IFT 6390 - Data Challenge 1

Axel Bogos
SID: 20091252
Kaggle username: axelbog
Département d'informatique et de recherche opérationnelle
Université de Montréal
Montreal, QC
axel.bogos@umontreal.ca

Introduction

Now more than ever, tackling climate change and minimizing the damage caused by extreme weather events is at the forefront of humanity's most pressing issues. As extreme climate events have and are predicted to continue becoming both more frequent and costly (NOAA National Centers for Environmental Information 2021), having reliable predictors of extreme climate events based on atmospherical and geographical data may be a useful tool to both the general population and policy-makers. In this Kaggle challenge, done in the context of IFT 6390, the aim is to develop such a predictor. The challenge provides a subset of the ClimateNet dataset (Prabhat et al. 2021) consisting of 47,760 training data points and 7,320 test data points spread over 120 locations, observed between 1996 and 2010. There are 3 distinct class labels in this subset of the data. In the following document, we describe the pre-processing and feature selection and engineering that was done, present the classification algorithms explored, namely multinomial logistic regression, random forest, Light Gradient Boosted Machine (henceforth referred to as LGBM) and Extreme Gradient Boosting (henceforth referred to as XGB) and finally discuss the evaluation methodology, results and possible future improvements.

Feature Design & Selection

The provided dataset consisted of 19 features columns; they are described in table 7. Note that this whole section, beside the very first sub-section regarding scaling is not applicable to the implementation and testing of the logistic regression predictor. A separate, minimally processed data-set was used for both the hyper-parameter search and the predictions submission of the logistic regression predictor. This data-set was solely scaled (according to the procedure below) and the meta-features *S.No* (which was really an observation index) and *time* (which we use for feature engineering in further models, but for now we consider as a meta-feature) were dropped.

Scaling Features

As the provided features represent a variety of range and units such as m/s, Kelvins, kg/m^2 and so forth, it was decided to scale features with a standard scaling with mean 0 and standard deviation of 1. To this effect, *sklearn*'s *StandardScaler* module was used. However; 3 features were not standardized in such a way:

1. *latitude* and *longitude* features were not standardized as they are used for geo-spatial visualizations and feature engineering later on;
2. The feature *time* was not standardized as it not a numerical feature; furthermore it is also used for feature engineering later in the process (or dropped in the case of logistic regression).

Collinearity

An analysis of the correlation between features revealed that some of the features were very correlated as shown in table 1.

Table 1: Distinct Pairs of Features with Correlation ≥ 0.9

	PSL	TREFHT	Z1000	Z200	ZBOT
QREFHT	NA	0.932	NA	NA	0.953
PS	0.999	NA	0.984	NA	NA
TS	NA	0.968	NA	NA	0.955
TREFHT	NA	NA	NA	NA	0.997

Hence, considering all the column features in table 1, we can reduce the redundancy and dimensionality of the dataset by dropping features *PSL, TREFHT, Z1000, Z200* and *ZBOT*.

Missing Values

There was no missing values in the dataset, hence no work was necessary to either impute or deal with the missing values.

Geographical Locations

The geographical data was provided in the form of latitudes and longitudes coordinates. Upon inspection through geospatial visualization, it became apparent that that 120 locations were in fact part of clusters.

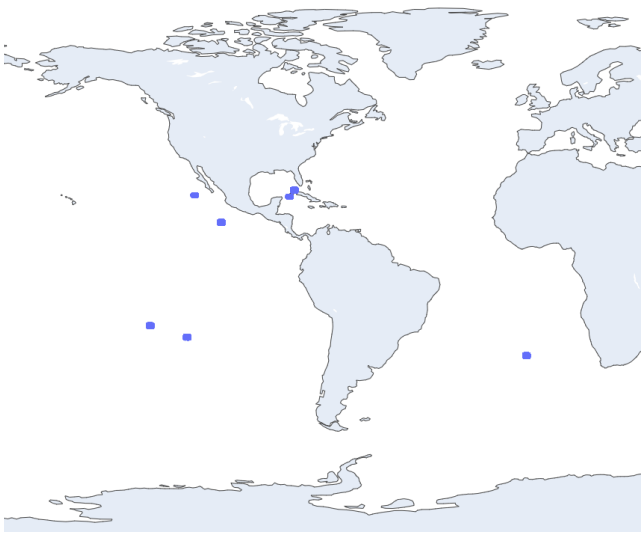

Figure 1: Geographic Locations of Data Points.

7 clusters could be clearly identified; 4 in the Pacific Ocean, 2 in the Caribbean Sea and 1 in the Southern Atlantic Ocean as shown in figure 1 (a higher resolution version of this figure is available in the appendix under figure 6). As such, K-Means clustering ($K = 7$) was done onto the latitudes and longitudes features in order to encode this feature with cluster labels ranging from 0 to 6. Visualizations of the K-Means results corroborated the clusters directly observable as shown in in figure 4, to be found in the appendix. Our handling of such categorical features will be shortly discussed.

Our intuition for using this feature encoding technique is that while latitudes and longitudes are continuous values which may not be strong features for a classifier, we expect physically nearby locations to be more similar to one-another and hence encoding the physical region as a categorical variable may be a more simpler representation of this geographical dimension.

Temporal Features

As the *time* feature is represented in the original data set in the form *YYYYMMDD*; our intuition is the following. The day should not be meaningful in any way; there is no sound reason for an extreme weather event to occur more on a given day (1-31) of an arbitrary month. If there are such patterns in the data, it would rather reflect a collection bias (for example, certain observation or measurement apparatus may only be available certain days), hence the ”day” part of the time feature may be safely discarded. Next let us consider the year. While a ”year” feature may be a meaningful feature to discriminate training data; ultimately our goal is to *predict* future events; hence discerning weather patterns that may be specific to a given year may not be a wise decision from a generalization point of view. Finally, we consider the months. As the weather is a cyclical pattern, months intuitively seem like the most correlated piece of temporal information with extreme weather events. Hurricanes, floods, droughts and so forth are all generally correlated with certain months of the year. Hence we proceed by re-encoding the *time* feature as solely the **month** of the observation as a digit from 1 to 12.

Categorical Features

Given the relatively low dimensionality of each of our 2 categorical features (the geographical clusters having 7 distinct values and the months having 12 distinct values), it was decided to represent these categorical features through one-hot encoding. Hence we create features *geo_cluster_X* where $X \in \{0, 1, 2, 3, 4, 5, 6\}$. and similarly with *month_X* with $X \in \{1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12\}$.

Algorithms

Logistic Regression

A multinomial logisitc regression classifier was implemented from scratch with the exception of the use of the Python Standard Library, *Numpy* and *Pandas*. The multinomial logistic regression classifier is an extension of the standard binary logistic classifier; it relies on cross-entropy loss instead of binary logistic loss and the *softmax* function serves a similar role as the sigmoid function in binary classification (that is to say, it transforms our input to a smooth function where we may either pick a threshold or apply *argmax* in the binary and multi-class cases respectively). As mentioned before, it was trained on a minimally processed version of the data-set.

Random Forest

The Random Forest algorithm is an ensembling method that relies on the aggregation-or the *bagging*- of numerous simple learners, in this case decision trees, where each simple learner is trained on a bootstrapped sample of our data set. The final class is determined by a majority-vote of these simple learners. Although simple in nature, this method is experimentally quite successful and is among the algorithms that have achieved our highest scores.

XGB & LBGM

Both XGB and LGBM are implementations of the extreme gradient boosting decision trees algorithm (and successors of Adaboost). While like random forests this algorithm is also based on ensembling, it uses boosting instead of bagging. This consists of iteratively training simple learners (in this case a decision tree) and *re-weighting* the cost associated with misclassified data points, increasing or decreasing this cost depending on whether it was misclassified or not by the simple learner. XGB and LGBM differ in the implementation rather than the theoretical background and motivation. They notably differ in how they implement the search for a new split in the decision tree; for our purpose this results in slightly different sets hyper-parameters to tune for each model and faster execution by the LGBM models. Each model also slightly differs in the input they accept; notably XGB cannot handle categorical variables if they are not one-hot or label encoded, while the LGBM implementation can.

Methodology

In this section, beside the general methodology, particular interest will be given to the methodology of the logistic regression, random forests and LGBM models. The reason for this is our methodology for XGB models is very similar to the one employed for LGBMs and it has not resulted in our best results, hence we will not dedicate a specific discussion to it.

Train-Validation Split

For all trials and model experimentation, *sklearn*'s *train-test-split* was used with a constant random seed, a 0.25 validation set size and stratified on the labels in order to ensure an equivalent class distribution in either train and validation sets. In practice, our test set was constituted of the provided test-set and the hidden labels on Kaggle.

Logistic Regression

In our implementation of multinomial logistic regression, a number of regularization and optimizations were also implemented; namely L_2 regularization, batched gradient descent, learning rate decay and early-stopping when the gradient approached zero. An overview of these hyper-parameters value can be seen in 2.

Table 2: Logistic Regression Hyper-parameters

Hyper-parameter	Value
Batch size	100
Learning rate	0.1
Regularization Constant	0
LR Decay	0.01
Number of epochs	1000
Grad. Tolerance	0.001

Notably, across multiple experiments, it was observed that the regularization constant did not help with getting better results. Quite the opposite, setting the regularization constant to any value ≥ 0 yielded worst results. One reason for this observation might be that given we are using a relatively simple discriminant function, our decision boundary has no occasion to overfit our relatively high-dimensional data set. Observing the training curves in figure 2, we see that despite using a high number of epochs, we see no signs of over-fitting, hence this seems to confirm our experimental intuition that L_2 regularization was not necessary.

The value of other hyper-parameters, most importantly the batch size and the learning rate (these two hyper-parameters had the largest impact on the final validation accuracy), was found by manual binary search.

Random Forest

Our Random Forest estimator used the *sklearn* library for both the implementation of the algorithm and the hyper-parameter tuning. After establishing a baseline with the default parameters of the *sklearn* Random Forest, Randomized

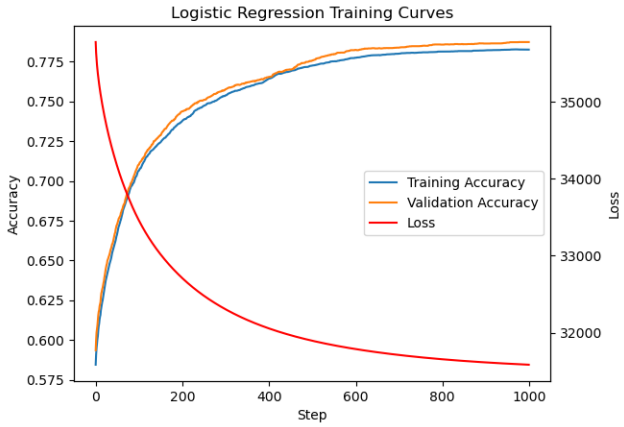

Figure 2: Logistic Regression Training Curves

Search with cross-validation was done to find the hyper-parameters shown in table 3. Those that are not mentioned are have been left to their default values.

Table 3: Random Forest Hyper-parameters

Hyper-parameter	Value
n_estimators	2000
min_samples_split	10
min_samples_leaf	4
max_depth	50
max_features	auto
bootstrap	True

Randomized Search with cross-validation was chosen instead of grid-search given that we wanted to search over a large hyper-parameter space. The randomized search was conducted with 200 iterations of 3 cross-validation splits for a total of 600 random fits. The range for which each hyper-parameter was searched and the number of random samples in each range if applicable can be found in table 9. Attempts at countering the imbalance in the data set by manipulating the *class_weight* attribute of the model did not seem to yield better result and it was hence left untouched since our model went over the baseline without this extra tuning.

LGBM

The LGBM hyper-parameter selection and optimization has been a more convoluted affair. Two methods have been experimented with, the initial being only an attempt at a quick trial with an optuna optimization, the second which was more thoroughly explored and experimented with a randomized grid search. optuna is a hyper-parameter search framework which optimizes the search process by pruning unpromising trials. Albeit more effort was spent with the Randomized Search method, this initial attempt with optuna surprisingly remained the best attempt on Kaggle (more-

over; this initial attempt was done before the complete pre-processing of the data). Hence both methods will be described.

Optuna Attempt As the this attempt was the very first one, the data it was ran on was slightly different in its pre-processing. There was two main differences: the features had not yet been scaled in a standard fashion, and the categorical variables were not one-hot-encoded. Both of these pre-processing method actually do not affect the LGBM model: as mentioned earlier, LGBMs have an inner processing function for dealing with categorical variables given the right data-type. The hyper-parameter search was done using *optuna*, which is a hyper-parameter search framework which accelerates hyper-parameter tuning by pruning unpromising random trials. It also uses cross-validation folds across the search process. In this case, 5 cross-validation folds were used with 20 trials for a total of 100 random fits. Table 4 displays all hyper-parameters found through the optuna search. The range of the search space can be found in the appendix in table 10. Note that due to optuna being quite a bit more optimized than Randomized Search, we are able to search a significantly larger parameter space. Unmentioned hyper-parameters have been left to the default values of the *lightgbm* library. The most important parameters to optimize were the learning rate, the number of estimators, the number of leaves and finally lambda_l1 and lambda_l2 for regularization.

Table 4: LGBM Hyper-parameters (Optuna Attempt)

Hyper-parameter	Value
learning_rate	0.12696
bagging_fraction	0.4
feature_fraction	0.9
lambda_l1	75
lambda_l2	35
max_depth	9
min_data_in_leaf	300
min_gain_to_split	0.96961
n_estimators	8840
num_leaves	1540

Randomized Search with Cross-Validation This attempt was made after the standard pre-processing described in the previous section, that is to say the features were scaled in a standard manner and the categorical features were one-hot encoded. Here, the hyper-parameter search was done using *sklearn*'s Randomized Search with cross-validation. An overview of the search space can be found in table 11. Overall, 30 trials were done with 5 cross-validation splits for a total of 150 random fits. The best found hyper-parameters are found in table 5. Unmentioned parameters are left to their default value. Since the data set is imbalanced, the scoring of random fits was done using the area under the curve metric, weighted in a one-versus-rest fashion instead of simple accuracy.

Table 5: LGBM Hyper-parameters (Randomized Search Attempt)

Hyper-parameter	Value
learning_rate	0.06
reg_lambda	1.333
reg_alpha	1.167
max_depth	11
min_data_in_leaf	230
n_estimators	200
num_leaves	40
colsample_bytree	1.0

Results

An overview of the performances and results in both the private and public scores can be found in table 6. Unfortunately, the TA baseline public scores are not available since the competition has closed and I had not taken note of them, hence only the final public score of the latter two baselines are shown in the table. Before discussing each models results in more detail, we point out that a few general observations about the results. First, the implemented logistic regression has beaten the logistic regression baseline, and both the best random forest and LGBM model have beaten the last baseline. Second, we see an improvement in the private scores for all models.

Table 6: Performances

Model	Best Public Score	Best Private Score
Log. Reg. Baseline	NA	0.73196
Implemented Log.Reg	0.74480	0.75136
XGB	0.74669	0.75218
TA Baseline	NA	0.76530
Random Forest	0.75081	0.76557
LGBM	0.76065	**0.76639**

Logistic Regression

The multinomial logistic regression result is quite good in multiple regards; it has beaten the baseline by a relatively significant margin, but it also nearly approaches the performance of a generally much more powerful and general model: XGB. The implication in this regard is two-fold: the hyper-parameter for the logistic regression have been aptly optimized (while still preventing over-fitting as the private score has also improved relative to the public score), while the XGB optimization has been more poorly achieved.

Moreover our intuition based on the training curves that the model was not in fact over-fitting despite a high number of epochs seems most likely.

Random Forest

The Random forest model achieved performances above the baseline with minimal hyper-parameter tweaking (a single-run of the randomized search was completed to beat the baseline). Furthermore, it shows the highest increase in performance (0.01476) between the public and private score, indicating good generalization of the model. Despite having a large number of estimators (2000) and a high maximum depth (50), the tree was not over-fitting. In fact, these are the maximum values for either parameter in our randomized search, hence perhaps even higher values (and hence a higher capacity) model may have been used. It was decided against this by reluctance to even slightly overfit the training data, since it was apparent that the test and training set had relatively different distribution of classes (or, in other words, different class imbalance). Due to the interpretable nature of random forest through an aggregation of tree predictors, we are also able to gain from our random forest predictor an idea of the learned feature importance. While this information was not utilized to enchance other models in this case, it still highlights the relevancy of random forests. The feature importance of our random forest classifier may be found in the appendix figure 5.

LGBM

While our LGBM model has achieved the best final performance, it has to be said that most efforts were (probably wrongly so) spent around optimizing it with sklearn's Randomized Search. Nevertheless, its best result beat the highest baseline and it improves its private score relative to the public score. Upon many trials, it has been observed that the LGBM model was likely sensible to over-fitting while searching for hyper-parameters in a reasonable range (based on search computation time). This intuition was the reason for the very first optuna result to seem suspiciously guilty of over-fitting, given its high number of estimators and number of leaves. Upon reflection, it is apparent that it was not in fact over-fitting, and rather simply found better combinations of hyper-parameters by searching a larger space than the more computationally expensive Randomized Search. Despite already yielding the best result, we believe the LGBM model has a lot of room for improvements compared to the logistic regression implementation and the random forest model. Figure 3 displays the multi logloss against the iterative trials of the optuna parameter search. As experimentally observed, it quickly converges to a local minima. Note that a larger version of this figure is available in the appendix at figure 7.

Discussion

Pros, cons, mistakes to avoid and ideas for improvements that were unfortunately not explored will be discussed for each of the main sections of the process.

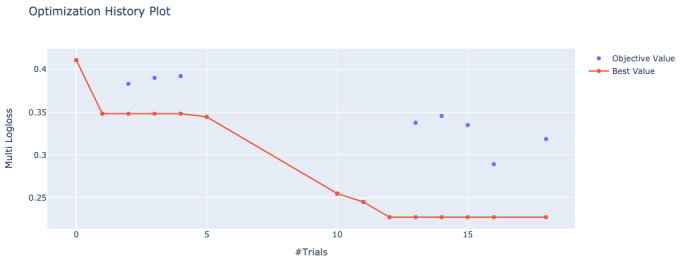

Figure 3: Optuna Hyper-Parameter Search

Data pre-processing

Decisions made at this stage have rippled through the project and may have had unintended consequences. More experimentation could have been done with different versions of the geographical cluster encoding: perhaps reducing the number of clusters (for example to 3: one in the Pacific, one in the southern Atlantic and one in the Caribbean) may have been a better dimensionality/information gain trade-off. Another additional information that may have been considered, particular with regards to the month feature encoding was whether or not the location was in the northern or southern hemisphere. Additionally, more domain knowledge may have helped engineer better features from the provided ones such as *U850*; the feature importance of the random forest predictor shown in figure 5 is a decent proxy for a relative idea of the value of each feature.

Methodology

Perhaps the most significant mistake in the current process was prioritizing the optimization of the LGBM model via Randomized Search over optuna. Better care should have been taken to convince oneself that the optuna optimization was not in fact over-fitting. Another suggestion for further projects and data challenges would be to use an external experiment monitoring framework such as *Weights and Biases* or *comet.ml*. Particularly for the logistic regression model where hyper-parameter search was done by manual binary search and manual metric tracking, the process was error prone and fastidious. Moreover, the initial steps such as data pre-processing should have been evaluated in parallel with a simple model to ensure that the steps taken are helping the classification. For example, a baseline random forest with default parameters could have been bench-marked each updates to the pre-processing pipeline, such as dropping particular features and engineering new ones. Another idea hinted in the random forest section would be to have leveraged the feature importance of a simpler model in the feature selection process.

Statement of Contributions

I hereby state that all the work presented in this report is that of the author.

References

NOAA National Centers for Environmental Information. 2021. U.s. billion-dollar weather and climate disasters.

Prabhat; Kashinath, K.; Mudigonda, M.; Kim, S.; Kapp-Schwoerer, L.; Graubner, A.; Karaismailoglu, E.; von Kleist, L.; Kurth, T.; Greiner, A.; Mahesh, A.; Yang, K.; Lewis, C.; Chen, J.; Lou, A.; Chandran, S.; Toms, B.; Chapman, W.; Dagon, K.; Shields, C. A.; O'Brien, T.; Wehner, M.; and Collins, W. 2021. Climatenet: an expert-labeled open dataset and deep learning architecture for enabling high-precision analyses of extreme weather. *Geoscientific Model Development* 14(1):107–124.

Appendix

Table 7: Data Set Features

Feature	Description
lat	latitude
lon	longitude
TMQ	total (vertically integrated) precipitable water [kg/m^2]
U850	zonal wind at 850 mbar pressure surface [m/s]
V850	meridional wind at 850 mbar pressure surface [m/s]
UBOT	lowest level zonal wind [m/s]
VBOT	lowest model level meridional wind [m/s]
QREFHT	reference height humidity [kg/kg]
PSL	sea level pressure [Pa]
T200	temperature at 200 mbar pressure surface [K]
T500	temperature at 500 mbar pressure surface [K]
PRECT	total (convective and large-scale) precipitation rate (liq + ice) [m/s]
TS	surface temperature (radiative) [K]
TREFHT	reference height temperature [K]
Z1000	geopotential Z at 1000 mbar pressure surface [m]
Z200	geopotential Z at 200 mbar pressure surface [m]
ZBOT	lowest modal level height [m]
time	YYYYMMDD format

Table 8: XGB Hyper-parameters

Hyper-parameter	Value
learning_rate	0.11
n_estimators	150
reg_lambda	1.0
reg_alpha	0.6667
colsample_bytree	0.625
booster	gbtree

Table 9: Random Forest Randomized Hyper-parameters Search Space

Hyper-parameter	Range	Num of Samples
n_estimators	100,2000	10
min_samples_split	[2,5,10]	NA
min_samples_leaf	[1,2,4]	NA
max_depth	[10,110]	11
max_features	[auto,sqrt]	NA
bootstrap	[True,False]	NA

Table 10: LGBM Optuna Hyper-parameters Search Space

Hyper-parameter	Range	Step
n_estimators	100,10000	20
learning_rate	0.01,0.3	10
colsample_bytree	0.5,1	5
num_leaves	20,1000	20
lambda_l1	0,100	5
lambda_l2	0,100	5
max_depth	3,12	10
min_data_in_leaf	20,10000	100
min_gain_to_split	0,15	10
bagging_fraction	0.2,0.9	0.1
feature_fraction	0.2,0.9	0.1

Table 11: LGBM Randomized Hyper-parameters Search Space

Hyper-parameter	Range	Num of Samples
n_estimators	[50,100,150,200,500,1000]	NA
learning_rate	0.01,0.2	20
colsample_bytree	0.5,1	5
num_leaves	10,100	10
reg_alpha	0,1.5	10
reg_lambda	0,1.5	10
max_depth	5,15	10
min_data_in_leaf	10,1000	10

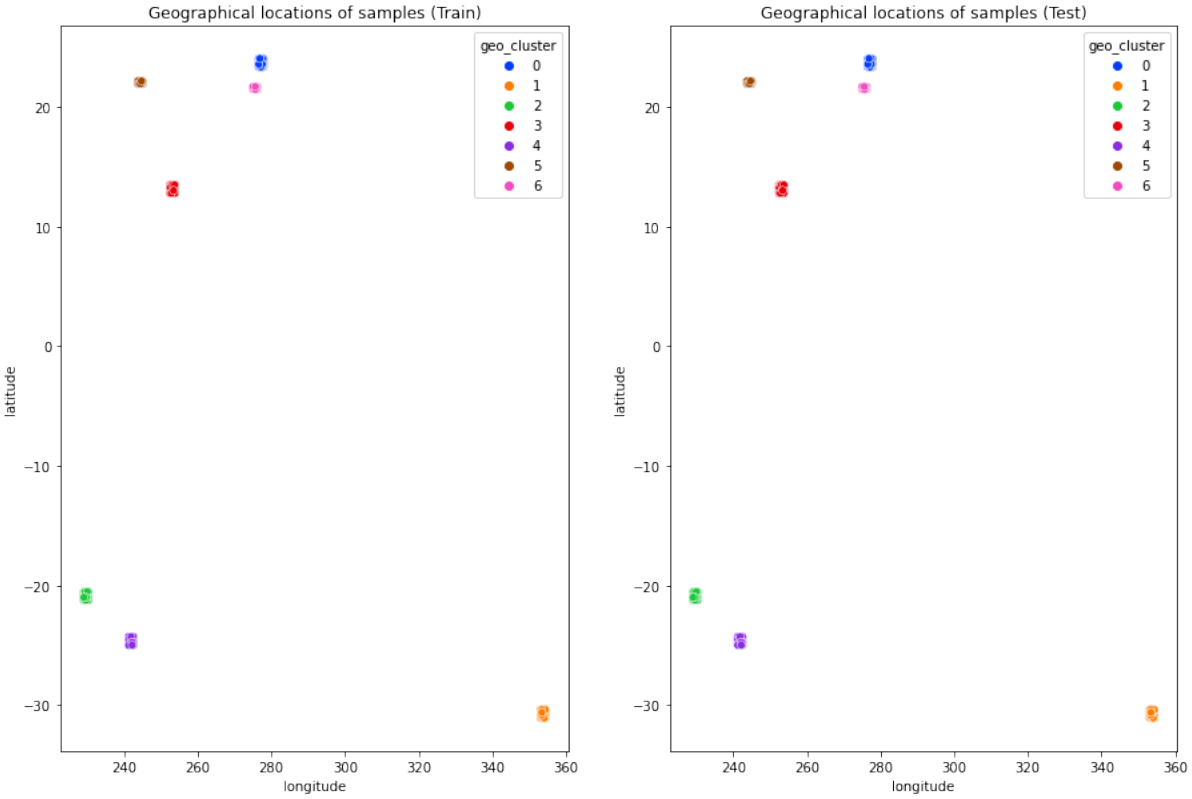

Figure 4: Geographical Clusters predicted by K-Means (K=7)

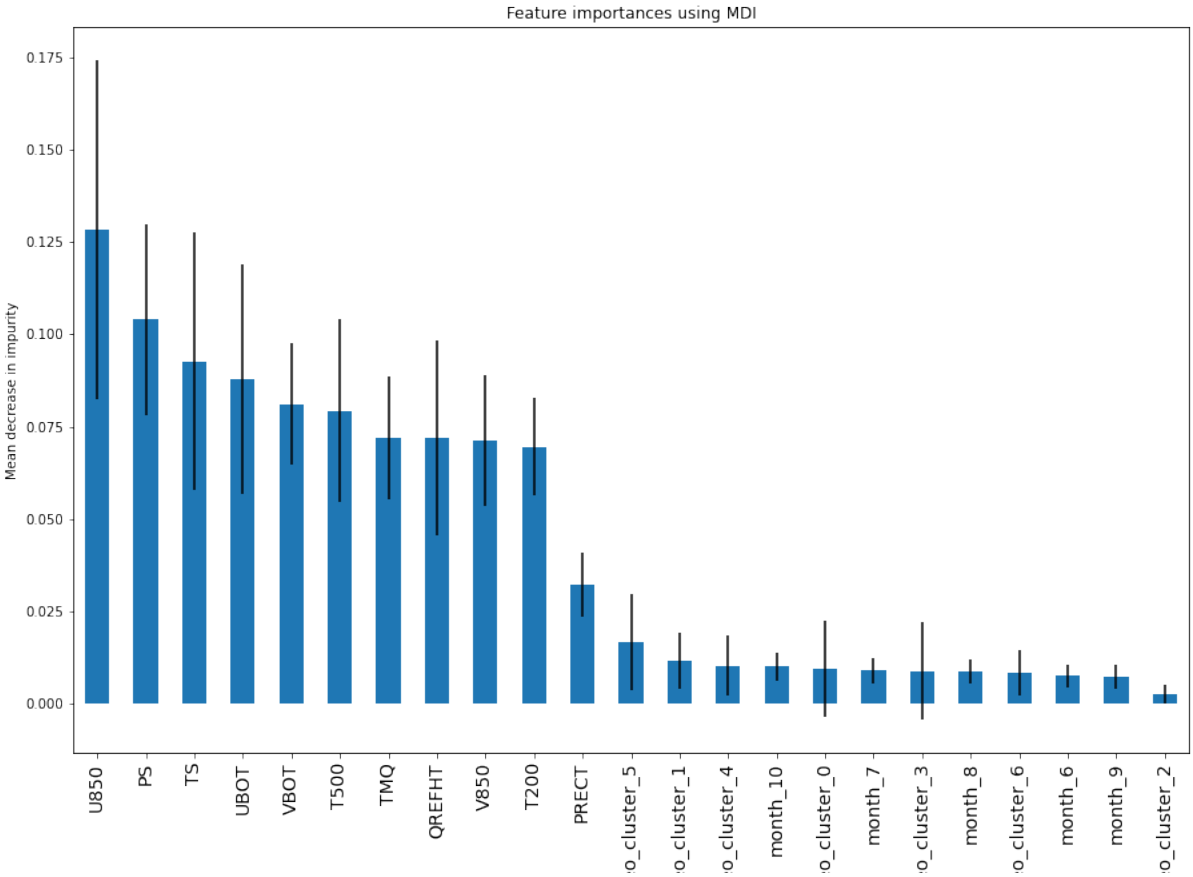

Figure 5: Random Forest Feature Importance by Mean Decrease in Impurity.

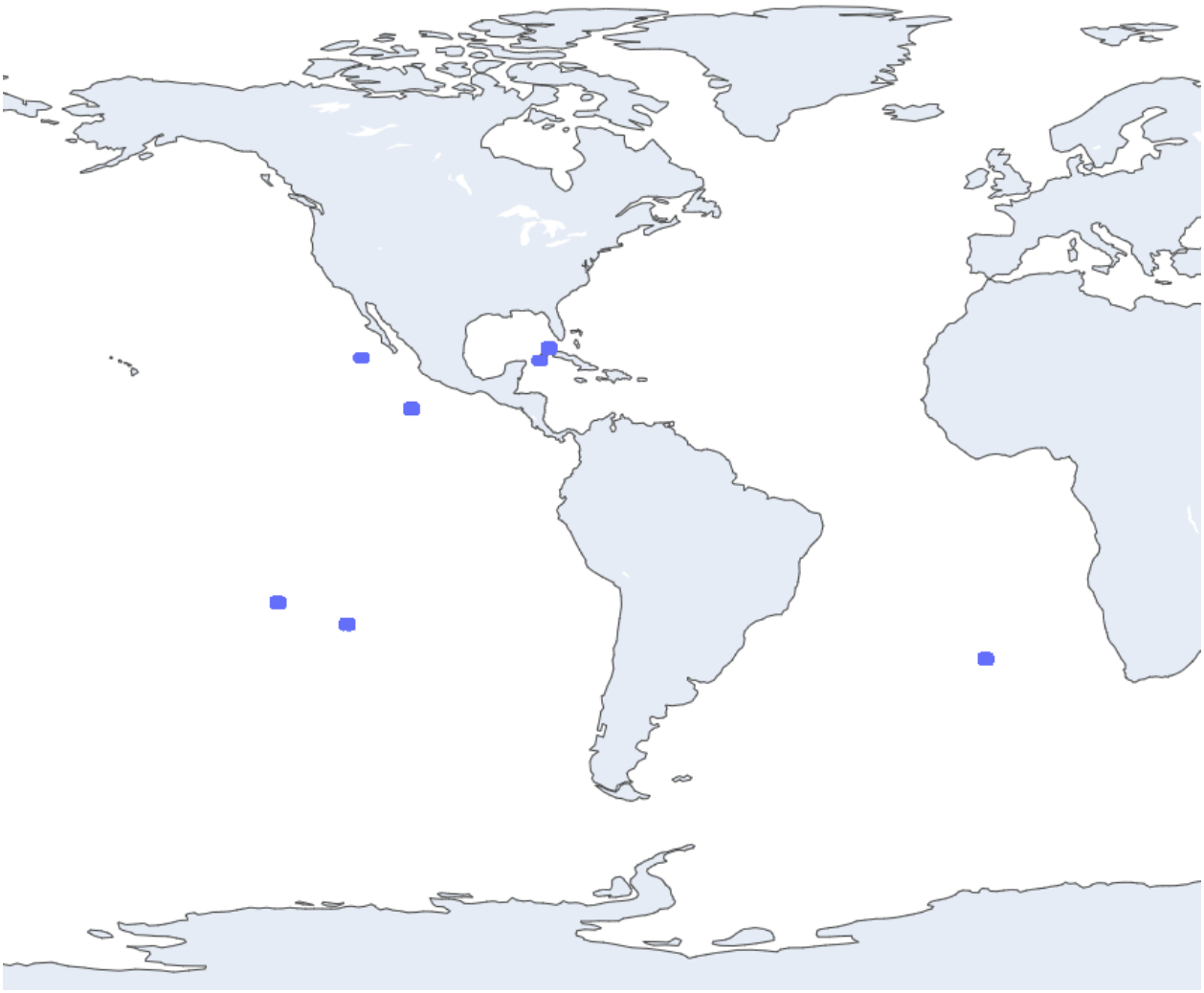

Figure 6: Geographic Locations of Data Points (Large)

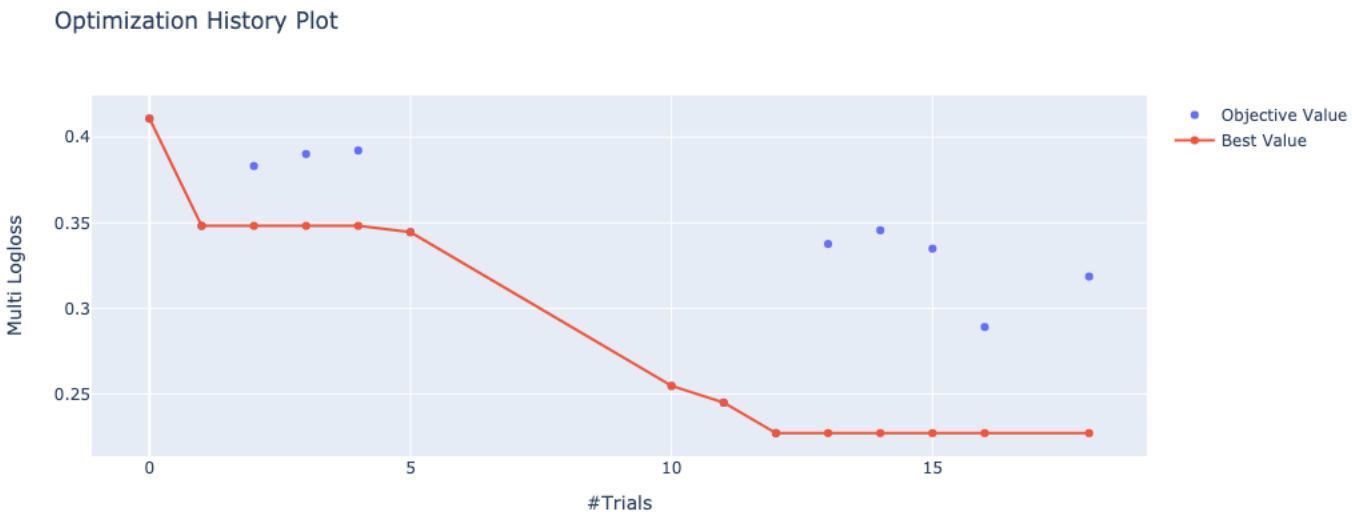

Figure 7: Optuna Hyper-Parameter Search (Large)