Abdiel Fernandez Alvarez
Kaggle competition 2
IFT 6390 – 2021

Report

Remote sensing is one of the more powerful tools for monitoring, planning, and managing on a large scale. The automation in detecting “Regions of Interest" (ROI) has been highly enhanced by artificial intelligence. The aim of the competition is to perform a classification of surfaces as “cropland” or “non-cropland", based on some satellite, radar and meteorological data. Two models were trained for classification, “Random Forest " (RF) and “XGBoost” (XGB). The RF was implemented with an exploratory search grid to find prediction a baseline. A RGB was trained with the objective of enhancing the prediction score and the generalization capacity of the RF model. To increase the confidence of prediction scores it was applied the cross-validation method. Surprisingly, after several attempts of parameter exploration the results from XGB didn’t represent an improvement on RF, which last as a better estimator for this problem.

Dataset description

The given dataset is composed of two sections, training and test(non_labeled). The training set contained 62000 samples, 216 features with labels and the test was composed of 1200 samples without labels. A segmentation on training (80%)/validation (20%) was performed on the training set. The distribution of classes turned out in 66% class 1 and 33% class 0 in both data sets. To reduce possible noise in the data, we applied a correction of missing values by replacing missing’s for the means of columns on the original dataset before segmentation.

Regarding the feature selection, it is well known in remote sensing literature that slope and elevation are both good estimators for the automatic detection of crop regions. However, taken those features with monthly frequency can be seen as redundancy (slope and elevation doesn’t change in a year). In addition, slope and elevations vary from month to month in the dataset, which could be considered as an inconsistency. The topological information was summarized in two new columns (ElevationMean, SlopeMean), replacing the month values (12 columns Slope, 12 columns Elevation) for its “annual average”. Those two columns were found between the three more important recurrently by all models.

Methodology:

A RF algorithm was applied, with a preliminary research grid (see below), as the first attempt to understand the data and establish a prediction baseline. The selection of RF was made based on several reasons: it’s appropriate for binary classification, has the capacity to deal with data in different scales; every tree consider just a few features so it’s not affected by the curse of dimensionality and in general it’s capable to avoid the problem of overfitting since output classes are chosen by majority vote or averaging(3). The exploratory search grid:

```
param_grid={'n_estimators': np.linspace(50, 400, 5).astype(int), 'max_depth': [None] + list(np.linspace(3,20,5).astype(int)), 'max_features': [None] + list(np.arange(0.5, 1, 0.1)), 'bootstrap': [True, False]}
```

From this research was extracted the model parameters of the best estimator, the list of feature importance and the first prediction on the test set (test_no_labels).

Since the result coming from RF was very promising, was decided to try to improve the RF performance by applying a boosting method. An XGBoost classification algorithm was applied (`XGBClassifier from xgboost`) with different configurations: first, passing the parameters of the best estimator form RF search; and in a second time, with a search grid.

param_grid = {'n_estimators':[200,1000,20],'max_depth': [None] + list(np.linspace(3,20,10).astype(int)), 'max_features': ['auto', 'sqrt', None] + list(np.arange(0.5, 1, 0.1)), 'max_leaf_nodes': [None] + list(np.linspace(5, 50,10).astype(int)), 'min_samples_split': [2, 5, 10],'bootstrap': [True]}

For avoiding overfitting of the XGB classifier it was applied the early stop technique. The bests classifiers found with the different configurations of XGBoost were fitted to the data and monitored in the process to find a good number of maximum iterations (max_iter = 10). In addition, the score of the algorithm was taken from the average of 3 rounds of cross validation (`StratifiedKFold from sklearn.model_selection`) every time.

Some tests were made with all previous described algorithm passing the data set standardized. Any interesting improvement were found, and for this reason this is not taking part in the discussion.

To evaluate the model's performance, it was taken into consideration two evaluation metrics, the F1 score and the *Receiver Operating Characteristics (ROC)* analysis and the *Area Under de Curve* (AUC) (7).

Result and discussion.

In the process to find the prediction as possible several experiences were performed. In this section we summarize the main founds and results. For the model of interest, we can find at the end of this section a summary of the considered metrics and the corresponding result from the Kaggle evaluation.

From the preliminary study (RF with search grid), the best estimator performed a score of 0,887 on the validation set and baseline of 0,97831 on unseen samples (Kaggle: CropHarvest - crop vs. non-crop) on the test_set without labels. The best classifier was composed as follow:

```
'n_estimators': 110, 'min_samples_split': 5, 'max_leaf_nodes': 43, 'max_features':
'auto', 'max_depth': 16, 'bootstrap': True
```

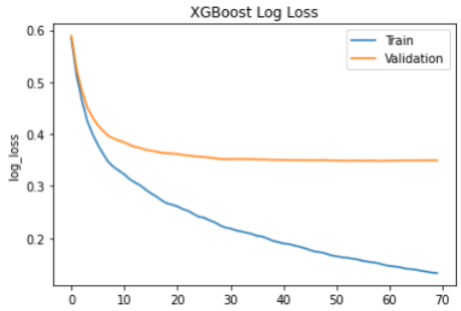
This is a very interesting preliminary approach, which at the end of the process result in the best estimation.

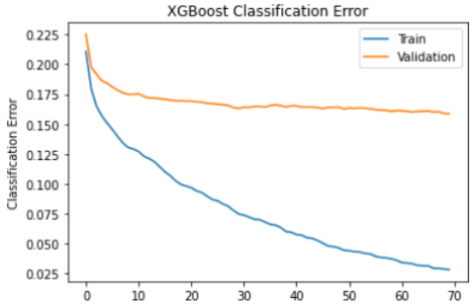
With the application of XGBoost configured the parameters of the best estimator from RF with search grid does not show an improvement for F1_score or Kaggle.

The best parameters from XGBoost implemented with a search are:

```
'n_estimators': 108, 'min_samples_split': 5, 'max_leaf_nodes': 17, 'max_features':
'auto', 'max_depth': 4, 'bootstrap': True.
```

This setting was tested applying the early stop technique and analyzed with both function "error" and "log-loss" as can be seen in the follow figure:





This setting gave a similar score on the validation set compare with the first applied model, however, once uploaded on Kaggle competition system, the score (0.95863) of prediction in the non-labeled dataset was lower than the RF score. Seems to be that XGBoost it's not making better the generalization capacity of the RF model.

The predictions with XGBoost don't perform better than RF in the non_labeled dataset. For this reason improvements were tried for the RF research in order to achieve a better score. A larger search grid was used to train and selected a new configuration for RF that better predicts on the on-labeled dataset, the dataset was standardized as mentioned before, bat nothing ends up with improvements.

Models Score summary:			
Model Name	F1-Score	ROC-AUC	Kaggle score
RFClassifier Random Search	0.887967	0.80941	0,97831
XGBCl_bestRFEstimator	0.88623	0.8185	0.94230
XGBCl_EarlyStop	0.8910	0.8264	0.95863

For all implemented methods we search for the most important features selected by the method. Here you can see a summary for the best-found estimator:

	feature	importance
193	topoElevationMean	0.027478
174	ERA5_temperature_2m_nov	0.018165
194	topoSlope	0.017292
0	Unnamed: 0	0.016241
30	ERA5_temperature_2m_feb	0.015528
...	...	...

CONCLUSIÓN

- The RF it's able to better explain the data. It seems to be a good option as first approach for classification, since it's less affected by data scale, course of dimensionality and do not trend to overfit.
- XGBoost makes good scores on the validation set but it's not always the best estimator to generalize and make predictions on unseen data. This is an unattended result, since this method by nature should improve the performance of a single decision tree.

Bibliography:

1. https://machinelearningmastery.com/random-forest-ensembles-with-xgboost/
2. https://towardsdatascience.com/xgboost-fine-tune-and-optimize-your-model-23d996fab663
3. https://www.analyticsvidhya.com/blog/2021/06/understanding-random-forest/
4. https://www.askpython.com/python/examples/k-fold-cross-validation
5. https://machinelearningmastery.com/how-to-calculate-precision-recall-f1-and-more-for-deep-learning-models/
6. https://machinelearningmastery.com/avoid-overfitting-by-early-stopping-with-xgboost-in-python/
7. https://machinelearningmastery.com/tour-of-evaluation-metrics-for-imbalanced-classification/