Kaggle Data Challenge 2

**Student names**: Charbel Sahyouni, Kaushik Moudgalya

**Student Number**: 20205112, 20176760 **Kaggle Team Name:** Charbel And Kaushik **Kaggle Username**: charbelsahyouni, diraizel

**Kaggle Display Name**: Charbel Sahyouni, Banana\_Leopard

**Kaggle Account Link**: <https://www.kaggle.com/charbelsahyouni>, <https://www.kaggle.com/diraizel>

# Introduction – Problem Description

Given meteorological and satellite data, predict land as either crop or non-crop land. The data is based on a processed subset of a recent NeurIPS dataset submission called CropHarvest [Tseng et al., 2021]. The dataset has around 60,000 points to be classified based on data from two satellites.

# Methodology

We tried a number of models on our data, including Random Forests (RF), CatBoost Classifier, Neural Networks, and a couple of AutoML methods, using a train test split of 85%-15%.

The classes were a little imbalanced and roughly 2:1 ratio. We did the following transforms on the dataset:

* Removed all the duplicates from the dataset.
* Removed the values from the dataset where the same features have different labels.
* Used sklearn to standardize the data by fitting standard scalar over train set.
* Transformed the features in the validation set using the transforms of the train set.
* Trained our models on the train set and evaluated the models using the validation set.
* The final submission data was then standardized using the entire sklearn feature transforms over the entire data given.
* Model selection intuition given in the following section.

# Results

Summary of PyCaret results, sorted by F1 scores:

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **Model** | **Accuracy** | **AUC** | **Recall** | **Precision** | **F1** | **Kappa** | **MCC** | **Time (s)** |
| **CatBoost** | 0.8447 | 0.9211 | 0.9099 | 0.8673 | 0.8881 | 0.6352 | 0.6372 | 25.5840 |
| **Extra Trees** | 0.8401 | 0.9150 | 0.9157 | 0.8577 | 0.8857 | 0.6203 | 0.6241 | 2.0598 |
| **XGB** | 0.8387 | 0.9147 | 0.9015 | 0.8658 | 0.8833 | 0.6227 | 0.6241 | 15.6225 |
| **LGBM** | 0.8377 | 0.9136 | 0.9072 | 0.8606 | 0.8833 | 0.6175 | 0.6199 | 1.6182 |
| **GBC** | 0.8114 | 0.8827 | 0.9155 | 0.8251 | 0.8679 | 0.5411 | 0.5503 | 152.8516 |
| **RF** | 0.8166 | 0.8846 | 0.8620 | 0.8664 | 0.8642 | 0.5819 | 0.5820 | 0.8204 |
| **KNN** | 0.8109 | 0.8742 | 0.8662 | 0.8561 | 0.8611 | 0.5648 | 0.5649 | 2.3084 |
| **AdaBoost** | 0.7739 | 0.8377 | 0.8771 | 0.8061 | 0.8400 | 0.4566 | 0.4615 | 31.8030 |
| **Decision Tree** | 0.7730 | 0.7419 | 0.8297 | 0.8341 | 0.8319 | 0.4824 | 0.4825 | 11.6026 |
| **Ridge** | 0.7514 | 0.0000 | 0.8926 | 0.7746 | 0.8294 | 0.3799 | 0.3932 | 0.1935 |

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **LDA** | 0.7529 | 0.8089 | 0.8831 | 0.7807 | 0.8287 | 0.3912 | 0.4010 | 1.2068 |
| **LR** | 0.7529 | 0.8071 | 0.8812 | 0.7816 | 0.8284 | 0.3926 | 0.4018 | 1.0275 |
| **Linear SVM** | 0.7378 | 0.0000 | 0.8446 | 0.7849 | 0.8135 | 0.3739 | 0.3771 | 2.5227 |
| **Naive Bayes** | 0.7094 | 0.7390 | 0.8367 | 0.7588 | 0.7958 | 0.2958 | 0.3002 | 0.1497 |
| **QDA** | 0.6961 | 0.7596 | 0.7035 | 0.8262 | 0.7543 | 0.3580 | 0.3716 | 0.7600 |

The above models were trained using the PyCaret AutoML library. The library takes in a set of features, but the feature scaling in the library is not that reliable from our empirical observations, so we scaled the features by hand.

Based on these results we performed hyperparameter tuning for some of the above models. We trained chose to train RFs in sklearn because of all the algortihms above RFs have the best F1 score with the least training time. We also tried our hand at the LightAutoML (LAMA) library.

Summary of non-PyCaret results:

|  |  |
| --- | --- |
| **Model** | **F1 Score** |
| **LAMA** | 0.8944 |
| **sklearn basic RF** | 0.88723 |

The curves associated with these results are in the Appendix.

# Discussion

Pros:

1. Our approach tried the whole kitchen sink of ML models.
2. Scaled our features properly according to the use-case.
3. Since our approach uses Blending and Stacking through AutoML (LAMA), there is high chance that our models will generalize well, since blended and stacked models generalize better than non-blended or non-stacked models.
4. Our approach is straightforward and simple to re-create.

Cons:

1. Could have plotted more curves for training and validation cost metrics.
2. Could have tried more hyperparameter tuning.
3. Since we used AutoML, most of our models are black boxes, and it is hard to interpret the hyperparameters provided when tuning models.

# Statement of Contributions

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

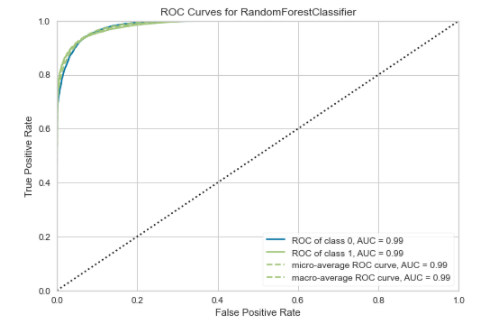
# References

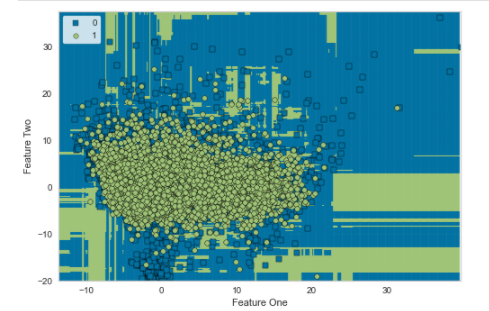
1. [PyCaret guide.](https://towardsdatascience.com/announcing-pycaret-2-0-39c11014540e)
2. [LightAutoML guide.](https://towardsdatascience.com/lightautoml-preset-usage-tutorial-2cce7da6f936)
3. [Our own previous code.](https://github.com/Etrama/IFT6390_Crop_Harvest)
4. All references used while coding are mentioned in the code itself.

# Appendix

1. [GitHub repository for this challenge.](https://github.com/Etrama/IFT6390_Crop_Harvest)
2. Glossary of acronyms: CatBoost = Categorical Boosting Classifier, Extra Trees = Extra Trees Classifier, XGB = eXtreme Gradient Boosting, LGBM = Light Gradient Boosting Machine, GBC

= Gradient Boosting Classifier, RF = Random Forest Classifier, KNN = K-Nearest Neighbours Classifier, AdaBoost = Adaptive Boosting Classifier, Ridge = Ridge Classifier, LDA = Linear Discriminant Analysis, LR = Logistic Regression, Linear SVM = Linear Support Vector Machines, QDA = Quadratic Discriminant Analysis.

1. Curves for the Random Forest Classifier trained:
2. ROC for Random Forest Classifier trained and tuned using PyCaret:
3. Decision Boundary plot for Random Forests trained and tuned using PyCaret:



1. Training-validation curve for Random Forests Classifier trained and tuned in PyCaret:

