Supporting Information for "Wildfire Danger Prediction and Understanding with Deep Learning"

Spyros Kondylatos[1,2,*], Ioannis Prapas[1,2,*], Michele Ronco[2], Ioannis Papoutsis[1], Gustau Camps-Valls[2], María Piles[2], Miguel-Ángel Fernández-Torres[2], Nuno Carvalhais [3,4]

[1] Institute for Astronomy, Astrophysics, Space Applications and Remote Sensing, National Observatory of Athens, Greece

[2]Image Processing Laboratory (IPL), Universitat de València, Spain

[3]Max Planck Institute for Biogeochemistry, Jena, Germany

[4]Departamento de Ciências e Engenharia do Ambiente, DCEA, Faculdade de Ciências e Tecnologia, FCT, Universidade Nova de Lisboa, Caparica, Portugal

[*]Authors contributed equally

Contents of this file

Introduction

This provides supplementary information for the main manuscript "Wildfire Danger Prediction and Understanding with Deep Learning". The first section presents a list with all the variables of the datacube as well as the details for all the pre-processing done. The second section provides the details about the architectures of the different models and the hyperparameters used in the experiments. The third section describes the different Explainability Artificial Intelligence (xAI) techniques used. The fourth is about the metrics used for evaluating the models and the fifth section describes the Fire Weather Index (FWI). The three figure sections provide figures regarding some supplementary analytics of the different fire drivers, more fire danger maps produced by the different models and xAI plots for all the variables.

Text S1. Data Extraction and Processing

To create the final datacube, we gathered the following data:

• **Daily weather data** from ERA-5 Land (Muñoz-Sabater et al., 2021), for 6 hours (00:00, 04:00, 08:00, 12:00, 16:00, 20:00) of 2 m temperature, 10 m wind *u*-component, 10 m wind *v*-component, total precipitation, 2 m dewpoint temperature and surface pressure available in 9 km spatial resolution.

• **Satellite variables** from MODIS downloaded from Nasa's portal `https://modis.gsfc.nasa.gov/data/`, including Leaf Area Index (LAI) and Fraction of Photosynthetically Active Radiation (Fpar), available in a 8-daily temporal and 500 m spatial resolution; Normalized Difference Vegetation Index (NDVI) and Enhanced Vegetation Index (EVI), available in a 16-daily temporal and 500 m spatial resolution; Evapotranspiration (ET) available in a 8-daily temporal and 500 m spatial resolution and Day/Night Land Surface Temperature (LST), available in a daily temporal and 1 km spatial resolution.

• **Soil moisture index & Anomaly** from EDO (European Drought Observatory) (Cammalleri et al., 2017) available in a 10-daily temporal and 5 km spatial resolution.

• **Roads distance** extracted from WorldPop (Tatem, 2017) at 1 km spatial resolution.

• **Waterway distance** extracted from WorldPop at 1 km spatial resolution.

• **Yearly population density** extracted from WorldPop at 1 km spatial resolution.

• **Land cover** extracted from Copernicus Corine Land Cover (CLC) (Büttner, 2014) for years 2006, 2012 and 2018, at 100 m spatial resolution.

• **Elevation** from Copernicus EU-DEM (Bashfield & Keim, 2011) at 30 m spatial resolution.

• **Daily Fire Weather Index** from Copernicus `https://cds.climate.copernicus.eu/cdsapp#!/dataset/cems-fire-historical?tab=overview` at 25km spatial resolution.

• **Historical burned areas** from EFFIS (San-Miguel-Ayanz et al., 2013) containing burned areas larger than 30 hectares (ha). The burned areas are intersected with the MODIS active fires product (Giglio et al., 2016) to recover the start date of the fire.

After, we post-process the gathered data in order to create a 1 km $\times$ 1 km $\times$ 1 day resolution datacube, that contains variables related to fire drivers as well as the output for the problem.

- **Weather Data.** We combine 2 m dewpoint temperature (DT) and 2 m temperature (T), to calculate relative humidity using the following equation:

$$100 * \frac{\exp \frac{17.625*DT}{243.04+DT}}{\exp \frac{17.625*T}{243.04+T}} \tag{1}$$

We also combine 10 m wind u-component and 10 m wind v-component to compute wind speed and direction. Then, we calculate the maximum, minimum and mean daily value for all the variables (2 m temperature, 10 m wind u-component, 10 m wind v-component, total precipitation, 2 m dewpoint temperature, surface pressure, relative humidity) based on the hourly values. For the wind, we also produce separate daily values for 10 m wind u-component, 10 m wind v-component, wind speed and direction calculated at the time of the maximum wind speed. Moreover, as land pixels near the sea have no values, because of the low native spatial resolution of the ERA-5 Land variables, we use nearest interpolation to fill these spatial gaps. Finally, we use nearest interpolation to map the 9 km spatial resolution to 1 km spatial resolution. We end up with **25** variables related to weather.

- **LAI, Fpar, NDVI, EVI, ET**. We use nearest interpolation to map these variables to 1 km spatial resolution. Moreover, we forward-fill the values in time, to fill the temporal gaps and let the variables have daily temporal resolution. These variables, together with LST Day and Night are the **7** variables from MODIS.

• **Soil Moisture & Anomaly**. We use nearest interpolation to map these variables to 1 km spatial resolution and we forward-fill them in time, to fill the temporal gaps. We end up with **2** variables.

• **Roads Distance** is **1** static variable that has no time dimension. No pre-processing needed.

• **Waterway Distance** is **1** static variable that has no time dimension. No pre-processing needed.

• **Yearly population density**. We collect each year's population density covering years 2009-2020. As year 2021 is still not available, we use year's 2020 value as a proxy for year 2021. We end up with **13** static (no time dimension) variables, each one depicting each year's population density.

• **Land Cover**. CLC comes in 100 m spatial resolution. In order to downscale it to 1 km spatial resolution, we use two separate approaches. First, we compute the majority class between all the spatially resolved 100 m pixels lying in the 1 km spatially resolved grid cell. Second, we create 10 variables, each one related to one out of 10 classes of interest, which are based on the predefined subclasses of Corine: **0 (Artificial surfaces)** : continuous urban fabric, industrial or commercial units, road and rail networks and associated land, port areas, airports, mineral extraction sites, dump sites, construction sites, green urban areas, sport and leisure facilities

1 (Discontinuous urban fabric): discontinuous urban fabric **2 (Arable land)** : non-irrigated arable land, permanently irrigated land, rice fields

3 (Permanent crops): vineyards, fruit trees and berry plantations

4 (Pastures): pastures

5 (General agriculture): annual crops associated with permanent crops, complex cultivation patterns, land principally occupied by agriculture with significant areas of natural vegetation, agro-forestry areas

6 (Forest): broad-leaved forest, coniferous forest, mixed forest

7 (Miscellaneous vegetation): natural grassland, moors and heathland, sclerophyllous vegetation, transitional woodland/shrub

8 (Miscellaneous No vegetation): beaches/dunes/sand, bare rock, sparsely vegetated areas, burnt areas, glaciers and perpetual snow

9 (Water): inland marshes, peatbogs, salt marshes, salines, intertidal flats, water courses, water bodies, coastal lagoons, estuaries, sea and ocean

In each 1 km spatially resolved cell, each of these variables has a value between 0 and 1, which represents the fraction of the class presence in the pixel. We repeat this process for each year that CLC is available. Thus, we end up with **33** static variables regarding CLC.

- **Elevation, Slope, Aspect, Roughness**. After gathering the elevation, we upscale it to 1 km spatial resolution by using mean aggregation. After having the 1 km spatially resolved elevation at hand, we calculate slope, aspect and roughness at 1 km spatial resolution, using GDAL `https://gdal.org/`. We end up with **4** topography variables.

- **FWI**. We use nearest interpolation to map the **1** value of FWI in a 1 km spatial resolution.

- **Burned areas.** The gathered shapefiles representing final burned areas, produced by different ignitions need some pre-processing to identify the date of the ignition of the fire and to combine several burned areas, that are produced by the same fire event, into one. To this end, we associate burned areas product to the active fires product. In order to identify the date of the ignition, we create a 1 km buffer around the burned areas shapefiles and we look for an active fire inside this area that has been produced within a week before the date that EFFIS provides for the burned area. Moreover, to combine several burned areas into one, we also use a buffer of 1 km and we do the following: We use as an anchor a burned area and we calculate its ignition point following the procedure above. If this ignition point is also lying inside buffered burned areas other than the anchor one and these burned areas have a date within a week before the anchor one, we consider that they belong to the same event as the anchor one. Having the ignition points and burned areas at hand, we rasterize them in the same 1 km spatial grid of all the input variables. We also produce a temporal variable that has no latitude or longitude dimension, that contains the number of fires that ignited daily. We end up with **3** variables related to fire events.

We end up with a total number of **90** variables.

The goal is to use input data available for the previous day of the prediction and associate it with the daily burned areas. We shift backward the variables related to fire events (burned areas, ignition points, number of fires, FWI). Moreover, we also shift backward the meteorological data, as we assume that will be available through forecasts for the day of prediction.

Text S2. Deep Learning Architectures and Hyperparameters

As stated in the main manuscript we leveraged a Random Forest (RF) classifier (Breiman, 2001), an XGBoost classifier (Chen & Guestrin, 2016), a Long-Short Term Memory Neural Network (Hochreiter & Schmidhuber, 1997) and Convolutional Long-Short Term Memory Neural Network (Shi et al., 2015) for treating the next day's fire danger forecasting problem.

Random Forest (RF) is a supervised ML algorithm, which consist of an ensemble of a large number of individual decision trees. We use RF as an ML baseline for comparison with more sophisticated DL architectures.

Extreme Gradient Boosting (XGBoost) is also a supervised decision tree ensemble learning algorithm. The difference between RF and XGBoost lies in how the different trees are built and combine. Gradient boosting comes from the idea of boosting a single weak model by combining it with a number of other weak models in order to generate a final stronger model. Thus, Gradient boosting decision trees iteratively train an ensemble of shallow decision trees, with each iteration using the error residuals of the previous model to fit the next model. The final prediction is a weighted sum of all of the tree predictions. XGBoost is a scalable and highly accurate implementation of gradient boosting that pushes the limits of computing power for boosted tree algorithms.

LSTM is a type of Recurrent Neural Networks, which are capable of modelling sequences of data. LSTM is constructed by a memory cell, an input gate, an output gate and a forget gate. The memory cell is responsible for remembering values over time intervals, while the other gates manipulate the flow of information of the cell. As fire danger forecasting is a problem where values of the variables in the past contribute to the final prediction, LSTM is exploited for its ability to model these temporal dependencies.

For the spatio-temporal dataset we train a Convolutional Long-Short Term Memory (ConvLSTM) Neural Network. ConvLSTM is a type of Neural Networks used for modelling sequences of images. A ConvLSTM layer is actually a Recurrent layer, but the internal matrix multiplications are replaced by convolution operations (Krizhevsky et al., 2012). ConvLSTM is used for its ability to model both the temporal and spatial dependencies between the input variables.

The hyperparameters of the RF model are chosen based on the best F1-score in the validation set. The final RF consists of 100 trees, with a maximum depth of 10, the minimum number of samples required to split is 2 and the minimum number of samples required to be at a leaf node is 1.

The hyperparameters of XGBoost are chosen based on the best F1-score in the validation set. The final XGBoost consists of 500 trees, with a maximum depth of 1, minimum split loss 0, learning rate set to 0.3, minimum sum of instance weight (hessian) needed in a child 2 and subsample ratio of columns when constructing each tree set to 0.5.

For the Deep Learning (DL) models, all linear layers, but the last, are followed by a dropout with probability $p = 0.5$ and the ReLU activation function. The data is min-max scaled before serving as input. We fill with the temporal aggregate of the time-series the nulls existing in the inputs of the LSTM, while the nulls existing in the inputs of the ConvLSTM are filled with a spatial average of the relative day in the time-series. If the value still remains null, we fill it with -1. The DL models are trained for 30 epochs with the binary cross-entropy loss with ℓ_2-norm regularization, the Adam optimizer and a batch size of 256.

Regarding the LSTM's architecture, a normalization layer is followed by a an LSTM layer with 64 neurons, which is followed by two linear hidden layers with 32 and 16 neurons, respectively, and an output 2-class softmax layer. The final model is trained with a 0.001 weight decay and 0.0025 learning rate, that is divided by 10 every 15 epochs.

Regarding the spatio-temporal dataset, a normalization layer is followed by a ConvLSTM layer with 32 filters and 3×3 kernels. This is followed by a convolutional layer with 32 filters and 3×3 kernels, padding of 1, and stride of 1 and a 2×2 max-pooling layer. Then two linear layers are added with 64 and 32 neurons, respectively, before the final 2-class softmax layer. The model is trained with a 0.01 weight decay and 0.0001 learning rate, that is divided by 10 after 15 epochs.

Text S3. Evaluation Metrics

For classification tasks, a prediction is defined as true positive (TP) when both the predicted label and ground truth are positive; as true negative (TN) when both the predicted label and ground truth are negative; as false positive (FP) when the predicted label is positive but the ground truth is negative; as false negative (FN) when the predicted label is negative but the ground truth is positive. In our case, a true positive means that the pixel of interest was burnt the given day and the prediction of the model was that it will get burnt. True negative means that the pixel of interest was not burnt the given day and the prediction of the model was that it will not get burnt. False positive means that the pixel of interest was not burnt the given day, but the prediction of the model was that it will get burnt. False negative means that the pixel of interest was burnt the given day, but the prediction of the model was that it will not get burnt.

Precision is the ratio between true positives and all predicted positives (true positives + false positives). That means that it measures how many of the pixels that predicted by the model that will get burnt, actually got burnt. Mathematically:

$$\frac{TP}{TP + FP} \tag{2}$$

Recall is the ratio between true positives and all actual positives (true positives + false negatives). That means that it measures how many of the pixels that actually got burnt burnt were predicted correctly by the model. Mathematically:

$$\frac{TP}{TP + FN} \tag{3}$$

F1-score is the harmonic mean between precision and recall and is used in problems where both of them are important, as in our case. Mathematically F1-score can be expressed as:

$$2 * \frac{Precision * Recall}{Precision + Recall} \tag{4}$$

When it comes to classification problems, we can measure the performance of the algorithm with the Receiver Operating Characteristics (ROC) Curve and the Area Under the Receiver Operating Characteristics Curve (AUROC).

ROC curve is created by plotting the true positive rate (TPR) against the false positive rate (FPR) at various threshold settings. The true positive rate is also known as sensitivity, recall (see S2). The false positive rate is also known as probability of false alarm and can be calculated as:

$$\frac{TN}{TN + FP} \tag{5}$$

In our case, we plot the ROC Curve for all ML models and FWI. So we compute the Recall and false positive rate for various thresholds and present them in a ROC curve.

AUROC represents the degree or measure of separability between two classes. It gives us information on how well the model can distinguish between the classes. Higher the AUROC, the better the model is at predicting 0 classes as 0 and 1 classes as 1. An excellent model has AUC near to the 1 which means it has a good measure of separability. A poor model has an AUC near 0 which means it has the worst measure of separability. And when AUC is 0.5, it means the model has no class separation capacity whatsoever. In our case, the higher the AUROC, the better the model is at giving higher values to the cells that burnt and lower values to the cells that did not burn.

Text S4. Explainability Methods

Machine learning models are becoming very difficult to understand. Given that, many methods exist in the field of XAI under different learning paradigms. Here we briefly describe the theory behind the methods we used to interpret the predictions of the models. The xAI computations for SHAP and IG were carried out using the pytorch package *captum* (Kokhlikyan et al., 2020), while for PDPs we directly coded a suitable version of the algorithm.

Shapley Values (SHAP). Shapley Value sampling theory solves the problem of attributing the output of an arbitrary machine learning model to its input features

(Lundberg & Lee, 2017). The Shapley value is a solution concept from cooperative game theory, which has found numerous applications in machine learning.

In XAI, the Shapley value is used to measure the contributions of input features to the output of a machine learning model at the instance level. Given a specific data point, the goal is to decompose the model prediction and assign Shapley values to individual features of the instance. Let $\mathcal{N} = \{1, \ldots, n\}$ be the finite set of players (or covariates in our case), each non-empty subset $\mathcal{S} \subseteq \mathcal{N}$ is called a *coalition* (optimal set of covariates) and $\mathcal{N}$ itself the *grand coalition.* A transferable utility (TU) game is defined by the pair $(\mathcal{N}, v)$ where $v : 2^{\mathcal{N}} \to \mathbb{R}$ is a mapping called the *characteristic function* or the *coalition function* of the game assigning a real number to each coalition and satisfying $v(\emptyset) = 0$. The Shapley value is a single-valued solution concept for cooperative games. The i^{th} component of the single solution vector satisfying this solution concept for any cooperative game $(\mathcal{N}, v)$ is given by

$$\phi_i^{Sh} = \frac{1}{|\Pi(\mathcal{N})|} \sum_{\pi \in \Pi(\mathcal{N})} \underbrace{[v(\mathcal{P}_i^\pi \cup \{i\}) - v(\mathcal{P}_i^\pi)]}, \tag{6}$$

that is, the average of all player's marginal contribution in permutation π. Therefore, the Shapley value of a player is the average marginal contribution of the player to the value of the predecessor set over every possible permutation of the player set.

The Shapley values of feature values are explanatory attributions to the input features. The problem is that the computation of Shapley values requires an exponential number of characteristic function evaluations (i.e. number of machine learning models to train), resulting in exponential time complexity. This is prohibitive in a machine learning context when each evaluation can correspond to training a machine learning model. For this reason, a wide variety of Shapley value approximations have been proposed. We denote with $\widehat{\phi}_i^{Sh}$ an approximated Shapley value for player $i \in \mathcal{N}$. The pioneering Shapley value-based universal explanation method SHAP (Lundberg & Lee, 2017) used in this work proposes a linear time approximation of the Shapley values. Their main insight is the computation of Shapley values by approximately solving an optimization problem:

$$w_{\mathcal{S}} = \frac{|\mathcal{N}| - 1}{\binom{|\mathcal{N}|}{|\mathcal{S}|} |\mathcal{S}|(|\mathcal{N}| - |\mathcal{S}|)}, \quad min_{\widehat{\phi}_0^{Sh},\dots,\widehat{\phi}_n^{Sh}} \sum_{\mathcal{S} \subseteq \mathcal{N}} w_{\mathcal{S}} \left(\widehat{\phi}_0^{Sh} + \textstyle\sum_{i \in \mathcal{S}} \widehat{\phi}_i^{Sh} - v(\mathcal{S}) \right) \tag{7}$$

$$s.t.\ \widehat{\phi}_0^{Sh} = v(\emptyset), \quad \widehat{\phi}_0^{Sh} + \textstyle\sum_{i \in \mathcal{N}} \widehat{\phi}_i^{Sh} = v(\mathcal{N}). \tag{8}$$

where $\mathcal{N} = \{1, \dots, n\}$ be the finite set of *players* (covariates), each non-empty subset $\mathcal{S} \subseteq \mathcal{N}$ is a *coalition* (optimal set of covariates), $\mathcal{N}$ itself the *grand coalition*, and $\phi(\mathcal{N}, v) \in \mathbb{R}^{\mathcal{N}}$ is the *solution vector* to the cooperative game $(\mathcal{N}, v)$.

The definition of weights in 7 and the objective function implies the evaluation of $v(\cdot)$ for 2^n coalitions. This complex problem has been addressed by subsampling the coalitions (Lundberg & Lee, 2017). Note that $w_{\mathcal{S}}$ is higher when coalitions are large or small.

The above method is implemented in captum package (Lundberg & Lee, 2017; Kokhlikyan et al., 2020), which we used to explain and interpret the predicted fire danger. To compute the SHAP each input is masked through the time dimension and, thus, the whole 10-day time series is considered to be a single variable to whom we associate a SHAP value. The neutral or baseline instance is computed by averaging the values of all the negative (i.e. non-burned) pixels per channel in the train set. The importance of the variables is calculated based on the average of the absolute values of the Shapley and the variables are ranked in order of decreasing importance. The SHAP values are have been produced for all covariates for all fire events in the test set. Additional SHAP are shown in Fig. S3 (a).

Partial Dependency Plots (PDPs). Machine learning models typically take a high dimensional vector as input. Thus, it becomes difficult to disentangle the contribution of each predictor and understand how changes in a given covariate affect the predicted output. A way to estimate the dependence of the model on each input feature separately consists in calculating the marginal learned distribution (Molnar et al., 2020). If the model predicts $\widehat{y}_t = \widehat{p}(E_t|x_{t'<t}) = F(x_{t'<t})$ where $x_{t'<t} = (x^1_{t'<t}, x^2_{t'<t}, \ldots, x^d_{t'<t})$, being d the number of features, then the PDP of a given covariate $x^k_{t'<t}$ is:

$$PDP(x^k_{t'<t}) = \int \widehat{p}(E_t|x_{t'<t})\mathrm{dP}(X_{s\neq k}) \simeq \frac{1}{n}\sum_{i=1}^{N} F(x^k_{t'<t}, x^{s\neq k}_{i,t'<t}) \tag{9}$$

where the integral is over the distribution $\mathrm{P}(X_{s\neq k})$ of all the other features except k, and it is approximated by averaging over a finite number of samples. Repeating the process for all the d covariates one obtains as much univariate approximations of the model as the number of features. Then, the PDPs tell us whether the relationship between $\widehat{y}_t$ and a feature $x^k_{t'<t}$ is linear or more complex. They also serve as a first guidance for understanding how the model behaves in different regimes, i.e. when we vary the input feature values. A main drawback of this approach is that it is rigorously valid only if all the covariates are independent. In presence of correlations the PDP as defined above simply ignores all the interactions. More refined versions that correct for this behaviour have

been introduced but they are typically much more costly from a computational point of view (see however (Molnar et al., 2020) and references therein). In all plots we use time-average values of the covariates on the x-axis in order to obtain a two-dimensional projection of the dependency. For estimating the PDPs we used a random subsample of 1500 points extracted from the test set including both fire and non-fire events. Additional PDPs are shown in Fig. S3 (b).

Integrated Gradients (IGs). Given the fact that DL models are differentiable, many XAI techniques are based on the gradients of the model with respect to the input. The main problem of using directly vanilla gradients as an estimate of feature importance comes from the fact that the activation functions of the models can easily saturate and once flat regions are reached then the gradients go to zero even for relevant inputs. To correct for this and other misbehaviour and to impose specific desirable properties of feature attributions, a cumulative gradient estimation has been introduced (Sundararajan et al., 2017):

$$IG(x^k_{t'<t}) = (x^k_{t'<t} - \overline{x}^k_{t'<t}) \times \int_0^1 \frac{\partial F[x + \alpha(x - \overline{x})]}{\partial x^k_{t'<t}} \tag{10}$$

$$\simeq (x^k_{t'<t} - \overline{x}^k_{t'<t}) \times \sum_1^n \frac{1}{n} \times \frac{\partial F[x + \frac{k}{n}(x - \overline{x})]}{\partial x^k_{t'<t}} \tag{11}$$

with $k = 1, \ldots, n$ abd where $\overline{x}$ represents a baseline input, generally the instance for which the model is least active. Besides obeying sensitivity to input feature changes and implementation invariance, IGs also satisfy the following completeness property:

$$F(x_{t'<t}) - F(\overline{x}_{t'<t}) = \sum_{s=1}^{d} IG(x^s_{t'<t}) \tag{12}$$

which tells us that the attributions as given by IGs equal the difference between the output of the model at x and the output at the baseline $\overline{x}$. Given the above formulas, we can get an IG per feature per time step and thus have additional information on the temporal evolution of the importance of each covariate. This gives us additional information we could not immediately obtain through SHAP or PDPs. In all plots we show the average and standard deviation for all the fire events in the test set. We show the remaining IGs in Fig. S3 (c).

Text S5. Fire Weather Index

FWI is an empirical index that relies on meteorological forecasts (temperature, relative humidity, wind speed and 24-hour total precipitation) to derive fire danger predictions. It consists of different components that account for the effects of fuel moisture and wind on fire ignition and spread. The final outcome is the mapping of fire danger in 6 classes from very low to extreme, with extreme fire danger represented by values larger than 50 in Europe.

FWI consists of six components. The three primary components (Fine Fuel Moisture Code, Duff Moisture Code and Drought Code) depend on temperature, total precipitation and relative humidity and follow daily the moisture contents of forest fuel. The three moisture codes plus the wind are linked in pairs to form the two intermediate components (Initial Spread Index and Adjusted Duff Moisture Code). These two intermediate layers represent the rate of spread and amount of available fuel to be burnt. Finally, the last component (Fire Weather Index) is a combination of Initial Spread Index and Adjusted Duff Moisture Code and represents the intensity of the spreading fire. Each of these components are calculated with simple mathematical equations.

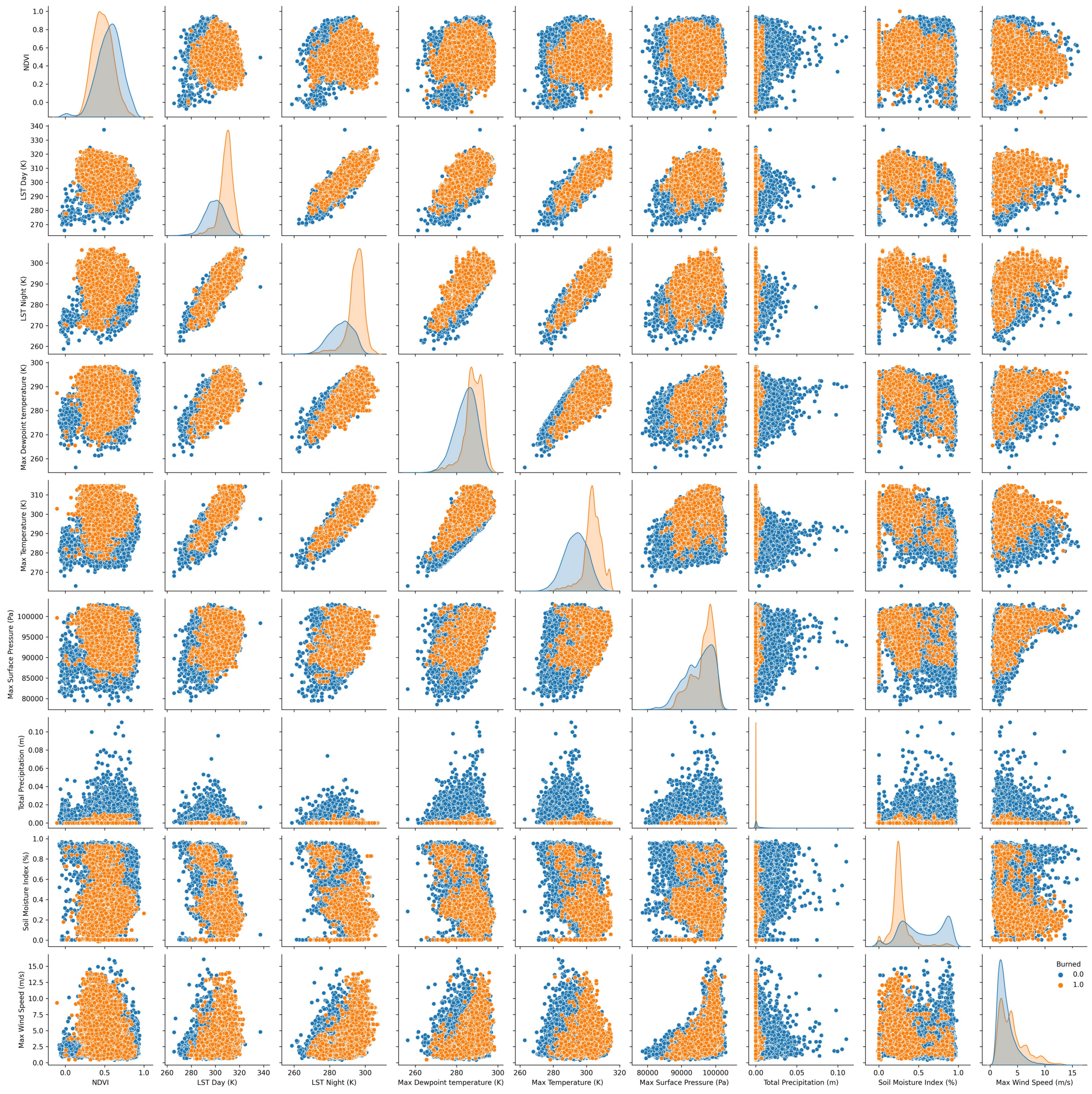

Figure S1. Interactions between the values of the variables of the test set. Each cell, except for the diagonal ones, represent the scatter plots between each pair of all the dynamic attributes. The cells in the diagonal represent the distributions of the variables, which are also presented in the main manuscript. Positives (burned pixels) are coloured in orange, while negatives (non-burned pixels) are coloured in green.

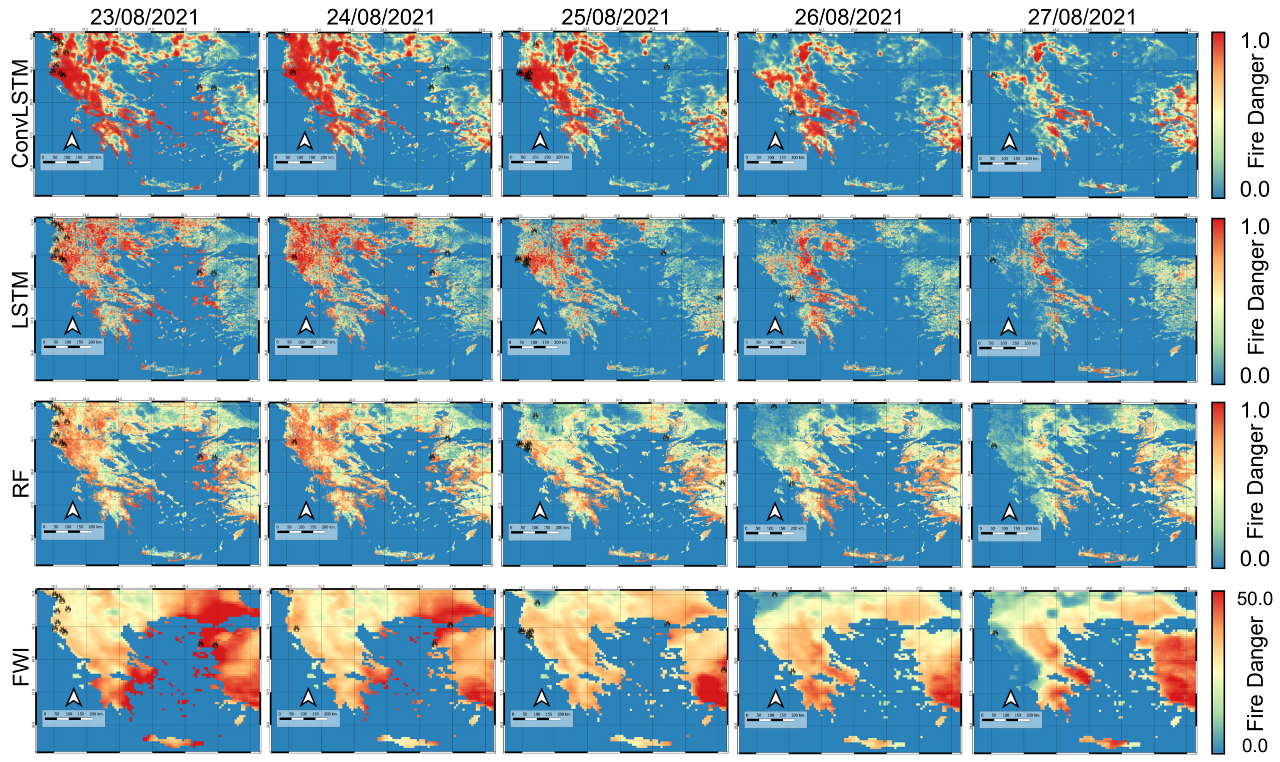

Figure S2. Maps produced by the four different models used in the main text (ConvLSTM, LSTM, RF, FWI) for 5 consecutive days of August 2021.

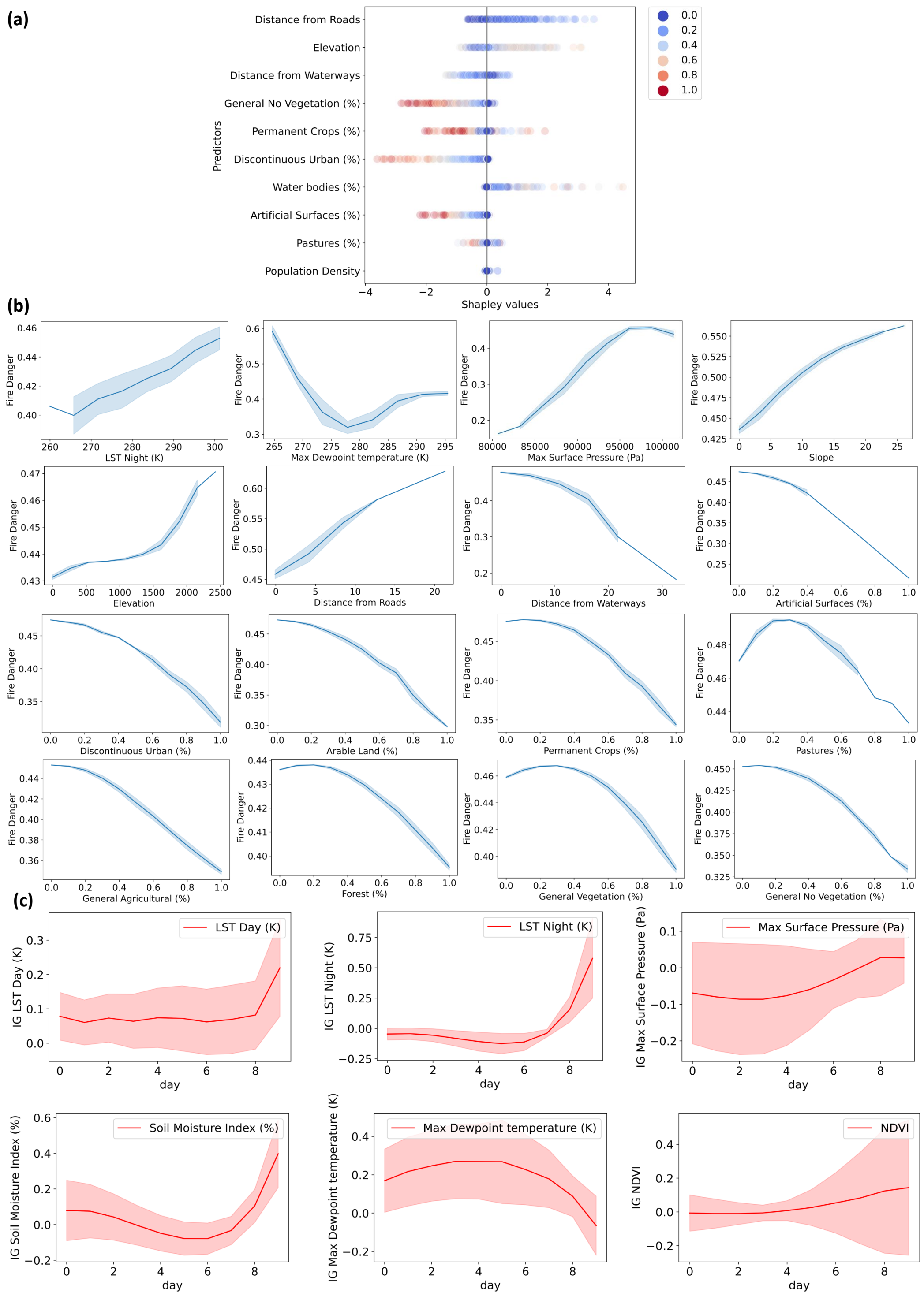

Figure S3. a) SHAP for variables missing from main text plot. Dots are coloured based on the scaled value of the relevant variable. b) PDP line plots (mean and standard deviation) for variables missing from main text. Values on the x axis are averaged over time and reported in the original units. c) The 10-day evolution of IGs (mean and standard deviation) over all burned pixels.

References

Bashfield, A., & Keim, A. (2011). Continent-wide DEM creation for the european union. In *34th international symposium on remote sensing of environment. the GEOSS era: Towards operational environmental monitoring. sydney, australia* (pp. 10–15). Citeseer.

Breiman, L. (2001, October). Random Forests. *Machine Learning*, *45*(1), 5–32. Retrieved 2021-09-16, from `https://doi.org/10.1023/A:1010933404324` doi: 10.1023/A:1010933404324

Büttner, G. (2014). Corine land cover and land cover change products. In *Land use and land cover mapping in europe* (pp. 55–74). Springer.

Cammalleri, C., Vogt, J. V., Bisselink, B., & de Roo, A. (2017, December). Comparing soil moisture anomalies from multiple independent sources over different regions across the globe. *Hydrology and Earth System Sciences*, *21*(12), 6329–6343. Retrieved 2022-04-20, from `https://hess.copernicus.org/articles/21/6329/2017/hess-21-6329-2017.html` (Publisher: Copernicus GmbH) doi: 10.5194/hess-21-6329-2017

Chen, T., & Guestrin, C. (2016). XGBoost: A scalable tree boosting system. In *Proceedings of the 22nd acm sigkdd international conference on knowledge discovery and data mining* (pp. 785–794). New York, NY, USA: ACM. Retrieved from `http://doi.acm.org/10.1145/2939672.2939785` doi: 10.1145/2939672.2939785

Giglio, L., Schroeder, W., & Justice, C. O. (2016). The collection 6 MODIS active fire detection algorithm and fire products. *Remote Sensing of Environment*, *178*, 31–41.

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

Kokhlikyan, N., Miglani, V., Martin, M., Wang, E., Alsallakh, B., Reynolds, J., ... Reblitz-Richardson, O. (2020). *Captum: A unified and generic model interpretability library for pytorch.*

Krizhevsky, A., Sutskever, I., & Hinton, G. E. (2012). ImageNet Classification with Deep Convolutional Neural Networks. In *Advances in Neural Information Processing Systems* (Vol. 25). Curran Associates, Inc. Retrieved 2021-09-16, from `https://papers.nips.cc/paper/2012/hash/c399862d3b9d6b76c8436e924a68c45b-Abstract.html`

Lundberg, S. M., & Lee, S.-I. (2017). A Unified Approach to Interpreting Model Predictions. In *Proceedings of the 31st International Conference on Neural Information Processing Systems* (pp. 4768–4777).

Molnar, C., Casalicchio, G., & Bischl, B. (2020). Interpretable Machine Learning–A Brief History, State-of-the-Art and Challenges. *arXiv preprint arXiv:2010.09337*.

Muñoz-Sabater, J., Dutra, E., Agustí-Panareda, A., Albergel, C., Arduini, G., Balsamo, G., ... others (2021). Era5-land: A state-of-the-art global reanalysis dataset for land applications. *Earth System Science Data Discussions*, 1–50.

San-Miguel-Ayanz, J., Schulte, E., Schmuck, G., & Camia, A. (2013). The european forest fire information system in the context of environmental policies of the european union. *Forest Policy and Economics*, *29*, 19-25. doi: https://doi.org/10.1016/j.forpol.2011.08.012

Shi, X., Chen, Z., Wang, H., Yeung, D.-Y., Wong, W.-K., & Woo, W.-c. (2015). Convolutional LSTM network: A machine learning approach for precipitation nowcasting. *Advances in neural information processing systems*, *28*, 802–810.

Sundararajan, M., Taly, A., & Yan, Q. (2017). Axiomatic attribution for deep networks. In

Proceedings of the 34th International Conference on Machine Learning (p. 3319–3328).

Tatem, A. J. (2017). Worldpop, open data for spatial demography. *Scientific data*, *4*(1), 1–4.