Accessible Drought Prediction

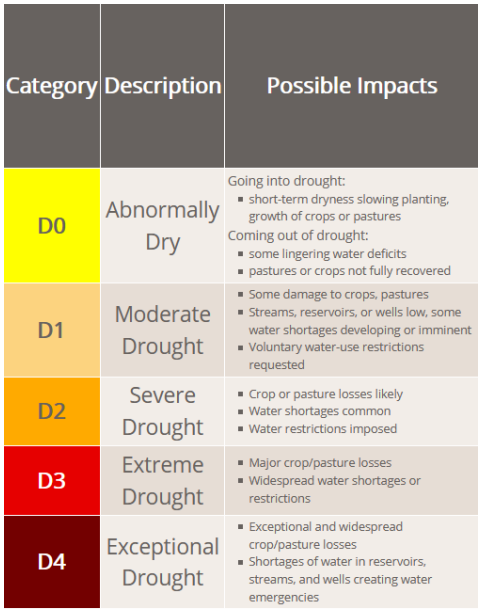
Dhamma Kimpara, Rick Nueve (Deep Learning class)

1 Introduction

The US drought monitor maintains and updates drought forecasts for each county in the US using an army of highly trained expert scientists who utilize a wide array of data, some of it proprietary. The goal of this project is to achieve the same forecast but with machine learning and readily available meteorological and soil data. This would enable areas with less resources to have a similar capacity to predict drought conditions.

The forecasting task is to predict drought levels (none, D0, ..., D4) for each week ahead of the prediction date, up to 6 weeks in the future. The data consists of widely available weather data and soil data.

While this dataset and task has received a few submissions on Kaggle, we differentiate our work with two main contributions:



Category	Description	Possible Impacts
D0	Abnormally Dry	Going into drought: ▪ short-term dryness slowing planting, growth of crops or pastures Coming out of drought: ▪ some lingering water deficits ▪ pastures or crops not fully recovered
D1	Moderate Drought	▪ Some damage to crops, pastures ▪ Streams, reservoirs, or wells low, some water shortages developing or imminent ▪ Voluntary water-use restrictions requested
D2	Severe Drought	▪ Crop or pasture losses likely ▪ Water shortages common ▪ Water restrictions imposed
D3	Extreme Drought	▪ Major crop/pasture losses ▪ Widespread water shortages or restrictions
D4	Exceptional Drought	▪ Exceptional and widespread crop/pasture losses ▪ Shortages of water in reservoirs, streams, and wells creating water emergencies

Figure 1: Drought Levels

1. Implementing, in the sci-kit learn framework, a simple estimator for Ordinal Classification [2]. We will call this estimator ORD throughout as a shorthand.
2. Using ORD, we show similar performance in the Macro F1 metric and superior performance in the Mean Absolute Error (MAE) metric, compared to other models.

In particular with contribution 2, we show that simple classifiers can match and surpass the performance of a Deep Learning Model. While the particular instance of ORD we use is simple, it can be made arbitrarily more powerful, which we suspect would improve performance. Furthermore, comparing to the Kaggle submission that uses Ridge Regression, we confirm the inclination in [2] that using a regression estimator is fundamentally *ad-hoc* for ordinal classification. Thus practitioners should lean towards using a model like ORD, that inherently handles the structure and specifics of ordinal classification.

2 Approach

While we settled on the ORD estimator, we tried a variety of approaches. These included Decision Trees (regression and classification) and Multi-layer Perceptrons (MLP). However all these approaches were impractical because of the size of the data and our computational limitations. As is standard, we normalized all of the features by mean and variance. We also

encoded the date as sin/cos transformed seasonal indicator features. We now describe contribution 1, implementation of ORD.

2.1 Ordinal Classification

Ordinal classification is a type of problem in between classification and regression. The prediction set is a discrete label set. However, unlike classification, there is an inherent ordering to the labels. This ordering is a feature of regression problems but regression does not assume a discrete label set.

Our problem falls into the class of ordinal classification problems. The task is to predict drought levels [none, D0, ..., D4] with a natural ordering (no drought to exceptional drought). Thus due to our computational limitations that we outline in section 2.3, we decided to focus on utilizing a simple instance of an ordinal classification estimator, ORD, that is tailored for the task [2].

We implemented ORD, following the sci-kit learn guidelines, in order to use ORD in sci-kit's ecosystem which critically includes MultiOutputClassifier and GridSearchCV. The code is included in the Appendix. We hope that this implementation might be useful for other practitioners. From this project learned the nuances of this special machine learning task and contributed something that might be useful to the wider community.

2.2 ORD

The idea of the estimator is to split the ordinal classification problem into separate binary classification tasks. Suppose we have k ordered classes 1, ..., k. Then we can use k-1 binary classifiers to predict the label of an example. These binary classifiers must be able to output a probability for each class.

First train a classifier for each label *i = 1, ..., k-1* that when given example *(x, y)*, predicts:

$$\begin{cases} 1 & \text{if } y > i, \\ 0 & \text{if } y \leq i. \end{cases}$$

Then, we compute the probability of each label, using the predict probability output of each classifier as follows:

$$\begin{aligned} \Pr[y = 1] &= 1 - \Pr[y > 1], \\ \Pr[y = 2] &= \Pr[y > 1] - \Pr[y > 2], \\ &\vdots \\ \Pr[y = i] &= \Pr[y > i - 1] - \Pr[y > i], \\ &\vdots \\ \Pr[y = k] &= 1 - \Pr[y > k - 1]. \end{aligned}$$

Due to our computational limitations, we selected Logistic Regression as our 'base' classifier for each of the k-1 binary problems. Even this simple classifier was not quick to train. Only two of the sci-kit provided solvers for training Logistic Regression was practical in terms of training time (LBFGS[1] and SAGA). Thus we decided to not explore more computationally intensive models such as MLPs, Gaussian Processes and Nearest Neighbors.

2.2.1 Multioutput Predictions

Since we implemented ORD in the sci-kit framework, we were able to use sci-kit classifiers as our base classifier, and use sci-kit wrappers for ORD to ease the training process. Since we must predict 6 different labels, one for each of the 6 weeks in the future, we wrapped ORD in sci-kit's MultiOutputClassifier wrapper that allowed parallel training of the sub-classifiers for each of the 6 labels. Furthermore, we were able to use GridSearchCV to obtain optimal hyper parameters.

2.3 Computational Limitations

Due to the number of features of the full data (180 days of past observations), it was impractical for us to use more sophisticated models due to our computational limitations. Furthermore, while we could use smaller versions of these more sophisticated models (less neurons, less trees etc.), these smaller versions performed poorly and still took an impractical amount of time to train.

When we tried a Multi-layer perceptron, the number of neurons we desired did not even fit into 32 GB of RAM. Smaller networks performed poorly, as mentioned previously. We also tried Gradient Boosted Trees (Regression and Classification). These models were slow at training due to the number of features and training set size. Furthermore, they performed poorly. Thus we decided to focus on the ORD classifier that used simple models, trained quickly, was tailored to the task, and succeeded at the task.

3 Data

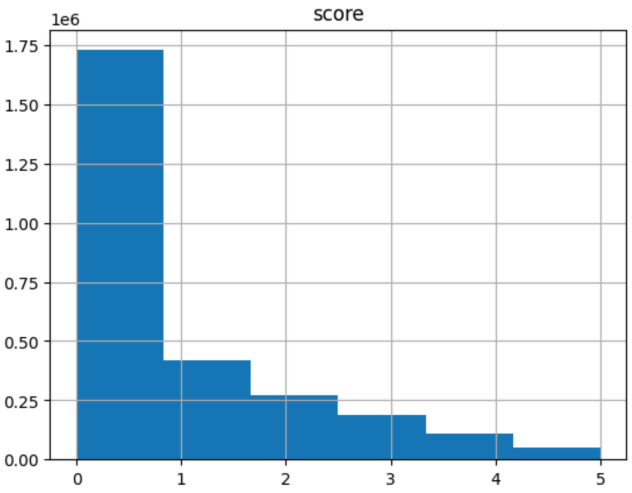
3.1 Raw Data

The main dataset is a time series dataset. The train/validation/test split was already given to us by the US drought monitor. Each dataset has 20 features of time series meteorological data for 3108 unique prediction sites. The goal is to predict the drought level (none to D4), for each week in the future, up to 6 weeks, throughout each time series for each unique prediction site. The drought level is the last column in each dataset. Another property of the dataset is that while the meteorological features are provided daily, the drought label is only provided weekly. So we we had to do some interpolation of the drought labels. The training set has 19,300,680 rows total which means 6210 time-series datapoints per prediction site. The validation and test sets are equal in size with 730 time-series datapoints per prediction site.

[1] Limited-memory Broyden–Fletcher–Goldfarb–Shanno

Furthermore, there is a separate dataset of 21 soil features for each of the prediction sites. From the histogram of the drought labels, one can see that true labels have a skewed distribution.

Histogram of Drought Labels



fips	date	PRECTOT	PS	QV2M	score
1001	2019-01-01	2.25	100.51	9.69	0.0

Figure 2: Example features from a row in the raw dataset

2.1 Pre-Processed Data

Since this is a forecasting task, we had to transform the data so that each datapoint corresponded to a row in the original dataset (a specific date and prediction site). The features for each datapoint consisted of 180 days of past meteorological observations, the past measured drought values, and the soil data for that prediction site. We also had to interpolate drought labels and create a vector to represent that true labels of the 6 future prediction weeks. We elected to use the code given in the Kaggle challenge to do the aforementioned data pre-processing to maintain comparability between our model and others.

We chose two past data windows to include as our features. The pre-processed data had the following dimensions:

Past data window	**# Features**	**# Training Samples**	**# Validation/Test Samples**
180 days	3810	100,000	9,000
14 days	324	1.4 Million	150,000

Table 1: Post-Processed data sizes

4 Results

In this section we will describe contribution 2, and show that our simpler model can match and even outperform more complex models.

Our ORD classifier with a Logistic Regression base classifier was able to match previous model's Macro F1 score (within 10%) and significantly improve upon the Mean Absolute Error (25% improvement). The LSTM approach is a deep learning approach, which we were able to improve upon with our Logistic Regression based classifier. We also suspect that our model is much faster to train. As was shown in [1], the last 14 days of weather data have almost all of the predictive power. Thus due to our computational constraints, we chose this version of the dataset for our model.

Model	Macro F1	MAE
LSTM [3]	0.64	0.28
Ridge Regression [1]	0.58	0.26
ORD w/ Logistic Regression	**0.58**	***0.20***

Table 2: Performance Results Comparison

As is illustrated by the confusion matrices below, our classifier consistently underestimated the drought severity. In particular it did the most underestimation for the most severe level, D4. Fortunately, our classifier almost always underestimated the drought level by only one level. This explains the improved MAE performance but equivalent Macro F1 performance. The scores under each confusion matrix also show that the classifier performs worse as the prediction is further out in time. This matches the observations of previous models listed in table 2.

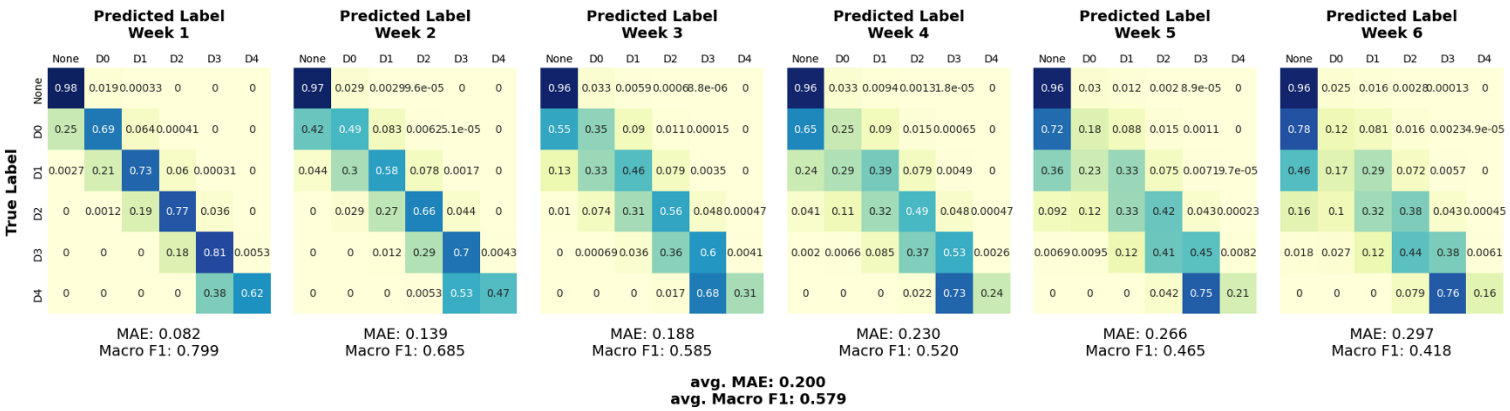


Figure 3: Confusion matrices of each prediction week

5 Discussion of Training Issues: Non-Convergence

As mentioned in section 2, more complex models were computationally infeasible for us to use. However, training was not straightforward even with our simpler Logistic Regression based ORD. We will examine and interpret the issue of non-convergence of our classifier that we faced.

While training all our ORD classifiers, we always received sci-kit learn's Convergence Warning. This means that the loss value during training never converged. We tried all the potential remedies to the situation including some feature engineering, increasing the number of training iterations, and changing the solver. As discussed previously, only two of the solvers for Logistic Regression were practical for us to use. So we were not able to confirm that some other solver would show convergence. We illustrate the convergence issue with the following graph:

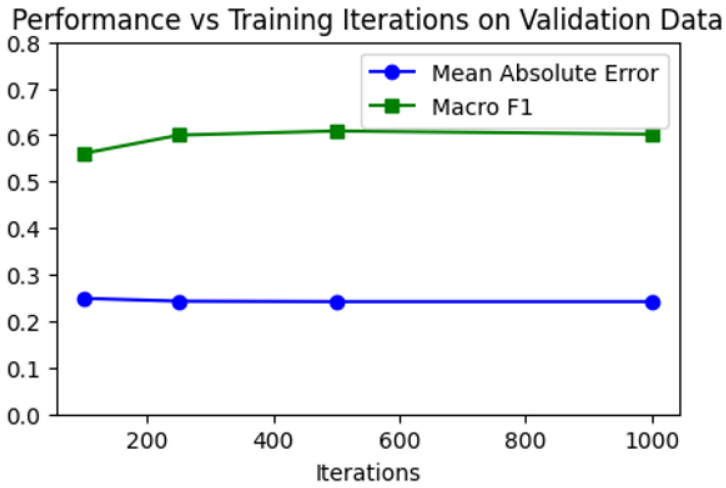


Figure 4: Performance versus training iterations

Our interpretation of this graph is that increasing the number of training iterations did not affect convergence, with the solvers and features we were able to use. This is because the performance of the classifier did not significantly change as more iterations are used.

From our investigations, we believe that Logistic Regression does not have enough expressive power to fit the data. However, due to our classifier matching and surpassing the performance of a more expressive model, LSTM, we believe that the labels are noisy. This means that *any* classifier cannot both fit and generalize to the data. Given that ORD performed so well with such a simple base classifier, we believe that further investigation with ORD, using a more powerful base classifier is highly warranted.

6 Conclusion

As shown in the results section, our classifier consistently underestimated the drought severity. We believe this is partly due to the class label imbalance shown in the histogram of drought labels in section 2. The more severe the drought label, the less prevalent it is in the dataset. Thus most classifiers will have the tendency to predict the more prevalent label, when there is high uncertainty in the prediction. Such is the case in a noisy dataset, that we suspect we have. It will be pertinent for the field to deal with these issues, and design loss functions to reflect different ways of handling prediction uncertainty. For example, a loss with an abstain prediction.

While our classifier did not improve upon the Macro F1 metric, it did improve significantly on the MAE. As is seen in the confusion matrices, when the classifier mispredicted, it did not mispredict by much (almost always by only 1 level of severity). We believe this shows that the structure of ORD, being able to take into account the nuances of the ordinal classification task, is highly successful at the task. Thus this confirms that ORD has an advantage over other *ad-hoc* models for ordinal classification [2]. We hope that our implementation of ORD will be useful to other practitioners as it is ready to be used in the sci-kit learn ecosystem.

6.1 Extensions

There are several avenues for extensions. These include: using more powerful classifiers, using a different loss function, using additional data, and taking advantage of the spatio-temporal nature of the data,

After showing the power of ORD, we suspect that the easiest gains in performance are to be had by using ORD with a more powerful base classifier. We showed that even with a base classifier that was not able to express the complexity of the data, we achieved state of the art performance. Thus a more powerful base classifier has high potential in ORD. Furthermore, one could use different classifiers for each of the prediction weeks *and* for each of the binary classifiers in an instance of ORD. Thus we believe that ORD has significant potential. We also believe that ORD can be easily integrated into other machine learning ecosystems such as PyTorch.

As previously mentioned, we suspect we have a noisy dataset. Thus in order to handle high uncertainty in the labels, it could be useful to use a different loss function. For example, if a classifier is able to report abstain, it could enable more informative predictions.

The issue that the classifier performed worse as the prediction was further into the future could be mitigated by using additional data such as weather forecasts. Finally, one direction is to use the spatio-temporal nature of the data to improve classifications. If we fed the data in a streaming fashion, like in a time series task, we could take advantage of Recurrent Neural Nets to make predictions. Furthermore, one could take advantage of the spatial aspect, such as distances between prediction sites, to encode more global features, perhaps with an Auto-encoder.

Acknowledgements

We thank the author of [1] for their helpful visualization and reporting functions.

References

[1] epistoteles (2021). Predicting Drought. https://github.com/Epistoteles/predicting-drought

[2] Frank, E., & Hall, M. (2001). A simple approach to ordinal classification. In European conference on machine learning (pp. 145-156). Springer, Berlin, Heidelberg.

[3] Minixhofer, Christoph (2021). Predict Droughts using Weather & Soil Data. From https://www.kaggle.com/datasets/cdminix/us-drought-meteorological-data

Appendix

```python
import numpy as np
from sklearn.base import BaseEstimator, ClassifierMixin, clone
from sklearn.utils.validation import check_X_y, check_array, check_is_fitted
from sklearn.utils.multiclass import unique_labels

class OrdinalClassifier(BaseEstimator, ClassifierMixin):
    def __init__(self, base_estimator=None, loss=None):
        self.base_estimator = base_estimator
        self.loss = loss #callable

    def fit(self, X, y):
        X, y = check_X_y(X, y)

        self.classes_ = np.sort(unique_labels(y))
        self.X_ = X
        self.y_ = y

        self.n_features_in_ = X.shape[1]

        self.estimators_ = {}

        for i in range(self.classes_.shape[0]-1):
            # for each k - 1 ordinal value we fit a binary classification problem
            binary_y = (y > self.classes_[i]).astype(np.uint8)
            clf = clone(self.base_estimator)
            clf.fit(X, binary_y)
            self.estimators_[self.classes_[i]] = clf
        return self

    def predict_proba(self, X):
        check_is_fitted(self)
        X = check_array(X)

        clfs_predict = {k: self.estimators_[k].predict_proba(X) for k in self.estimators_}
        predicted = []

        if len(self.classes_) == 1:
            return np.ones((X.shape[0],1))

        # V1 = 1 - Pr(y > V1)
        predicted.append(1 - clfs_predict[self.classes_[0]][:,1])

        for i in range(1, len(self.classes_) - 1): #sorted classes
            # Vi = Pr(y > Vi-1) - Pr(y > Vi)
             predicted.append(clfs_predict[self.classes_[i-1]][:,1] -
                              clfs_predict[self.classes_[i]][:,1])

        # Vk = Pr(y > Vk-1)
        predicted.append(clfs_predict[self.classes_[-2]][:,1])
        return np.vstack(predicted).T

    def predict(self, X):
        check_is_fitted(self)
        X = check_array(X)
        return self.classes_[np.argmax(self.predict_proba(X), axis=1)]

    def score(self, X, y, sample_weight=None):
        _, indexed_y = np.unique(y, return_inverse=True)
        return self.loss(indexed_y, self.predict(X), sample_weight=sample_weight)
```