CropHarvest - Crop vs. Non-Crop (crop land detection from remote sensing data)

Kaggle Competition 2 - Team - SuperAkelius

Abhay Puri (20209505)
Kaggle User - https://www.kaggle.com/abhaypuri98

Saurabh Bodhe (20208545)
Kaggle User - https://www.kaggle.com/astatine404

1 Introduction

In this competition, the task is to design a binary classifier to classify between the two classes: Crop (1) and Non-Crop (0). The dataset, called CropHarvest contains data about geographical coordinates at different months of the year and other data like topography, temperature, and precipitation. There are a total of 216 features.

Major challenges:

1. Large number of features slowing down the training process and as a result, slowing down the hyperparameter tuning.
2. Very large hyperparameter space to tune with.
3. Limited availability of compute and running time limits on Colab.
4. Unexpectedly high F1-score on the test set (Kaggle) compared to the validation set, showing signs of overfitting the test set.

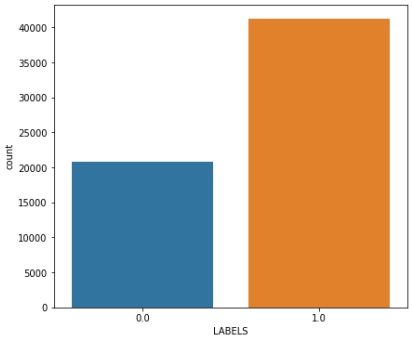
Approach:

1. Analysed the dataset for possible cleaning and preprocessing.
2. Since the dataset has 216 features, assessed the feature relevance scores like chi2 and mutual info to find possibilities of dropping irrelevant features.
3. Any attempt to drop irrelevant features either resulted in no effect or reduction of cross-validation F1 score.
4. Tried training and testing all major classifiers in SKLearn to obtain a manageable set of classifiers for further hyperparameter tuning.
5. LightGBM, RandomForests, and ExtraTrees appeared to work the best.
6. Optuna was used to tune the hyperparameters for the three classifiers using 5 split Stratified-KFold cross-validation with F1 score as the decision metric.
7. Best F1 cross-validation scores obtained: RandomForest > ExtraTrees > LightGBM

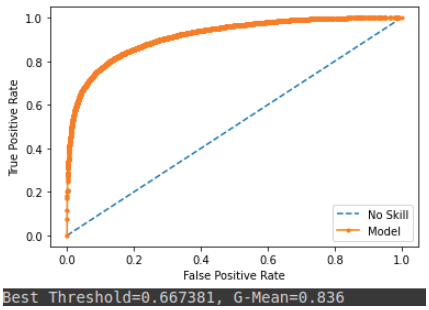
2 Methods

2.1 Data Preprocessing

There wasn't any major data preprocessing done on the dataset. The data is numerical throughout so no conversion is needed there. Since the algorithms chosen below are all tree-based, there was no



(a) Label Imbalance



(b) Threshold Shifting using ROC curve

Figure 1

need for any scaling or standardization. Although there is data imbalance to a certain extent with Crop (1) being the majority class (Fig 1. (a)), we concluded that the imbalance is not severe enough to try balancing the data. Additionally, balancing techniques like undersampling, oversampling, SMOTE, etc. have performance costs down the road which will likely outweigh the benefits.

2.2 Random Forest

Random forest is a type of ensemble method for classification, regression, and other tasks. It works by constructing many decision trees at training time. For classification tasks, the output of the random forest is the class selected by most trees. The main principle used here is that of bagging, where bootstrap sampling is used to sample the dataset to contruct each tree. In addition to dataset, even the features are sampled. Random forest help correct decision trees' habit of overfitting to their training set. Random forests generally outperform decision trees, primarily due to their resistance to overfitting.

2.3 ExtraTrees

Adding one more step of randomization to random forests gives us the extremely randomized trees or ExtraTrees. While very similar to random forests in that they are an ensemble of individual trees, there are some differences: first, each tree is trained using the whole learning sample (instead of bootstrapping), and second, the top-down splitting in the tree learner is randomized. At each node, instead of computing the optimal split-point for each feature, a random split-point is selected. This value is selected from a uniform distribution within the feature's range in the training set. Then, of all the randomly generated splits, the split that yields the highest score is chosen to split the node. The max number of features to consider at the split is a hyperparameter similar to random forest.

2.4 Threshold Shifting

By default, the threshold using be the prediction function is 0.5. Since this is a binary classification problem, threshold shifting is not too difficult. So we attempted to tune the decision threshold by using the ROC curves method (Fig 1. (b)). But the observed performance scores were always similar or worse than scores with default threshold when tested on each of the best-tuned classifiers. Our hypothesis for this failure is that the tuned hyperparameters may no longer be the optimum ones. This is because the tuning process itself uses the default threshold internally to calculate the cross-validation scores. If we ran extra rounds of hyperparameter tuning with the different thresholds incorporated as an extra hyperparameter within the tuning, it could have been possible to slightly improve the performance. Considering this is not supported out of the box by Optuna, we concluded this would take too much time and result in a little gain.

2.5 Optuna

Optuna is the key to fast hyperparameter tuning on a large set of hyperparameters. SkLearn's GridSearch takes a prohibitive amount of time with a large set and often runs over the Colab usage

Table 1: Results - F1 Scores

Algorithm	Local Cross Validation	Kaggle Public	Kaggle Private
Random Forest	0.88883	0.99757	0.99669
Extra Trees	0.88421	0.99516	0.99779
LightGBM	0.84645	0.97572	0.97582

limits. RandomSearch more than often produces sub-optimum results. Optuna intelligently tries out various hyperparameter values and gradually moves in the direction of optimum using the information obtained in previous trials. It has state-of-the-art algorithms to sample hyperparameter spaces to quickly learn about the direction in which it needs to move to get better performance. It also effectively prunes unpromising trials. It has inbuilt support for parallelization and the multiple threads running trials share information among themselves to further increase the search speed.

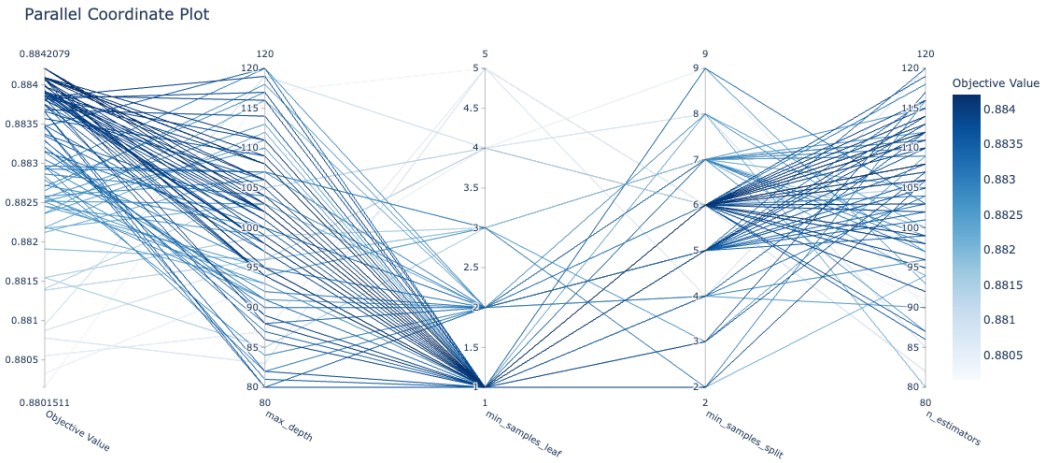


Figure 2: Hyperparameter tuning for ExtraTrees

During initial analysis, the three classifiers: RandomForest, ExtraTrees, and LightGBM were found to be the best candidates for further tuning. Optuna was used to tune these with the hyperparameter ranges provided.

3 Results

Table 1 shows the final results containing the cross-validation scores we obtained on the complete train dataset and the Kaggle public and private scores. We strictly used stratified 5-fold cross-validation for assessment and used the complete training set for the final trained model. We avoided train-validation split since in this task every small gain is important, so we didn't want to risk losing data to validation. We had concluded Random Forest to be our best model since it had the highest cross-validation score as well as the Kaggle public score. But surprisingly, ExtraTrees somehow surpassed Random Forest on the private set. The reasoning for this is not very clear but our initial hypothesis is that the extra degree of randomization in ExtraTrees compared to Random Forest made it a better-generalized model and hence better private score.

4 Discussion

Being a very clean dataset and a binary classification problem, this was a fairly straightforward task and the only major work was hyperparameter tuning. In review, a final rank of 6/50 on Kaggle was a good indicator of the task being accomplished. Optuna, being the game-changer here, significantly increased productivity and enabled us to try out more experiments than is usually feasible. There

was certainly scope for improvement. The subset of models chosen for tuning was fairly small and there was space for more experimentation there. We did not use any data-specific techniques when there were certain approaches possible. One such example is that since the data was for different months, there was a possibility of using models specialized for time-series data like LSTMs and RNNs. Even if such special models were not used, there was certainly scope for exploiting the additional information that can be extracted from the time-series data. Lastly, the unexpected better performance of ExtraTrees has no clear explanation, the above hypothesis could be true or it could be just due to the variations in the data splits.

5 Statement of Contributions

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

References

[1] https://scikit-learn.org/stable/modules/classes.html

[2] https://optuna.readthedocs.io/en/stable/

[3] https://www.analyticsvidhya.com/blog/2020/11/hyperparameter-tuning-using-optuna/

[4] https://machinelearningapplied.com/hyperparameter-search-with-optuna-part-1-scikit-learn-classification-and-ensembling/

[5] https://lightgbm.readthedocs.io/en/latest/pythonapi/lightgbm.LGBMClassifier.html

[6] https://machinelearningmastery.com/threshold-moving-for-imbalanced-classification/

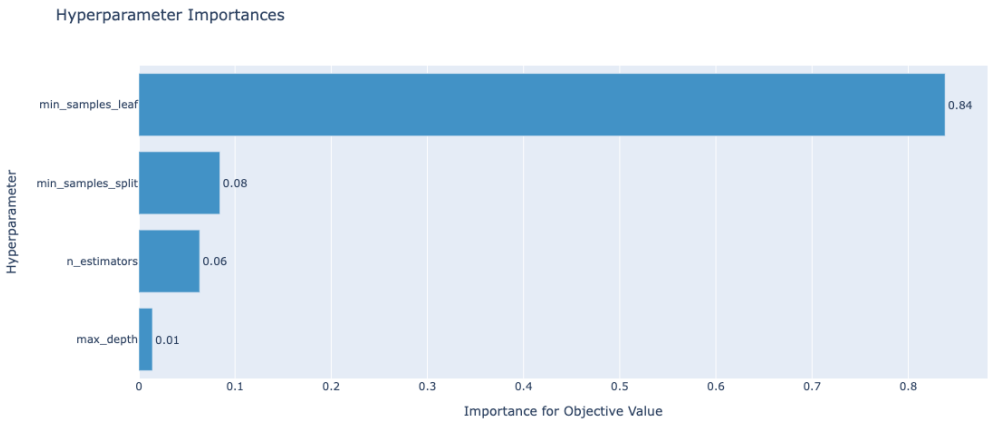
[7] https://machinelearningmastery.com/hyperopt-for-automated-machine-learning-with-scikit-learn/

[8] https://towardsdatascience.com/5x-faster-scikit-learn-parameter-tuning-in-5-lines-of-code-be6bdd21833c

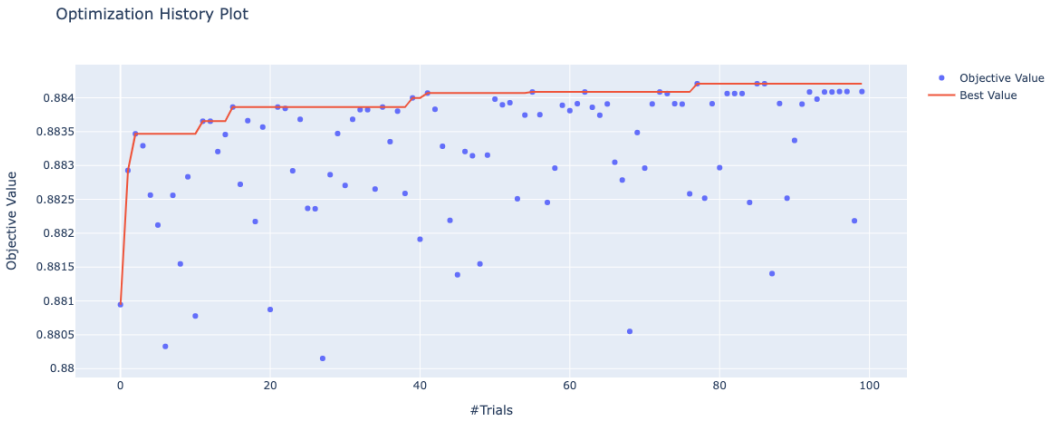
[9] https://scikit-optimize.github.io/stable/modules/classes.html

[10] https://en.wikipedia.org/wiki/Random_forest

A Appendix



(a)



(b)

Figure 3: Hyperparameter Tuning and Analysis of ExtraTrees