CentraleSupélec

2EL1730 Machine Learning Project

BOUMOUSSOU JIHAD YOUNES | OUARHIM YOUNES
ADIB AYMANE CHAOUI

THE YYA TEAM

MACHINE LEARNING
ENGINEERING CYCLE - 2^{nd} YEAR
SG6

February 2025

Introduction

The goal of this project is to develop a machine learning model capable of detecting changes in land areas. The challenge involves classifying geographical regions into six predefined categories based on multiple time periods using multi-date remote sensing data. The dataset has been preprocessed using computer vision techniques to generate numerical features, which will be used for training and evaluation.
We will delve in this report into the feature engineering process and into the model selection and evaluation part of the challenge.
Through this study, we aim to explore the challenges of change detection in remote sensing data and the effectiveness of machine learning techniques in addressing them.

1 Feature Engineering

In this section, we will address the feature engineering process and all of our ETL (extract, transform, load) process.

1.1 Dealing with missing values

We discovered that the dataset had missing values. Initially, we compared the class distributions: `Demolition`, `Road`, `Residential`, `Commercial`, `Industrial`, and `Mega Projects`. We observed that removing all missing data resulted in a decrease of less than 5% in each class. To avoid introducing bias through interpolation or forward/backward filling, we decided to remove the missing data.

1.2 Dealing with multilabel variables

In the provided datasets, we have two multilabel variables: `urban_type` and `geography_type`. We processed the multilabels for each data point to extract the possible urban and geography types. These urban and geography types were subsequently converted into binary variables.

1.3 Feature creation

To enhance our model, we generated a diverse set of features, capturing different aspects of the data, including geometric properties, time-related attributes, and color characteristics. Our objective was to create as many relevant features as possible to improve classification performance. Since our final model is based on decision trees, it can automatically determine the most important features and utilize them effectively for classification.

Geometric features

Using the provided geometry field, we derived various geometric features, including area, perimeter, bounding box coordinates, centroid positions, aspect ratio, compactness, and elongation. These features helped both our analysis and the model capture the spatial characteristics and structural properties of the geographical site.

Temporal features

For temporal features, we generated four variables representing the differences between consecutive dates (date0, date1, date2, date3, date4). Additionally, we created a binary variable indicating whether the status of the geographical site changed between two successive dates. Lastly, we introduced a feature to count the total number of status changes across the five given dates.

Color features

The original color features available in the dataset included the mean and standard deviation of each RGB channel across the five given dates. To enhance these features, we first computed the differences in the RGB channels' means and standard deviations between consecutive dates. Next, to capture temporal variability, we introduced variance ratio features, representing the fraction of the standard deviation to the mean for each RGB channel. Finally, we created weighted temporal mean and standard deviation features to better reflect trends over time.

1.4 Dealing with an imbalanced dataset

The dataset exhibits a highly imbalanced class distribution across the six change types. The majority class is `Residential` with 146,453 instances, followed by `Commercial` with 99,500 instances. `Demolition` and `Road` changes are significantly less frequent, with 31,156 and 14,190 occurrences, respectively. The `Industrial` category has only 1,310 samples, while `Mega Projects` is the rarest class, with just 149 instances. This class imbalance poses a challenge for model training, as it may lead to biased predictions favoring the majority classes.

This led us to adopt an oversampling strategy for the underrepresented classes. However, we observed that random oversampling did not significantly improve the model's predictive performance for the `Mega Projects` class due to its extreme imbalance. As a result, our approach focused on increasing the number of `Road` and `Industrial` instances through random oversampling using the `imblearn` library, which had a positive effect on the predicted accuracy of both of these classes.

2 Model tuning and comparison

In this section, we will explain how we selected our final classifier, the strategy used for hyperparameter tuning, feature filtering, and the resulting F1 score.

2.1 Choosing the appropriate model

For model selection, we experimented with various classifiers. Initially, we tried logistic regression, but it failed to exceed a 50% F1 score, despite removing correlated features and ensuring that the assumptions of logistic regression were met. This could be attributed to the fact that logistic regression is a linear classifier, while our independent variables do not exhibit a linear relationship with the outcome variable, `change_type`.
Next, we tested a Support Vector Machine (SVM) for multiclass classification, but this model also produced a low F1 score of just 20%.
We then shifted to neural networks. However, due to the challenges in selecting the optimal number of hidden layers and neurons, as well as the lack of interpretability of neural networks, we were unable to surpass a 50% F1 score.
Ultimately, we turned to one of the most widely used classifiers: decision trees. While decision trees did managed to exceed the 50% F1 score, the improvement was minimal. As a result, we decided to explore ensemble methods. We tested Bagging with Random Forest and Boosting with Extreme Gradient Boosting (XGBoost). Our findings revealed that boosting significantly outperformed bagging in terms of F1 score.
Therefore, our final model choice was the XGBoost classifier, see [1].

2.2 Hyperparameter tuning

For the XGBoost model, there were several hyperparameters to tune, including the maximum depth of the weak classifiers (decision trees), the learning rate, the fraction of columns each weak classifier considers, and the Lasso and Ridge regularization coefficients.
Given the large number of possible hyperparameter combinations, we decided not to use k-fold cross-validation due to its high computational cost. Additionally, we considered the evaluation on the shuffled 20% validation set to be a reliable estimate of the model's generalization error. As a result, we manually explored a selection of promising hyperparameter combinations and ultimately found a satisfactory set of parameters.
We train our final XGboost model on 800 boosting rounds.

2.3 Filtering Features: Impact on Model Performance

After training the first instance of XGboost with 80% of training data. We used the feature importances to gauge the number of features to be considered. We sorted features by their importance and then removed at each iteration 10% of the less important features. Then we trained and evaluated an XGboost model on the left variables.
As anticipated, we observed that an increased number of features led to a higher F1 score. This can be attributed to XGBoost's ability to perform automatic feature selection, driven by the importance metric of each feature. This metric reflects the reduction in impurity achieved by using a particular feature, as well as how frequently the feature is used to split decision tree nodes.
As a result, we include all features when training our XGBoost model on the full training dataset, despite the potential computational cost associated with it.

2.4 Kaggle score on test data

After submitting our predictions on the Kaggle platform, our XGBoost classifier achieved an impressive F1 score of **97.091% !**

References

[1] T. Chen and C. Guestrin, "XGBoost: A scalable tree boosting system," *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pp. 785–794, 2016.