OK – no error
- 404 Not Found – not found

GET /meior/api/v1.0/gspace/(int: *gspace_id*)**/irrigation/predict/**

int: *num_days*

GET /meior/api/v1.0/gspace/<int:gspace_id>/irrigation

/predict/<int:num_days>

Returns irrigation predictions in mm for the number of days requested, up to a maximum of 7 days. Day zero represents current day prediction.

Example response:

```
{
    <int:day_number> : {
        <int:zone_id> : <float:irrigation_prediction>,
        <int:zone_id> : <float:irrigation_prediction>,
        ...
    },
    <int:day_number> : {
        <int:zone_id> : <float:irrigation_prediction>,
        <int:zone_id> : <float:irrigation_prediction>,
        ...
    },
    ...
}
```

Status Codes

- 200 OK – no error
- 404 Not Found – not found

GET /meior/api/v1.0/gspace/(int: *gspace_id*)**/meteo_api/forecast/**

int: *start_date***/int:** *end_date*

GET /meior/api/v1.0/gspace/<int:gspace_id>/meteo_api/forecast/<int:start_date>/<int:end_date>

Returns a JSON with stored (past) weather forecasts between the requested dates. By definition stored weather forecasts are done everyday at 7 p.m. (local time). For each <int:timestamp> in the response an array of dictionaries is returned, one dictionary for each of the forecasted days.

Example response:

```
{
    <int:timestamp> : {
        <int:day_number> : {
            "Reference_Time" : <int:unixtime>,
            "Detailed_Status" : <str>,
            "Temp_Max[degC]" : <float>,
            "Temp_Min[degC]" : <float>,
            "Humidity[%]" : <int>,
            "Clouds[%]" : <float>,
            "Dewpoint[degC]" : <float>,
            "Precipitation[mm]": <float>,
            "Wind Speed[m/s]" : <float>,
            "Wind gust[m/s]" : <float>,
            "Wind Dir[deg]" : <int>,
            "UV Index[0->11+]" : <float>,
            "Pressure[mbar]" : <int>,
            "Sunrise_Time" : <int:unixtime>,
            "Sunset_Time" : <int:unixtime>,
            "source" : <str>
        },
        <int:day_number> : { ... },
        ...
    },
    <int:timestamp> : {...},
    ...
}
```

Example (for a single day):


```
{
    "1637841600" : {
        0: {
            "reference_time" : 1637841600,
            "Detailed_Status" : "clear sky",
            "Temp_Max[degC]" : 15.52,
            "Temp_Min[degC]" : 8.72,
            "Humidity[%]" : 44,
            "Clouds[%]" : 0,
            "Dewpoint[degC]" : 2.28,
            "Precipitation[mm]": 0,
            "Wind_Speed[m/s]" : 6.25,
            "Wind_gust[m/s]" : 10.35,
            "Wind_Dir[deg]" : 29,
            "UV_Index[0->11+]" : 3,
            "Pressure[mbar]" : 1016,
            "Sunrise_Time" : 1637824890,
            "Sunset_Time" : 1637860650,
            "source" : "OWM"
        },
        1: { ... }
    }
}
```


Status Codes

- 200 OK – no error
- 404 Not Found – not found

GET /meior/api/v1.0/gspace/(int: *gspace_id*)**/meteo_api/forecast/**

int: *num_days*

GET /meior/api/v1.0/gspace/<int:gspace_id>/meteo_api/

forecast/<int:num_days>

Returns a JSON that contains predicted meteorological data for the number of days requested, up to a maximum of 7 days. Day zero represents current day prediction.

Example response:

```
{
    <int:day_number> : {
        "Prediction_Date" : <int:unixtime>
        "Detailed_Status" : <str>
        "Temp_Max[degC]" : <float>
        "Temp_Min[degC]" : <float>
        "Humidity[%]" : <int>
        "Clouds[%]" : <float>
        "Dewpoint[degC]" : <float>
        "Precipitation[mm]": <float>
        "Wind Speed[m/s]" : <float>
        "Wind gust[m/s]" : <float>
        "Wind Dir[deg]" : <int>
        "UV Index[0->11+]" : <float>
        "Pressure[mbar]" : <int>
        "Sunrise_Time" : <int:unixtime>
        "Sunset_Time" : <int:unixtime>
    },
    <int:day_number> : {...},
    ...
}
```

Status Codes

- 200 OK – no error
- 404 Not Found – not found

GET /meior/api/v1.0/gspace/(int: *gspace_id*)**/ETO/FAO-56PM/**

int: *start_date*/**int:** *end_date*

GET /meior/api/v1.0/gspace/<int:gspace_id>/ETO/FAO-56PM/<int:start_date>/<int:end_date>

Returns ETO in mm between the requested (past) dates. Time needs to be provided in Unix time format.

Example response:

```
{
    <timestamp> : <float:ETO>,
    <timestamp> : <float:ETO>,
    <timestamp> : <float:ETO>
}
```

Status Codes

- 200 OK – no error
- 404 Not Found – not found

GET /meior/api/v1.0/gspace/(int: *gspace_id*)**/solar_radiation/int:** *start_date*/**int:** *end_date*

GET /meior/api/v1.0/gspace/<int:gspace_id>/solar_radiation/<int:start_date>/<int:end_date>

Returns a JSON with stored (past) solar radiation data in W/m^2/day between the requested dates. Time needs to be provided in Unix Time format.

Example response:

```
{
    <timestamp> : <float>,
    <timestamp> : <float>,
    ...
}
```

Status Codes

- 200 OK – no error
- 404 Not Found – not found

GET /meior/api/v1.0/gspace/(*gspace_id*)**/weather_station/***start_date*/*end_date*

GET /meior/api/v1.0/gspace/<gspace_id>/weather_station/<start_date>/<end_date>

Returns a JSON that contains historical weather station data between the requested dates. Weather parameters depend on the used station. The example's structure is for Vale do Lobo weather station.

Example response:

```
{
    <int:timestamp> : {
        "Reference_Time" : <int:unixtime>,
        "Date" : <str>,
        "Day" : <int>,
        "TempAvg[degC]" : <float>,
        "TempMin[degC]" : <float>,
        "TempMinTime" : <str>,
        "TempMax[degC]" : <float>,
        "TempMaxTime" : <str>,
        "DewpointAvg[degC]" : <float>,
        "DewpointMin[degC]" : <float>,
        "DewpointMax[degC]" : <float>,
        "HumididtyAvg[%]" : <float>,
        "HumidityMin[%]" : <float>,
        "HumidityMax[%]" : <float>,
        "PressureAvg[hPa]" : <float>,
        "PressureMin[hPa]" : <float>,
        "PressureMax[hPa]" : <float>,
        "SolarRadiationAVG[W/m^2]" : <float>,
        "SolarRadiationMax[W/m^2]" : <float>,
```

```
        "EvapotranspDaily[mm]" : <float>,
        "RainIntensity[mm/h]" : <float>,
        "Precipitation[mm]" : <float>,
        "WindAvg[Km/h]" : <float>,
        "WindMax[Km/h]" : <float>,
        "WindGust[Km/h]" : <float>,
        "WindDirection" : <str>,
        "source" : <str>
    },
    <int:timestamp> : {...},
    ...
}
```

Status Codes

- 200 OK – no error
- 404 Not Found – not found

GET /meior/api/v1.0/gspace/(int: *gspace_id*)/irrigation/int: *start_date*/int: *end_date*

GET /meior/api/v1.0/gspace/<int:gspace_id>/irrigation/<int:start_date>/<int:end_date>

Returns irrigation in mm between the requested (past) dates. Time needs to be provided in Unix time format.

Example response:

```
{
    <timestamp> : {
        <int:zone_id> : <float:irrigation>,
        <int:zone_id> : <float:irrigation>,
        ...
    },
    <timestamp> : {
```

```
        <int:zone_id> : <float:irrigation>,
        <int:zone_id> : <float:irrigation>,
        ...
    },
    ...
}
```

Status Codes

- 200 OK – no error
- 404 Not Found – not found

GET /meior/api/v1.0/gspace/(*gspace_id*)**/meteo_api/***start_date*/*end_date*

GET /meior/api/v1.0/gspace/<gspace_id>/meteo_api/<start_date>/<end_date>

Returns a JSON that contains historical "current/read" weather for data between the requested dates.

Example response:

```
{
    <int:timestamp> : {
        "Reference_Time" : <int:unixtime>,
        "Detailed_Status" : <str>,
        "Temp[degC]" : <float>,
        "Feels_Like[degC]" : <float>,
        "Humidity[%]" : <int>,
        "Clouds[%]" : <float>,
        "Dewpoint[degC]" : <float>,
        "Precipitation[mm]": <float>,
        "Wind Speed[m/s]" : <float>,
        "Wind Dir[deg]" : <int>,
        "UV Index[0->11+]" : <float>,
```

```
        "Pressure[mbar]" : <int>,
        "Sunrise_Time" : <int:unixtime>,
        "Sunset_Time" : <int:unixtime>,
        "source" : <str>
    },
    <int:timestamp> : {...},
    ...
}
```

Status Codes

- 200 OK – no error
- 404 Not Found – not found

DELETE /meior/api/v1.0/gspace/(int: *gspace_id*)**/sensors/**

int: *sensor_id*/**string:** *security_token*

DELETE /meior/api/v1.0/gspace/<int:gspace_id>/

sensors/<int:sensor_id>/<string:security_token>

Deletes the selected Sensor. A security token is mandatory.

Example response:

```
{
    "deleted" : "True"
}
```

Status Codes

- 200 OK – no error
- 404 Not Found – not found

PATCH /meior/api/v1.0/gspace/(int: *gspace_id*)**/sensors/**

int: *sensor_id*/

PATCH /meior/api/v1.0/gspace/<int:gspace_id>/sensors/<int:sensor_id>/

Updates the selected sensor's register fields.

Request format:

form Payload

```
{
    "sensor_name" : <str:optional>,
    "zone_id" : <int:optional>,
    "unit_description" : <str:optional>,
    "sensor_description : <str:optional>
}
```

Example response:

```
{
    "sensor_id" : <int>,
    "zone_id" : <int>|None,
    "unit_description" : <str>,
    "sensor_description" : <str>|None
}
```

Status Codes

- 202 Accepted – no error
- 404 Not Found – not found

POST /meior/api/v1.0/gspace/(int: *gspace_id*)**/sensors/**

int: *sensor_id*

POST /meior/api/v1.0/gspace/<int:gspace_id>/sensors/<int:sensor_id>

Inserts a sensor's reading.

Request format:

form Payload

```
{
    "readings" : [
        { "value" : "<float:mandatory>",
          "timestamp" : <timestamp:mandatory>
        },
        {...}
    ]
}
```

Example response:

```
[
    {
        "value" : "<float:mandatory>",
        "timestamp" : <timestamp:mandatory>
    },
    {...}
]
```

Status Codes

- 201 Created – no error
- 404 Not Found – not found

GET /meior/api/v1.0/gspace/(int: *gspace_id*)/sensors/

int: *sensor_id*

GET /meior/api/v1.0/gspace/<int:gspace_id>/sensors/<int:sensor_id>

Returns a JSON with information and readings of the requested sensor.

Example response:

```
{
    "sensor_id" : <int>,
    "sensor_name" : <str>,
    "zone_id" : <int>,
```

```
    "unit_description" : <str>,
    "sensor_description" : <str>,
    "readings" : [
        {
            "value" : "<float:reading>",
            "timestamp" : <timestamp>
        },
        {...}
    ]
}
```

Status Codes

- 200 OK – no error
- 404 Not Found – not found

`POST /meior/api/v1.0/gspace/(int: gspace_id)/sensors`

`POST /meior/api/v1.0/gspace/<int:gspace_id>/sensors`

Creates a new sensor on the provided gspace_id and return its ID.

Request format:

form Payload

```
{
    "sensor_name" : <str:mandatory>,
    "zone_id" : <int:optional>,
    "unit_description" : <str:mandatory>,
    "sensor_description : <str:optional>
}
```

Example response:

```
{
    "sensor_id" : <int>
}
```

Status Codes

- 201 Created – no error
- 404 Not Found – not found

GET /meior/api/v1.0/gspace/(int: *gspace_id*)/sensors

GET /meior/api/v1.0/gspace/<int:gspace_id>/sensors

Returns a JSON that contains information and readings of all sensors.

Example response:

```
[
    {
        "sensor_id" : <int>,
        "sensor_name" : <str>,
        "grid_location" : <str>,
        "unit_description" : <str>,
        "sensor_description" : <str>,
        "readings" : [
            {
                "value" : "<float:reading>",
                "timestamp" : <timestamp>
            },
            {...}
        ]
    },
    {...}
]
```

Status Codes

- 200 OK – no error
- 404 Not Found – not found

DELETE /meior/api/v1.0/gspace/(int: *gspace_id*)/

string: *security_token*

DELETE /meior/api/v1.0/gspace/<int:gspace_id>/<string:security_token>

Deletes the selected Green Space and related data (CASCADE). A security token is mandatory.

Example response:

```
{
    "deleted" : "True"
}
```

Status Codes

- 200 OK – no error
- 404 Not Found – not found

PATCH /meior/api/v1.0/gspace/(int: *gspace_id*)

PATCH /meior/api/v1.0/gspace/<int:gspace_id>

Updates the registered fields for Green Space with :attr:gspace_id.

Request format:

form Payload

```
{
    "gspace_name" : <str:optional>,
    "latitude" : <str:optional>,
    "longitude" : <str:optional>,
    "address" : <str:optional>,
    "description" : <str:optional>,
    "zones" : [
        {
            "zone_id" : <int:mandatory>,
            "area" : <float:optional>,
            "Kc" : <float:optional>,
            "description" : <str:optional>,
        },
        {...}
```

```
    ] <list: optional>
}
```

Example response:

```
{
    "gspace_id" : <int>,
    "gspace_name" : <str>,
    "latitude" : <float>,
    "longitude" : <float>,
    "address" : <str>,
    "description" : <str>,
    "zones" : [
        {...},
        {...}
    ]
}
```

Status Codes

- 202 Accepted – no error
- 404 Not Found – not found

GET /meior/api/v1.0/gspace/(int: *gspace_id*)

GET /meior/api/v1.0/gspace/<int:gspace_id>

Returns a JSON with selected Green Space's info.

Example response:

```
{
    "gspace_id" : <int>,
    "gspace_name" : <str>,
    "latitude" : <float>,
    "longitude" : <float>,
    "address" : <str>,
    "description" : <str>,
    "zones" : [
```

```
        {...},
        {...}
    ]
}
```

Status Codes

- 200 OK – no error
- 404 Not Found – not found

C.1.2 Submodules

C.1.2.1 app.api_routing module

app.api_routing.**append_sensor_reading**(*gspace_id*, *sensor_id*, *valor*)

POST /meior/api/v1.0/gspace/<int:gspace_id>/sensors/<int:sensor_id>

Inserts a sensor's reading.

Request format:

form Payload

```
{
    "readings" : [
        {
            "value" : "<float:mandatory>",
            "timestamp" : <timestamp:mandatory>
        },
        {...}
    ]
}
```

Example response:

```
[
    {
```

```
        "value" : "<float:mandatory>",
        "timestamp" : <timestamp:mandatory>
    },
    {...}
]
```

Status Codes

- 201 Created – no error
- 404 Not Found – not found

app.api_routing.**create_gspace**()

POST /meior/api/v1.0/gspace

Creates a new Green Space.

Request format:

form Payload

```
{
    "gspace_name" : <str:mandatory>,
    "latitude" : <str:mandatory>,
    "longitude" : <str:mandatory>,
    "address" : <str:optional>,
    "description" : <str:optional>,
    "zones" : [
        {
            "zone_id" : <int:mandatory>,
            "area" : <float:optional>,
            "Kc" : <float:mandatory>,
            "description" : <str:optional>
        },
        {...}
    ]
}
```

Example response:

```
{
    "gspace_id" : <int>
}
```

Status Codes

- 201 Created – no error
- 404 Not Found – not found

app.api_routing.**create_sensor**(*gspace_id*)

POST /meior/api/v1.0/gspace/<int:gspace_id>/sensors

Creates a new sensor on the provided gspace_id and return its ID.

Request format:

form Payload

```
{
    "sensor_name" : <str:mandatory>,
    "zone_id" : <int:optional>,
    "unit_description" : <str:mandatory>,
    "sensor_description : <str:optional>
}
```

Example response:

```
{
    "sensor_id" : <int>
}
```

Status Codes

- 201 Created – no error
- 404 Not Found – not found

app.api_routing.**delete_gspace**(*gspace_id*, *security_token*)

DELETE /meior/api/v1.0/gspace/<int:gspace_id>/<string:security_token>

Deletes the selected Green Space and related data (CASCADE). A security token is mandatory.

Example response:

```
{
    "deleted" : "True"
}
```

Status Codes

- 200 OK – no error
- 404 Not Found – not found

app.api_routing.**delete_sensor**(*gspace_id*, *sensor_id*, *security_token*)

DELETE /meior/api/v1.0/gspace/<int:gspace_id>/
/sensors/<int:sensor_id>/<string:security_token>

Deletes the selected Sensor. A security token is mandatory.

Example response:

```
{
    "deleted" : "True"
}
```

Status Codes

- 200 OK – no error
- 404 Not Found – not found

app.api_routing.**documentation**()

app.api_routing.**get_ETO_between_dates**(*gspace_id*, *start_date*, *end_date*)

GET /meior/api/v1.0/gspace/<int:gspace_id>/

/ETO/FAO-56PM/<int:start_date>/<int:end_date> Returns ETO in mm between the requested (past) dates. Time needs to be provided in Unix time format.

Example response:

```
{
    <timestamp> : <float:ETO>,
    <timestamp> : <float:ETO>,
    <timestamp> : <float:ETO>
}
```

Status Codes

- 200 OK – no error
- 404 Not Found – not found

app.api_routing.**get_about**()

This endpoint returns more detailed information about the API version.

app.api_routing.**get_all_sensor_readings**(*gspace_id*)

GET /meior/api/v1.0/gspace/<int:gspace_id>/sensors

Returns a JSON that contains information and readings of all sensors.

Example response:

```
[
    {
        "sensor_id" : <int>,
        "sensor_name" : <str>,
        "grid_location" : <str>,
        "unit_description" : <str>,
        "sensor_description" : <str>,
        "readings" : [
            {
                "value" : "<float:reading>",
```

```
                "timestamp" : <timestamp>
            },
            {...}
        ]
    },
    {...}
]
```

Status Codes

- 200 OK – no error
- 404 Not Found – not found

app.api_routing.**get_gspace_info**(*gspace_id*)

GET /meior/api/v1.0/gspace/<int:gspace_id>

Returns a JSON with selected Green Space's info.

Example response:

```
{
    "gspace_id" : <int>,
    "gspace_name" : <str>,
    "latitude" : <float>,
    "longitude" : <float>,
    "address" : <str>,
    "description" : <str>,
    "zones" : [
        {...},
        {...}
    ]
}
```

Status Codes

- 200 OK – no error

- 404 Not Found – not found

app.api_routing.**get_gspaces_infos**(*gspace_id*)

GET /meior/api/v1.0/gspace/

Returns a JSON with a list of Green Spaces' infos.

Example response:

```
[
    {
        "gspace_id" : <int>,
        "gspace_name" : <str>,
        "latitude" : <float>,
        "longitude" : <float>,
        "address" : <str>,
        "description" : <str>,
        "zones" : [
            {...},
            {...}
        ]
    },
    {...}
]
```

Status Codes

- 200 OK – no error
- 404 Not Found – not found

app.api_routing.**get_irrigation_between_dates**(*gspace_id*, *start_date*, *end_date*)

GET /meior/api/v1.0/gspace/<int:gspace_id>/irrigation/<int:start_date>/<int:end_date>

Returns irrigation in mm between the requested (past) dates. Time needs to be provided in Unix time format.

Example response:

```
{
    <timestamp> : {
        <int:zone_id> : <float:irrigation>,
        <int:zone_id> : <float:irrigation>,
        ...
    },
    <timestamp> : {
        <int:zone_id> : <float:irrigation>,
        <int:zone_id> : <float:irrigation>,
        ...
    },
    ...
}
```

Status Codes

- 200 OK – no error
- 404 Not Found – not found

app.api_routing.**get_meteo_api_between_dates**(*gspace_id*, *start_date*, *end_date*)

GET /meior/api/v1.0/gspace/<gspace_id>/meteo_api/<start_date>/<end_date>

Returns a JSON that contains historical "current/read" weather for data between the requested dates.

Example response:

```
{
    <int:timestamp> : {
        "Reference_Time" : <int:unixtime>,
        "Detailed_Status" : <str>,
        "Temp[degC]" : <float>,
```

```
        "Feels_Like[degC]" : <float>,
        "Humidity[%]" : <int>,
        "Clouds[%]" : <float>,
        "Dewpoint[degC]" : <float>,
        "Precipitation[mm]": <float>,
        "Wind Speed[m/s]" : <float>,
        "Wind Dir[deg]" : <int>,
        "UV Index[0->11+]" : <float>,
        "Pressure[mbar]" : <int>,
        "Sunrise_Time" : <int:unixtime>,
        "Sunset_Time" : <int:unixtime>,
        "source" : <str>
    },
    <int:timestamp> : {...},
    ...
}
```

Status Codes

- 200 OK – no error
- 404 Not Found – not found

app.api_routing.**get_meteo_between_dates**(*gspace_id*, *start_date*, *end_date*)

GET /meior/api/v1.0/gspace/<int:gspace_id>/meteo_api/forecast/<int:start_date>/<int:end_date>

Returns a JSON with stored (past) weather forecasts between the requested dates. By definition stored weather forecasts are done everyday at 7 p.m. (local time). For each <int:timestamp> in the response an array of dictionaries is returned, one dictionary for each of the forecasted days.

Example response:

```
{
    <int:timestamp> : {
```

```
        <int:day_number> : {
            "Reference_Time" : <int:unixtime>,
            "Detailed_Status" : <str>,
            "Temp_Max[degC]" : <float>,
            "Temp_Min[degC]" : <float>,
            "Humidity[%]" : <int>,
            "Clouds[%]" : <float>,
            "Dewpoint[degC]" : <float>,
            "Precipitation[mm]": <float>,
            "Wind Speed[m/s]" : <float>,
            "Wind gust[m/s]" : <float>,
            "Wind Dir[deg]" : <int>,
            "UV Index[0->11+]" : <float>,
            "Pressure[mbar]" : <int>,
            "Sunrise_Time" : <int:unixtime>,
            "Sunset_Time" : <int:unixtime>,
            "source" : <str>
        },
        <int:day_number> : { ... },
        ...
    },
    <int:timestamp> : {...},
    ...
}
```

Example (for a single day):

```
{
    "1637841600" : {
        0: {
            "reference_time" : 1637841600,
            "Detailed_Status" : "clear sky",
            "Temp_Max[degC]" : 15.52,
            "Temp_Min[degC]" : 8.72,
            "Humidity[%]" : 44,
            "Clouds[%]" : 0,
```

```
            "Dewpoint[degC]" : 2.28,
            "Precipitation[mm]": 0,
            "Wind_Speed[m/s]" : 6.25,
            "Wind_gust[m/s]" : 10.35,
            "Wind_Dir[deg]" : 29,
            "UV_Index[0->11+]" : 3,
            "Pressure[mbar]" : 1016,
            "Sunrise_Time" : 1637824890,
            "Sunset_Time" : 1637860650,
            "source" : "OWM"
        },
        1: { ... }
    }
}
```

Status Codes

- 200 OK – no error
- 404 Not Found – not found

`app.api_routing.`**`get_sensor_readings`**(*gspace_id*, *sensor_id*)

`GET /meior/api/v1.0/gspace/<int:gspace_id>/sensors/<int:sensor_id>`

Returns a JSON with information and readings of the requested sensor.

Example response:

```
{
    "sensor_id" : <int>,
    "sensor_name" : <str>,
    "zone_id" : <int>,
    "unit_description" : <str>,
    "sensor_description" : <str>,
    "readings" : [
        {
            "value" : "<float:reading>",
            "timestamp" : <timestamp>
```

```
        },
        {...}
    ]
}
```

Status Codes

- 200 OK – no error
- 404 Not Found – not found

app.api_routing.**get_solar_radiation_between_dates**(*gspace_id*, *start_date*, *end_date*)

GET /meior/api/v1.0/gspace/<int:gspace_id>/solar_radiation/<int:start_date>/<int:end_date>

Returns a JSON with stored (past) solar radiation (prediction?) data in W/m^2 between the requested dates. Time needs to be provided in Unix Time format.

Example response:

```
{
    <timestamp> : <float>,
    <timestamp> : <float>,
    ...
}
```

Status Codes

- 200 OK – no error
- 404 Not Found – not found

app.api_routing.**get_weather_station_between_dates**(*gspace_id*, *start_date*, *end_date*)

GET /meior/api/v1.0/gspace/<gspace_id>/

weather_station/<start_date>/<end_date>

Returns a JSON that contains historical weather station data between the requested dates. Weather parameters depend on the used station. The example's structure is for Vale do Lobo weather station.

Example response:

```
{
    <int:timestamp> : {
        "Reference_Time" : <int:unixtime>
        "Date" : <str>,
        "Day" : <int>,
        "TempAvg[degC]" : <float>,
        "TempMin[degC]" : <float>,
        "TempMinTime" : <str>,
        "TempMax[degC]" : <float>,
        "TempMaxTime" : <str>,
        "DewpointAvg[degC]" : <float>,
        "DewpointMin[degC]" : <float>,
        "DewpointMax[degC]" : <float>,
        "HumididtyAvg[%]" : <float>,
        "HumidityMin[%]" : <float>,
        "HumidityMax[%]" : <float>,
        "PressureAvg[hPa]" : <float>,
        "PressureMin[hPa]" : <float>,
        "PressureMax[hPa]" : <float>,
        "SolarRadiationAVG[W/m^2]" : <float>,
        "SolarRadiationMax[W/m^2]" : <float>,
        "EvapotranspDaily[mm]" : <float>,
        "RainIntensity[mm/h]" : <float>,
        "Precipitation[mm]" : <float>,
        "WindAvg[Km/h]" : <float>,
        "WindMax[Km/h]" : <float>,
        "WindGust[Km/h]" : <float>,
```

```
        "WindDirection" : <str>,
        "source" : <str>
    },
        <int:timestamp> : {...},
        ...
}
```

Status Codes

- 200 OK – no error
- 404 Not Found – not found

app.api_routing.**not_found**(*error*)

app.api_routing.**predict_ETO**(*gspace_id*, *num_days*)

GET /meior/api/v1.0/gspace/<int:gspace_id>/ETO/FAO-56PM/predict/<int:num_days>

Returns ETO predictions in mm for the number of days requested, up to a maximum of 7 days. Day zero represents current day prediction.

Example response:

```
{
    <int:day_number> : <float:ETO_prediction>,
    <int:day_number> : <float:ETO_prediction>,
    ...
}
```

Status Codes

- 200 OK – no error
- 404 Not Found – not found

app.api_routing.**predict_irrigation**(*gspace_id*, *num_days*)

GET /meior/api/v1.0/gspace/<int:gspace_id>/irrigation/predict/<int:num_days>

Returns irrigation predictions in mm for the number of days requested, up to a maximum of 7 days. Day zero represents current day prediction.

Example response:

```
{
    <int:day_number> : {
        <int:zone_id> : <float:irrigation_prediction>,
        <int:zone_id> : <float:irrigation_prediction>,
        ...
    },
    <int:day_number> : {
        <int:zone_id> : <float:irrigation_prediction>,
        <int:zone_id> : <float:irrigation_prediction>,
        ...
    },
    ...
}
```

Status Codes

- 200 OK – no error
- 404 Not Found – not found

app.api_routing.**predict_meteo**(*gspace_id*, *num_days*)

GET /meior/api/v1.0/gspace/<int:gspace_id>/meteo_api/forecast/<int:num_days>

Returns a JSON that contains predicted meteorological data for the number of days requested, up to a maximum of 7 days. Day zero represents current day prediction.

Example response:

```
{
    <int:day_number> : {
        "Prediction_Date" : <int:unixtime>
        "Detailed_Status" : <str>
        "Temp_Max[degC]" : <float>
        "Temp_Min[degC]" : <float>
        "Humidity[%]" : <int>
        "Clouds[%]" : <float>
        "Dewpoint[degC]" : <float>
        "Precipitation[mm]": <float>
        "Wind Speed[m/s]" : <float>
        "Wind gust[m/s]" : <float>
        "Wind Dir[deg]" : <int>
        "UV Index[0->11+]" : <float>
        "Pressure[mbar]" : <int>
        "Sunrise_Time" : <int:unixtime>
        "Sunset_Time" : <int:unixtime>
    },
    <int:day_number> : {...},
    ...
}
```

Status Codes

- 200 OK – no error
- 404 Not Found – not found

app.api_routing.**predict_solar_radiation**(*gspace_id*, *num_days*)

GET /meior/api/v1.0/gspace/<int:gspace_id>/solar_radiation/predict/<int:num_days>

Returns a JSON with predicted solar radiation data in W/m^2 for the number of days requested, up to a maximum of 7 days. Day zero represents current day prediction.

Example response:

```
{
    <int:day_number> : <float>,
    <int:day_number> : <float>,
    ...
}
```

Status Codes

- 200 OK – no error
- 404 Not Found – not found

app.api_routing.**update_gspace**(*gspace_id*)

PATCH /meior/api/v1.0/gspace/<int:gspace_id>

Updates the registered fields for Green Space with :attr:gspace_id.

Request format:

form Payload

```
{
    "gspace_name" : <str:optional>,
    "latitude" : <str:optional>,
    "longitude" : <str:optional>,
    "address" : <str:optional>,
    "description" : <str:optional>,
    "zones" : [
        {
            "zone_id" : <int:mandatory>,
            "area" : <float:optional>,
            "Kc" : <float:optional>,
            "description" : <str:optional>,
        },

    ]
    <list: optional>
```

```
}
```

Example response:

```
{
    "gspace_id" : <int>,
    "gspace_name" : <str>,
    "latitude" : <float>,
    "longitude" : <float>,
    "address" : <str>,
    "description" : <str>,
    "zones" : [
        {...},
        {...}
    ]
}
```

Status Codes

- 202 Accepted – no error
- 404 Not Found – not found

app.api_routing.**update_sensor**(*gspace_id*, *sensor_id*)

PATCH /meior/api/v1.0/gspace/<int:gspace_id>/sensors/<int:sensor_id>/

Updates the selected sensor's register fields.

Request format:

form Payload

```
{
    "gspace_id" : <int>,
    "gspace_name" : <str>,
    "latitude" : <float>,
    "longitude" : <float>,
    "address" : <str>,
    "description" : <str>,
```

```
    "zones" : [
        {...},
        {...}
    ]
}
```

Example response:

```
{
    "sensor_name" : <str:optional>,
    "zone_id" : <int:optional>,
    "unit_description" : <str:optional>,
    "sensor_description" : <str:optional>
}
```

Status Codes

- 202 Accepted – no error
- 404 Not Found – not found