PREDICTING FOREST FIRES
A guide to using machine learning models to predict forest fires and understand their causes

1 Introduction

This report will examine how best to predict forest fires, based on the Forest Fires data set from the UCI machine learning repository. It contains 517 instances of the area of forest fires, along with a selection of 12 explanatory features.

Naturally, as with any natural disaster, being able to predict areas where forest fires are likely to break out and identifying which variables contribute to an increased risk of forest fire would allow resources to be spent at the right times and locations, potentially saving lives and nature. In the following sections, this report explores the dataset and its variables, introduces the machine learning models used to predict forest fires, and discusses the most important findings and takeaways.

2 Data & exploration

The data was collected at the Montesinho Natural Park, located at the northern border of Portugal. The target variable `area` ranges from 0.0 to 1090.84 and measures the area of the fire in hectares (ha). The variables `X` and `Y` range from 1 to 9 and 2 to 9, respectively, and locate the fire on the Montesinho Natural Park map, as depicted in figure 1 below. The fires are located with some added jitter in the middle of the coordinate squares, and the size indicates the area burned.

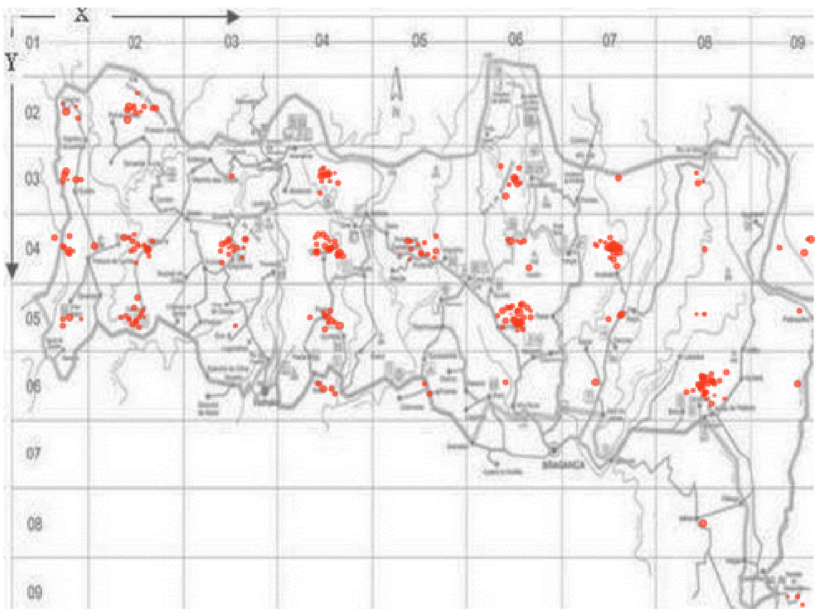
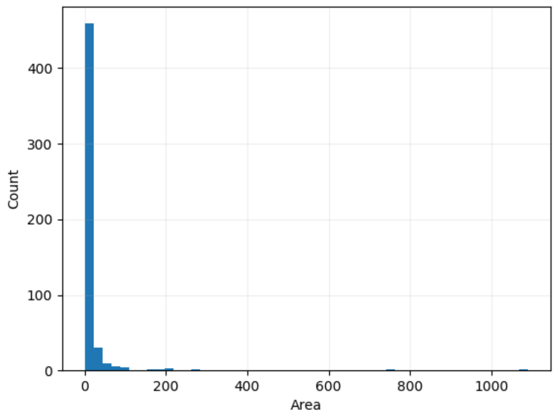


Figure 1: Map of fires in Montesinho Natural Park

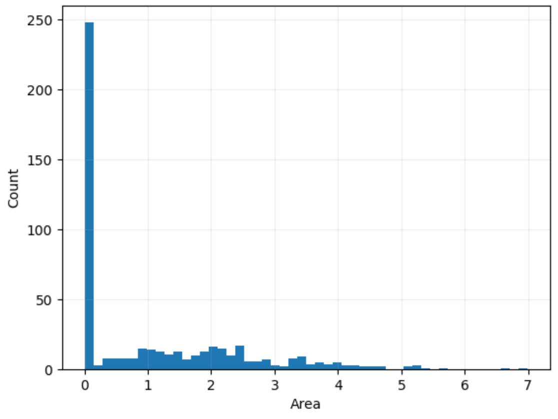
The variables `month` and `day` are categorical and denote the month ('jan' to 'dec') and day ('mon' to 'sun') of the incident. The variables `FFMC`, `DMC`, `DC` and `ISI` all come from the Forest Fire Weather Index (FWI), which comprises of several different metrics and measures the risk of wildfire incidents. `FFMC` (Fine Fuel Moisture Code) measures the humidity/moisture of litter and other small components that could fuel a fire, `DMC` (Duff Moisture Code) relates to the moisture of larger components, `DC` (Drought Code) measures the humidity of deeper layers of the environment, and `ISI` (Initial Spread Index) is a measure of the wind which would help a fire spread. The variables `temp` and `RH` measure temperature (in Celcius) and relative humidity (in %), respectively, while `wind` measures wind speed in km/h and `rain` denotes rain in mm/m^2.

Processing To feed the data to machine learning models, some processing needs to be done. The variables `month` is categorical and in string-form and will be recoded as integers, ranging from 1 to 12. Similarly, `day` will be recoded to an integer in 1 to 7. Moreover, the target variable `area` is heavily skewed towards 0 and will be transformed through $\texttt{area} = \ln(\texttt{area}+1)$, as done in the original paper by Cortez and Morais (2007). This transformation ensures that the data appears less skewed and has less extreme outcomes. To inspect the final predicted outcomes, the inverse of $\ln(x + 1)$ is taken, which is $e^x - 1$.

Figure 2 shows the histograms of the outcome variable before and after transformation. While the area had values above 1000, the (natural) log-transformation has maximum values only around 7. While the majority of observations are still at or around 0, there are more observations around values 1-4, which should help model predictions.



(a) Original area



(b) log-transformed area

Figure 2: Histograms of the outcome variable `area` before and after transformation

To investigate which variables seem to be related to the area, we can turn to pairwise scatterplots of the outcome variable against all other variables.

Looking at the scatterplots in figure 3, the majority of features do not show a clear relation with the outcome although it seems that the outcome goes up with increasing months or temperatures. To investigate how variables are correlated with each other and the outcome, a correlation heatmap can be used.

Interestingly, the majority of variables do not show any correlation amongst each other and with the area (before transformation). However, the month and temperature are correlated positively with each other and all of the four indicators from the Forest Fire Weather Index (`FFMC`, `DMC`, `DC` and `ISI`). All of those indicators are also positively correlated amongst themselves, although the highest correlation (0.87) was between `DC` and `month`. Moreover, correlations with the outcome `area` are very small in magnitude for all variables.

3 Models & methodology

To try and predict the area of forest fires, I will use a simple linear regression, random forest, XGBoost, a Gaussian process a generalized linear model for gamma distributed outcomes. Those models have all been widely and successfully used in the literature.

Linear Regression Linear regression, based on ordinary least squares (OLS) is a relatively simple model to model the relationship between target y and predictors X, where X denotes the design matrix of variables

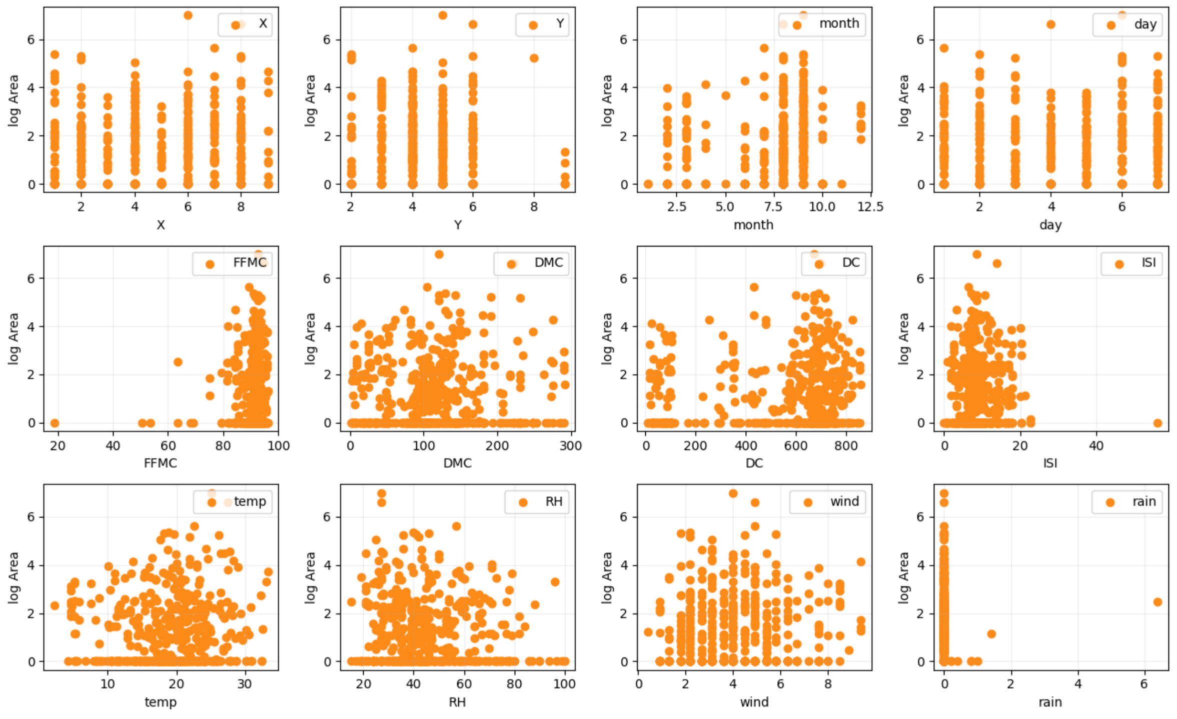


Figure 3: Scatter plots of outcome against all feature variables

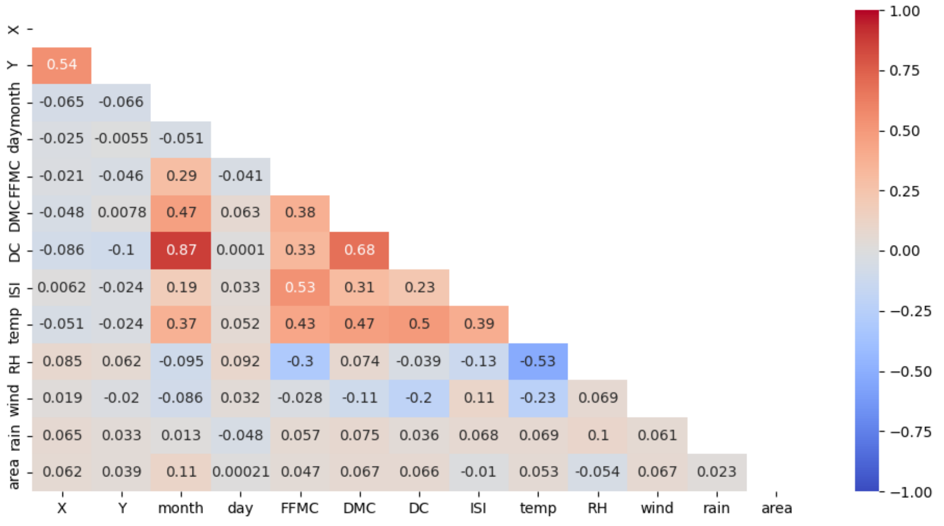


Figure 4: Correlation heatmap

and intercept, and all observations. The target variable is assumed to have a normal distribution with variance σ^2 and mean $X\beta$ and can thus be expressed as

$$y = X\beta + \epsilon, \tag{1}$$

where each ϵ_i has variance σ^2 and thus $y_i \sim \mathcal{N}(x_i\beta, \sigma^2)$. The estimate of coefficient vector β is $\hat{\beta} = (X'X)^{-1}X'y$. The advantage of the OLS estimator $\hat{\beta}$ is that it can easily be interpreted as partial effects of the variables and leads to interpretable results.

Gaussian Process Gaussian Processes (GPs) are a powerful, non-parametric Bayesian technique that can be used for regression tasks. At its core, a Gaussian Process is an infinite collection of random variables, any finite number of which have a joint multivariate Gaussian distribution. In the context of machine learning, GPs are used to model the function that maps inputs to outputs. The key idea is that instead of assuming the function is known up to a finite-dimensional parameter vector (as in parametric methods), GPs assume the function is drawn from a prior distribution that is Gaussian. This means that the function itself is left unspecified, allowing any kind of function, making them particularly useful in scenarios where the underlying process is complex and the data is limited. By placing a Gaussian prior over the function space, GPs can predict not just the mean of the output variable, but also its variance, giving a measure of the uncertainty in the prediction. The GP assumes a prior zero mean, so the outcome variable should be standardized, as performance deteriorates greatly otherwise. The package `sklearn.gaussian_process.GaussianProcessRegressor` natively implements the standardization through a keyword. The most important part of a GP is the covariance function, which essentially determines the distance between observations and thus how new predictions are formed based on previously observed values. I chose the default option of a constant kernel, multiplied with an RBF (also known as squared exponential) kernel, defined as $k_{RBF}(x_i, x_j) = \sigma_f^2 \exp\left(-\frac{(x_i - x_j)^\top (x_i - x_j)}{l^2}\right)$, with l the length scale and σ_f^2 the amplitude. The optimal parameters are automatically found by optimizing the log marginal likelihood (Schulz et al., 2018).

Tree-based models

Random Forest Random Forests (RF) are an ensemble learning method that operates by constructing multiple decision trees during training and outputting the mean prediction of the individual trees. By training a multitude of decision trees on various sub-samples of the dataset and using averaging to improve the predictive accuracy and control over-fitting, RFs offer robust predictions (Breiman, 2001). This model can be implemented easily with the class `sklearn.ensemble.RandomForestRegressor`.

XGBoost XGBoost, short for eXtreme Gradient Boosted trees, is a very successful machine learning model. It operates by combining the predictions from several trees and is thus an ensemble model. Contrary to e.g. a Random Forest, trees are trained in a sequential manner, where each additional tree in the ensemble is trained to predict the residual error of the ensemble of previous trees. Moreover, the predictions from each tree are scaled by a factor ν (typically below 0.1) to improve generalization and reduce overfitting (Friedman, 2001). This model is implemented in the class `xgboost.XGBRegressor`.

Hyperparameter tuning Both RF and XGBoost have hyperparameters that have to be tuned to obtain the best results. Because a wide range of values are possible for each hyperparameter, a grid search would have to be extremely sparse or is entirely unfeasible computationally, so I used Bayesian Optimization (BO) to find the best set of hyperparameters, which leverages Bayesian inference to construct a GP model of the objective function (MSE on a validation set, given the hyperparameter values), allowing for efficient exploration and exploitation of the search space. This GP model predicts the expected outcome and uncertainty of the objective function at any given point in the search space. By iteratively selecting the next point to evaluate based on maximizing the expected improvement or acquiring the most information, BO finds the optimal solution with fewer evaluations compared to e.g. grid search (Snoek et al., 2012). For RF, the parameters are the number of trees `n_estimators` (optimal value 126), the maximum depth for each tree `max_depth` (2), the maximum subset of features to use at each split `max_features` (0.7721) and the minimum number of samples in a leaf node required to allow a split `min_samples_split` (2). For XGBoost, the hyperparameters to tune are also `n_estimators` (279) and `max_depth` (4), the subset of observations to use at each iteration `subsample` (0.6591) and the learning rate ν (`learning_rate`, optimal value 0.0047).

Generalized linear model So far, all models above either do not or only implicitly assume the data distribution and might hence not be optimally suited. The first step is to determine the likely data distribution.

Looking at figure 2, the outcome is heavily right-skewed and continuous, so possible target distributions are Gamma, Beta, Weibull, Exponential or Log-Normal. One way to test whether data comes from a given distribution is the Kolmogorov-Smirnov test, which compares the theoretical CDF $F_0(x)$ with the empirical CDF $\hat{F}_n(x)$, as described in Massey Jr (1951). The test statistic is given by

$$D_n = \sup_{x \in \mathbb{R}} |\hat{F}_n(x) - F_0(x)|, \tag{2}$$

where $\hat{F}_n(x) = \frac{1}{n} \sum_{i=1}^{n} 1_{(-\infty,x]}(X_i)$. The corresponding p-value can then be obtained for desired confidence level, given sample size n.

Applying this methodology to the outcome variable `ln(area)`, the best fit is the Gamma distribution with parameters $\alpha = 0.4$ and scale $\theta = 1$. Still, the p-value was extremely low, indicating that this best fit is likely still not the true data distribution. Figure 5 below shows a histogram of the true outcomes along with the density. While the density fits reasonably well and displays the spike for outcomes around 0, it does not accurately fit the tail of the distribution.

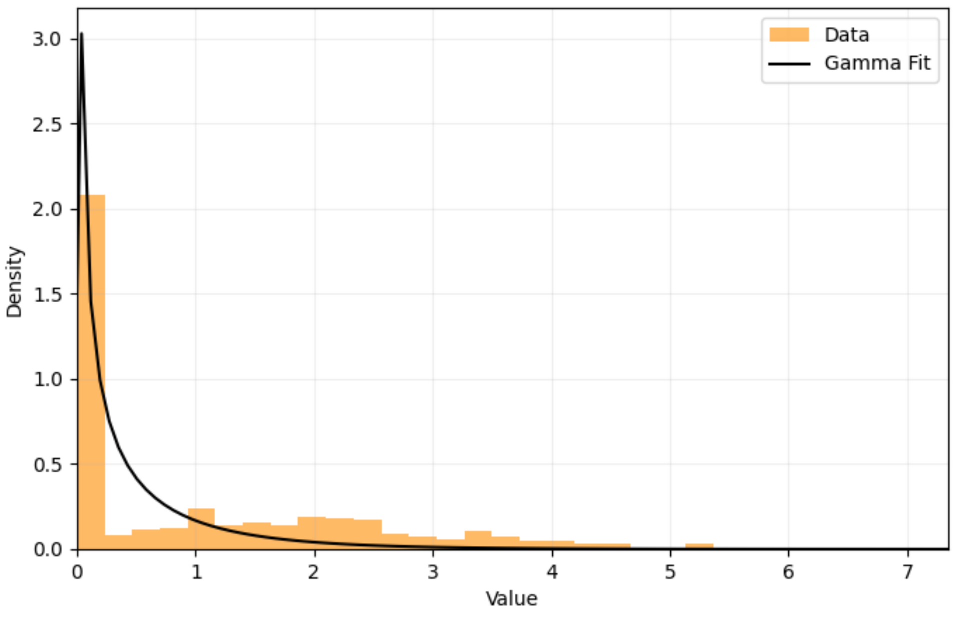


Figure 5: Outcome variable with density plot of best-fitting distribution

Assuming a gamma distribution for the outcome, a general linear model can be employed, which takes this distribution into account (Fahrmeir et al., 1994). The package `sklearn.linear_model.GammaRegressor` can then be used to predict the outcome as a function of the predictors. Because the data has strictly non-negative values, the log-link is used, meaning predictions are formed as $\hat{y}_i = \exp(\beta_0 + \beta_1 x_{1i} + ... + \beta_k x_{ki})$. Note that the negative-inverse link could also be used for the Gamma distribution. Additionally, the package includes a tuneable penalty term (also named α) to avoid overfitting. As this is a single parameter, a standard grid search can be employed to find the parameter which performs best. Because the dataset is relatively small, dividing the training set into smaller training and validation sets might make the results highly dependent on the split chosen. To circumvent this, I use cross validation, which (for each value of α) calculates the MSE for each of 5 folds and returns the average. That way, the results do not depend on a specific split. Figure 6 shows the average MSE for 5 folds for each value of α, along with the best value of around 2.6, indicating light penalization of model coefficients.

3.1 Model performance

All models above have been fit and tuned exclusively on the training set, which used 80% of observations. All the tuning was done by either subdividing the training set into another training and validation set, or cross validation on the training set. To compare the predictive performance, the hold-out test set is used. This separation of training and test set is vital to ensure that the scores of the models reflect performance on unseen data. To later predict the area of forest fires and identify high-risk areas/cases, the best model should be used.

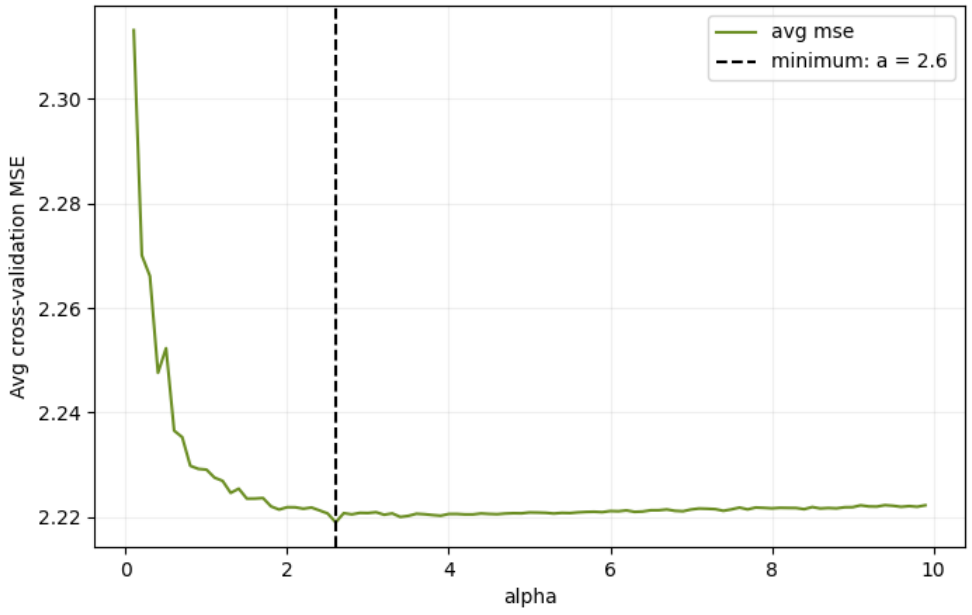


Figure 6: Average MSE for 5-fold cross validation for values of α

Table 1: MSE and MAE on the test set for all models

	Model				
Metric	**OLS**	**Gaussian Process**	**Random Forest**	**XGBoost**	**Gamma Regressor**
MSE	1.594	1.556	1.568	1.503	1.584
MAE	1.107	1.077	1.092	1.039	1.114

Interestingly, the best MSE and MAE was achieved by XGBoost, followed by the Gaussian Process, although the RF is very close behind. OLS (when trained on standardized data) has a very intuitive way of analyzing the coefficients as partial effects, while XGBoost and RF also have built-in feature importance methods.

4 Causes of forest fires

To investigate how the different variables affect the area of a forest fire, we can first look at the coefficients of the OLS model. Note that these are coefficients for the model trained on standardized inputs, so larger coefficients directly correspond to a larger influence on the outcome. In practice, training on standardized versus unstandardized features has no impact on predictive performance, as any scaling can be absorbed in the coefficients.

Table 2: Coefficients from the OLS model trained on standardized data

	Feature												
	intercept	**X**	**Y**	**month**	**day**	**FFMC**	**DMC**	**DC**	**ISI**	**temp**	**RH**	**wind**	**rain**
Coef	1.175***	0.1212	-0.0375	0.4379**	0.0089	-0.0259	0.1749	-0.4049	-0.0233	0.0804	-0.0224	0.0541	0.0195
p-value	0.000	0.1798	0.6592	0.0106	0.8964	0.3736	0.1587	0.1651	0.5231	0.1980	0.4527	0.3168	0.6229

Note: *** significant at 1% level, ** significant at 5% level, * significant at 10% level

Positive coefficients of the features indicate that an increase in the variable also increases the area of the forest fire. Interestingly, the coefficient of rain is positive, which one wouldn't expect. However, the coefficient is not significant and there are a lot of values in a cloud for very low values of rain, meaning that the line of best fit (so the value of the coefficient) is relatively arbitrary as the data does not show a linear trend. Apart from month, only the intercept is statistically significant, meaning that the majority of features are not very useful in predicting the area. Figure 7 shows the coefficients as a bar plot. The largest coefficients are the intercept and month, although DC (which measures the humidity of deeper layers of the environment) also has a comparatively large value, with a negative coefficient.

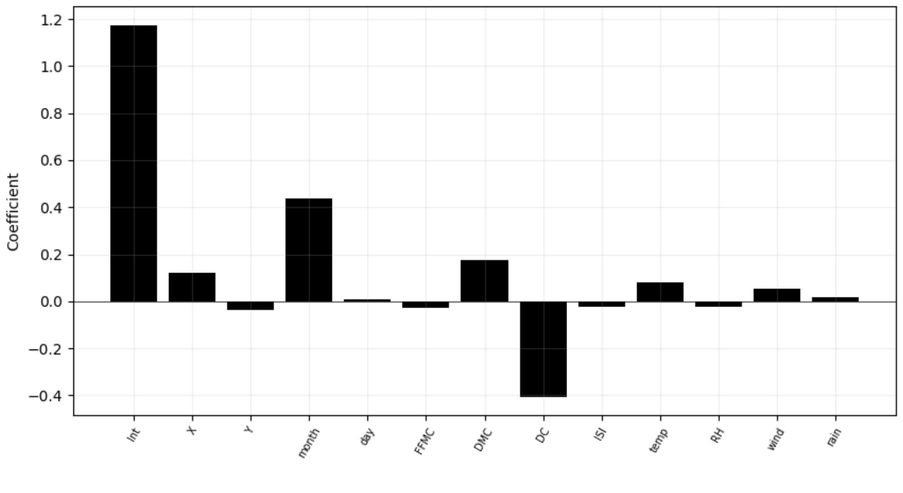


Figure 7: Coefficient estimates from OLS

One problem of just analyzing the coefficients of the linear regression is the enforced linearity. Looking at the fires on the map, they seem to increase as `Y` increases first, but then decrease again for higher values because the park only stretches down into coordinate 7 for most of the area. Moreover, most fires would be expected in the summer months, with less during November or December again, but this is not possible to model with a linear regression (cross-terms or squares could be added but with relatively few observations this could harm the fit further).

Turning to the XGBoost and RF models, it's possible to investigate how often a variable was used in splits during the training process, indicating relative importance of that variable. Figure 8 shows these importances. While temperature has the highest feature importance for both models, its importance is much larger for the random forest. For XGBoost, the feature importances are relatively equal, which makes more sense as the maximum depth for XGBoost is higher, so more features can be used. The variables with

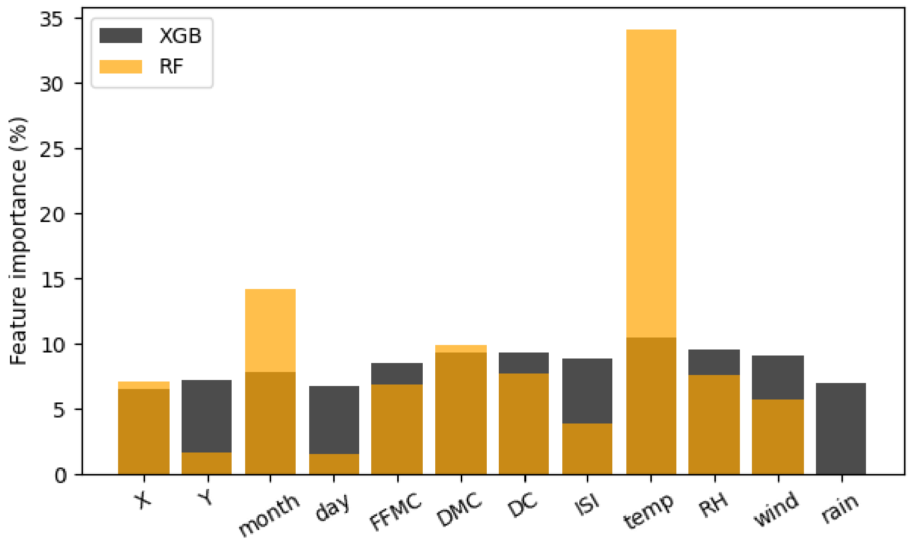


Figure 8: Feature importances for of XGB and RF

the lowest importances are `X`, `Y` `day` and `rain`, which intuitively makes sense as the day of the week is much less important than e.g. the month of the year and looking at Figure 1, the fires seem to be spread out relatively evenly across the coordinates. As mentioned above, `rain` does not show a clear relation to the outcome variable and it makes sense that it has a lower feature importance as well.

Instead of reducing each variable to a single coefficient or measure of importance, the Individual Conditional Expectation (ICE) plot shows the relation of the predicted area and a feature of interest. A Partial Dependence Plot (PDP) is the average of individual ICE lines (one per observation). This allows us to see how the predicted area of a fire changes as the variable changes from its minimum to maximum value. Roughly following the notation of the `sklearn` library, denote X_i as all observations of feature i, x_i individual observations, and X_{-i} all other features except i (its complement) and x_{-i} observations of X_{-i}. Then, the partial dependence of a function f (here the predictive function of XGBoost) at point x_i is defined as

$$pd_{X_i}(x_i) = \mathbb{E}_{X_{-i}}[f(x_i, X_{-i})] = \int f(x_i, x_{-i})p(x_{-i})dx_{-i}. \tag{3}$$

In practice, this can be calculated as

$$pd_{X_i}(x_i) \approx \frac{1}{n}\sum_{j=1}^{n} f(x_i, x_{-i}^{(j)}). \tag{4}$$

Note that x_i here denotes a point of feature X_i. To obtain the whole plot, x_i is varied from the minimum to maximum value, to obtain the model predictions given the changing feature value. Each orange ICE line is based on one sample (where x_i is varied and $x_{-i}^{(j)}$ kept constant), while the average is the PDP line.

Figure 9 paints a much more complete picture of how the individual features are related to the outcome. While some features like `Y`, `day` or `rain` (which also had the lowest feature importances) show relatively constant lines, the other features show much more nuanced interaction. As expected, as `temp` increases, predictions increase, but the interaction is not as linear as one might expect. Similarly, `month`, `DMC` or `RH` show more intericate interactions. Moreover, the relative humidity `RH` first has prediction values increasing, before flattening out again, indicating that very low humidity actually decreases the expected area of forest fires, given the XGBoost model's fit.

Because a lot of values of the log area are 0, I further investigated whether classifier models could be of use, predicting whether a fire might take place or not (binary outcomes of 0 and 1). However, both logistic regression and the classifier version of XGBoost performed badly, resorting to constant predictions of 0. Investigating the probability predicted by the models, that too is constant at around 45%. Looking at the distribution of the data (figure 2), a lot of values are very close to 0, so differentiating between exact 0 value or slightly higher proved too difficult. In general the regression models performed substantially better and should thus be preferred, as they can still predict very low values of area if it appears unlikely but is not constrained by a simple yes or no. Additionally, even a small difference between 0 and 0.1 makes a big difference in the response and potential area burned, so zero predictions of the classifiers are undesirable.

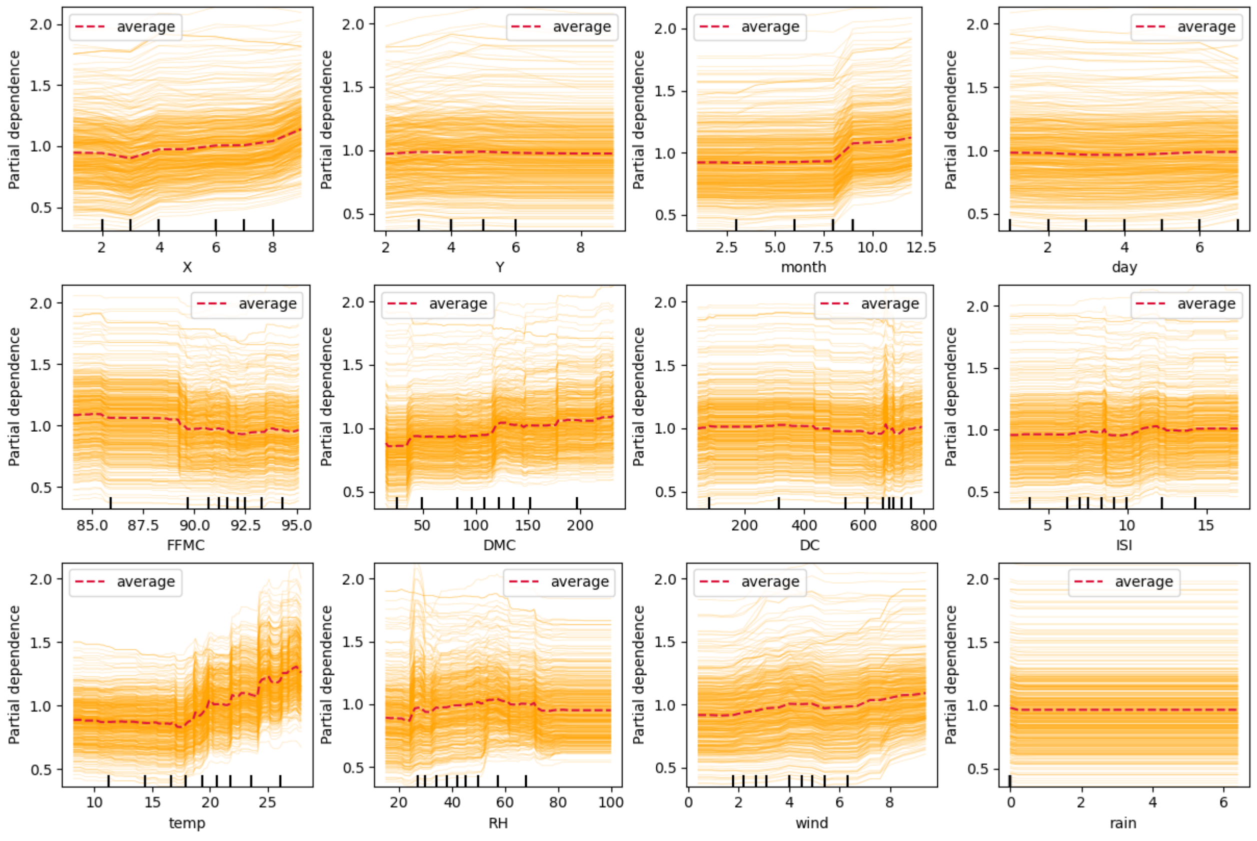


Figure 9: Partial Dependence Plots for all variables using the XGBoost model

When using the model to predict the likely area or size of a fire given input data, the regression models thus offer a much more complete and accurate picture of the outcome than the classification models could.

5 Location of forest fires

Using the XGB model for predictions, it is now possible to investigate which areas are most at risk of fires (a larger predicted area), given input features. The simplest way is to set all variables at their mean and then plot the predicted area for each coordinate pair. Figure 10 shows these predictions.

Looking at the predicted outcome per coordinate rectangle, we can see that prediction values increase as `X` increases (towards the east). Values on the west side of the park have comparatively low values. Comparing this to the incidents on the map in figure 1, there are relatively few fires in the west and the majority seems to be concentrated around the center to east area. So if all features are at their respective mean values, the eastern side of the park is at much higher risk.

Another way is to compute the expected area per coordinate pair in a similar fashion as the ICE. For each (`X`, `Y`) pair, the model can predict the area for each observation in the dataset, just substituting in the respective coordinates. However, this full data pass lead to relatively constant predictions, as the outcomes were all around 1.52-1.53. This could be because other features are much more relevant so averaging over all the data will always lead to relatively similar predictions.

To investigate how the predicted area changes as more extreme feature values are observed, the map can be plotted for feature values that should produce very low or high output. To do this, I selected the feature values which had the lowest and highest partial dependence in figure 9, meaning each feature at the value which individually leads to the lowest/highest model predictions. Figure 11 shows these maps.

As expected, the predicted outcome (transformed back to original scale in hectares) differs significantly. While in the plot with lowest values, the maximum values are around 0.8 hectares, for the other map that is around 4.2 hectares (over five times as much). Still, for both of these plots, the predicted values increase towards the center/right side of the map, although the map with maximizing inputs looks closer to the original map of forest fires. That the map with minimizing inputs has such a strong increase along the x-coordinates, could indicate that for generally adverse conditions for fires, rectangles towards the east end of the map are at higher risk of fires.

The publicly available code also contains a file called `forecast.py` in the folder `analyses`, which allows

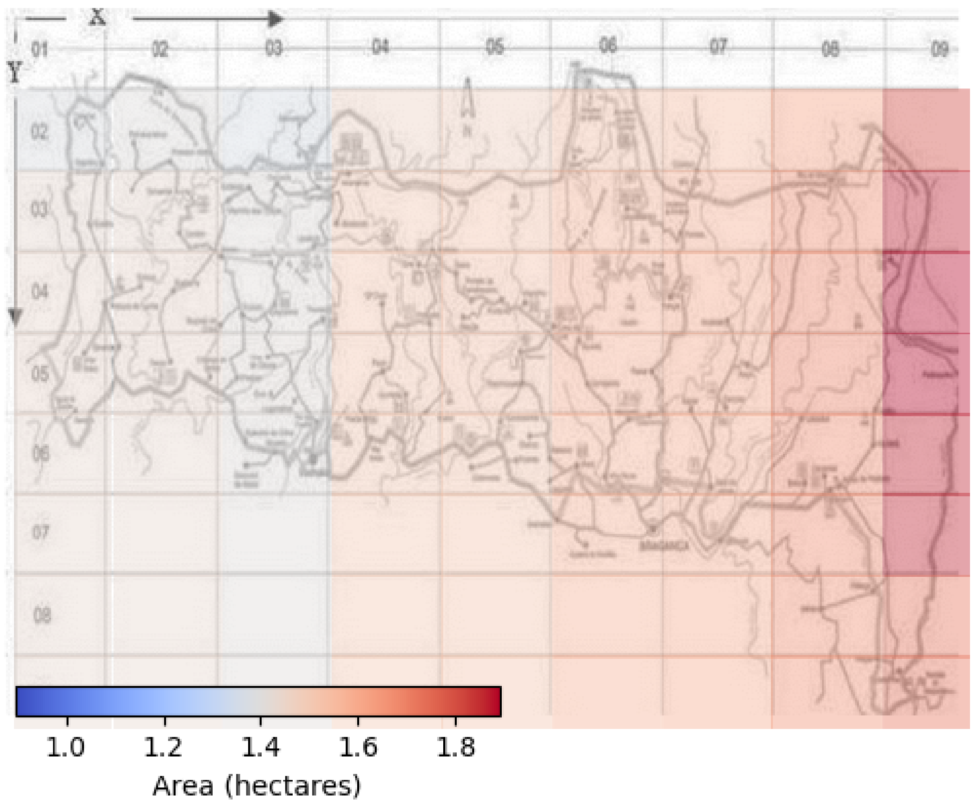
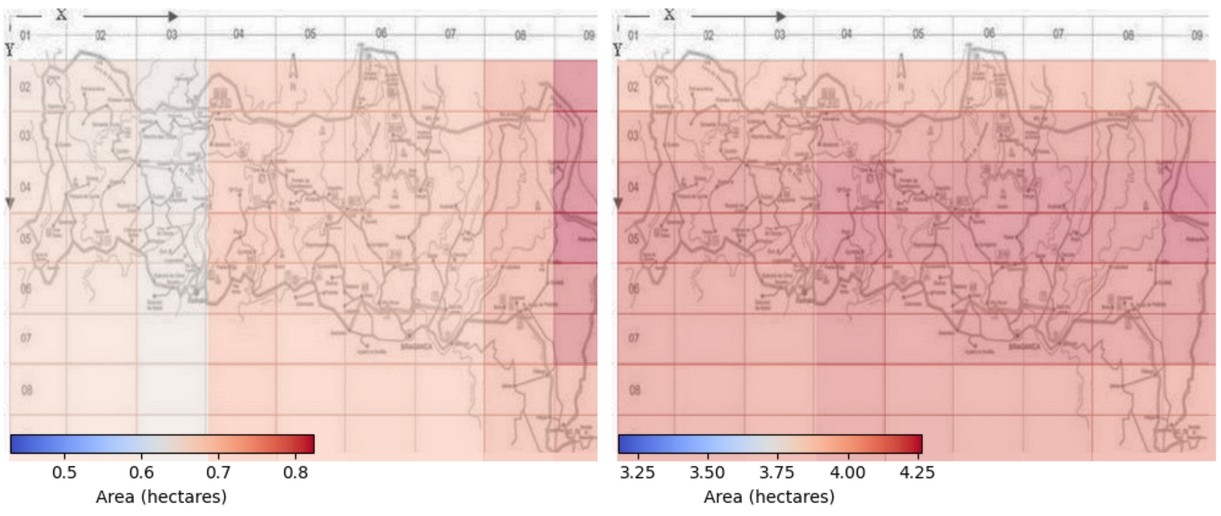


Figure 10: Predicted outcome per coordinate pair, XGBoost



(a) Lowest

(b) Highest

Figure 11: Predicted outcome per coordinate pair, setting feature values to minimizing and maximizing values

the user to specify input features and have either the whole map plotted, or the area for a specific rectangle returned. Also available are the function calls used to generate the plots with minimizing and maximizing feature inputs.

6 Conclusion

In conclusion, this article showed how to use machine learning models to predict forest fires and understand the main causes. As a first step, it is important to get an understanding of the dataset and the relation of the outcome variable with the features, potentially transforming the data in some way if necessary.

After understanding the models used, tuning the hyperparameters with Bayesian Optimization or grid search provides a way of finding the best model specifications. Apart from that, a KS test can be used to determine the most likely data distribution and use that for a generalized linear model.

After comparing the models in terms of predictive power in section 3.1, either regression coefficients, built-in feature importance tools or partial dependence plots can be used to understand how each variables influences the outcome. Here, the month, temperature and humidity of the environment (`DC`) were the most important variables. Additionally, the center and east end of the park seemed to be at higher risk of forest fires. Finally, this project investigates the most likely locations of forest fires and provides a tool to show the map for specified inputs. The code for all of this is available, so feel free to experiment with the setup, different models, or the interactive map!

References

Paulo Cortez and Aníbal de Jesus Raimundo Morais. A data mining approach to predict forest fires using meteorological data. *Associação Portuguesa para a Inteligência Artificial (APPIA)*, 2007.

Eric Schulz, Maarten Speekenbrink, and Andreas Krause. A tutorial on gaussian process regression: Modelling, exploring, and exploiting functions. *Journal of Mathematical Psychology*, 85:1–16, 2018.

Leo Breiman. Random forests. *Machine learning*, 45:5–32, 2001.

Jerome H Friedman. Greedy function approximation: a gradient boosting machine. *Annals of statistics*, pages 1189–1232, 2001.

Jasper Snoek, Hugo Larochelle, and Ryan P Adams. Practical bayesian optimization of machine learning algorithms. *Advances in neural information processing systems*, 25, 2012.

Frank J Massey Jr. The kolmogorov-smirnov test for goodness of fit. *Journal of the American statistical Association*, 46(253):68–78, 1951.

Ludwig Fahrmeir, Gerhard Tutz, Wolfgang Hennevogl, and Eliane Salem. *Multivariate statistical modelling based on generalized linear models*, volume 425. Springer, 1994.