Classification of Extreme Weather Events

January 3, 2022

1 Student Information

Full Name: Reza Bayat
Kaggle Username: rezabyt

2 Introduction

Climate change is arguably one of the most pressing challenges facing humanity in the 21st century. Identifying weather patterns that frequently lead to extreme weather events is a crucial first step in understanding how they may vary under different climate change scenarios [1]. In this subset problem, we are going to design an algorithm to classify whether a set of climate variables corresponding to a time point and location is associated with i) standard (background) conditions, ii) a tropical cyclone, or iii) an atmospheric river.

Since the ClimateNet data set is labeled, we will work on a Supervised problem; therefore, we used a few subsets of classification algorithms such as Multiclass Logistic Regression and Random Forest. In the first part, we implemented the Logistic Regression from scratch, which was inspired by [2] GitHub repository. Then, we chose the RandomForest model and trained it with the GridSearch approach to finding a good setup of hyper-parameters that enhance the results. The validation accuracy of MultiClass Logistic Regression is 73.55%, and the accuracy of RandomForest, which was tuned with GridSearch on hyperparameters, is 76.66%.

3 Feature Design

Generally, for all datasets, I follow some steps to get a sense of how good the dataset is:

- Plotting count plot of classes to see if the dataset is balanced or not. (figure 1)
- Plotting correlation matrix using "data.corr()" to see which features are more relevant. (figure 4)
- Looking at some statistics(count, mean, std) of the dataset by using "data.describe().T" (figure 5)
- Checking if features have Null values or not by using "data.isnull().sum()" in order for handling missing values or maybe dropping some useless features.(figure 6)

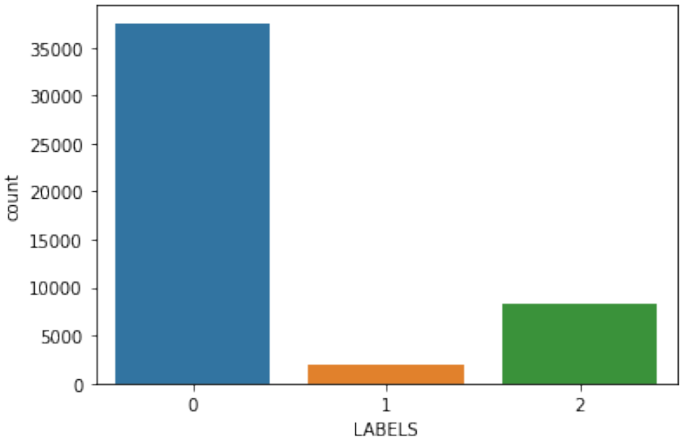


Figure 1: Count Plot

3.1 Logistic Regression

For this part, I didn't make any feature selection, and I just normalized the data set with the z-score.

$$z = \frac{x - \mu}{\sigma}$$

$$\mu = Mean, \quad \sigma = Standard\ Deviation$$

3.2 Random Forest

I only removed the time column since its format(yy-mm-dd) wasn't proper for branching trees, and I also did normalization on all features using z-score.

3.3 Others

I also trained some other algorithms such as Naive Bayes and SVM; however, the validation accuracy was lower than Random Forest, but for all of them also only normalization on the dataset was applied.

4 Algorithms

4.1 Multiclass Logistic Regression

Instead of using the Lab implementation of Logistic Regression, defined for binary class problems, I used another approach that uses Softmax instead of the Sigmoid and also used the Cross-Entropy Loss; therefore, the model's output will be probabilities. In other words, the softmax function computes the possibility that the training sample x(i) belongs to class j given the weight and net input z(i).

Softmax:

$$P\left(y = j \mid z^{(i)}\right) = \phi_{sofmax}\left(z^{(i)}\right) = \frac{e^{z^i}}{\sum_{j=0}^{k} e^{q_k^{(i)}}}$$

Cross Entropy Loss:

$$Loss = -\sum_{i=1}^{n} y_i \cdot \log \hat{y}_i$$

4.2 Random Forest

Random forests or random decision forests are an ensemble learning method for classification, regression, and other tasks that operate by constructing many decision trees at training time. For classification tasks, the output of the random forest is the class selected by most trees.

4.2.1 Grid Searching

Grid search is a tuning technique that attempts to compute the optimum values of hyperparameters. It is an exhaustive search that is performed on the specific parameter values of a model. I used the GridSearch technique on Random Forest to find the good enough hyper-parameter set that outperforms random or default settings.

5 Methodology

5.1 Training/validation split

Multi-Class Logistic Regression: before training the model, I split the dataset into two separate data sets, validation(20% of the dataset) and training(80% of the dataset). And then, I trained multiple settings of my model, including a number of epochs and learning rate, to find the best model that performs well on the validation set.

RandomForest: Since the library I used for GridSearch finds the best model based on the k-fold-cross-validation, I only took 5% of the dataset for final validation to check the model's performance.

5.2 Regularization strategies

Weights: Only for Logistic Regression, the l2 regularization on weights is tried, but it didn't improve the performance of the model, so I decided not to use it.

Dataset: For both models, I regularized the dataset based on z-score; because some of the algorithms that I tried are dependent on the magnitude of each feature, I normalized the dataset to make sure every algorithm runs with the same dataset settings.

5.3 Optimization tricks

Optimization tricks: if it is considered as an optimization trick, the mini-batch approach is used for training Logistic Regression since it was much more computationally efficient and faster on a regular laptop.

Hyper Parameter	Value
max depth	30
min samples leaf	20
n estimators	100
bootstrap	True

Table 1: Random Forest Hyper Parameters.

5.4 Choice of hyper-parameters

The automatic hyper-parameters search has not been used for training Logistic Regression; just some combinations of learning-rate, batch-size, and number of epochs were used; and the best one with the higher validation accuracy is selected for Kaggle submission.

But for Random Forest: the GridSearch approach is applied along with k-fold-cross-validation, and the best model among all combinations is selected for final results on Kaggle. The selected hyper parameters using Grid Searching are presented in the table 1.

6 Results

In this section, the comparison of different models' performance is presented using Accuracy and Balance Accuracy.

Note: As we observed in figure 1 classes are imbalanced, so the Balanced Accuracy shows how well the model is performing on each separate class.

6.1 Accuracy

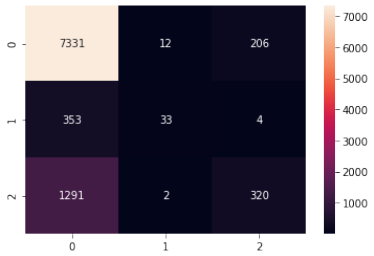
Model	Accuracy	Balanced Accuracy
Logistic Regression	82.04%	76.11%
Random Forest(Without GridSearch)	85.35%	73.63%
Random Forest(Wtih GridSearch)	**88.87**%	**83.15**%
Naive Bayes(Gaussian)	57.67%	48.05%
SVM(linear)	80.97%	73.89%

6.2 Confusion Matrix

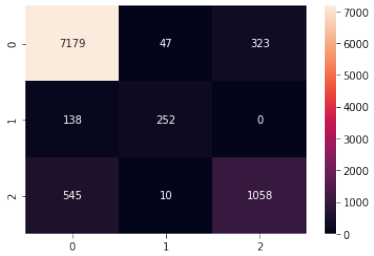
The Confusion Matrix of four models is presented in the figure 2. Similar to the accuracy of the Random Forest(With GridSeach), the confusion matrix of this model also shows better performance.

6.3 Loss Curve

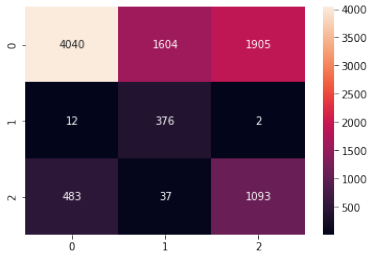
Figure 3 shows the loss curve during training of the Logistic Regression.



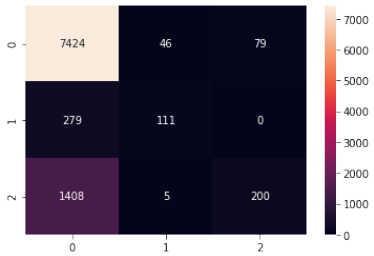
(a) Logistic Regression



(b) Random Forest with GridSeach



(c) Naive Bayes



(d) SVM

Figure 2: Confusion Matrix

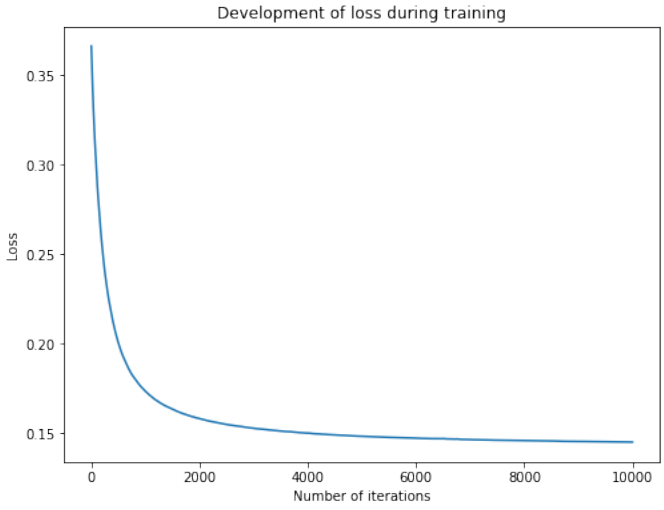


Figure 3: Logistic Regresion Loss

7 Discussion

First of all, since hyperparameters searching needs a lot of computing resources, I only could do it on Random Forest; therefore, if more resources are available, we can try more hyperparameters searching on different kinds of algorithms such as XGBoost, SVM, etc.

In the following some suggestions for future works and improving our approach are also presented:

- Considering the nature of features and using feature selection techniques:
  - Creating new features: using existing features to create new features to determine if they provide new signals to predict our outcome.
  - Converting features format to a proper structure; for example, converting "time" into three separate features(day, month, year)
- Customizing algorithms with imbalance datasets by using some methods such as weighted loss or re-sampling data using methods like Under-Sampling and Over-Sampling.
- Doing more searches on hyperparameters which requires more resources.
- Trying other kinds of ensemble methods; for example, a voting method using different types of algorithms including decision tree, SVM, etc.
- Last but not least, by using a large portion of the original ClimateNet dataset, we can apply much more complex models and features analyses; in this case, we also can use some pre-trained and tuning them in our problem domain.

8 Statement of Contributions

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

References

[1] ClimateNet, https://gmd.copernicus.org/articles/14/107/2021/, 03 1 2021.

[2] Github, https://github.com/bamtak/machine-learning-implemetation-python, 03 1 2021.

[3] Softmax Regression, https://www.kdnuggets.com/2016/07/softmax-regression-related-logistic-regression.html, 03 1 2021.

[4] Random Forest, https://en.wikipedia.org/wiki/Random_forest, 03 1 2021.

A Appendix

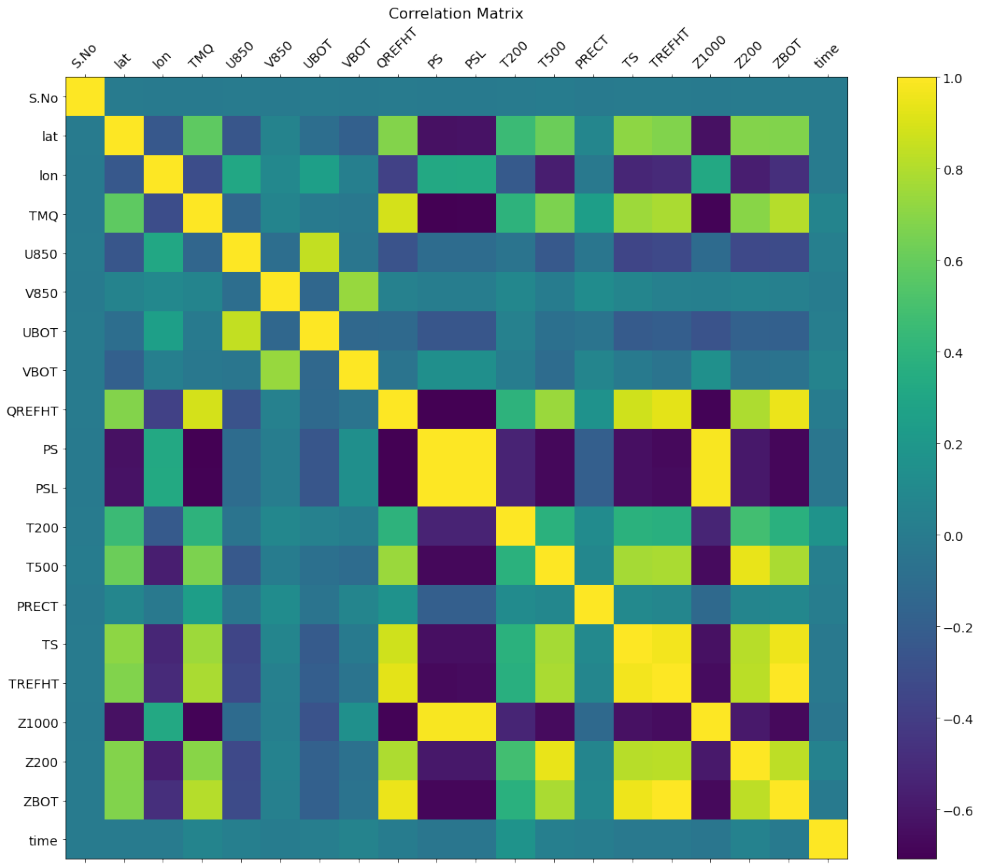


Figure 4: Correlations Matrix

	count	mean	std	min	25%	50%	75%	max
S.No	47760.0	2.387950e+04	1.378727e+04	0.000000e+00	1.193975e+04	2.387950e+04	3.581925e+04	4.775900e+04
lat	47760.0	-2.933507e+00	2.290675e+01	-3.109518e+01	-2.458279e+01	-3.872229e+00	2.176662e+01	2.405476e+01
lon	47760.0	2.692708e+02	4.104659e+01	2.290625e+02	2.418750e+02	2.531250e+02	2.771875e+02	3.543750e+02
TMQ	47760.0	3.483568e+01	1.286955e+01	6.960679e+00	2.403308e+01	3.489657e+01	4.472710e+01	8.064136e+01
U850	47760.0	-6.202928e-01	6.847307e+00	-5.880523e+01	-5.507042e+00	-1.671802e+00	3.621788e+00	3.227734e+01
V850	47760.0	1.040469e-02	4.332385e+00	-2.621691e+01	-2.307535e+00	1.571722e-01	2.341779e+00	5.806025e+01
UBOT	47760.0	-2.152124e+00	6.043284e+00	-5.826048e+01	-6.589128e+00	-2.687853e+00	1.720815e+00	3.021539e+01
VBOT	47760.0	-7.594555e-01	4.761139e+00	-2.076045e+01	-3.800056e+00	-7.749922e-01	2.072186e+00	3.110628e+01
QREFHT	47760.0	1.437782e-02	3.969209e-03	4.363584e-03	1.124537e-02	1.456635e-02	1.798656e-02	2.236619e-02
PS	47760.0	1.015541e+05	5.458636e+02	9.661160e+04	1.011521e+05	1.015280e+05	1.018875e+05	1.039970e+05
PSL	47760.0	1.015563e+05	5.448553e+02	9.661160e+04	1.011576e+05	1.015312e+05	1.018889e+05	1.039970e+05
T200	47760.0	2.168945e+02	2.307294e+00	2.088242e+02	2.155043e+02	2.171624e+02	2.184307e+02	2.291729e+02
T500	47760.0	2.643758e+02	3.507889e+00	2.491460e+02	2.629101e+02	2.652451e+02	2.668773e+02	2.735808e+02
PRECT	47760.0	4.629730e-08	2.385750e-07	-4.400000e-23	1.610000e-14	5.280000e-09	3.120000e-08	1.440000e-05
TS	47760.0	2.983095e+02	3.794155e+00	2.893307e+02	2.958607e+02	2.986852e+02	3.017601e+02	3.038698e+02
TREFHT	47760.0	2.968927e+02	4.021797e+00	2.854267e+02	2.942212e+02	2.973013e+02	3.004741e+02	3.043642e+02
Z1000	47760.0	1.355229e+02	4.251810e+01	6.333521e+01	1.018365e+02	1.337730e+02	1.626831e+02	3.281237e+02
Z200	47760.0	1.229162e+04	1.309269e+02	1.171825e+04	1.222998e+04	1.232195e+04	1.239146e+04	1.251103e+04
ZBOT	47760.0	6.537814e+01	1.041089e+00	6.243359e+01	6.465773e+01	6.547276e+01	6.632780e+01	6.713165e+01
time	47760.0	2.002393e+07	3.658871e+04	1.996061e+07	2.000091e+07	2.001121e+07	2.004082e+07	2.010103e+07
LABELS	47760.0	3.862647e-01	7.625111e-01	0.000000e+00	0.000000e+00	0.000000e+00	0.000000e+00	2.000000e+00

Figure 5: Describing the dataset

```
S.No      0
lat       0
lon       0
TMQ       0
U850      0
V850      0
UBOT      0
VBOT      0
QREFHT    0
PS        0
PSL       0
T200      0
T500      0
PRECT     0
TS        0
TREFHT    0
Z1000     0
Z200      0
ZBOT      0
time      0
LABELS    0
dtype: int64
```

Figure 6: Check null values