Drought prediction using weather data

Furfaro Bernard Antonio, Marossi Clara, Rabuzzi Giacomo

May 7, 2025

Abstract

Drought effects are often underestimate, yet they can have a significant impact on agriculture, domestic water supply, energy production, public health, wildlife, food security, energy and communities. That's why is important to have regularly updated sources of information on drought, even though studying these data requires a lot of work for researchers. During this study we fitted various statistical models, for both regression and classification, to see whether they are able to replace experts' interpretation and work, reducing the human effort necessary to analyze drought data. In particular, we employed both traditional statistical models and machine learning techniques, such as random forest, to assess their predictive capabilities and reliability. This approach allowed us to compare the performance of different methodologies and evaluate the potential of automated systems in drought monitoring and analysis.

1 Introduction

The U.S. Drought Monitor (USDM) provides a big picture look at drought conditions in the U.S. and its territories. Every Thursday a map is released, labeling areas according to their intensity of drought. The map uses six classifications: normal conditions, abnormally dry, showing areas that may be going into or are coming out of drought, and four levels of drought, as shown in Table 1.

Category	Description
None	Normal or wet conditions
D0	Abnormally Dry
D1	Moderate Drought
D2	Severe Drought
D3	Extreme Drought
D4	Exceptional Drought

Table 1: Drought Classification Table

The USDM differentiates between short-term drought, that varies easily and impacts mostly on agriculture and grasslands, and long-term drought, that is more persistent and impacts on hydrology and ecology. An area can also be impacted by both.

This drought level is versatile and reliable, since is obtained as a combination of many sources: local observations, local expert feedback and physical indicators. These last measures comes from different areas, such as climate, weather and hydrology, and vary also across different time spans. Thus, it goes beyond just rain and snow, and considers also humidity, vegetation health, steamflow, soil moisture and so on.

Having both short and long term conditions makes it challenging to balance short-term rainfall with long-term dryness and vice versa, making it necessary to analyse different time intervals. Furthermore, depending on the specific place, the authors should weight indicators differently. For example, snowpack has a different impact in the West and in the East.

The USDM was born in 1999 from the collaboration between the National Drought Mitigation Center at the University of Nebraska-Lincoln, the National Oceanic and Atmospheric Administration and the U.S. Department of Agriculture. Meteorologists and climatologists' job is to make sense out of the data even when they seem to lack it.

Monitoring drought level helps state agencies, non-governmental organizations and local decision makers recognizing emerging drought, to identify disasters and emergencies correlated to it.

1.1 Goals

The main goal of this paper is to investigate if droughts can be predicted using only meteorological data. First, we computed an exploratory analysis of our dataset, in order to understand and interpret its components and structure. Then we tackled the problem using two different approaches: regression and classification. We trained both to see whether "simplifying" the problem into discrete classes, rather than considering a continuous response, could facilitate the process and improve the predictions.

2 Dataset

The dataset used in this experiment is publicly available on the Kaggle platform [1] and it combines open data offered by the NASA POWER Project and the US Drought Monitor authors. An additional dataframe, taken from the Harmonized World Soil Database, is included.

The soil data contains non-timeseries information on each county, such as FIPS code, latitude, longitude, elevation, ecc. After testing its integration with the original dataset, we found out that it did not give any additional insight. In fact, it contains specific information about the county where the observation was recorded. However, since the main goal of this study is to investigate whether droughts can be predicted using only meteorological data, we chose to exclude soil characteristics. Including them would not align with our objective and could introduce unnecessary complexity without contributing to our intended analysis. Therefore, to maintain a focused approach and optimize computational efficiency, we decided not to incorporate these variables.

The main dataset consists of spatiotemporal observations, where each entry corresponds to a specific day in a U.S. county, reporting the meteorological indicators in Table 2. Since drought classification is assigned once a week, the label variable presents missing values for six days a week.

Indicator	Description
WS10M_MIN	Minimum Wind Speed at 10 Meters (m/s)
QV2M	Specific Humidity at 2 Meters (g/kg)
T2M_RANGE	Temperature Range at 2 Meters (C)
WS10M	Wind Speed at 10 Meters (m/s)
T2M	Temperature at 2 Meters (C)
WS50M_MIN	Minimum Wind Speed at 50 Meters (m/s)
T2M_MAX	Maximum Temperature at 2 Meters (C)
WS50M	Wind Speed at 50 Meters (m/s)
TS	Earth Skin Temperature (C)
WS50M_RANGE	Wind Speed Range at 50 Meters (m/s)
WS50M_MAX	Maximum Wind Speed at 50 Meters (m/s)
WS10M_MAX	Maximum Wind Speed at 10 Meters (m/s)
WS10M_RANGE	Wind Speed Range at 10 Meters (m/s)
PS	Surface Pressure (kPa)
T2MDEW	Dew/Frost Point at 2 Meters (C)
T2M_MIN	Minimum Temperature at 2 Meters (C)
T2MWET	Wet Bulb Temperature at 2 Meters (C)
PRECTOT	Precipitation (mm day^{-1})

Table 2: Description of the indicators and data format

The dataset is already divided into training, validation, and test sets according to the year of each observation

2.1 EDA and Pre-processing

For computational efficiency and feasibility, we limited our analysis only to the most recent 10 years available, from 2010 to 2020, even though the original dataset covers a longer period of time.

We also removed the following variables: T2M_RANGE, WS50M_RANGE and WS10M_RANGE, as their corresponding point value, minimum and maximum are already included, making them noisy and redundant; T2M and T2MDEW, due to their high correlation with other variables, as highlighted in the correlation matrix in Figure 1.

The correlation matrix shows two distinct clusters of highly correlated variables: one related to temperature and humidity (including T2M, T2MWET, T2MDEW, and T2M MIN/MAX) and another associated with wind speed at different altitudes (WS10M, WS50M, and their respective min/max values). This pattern suggests that temperature and humidity variables strongly influence each other, while wind-related variables exhibit a similar internal correlation. This structure will likely be reflected in the modeling phase as well.

On the other hand, we created a one-hot encoded variable representing the month, derived from the original date variable. Each month has unique and specific meteorological characteristics, such as temperature variations, precipitation, and weather conditions, which can significantly influence data patterns. However, these features might not be directly captured by the variables in the dataset. By including the month as a one-hot encoded variable, we can account for these seasonal peculiarities, which may not be otherwise detected,

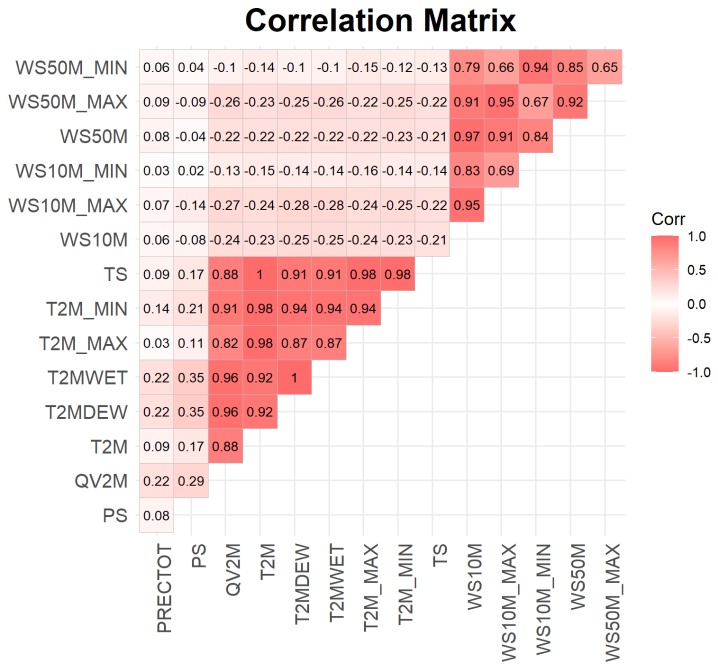

Figure 1: Correlation matrix for the data

and improve the performance of predictive models by incorporating these recurring patterns throughout the year.

We can notice how the drought score has been detected only once a week for each county, so we have meteorological data for the entire week but in the absence of the target value. To address the problem of missing values in the response variable (drought class), we added the one-week lags for the weather variables and deleted all the observations with missing label, so that we have rows summarizing the data of the weeks. Furthermore, to add even more information and be able to capture more complex relationship between the predictors and the response, without significantly increasing the number of variables, we incorporated weekly averages of the past eight weeks for each row.

In addition, we analyzed the relationship between the lags within the same variable. Based on the correlation matrix, we decided not to include the lags for the variable PS, as its correlation over time remained close to 1, even for distant periods and weeks.

The drought score is a continuous variable suitable for regression tasks. However, to enable the use of classification models, we derived a discrete variable discretizing it as outlined in Table 3.

Score	**Class Score**
0	None
$(0, 1]$	D0
$(1, 2]$	D1
$(2, 3]$	D2
$(3, 4]$	D3
$(4, 5]$	D4

Table 3: Mapping of the *score* variable

The classification problem is highly imbalanced: the majority of the observations belong to the class "None" and the frequency decreases across the following classes, as we can see from Figure 2.

To address the imbalance and redundancy across counties, we applied k-means clustering to select a representative subset. The underlying assumption is that geographically closer counties likely report similar information; therefore, we prioritized selecting "distant" counties. First, we clustered the counties into three groups (west, middle, and east) based on longitude and latitude. Within each cluster, we retained only counties that were separated by at least a predetermined distance: 200 km in the west, 220 km in the middle, and 240 km in the east. These distance thresholds were chosen according to the population density of each region. This approach significantly reduced the number of counties (necessary for our computational capabilities) while preserving the original data structure and mitigating the imbalance issue, ensuring a more feasible and meaningful analysis.

Finally, we divided the dataset into training and test set using a time-based slit to respect the temporal order. The training set contains all the observations from 2010 to 2019, while the test set contains data from the last year available, 2020.

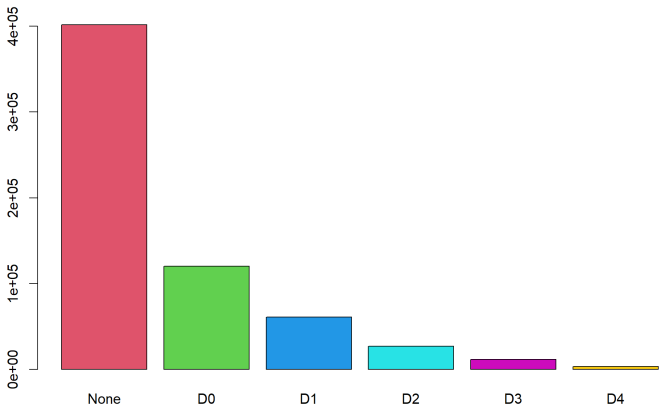

Figure 2: Distribution of target variable

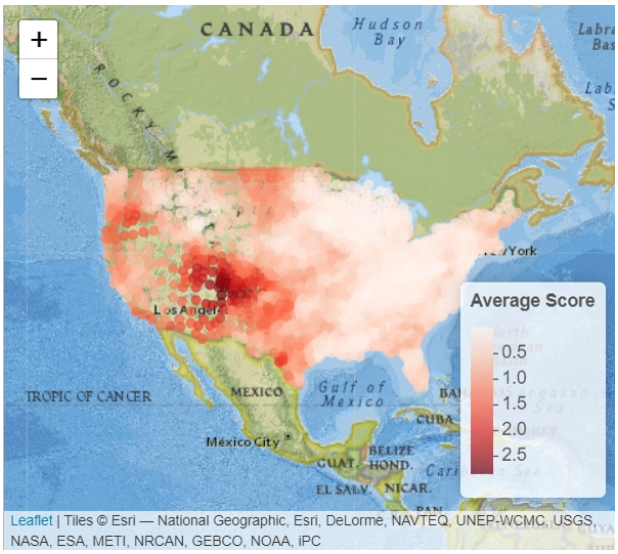

Figure 3: US MAP before counties' selection

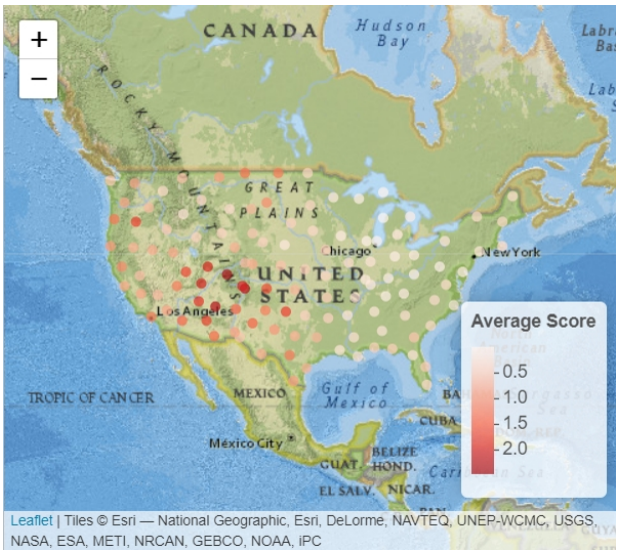

Figure 4: US MAP after counties' selection

3 Regression

We can now consider the problem from a regression point of view, using as target variable the continuous score detected by the creators of the dataset.

3.1 Beta Regression

The beta regression model is tailored for situations where the variable of interest is continuous and restricted to the interval (0,1). It assumes that the response follows a Beta distribution, whose probability density function (PDF) is given by:

$$\pi(y; p, q) = \frac{\Gamma(p+q)}{\Gamma(p)\Gamma(q)} y^{p-1}(1-y)^{q-1}, \quad 0 < y < 1, \tag{1}$$

where $p > 0$, $q > 0$, and $\Gamma(\cdot)$ is the gamma function. The mean and variance of y are, respectively,

$$E(y) = \frac{p}{p+q}, \qquad \text{var}(y) = \frac{pq}{(p+q)^2(p+q+1)}. \tag{2}$$

However, for regression tasks, it's more common to work with a different parametrization to model the mean of the response and include a precision parameter. Let

$$\mu = \frac{p}{p+q} \quad \text{and} \quad \phi = p+q, \quad \text{i.e.,} \quad p = \mu\phi \quad \text{and} \quad q = (1-\mu)\phi.$$

It follows from (2) that

$$E(y) = \mu, \qquad \text{var}(y) = \frac{V(\mu)}{1+\phi}. \tag{3}$$

where $V(\mu) = \mu(1-\mu)$, so that μ is the mean of the response variable and ϕ can be interpreted as a precision parameter: for fixed μ, the larger the ϕ, the smaller the variance of y.

The density of y can be written as

$$f(y; \mu, \phi) = \frac{\Gamma(\phi)}{\Gamma(\mu\phi)\Gamma((1-\mu)\phi)} y^{\mu\phi-1}(1-y)^{(1-\mu)\phi-1}, \quad 0 < y < 1, \tag{4}$$

where $0 < \mu < 1$ and $\phi > 0$.

The model assumes that the response can be written as

$$g(\mu_t) = \sum_{i=1}^{k} x_{ti}\beta_i = \eta_t, \tag{5}$$

where $\beta = (\beta_1, \ldots, \beta_k)^\top$ is a vector of unknown regression parameters ($\beta \in \mathbb{R}^k$), $x_{t1}, \ldots, x_{tk}$ are observations on k covariates and g is a link function (strictly monotonic and twice differentiable, for example logit or probit).

The parameters are estimated using maximum likelihood procedure.

To fit this regression model on our data, we had to adjust the response range, reducing its variability within the standard unit interval. Moreover, we addressed the issue of extreme values for the response (0 and 1), simply adding a small constant to the 0 values and subtracting it from the 1 values.

We fitted the model on the training and then made predictions on the test. The model's performances are presented in Table 4.

Metric	**Training Data**	**Test Data**
Standard Deviation (SD)	0.1943	0.2514
RMSE	0.1724	0.2249
MAE	0.1311	0.1681

Table 4: Beta Regression Performance on Training and Test Data

These metrics seems to indicate quite strong performances: RMSE and MAE are lower than the standard deviation in both training and test set. As expected, the error measures are lower in the training than in the test, but the difference is not that significant, indicating that the model generalizes quite well even on unseen data. However, even though these metrics suggest good performances, comparing the fitted and the real values through a scatter plot highlights some issues, especially due to the high concentration of values close to zero. This suggests that our model is struggling to predict values closer to 1 and generally tends to underestimate the response: the model most likely overfit to the lower values while fails when the scorse gets closer to 1. The high presence of low values skews the measures downwards, whereas they are not that much affected by the errors in predicting the higher ones. We concluded that a standard model is not well suited for an excess of extreme values in our data, so we tried to focus more on them and improve the performances across the entire range of the response fitting a zero/one inflated beta (ZOIB) regression model.

3.2 Beta Regression Inflated

This model is an extension of the base implementation and differs from it since it handles a response variable that takes values from closed unit interval [0, 1]. It consists of three parts:

1. A **logistic model** for the probability of $Y = 0$
   Thus, $p_0 = P(Y = 0)$ = probability of zeros;
2. A **logistic model** for the probability of $Y = 1$.
   Thus, $p_1 = P(Y = 1)$ = probability of ones;
3. A **Beta regression model** for values in $(0, 1)$.
   Thus, $Y \mid (0 < Y < 1) \sim \text{Beta}(\mu, \phi)$

The probability density function (PDF) of Y is now given by:

$$f(y) = \begin{cases} p_0, & \text{if } y = 0 \\ p_1, & \text{if } y = 1 \\ (1 - p_0 - p_1) \cdot \frac{\Gamma(\phi)}{\Gamma(\mu\phi)\Gamma((1-\mu)\phi)} y^{\mu\phi-1}(1-y)^{(1-\mu)\phi-1}, & \text{if } 0 < y < 1 \end{cases} \tag{6}$$

where $\mu = E[Y \mid 0 < Y < 1]$ is the mean of the Beta component, ϕ is the precision parameter of the Beta distribution, p_0 and p_1 are estimated separately using logistic regression.

The results presented in Table 5 indicate no improvement from the previous model: the RMSE on the test set is unchanged, while the MAE is slightly higher. Thus, the Beta Inflated model is still not well-suited for handling the excess of zeros. Moreover, the comparison of the real values versus the predicted ones continues to show issues, without any significant improvement from before. Predicting the response variable as continuous is probably too complex and may require additional predictors or more advanced techniques.

Metric	Training Data	Test Data
Standard Deviation (SD)	0.1982	0.2565
RMSE	0.2043	0.2234
MAE	0.1779	0.1882

Table 5: Beta Regression Performance on Training and Test Data

4 Classification

As mentioned earlier, we tried to simplify the problem, approaching it from a different and simpler prospective: we divided the score variable into classes and fitted classification models. We applied both multi-class and binary models. For the binary models we created a new variable indicating the presence/absence of drought, obtained aggregating all drought levels into one class.

4.1 Ordered Logistic Regression

Since the response variable has multiple possible categories, fitting a logistic model requires the use of multinomial models. Given the ordered nature of the target variable, this necessitates the application of ordered multinomial logistic regression. We focus on the probabilities:

$$P(G \leq G_j \mid X = x) = \sum_{l=1}^{j} P(G = G_l \mid X = x)$$

In particular, we model the logit of the probability following a linear trend with common β coefficients and varying intercepts across classes. The model can be expressed as:

$$\text{logit}(P(Y \leq G_j \mid X = x)) = \alpha_j + \beta_1 x_1 + \beta_2 x_2 + \cdots + \beta_k x_k$$

where α_j represents the intercept for class j, $\beta_1, \beta_2, \ldots, \beta_k$ are the common regression coefficients. The parameters in an ordered multinomial logistic regression model are estimated using Maximum Likelihood

Estimation; the log-likelihood function is maximized with respect to the model parameters, using numerical optimization methods like Newton-Raphson.

Furthermore we addressed class imbalance toward class None by applying weights. The weights are calculated as the inverse of the relative frequency of each class in the training data. Specifically, less frequent classes are assigned higher weights, while more frequent classes receive lower weights. These weights are then incorporated into the model to ensure that the influence of underrepresented classes is not overshadowed by the majority classes. So we want to maximize the following log-likelihood function:

$$\ell(\beta) = \sum_{i=1}^{N} w_i \log P(y_i \mid x_i, \beta)$$

where:

- w_i are the weights assigned to the observations;
- $P(y_i \mid x_i, \beta)$ is the probability of the observed class y_i, given the covariate x_i and the parameters β.

This approach effectively improved the model's performance on less frequent classes, even if the results are not so satisfactory for minority classes.

After training the model on data up to 2019, we made predictions for the year 2020 and obtained the following results in terms of sensitivity and balanced accuracy:

Metric	**Class: None**	**Class: D0**	**Class: D1**	**Class: D2**	**Class: D3**	**Class: D4**
Sensitivity	0.662	0.198	0.094	0.240	0.295	0.392
Balanced Accuracy	0.733	0.528	0.514	0.539	0.597	0.658

Table 6: Model performance metrics for each class

The model appears to perform better for the extreme classes (those at the ends of the response scale), while the more moderate classes tend to be misclassified more frequently. This could be due to a number of factors, including overfitting to the more frequent classes or irrelevant variables influencing the model.

A possible solution to address this issue could be the use of lasso regularization, which introduces a penalty term on the coefficients in the objective function that we aim to maximize. This can help by shrinking the coefficients of less important predictors to zero. As result we could effectively remove unnecessary variables and reduce model complexity, potentially improving generalization and overall classification performance, especially for the more moderate classes.

4.2 Ordered Logistic Regression With Lasso Penalization

While the structure of this model remains the same as the previous one, the primary difference lies in the estimation of the parameters. Specifically, we introduce a penalization term to the objective function as follows:

$$\min_{\beta} \left\{ -\sum_{i=1}^{n} \log p(y_i \mid x_i, \beta) + \lambda \sum_{j=1}^{p} |\beta_j| \right\}$$

The first term corresponds to the standard log-likelihood (actually it is negative log-likelihood because we set a minimization problem), while the second term is the L1-penalty (lasso regularization) applied to the coefficients.

The degree of shrinkage applied to the coefficients is governed by the hyperparameter λ, which we select using a cross-validation procedure with 5 folds. To reduce computational cost, we consider only five possible values of λ during the tuning process. The same sequence of λ values is used in each iteration of the cross-validation. For each choice of λ, we evaluate the model's performance on the held-out validation set by calculating the out-of-sample log-likelihood, misclassification rate, Brier score, and the percentage of deviance explained.

We note that some of these performance metrics are sensitive to imbalances in the class distribution. To mitigate this issue, the function used for model fitting (ordinalNet) does not support a weighting mechanism. Therefore, we address this problem by employing an undersampling technique for the three most frequent classes and an oversampling method for the three least frequent classes, adjusting the frequencies to the median class size. This procedure results in a more balanced training set, which improves the model's performance and mitigates the impact of class imbalance.

As shown in Figure 5, very small lambda values appear to be more effective across all evaluation metrics. This suggests that the penalization and shrinkage should have a minimal impact on the estimates.

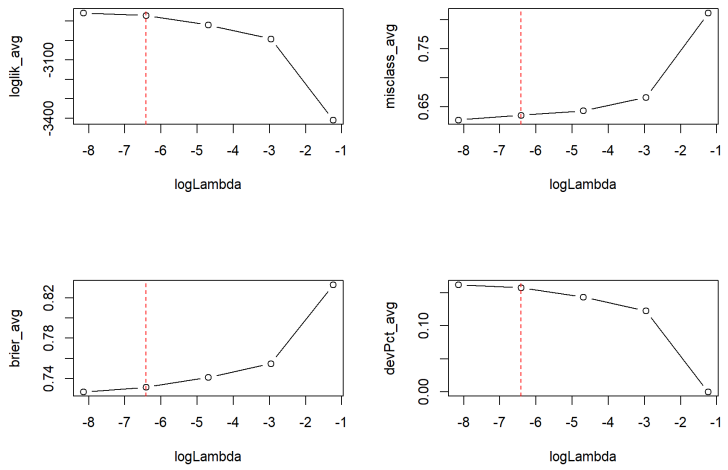

Figure 5: log(λ) vs Average performance metrics with 5-CV

We selected the highest value of λ (most parsimonious model) for which the log-likelihood and the percentage of deviance remain at a plateau and have not yet started to decline significantly. At the same time, the misclassification error and the Brier score have not yet shown a noticeable increase.

We can then train the best model on the entire training set (we obtain a model with 109 non zero coefficients out of 180) and assess its performance on the test set. The results are presented in the following table, where

they are compared to those obtained using a weighting mechanism (without undersampling or oversampling) and without lasso penalization:

Metric	Class: None	Class: D0	Class: D1	Class: D2	Class: D3	Class: D4
Sensitivity without Lasso	0.662	0.198	0.094	0.240	0.295	0.392
Sensitivity with Lasso	0.680	0.259	0.149	0.189	0.282	0.504
Balanced Accuracy without Lasso	0.733	0.528	0.514	0.539	0.597	0.658
Balanced Accuracy with Lasso	0.757	0.552	0.531	0.535	0.590	0.713

Table 7: Model performance metrics for each class

The lasso regularization slightly improved the model's performance; however, satisfactory results were only achieved for the extreme classes (None and D4). Therefore, we could try simplifying the problem by reducing the target variable to a binary classification, focusing solely on the presence or absence of droughts.

4.3 Logistic Regression

In logistic regression, in the case of a binary response, the posterior probabilities of the 2 classes are modeled by linear functions in x, with the constraint that the probabilities remain in the interval [0, 1] and that they sum up to 1.

$$\log(P(G = 0|x)) - \log(P(G = 1|x)) = \beta_0 + \beta_1^\top x$$

And so we obtain:

$$P(G = 0|x) = \frac{\exp\{\beta_0 + \beta_1^\top x\}}{1 + \exp\{\beta_0 + \beta_1^\top x\}}$$

$$P(G = 1|x) = \frac{1}{1 + \exp\{\beta_0 + \beta_1^\top x\}} = p$$

The parameters can be estimated using the ML method. The log-likelihood function is:

$$l(\beta) = \sum_{i=1}^{n} \log p_{g_i}(x_i; \beta) = \sum_{i=1}^{n} [g_i \log p_i + (1 - g_i) \log(1 - p_i)]$$

Where p_i is the conditional probability $P(G_i = 1 \mid x_i)$.

We use numerical methods like IRLS (Iterative Reweighted Least Squares) to estimate the values of the parameters.

By default, logistic regression classifies an observation into one of two categories (0 or 1) using a standard decision threshold of 0.5. Specifically: if the predicted probability $P(G = 1|x)$ is >= 0.5, the model classifies the observation as 1 (the event occurs); if the predicted probability is < 0.5, the model classifies the observation as 0 (the event does not occur). In our implementation, we used 5-fold cross-validation to determine the optimal decision threshold for predictions. Specifically, we explored possible threshold values ranging from 0.1 to 0.9, incrementing by 0.1 at each step. The optimal threshold (0.4) was selected by maximizing the average balanced accuracy across the validation folds as shown in Figure 6.

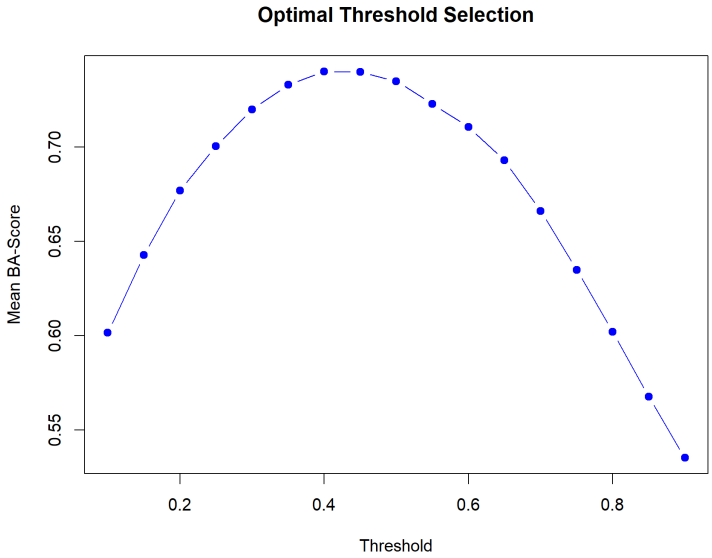

Figure 6: Selecting the threshold

	Reference	
Prediction	**0**	**1**
0	1947	689
1	868	2684

Table 8: Confusion matrix comparing predicted and ground truth values.

After selecting the optimal threshold, we applied the trained model to predict the response variable on the test set and compared the results with the ground truth target values. This evaluation produced the following confusion matrix and performance metrics:

We can observe that simplifying the problem by reducing it to two classes has significantly improved prediction performance, making the results much more acceptable. As a result, we can consider the model useful for predicting the presence or absence of droughts.

Next, we will examine whether incorporating LASSO regularization into the objective function, thus shrinking some coefficients, can enhance the model's performance, given the high number of variables involved.

4.4 Logistic Regression With Lasso Penalization

As we did for ordered logistic regression, we incorporated Lasso penalization into the objective function:

$$\min_{\beta} \left\{ -l(\beta) + \lambda \sum_{j=1}^{p} |\beta_j| \right\}$$

Metric	Value
Sensitivity	0.7957
Balanced Accuracy	0.7437

Table 9: Performance metrics of the model.

The L1 penalty promotes variable selection, enhancing model interpretability and reducing the risk of overfitting.

The model was optimized by selecting the regularization parameter λ through cross-validation. The optimal λ was chosen as the value that minimized classification error within one standard deviation. Essentially, we selected the most parsimonious model that was not significantly different from the one that achieved the lowest misclassification error. As shown in Figure 7, this corresponds to the second vertical dotted line from the left. The model is now way more simple than before, the number of variables almost halved.

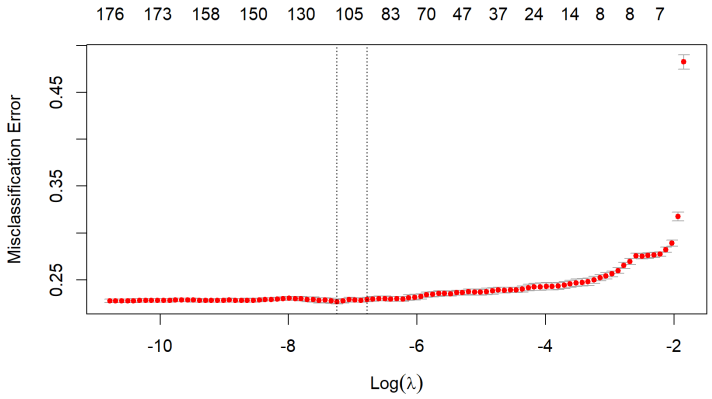

Figure 7: Selecting the λ value

Once the model was trained on the training set (data up to 2018) with the optimal lambda value, we evaluated different decision thresholds to convert the predicted probabilities into binary classes, focusing on maximizing balanced accuracy. To achieve this, we used the data from 2019 as the validation set to identify the threshold that best balanced model performance (0.45 as we can see in Figure 8).

After identifying the optimal threshold, we retrained the model on the entire training set (former training set + validation) and applied the selected threshold to obtain predictions on the test data (2020). The final evaluation was conducted using a confusion matrix, calculating metrics such as balanced accuracy and sensitivity, and specificity, confirming the model's predictive ability.

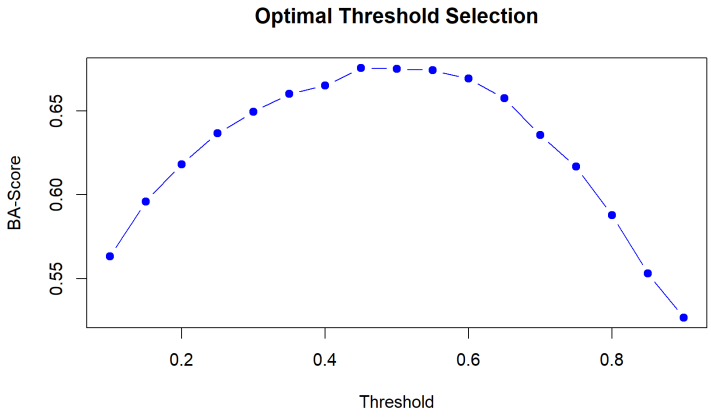

Figure 8: Selecting the threshold value

	Reference	
Prediction	**0**	**1**
0	2105	835
1	710	2538

Table 10: Confusion matrix comparing predicted and ground truth values.

Metric	**Value**
Sensitivity without Lasso	0.7957
Sensitivity with Lasso	0.7524
Balanced Accuracy without Lasso	0.7437
Balanced Accuracy with Lasso	0.7501

Table 11: Performance metrics of the model.

We can highlight that the Balanced Accuracy has slightly improved. This occurs even if sensitivity decreases because the new model performs better at correctly predicting the 0 values.

4.4.1 Group Lasso Penalization

Another approach within the setting of logistic regression with lasso penalization is to apply the group lasso method, where variables are grouped based on their meaning (*Meier, van de Geer and Bühlmann, 2008* [6]). Specifically, we can group all lagged versions of the same variable together and decide whether to remove them entirely as a group rather than individually.

While the structure of the model remains the same as standard logistic regression, the objective function changes to accommodate the group lasso penalty. Instead of penalizing each coefficient separately, the

variables are divided into predefined groups, and the penalty is applied to the entire group of coefficients. The modified objective function becomes:

$$\min_{\beta} \left\{ -\sum_{i=1}^{n} \log p_{g_i}(x_i; \beta) + \lambda \sum_{k=1}^{m} w_k \|\beta_{V_k}\|_2 \right\}$$

where:

$$\|\beta_{V_k}\|_2 = \sqrt{\sum_{j \in V_k} \beta_j^2}$$

and w_k are the square roots of the group sizes. They are used to rescale the penalty with respect to the dimensionality of the parameter vector β_{V_k}.

This technique allowed us to identify which groups of variables are less important for prediction purposes. We performed cross-validation on the training set to tune the regularization parameter λ, controlling the degree of shrinkage of the coefficients. As shown in Figure 9, as the value of the lambda coefficient increases, the weight of the Lasso penalty on the objective function also rises, causing the variables to be increasingly shrunk toward zero. For this model, we are not primarily concerned with the exact results and performance

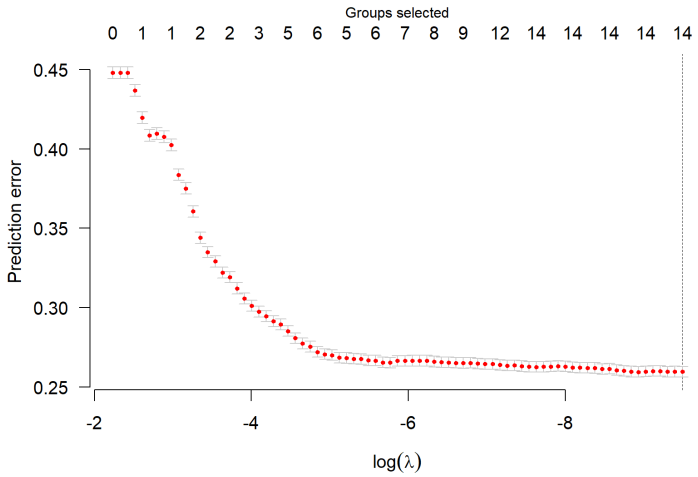

Figure 9: Variable shrinkage when λ changes

but rather with interpretability. Specifically, we want to understand the order in which groups of variables are shrunk to zero, allowing us to identify the most important variables for the classification task.

Lambda	Variable
0.00058	TS
0.00064	WS50M
0.00085	WS50M_MAX
0.00093	WS10M_MIN
0.00123	T2MWET
0.00236	WS50M_MIN
0.00342	QV2M
0.00375	T2M_MIN
0.00790	WS10M_MAX
0.01381	month
0.01381	WS10M
0.02198	T2M_MAX
0.04627	PRECTOT
0.09740	PS

Table 12: λ values that shrink each variable to zero

We can observe that when the penalty term is introduced, the first variables to shrink are the Earth skin temperature and wind-related variables. Interestingly, only one wind variable (wind speed at 10 meters) remains important, suggesting that the wind variables may provide similar information and only one is necessary (we could also presume this form the correlation matrix in Figure 1). This aligns with the expectation that wind has a smaller impact on drought prediction compared to other factors, such as precipitation. On the other hand, temperature variables, pressure, the month of the year, and, of course, precipitation, seem to play a much more significant role in predicting drought levels.

4.5 Decision Tree

Finally, we applied a machine learning technique for the classification task. These methods were chosen for their interpretability, as they allow us to identify the most relevant variables in our analysis.

We began by implementing a Decision Tree classifier. In this approach, the algorithm sequentially determines the best split at each node, selecting the most informative variable and the corresponding threshold. The goal is to minimize an impurity measure, which in our case is the Gini index. This process continues recursively until a stopping criterion is met, such as a minimum number of observations per node or a maximum tree depth.

One of the key aspects of Decision Tree models is the selection of hyperparameters, which significantly impact performance. To optimize the model, we applied Bayesian optimization to tune three main hyperparameters: the maximum depth of the tree, the minimum number of observations required in each node, and a complexity parameter that determines whether a split should be performed. The optimization objective was to minimize the Balanced Error Rate (BER), a metric that accounts for class imbalance, using a 5-fold cross-validation procedure.

Once the optimal hyperparameters were identified, we trained the model while applying observation weighting to mitigate class imbalance. The final model was then fitted to the test set.

The classification results revealed that the model performed reasonably well for the majority class None, but struggled to accurately classify the remaining classes. To further analyze the model's decision-making process, we visualized the first few splits of the Decision Tree (Figure 10). From this, we observed that the variable PRECTOT and its lagged values played a crucial role in the classification, highlighting their significance in predicting the target variable.

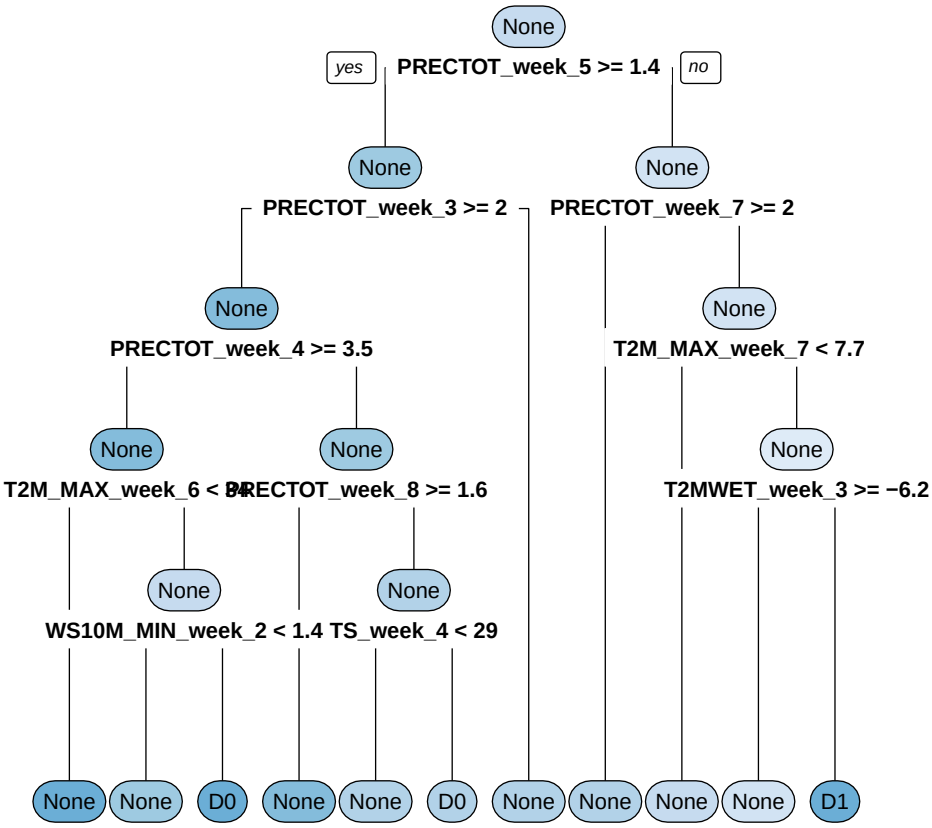

Figure 10: First nodes of the decision tree

4.6 Random Forests

Given the limitations observed with the Decision Tree approach, we extended our analysis using Random Forests, an ensemble method that aggregates multiple Decision Trees to improve overall classification accuracy and robustness. In Random Forests, each tree is trained on a bootstrap sample of the data, and at each split, only a subset of the total features is considered. This method reduces variance and enhances generalization, making it a powerful alternative to a single Decision Tree. The final classification is determined

through majority voting across all trees.

As with the Decision Tree, we applied Bayesian optimization to tune the key hyperparameters of the Random Forest model. The parameters optimized included the number of trees in the forest, the maximum depth of each tree, and the minimum number of observations required at each split. The objective remained the minimization of the Balanced Error Rate (BER), using 5-fold cross-validation to ensure a robust selection of hyperparameters.

After tuning, we trained the model on the training set and applied it to the test data. Initially, we ran the Random Forest without any adjustment for class imbalance. As a result, the vast majority of observations were classified as None, leading to poor overall performance. To address this issue, we applied class weighting, assigning higher weights to underrepresented classes and lower weights to more frequent ones. This approach aimed to make the model more sensitive to minority classes by increasing their influence during training.

The weighted Random Forest model showed slight improvements in classification performance. In particular, there was an increase in predictions for classes D0 and D1, though the model still struggled to correctly classify the rarest classes. Given these limitations, we explored an alternative strategy to refine class assignments.

To further address the imbalance in the target variable, we implemented a threshold-based approach. In standard classification, an observation is assigned to the class with the highest predicted probability. Instead, we introduced class-specific probability cutoffs, meaning that an observation was only assigned to a class if its probability exceeded a predefined threshold. More frequent classes were given higher cutoffs, ensuring that rare classes were not overshadowed.

Despite these refinements, the performance remained similar to the weighted approach, with the model still struggling to classify the minority classes effectively.

Overall, our machine learning implementations did not lead to a substantial improvement in classification performance compared to statistical models. The sensitivity results for these methods are summarized in Table 13.

Finally, we computed feature importance measures for the weighted Random Forest model using both the Gini Index and Accuracy criteria. As shown in Figure 11, the most influential variable was PS, followed by PRECTOT, T2M_MAX, and month. These findings align with the results from the Decision Tree model and Group Lasso analysis, reinforcing the significance of these variables in predicting the target outcome.

Sensitivity	**Class: None**	**Class: D0**	**Class: D1**	**Class: D2**	**Class: D3**	**Class: D4**
Decision Tree	0.702	0.275	0.173	0.076	0.062	0.000
Random Forest	0.934	0.187	0.085	0.006	0.009	0.000
Random Forest using weights	0.562	0.486	0.296	0.090	0.122	0.016
Random Forest using different cutoff	0.662	0.310	0.204	0.118	0.180	0.080

Table 13: Model sensitivity for each class

5 Combining Regression And Classification

As last approach, we combined regression and classification models to separately handle the zero observations and the positive counts. This should help dealing with the excess of zeros and allows to work with a response

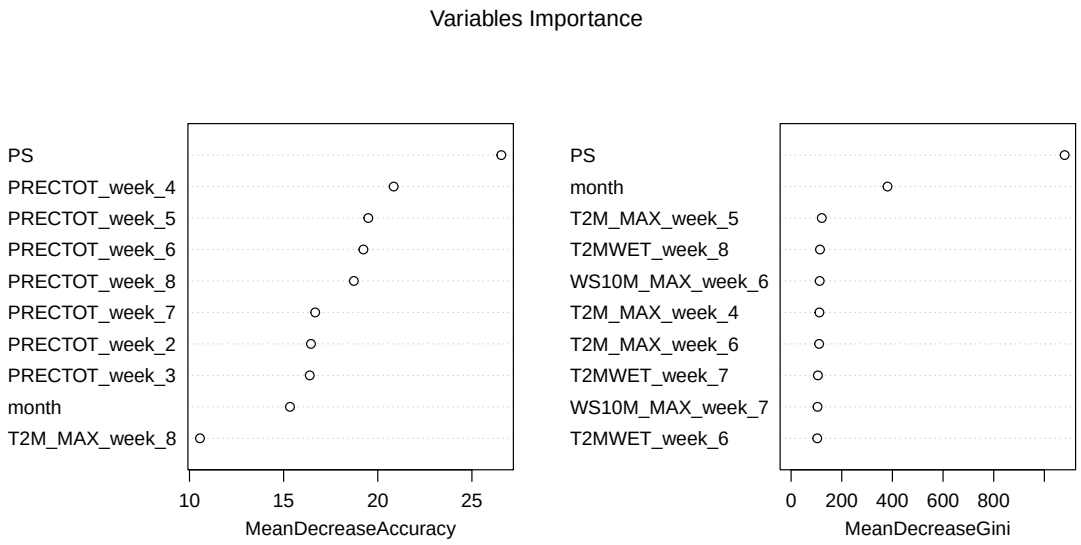

Figure 11: Variables importance in weighted Random Forest

variable that takes values from closed unit interval [0,1]. We considered the binary variable equal to one if the score is exactly zero, and zero otherwise and trained a logistic regression model using this as response. Then, we selected only the observations with positive score and fitted a beta regression model on this continuous response. The predictions for new observations are made combining these two parts: if the probability of the logistic model of being zero is greater than 0.5 we assign zero, otherwise we use the other estimated model.

This model accounts for the zero observations, such as the Beta Inflated model, but they differ since in this case we actually have predicted values equal to 0, whereas the Beta Inflated model didn't produce exact zeros.

Once again, we fitted the model on the training and then made predictions on the test. The model's performances are presented in Table 14.

Metric	**Training Data**	**Test Data**
Standard Deviation (SD)	0.1982	0.2565
RMSE	0.1806	0.2176
MAE	0.1121	0.1445

Table 14: Binary Model Performance on Training and Test Data

We can see a slight improvement of both RMSE and MAE on the test set: this means that this model performances better on unseen data than the simple beta model. This confirms that separately dealing with the zero values helps during the prediction step.

6 Conclusions

In this study, we explored the potential of meteorological data to predict drought conditions using both traditional statistical models and machine learning techniques, such as random forest. Our goal was to assess whether these models could provide reliable predictions, reducing the human effort required for drought monitoring and analysis.

First, we tried to predict the exact drought score using regression models. Given the constrained nature of the response, we fitted a Beta regression. However, it had limited efficacy in predicting drought levels, often underestimating the response. Even its inflated version, designed to handle zero values, failed to provide a significant improvement. These results suggest that beta regression may not be well-suited for this type of data, particularly given the complexity and variability of drought conditions.

As next step, we approached the prediction task from a different point of view, using the categorized version of the drought level. Nevertheless, the classification models still struggled to accurately predict the ground truth. In particular, we observed that intermediate classes were rarely identified correctly by the model. Techniques such as penalization and strategies to address class imbalance provided only marginal improvements, indicating that a simple classification approach might not be sufficient for this problem.

We were actually able to achieve a significant improvement in the performances simplifying the problem to a binary classification task. However, this simplification may be too drastic, as it overlooks the important distinctions between mild and severe drought levels, which could provide valuable information for more intermediate predictions.

One potential solution we explored was the combination of regression and classification models, as discussed in the previous section. While this hybrid approach led to some performance improvements, the overall reliability of the predictions remained limited. These findings suggest that the assumptions behind our models may not align well with the data at hand.

To further test this hypothesis, we applied machine learning algorithms such as decision trees and random forests, as they are more flexible and capable of capturing non-linear patterns in the data. Even though they showed greater adaptability, they still did not achieve the level of accuracy required to replace human labor.

We believe that the main obstacle in this study was the computational cost. Due to resource constraints, we had to make several compromises during the analysis: we restricted to a subset of counties, considered a limited time span, and reduced the use of dummy variables (excluding those for individual counties and limiting time-related ones to monthly dummies). Including these additional predictors would have drastically increased the model's complexity, making computations unmanageable with our resources.

Looking ahead, potential improvements to this study could focus on addressing these limitations by including more counties, extending the temporal span and incorporating additional predictors (such as soil moisture or satellite-derived vegetation indices). Moreover, leveraging more computationally efficient modeling techniques would help optimize the performance. Future research could also explore deep learning approaches, which might better capture complex spatial and temporal dependencies in drought data.

While our findings suggest that the models tested in this study are not yet suitable for fully automating drought prediction, they provide a introductory view into the challenges of using meteorological data for this purpose. Further research and efforts will be necessary to develop more accurate and efficient solutions for drought monitoring.

References

[1] *US Drought Meteorological Data*. Available at: https://www.kaggle.com/datasets/cdminix/us-drought-meteorological-data.

[2] U.S. Drought Monitor, *What is the U.S. Drought Monitor?*, 2024. Available at: https://droughtmonitor.unl.edu/About/WhatistheUSDM.aspx.

[3] National Drought Mitigation Center, *Drought Impacts*, 2024. Available at: https://droughtimpacts.unl.edu/Home.aspx.

[4] S. L. P. Ferrari and F. Cribari-Neto, *Beta Regression for Modeling Rates and Proportions*, Available at: https://www.ime.usp.br/~sferrari/beta.pdf.

[5] Y. Min and A. Agresti, *Random effect models for repeated measures of zero-inflated count data*. Statistical Modelling. 2005; Volume 5, Issue 1:1-19. Available at: https://journals.sagepub.com/doi/abs/10.1191/1471082X05st084oa?journalCode=smja.

[6] L. Meier, S. van de Geer and P. Bühlmann, *The group LASSO for logistic regression*. Journal of the Royal Statistical Society Series B. 2008. Available at: https://stat.ethz.ch/Manuscripts/buhlmann/logistic-grouplasso-final.pdf.