-

– CENTRALESUPELEC –
MACHINE LEARNING Project

BOUTOUATOU Ouissal
YARTAOUI Farouk
OUCHNA Yassine
TBATOU Hamza
Team's Name : Mchache

1 Introduction

In this report, we are pleased to present the findings and methodologies employed in our machine learning project, which aimed at categorizing geographical areas into six distinct classes. Our approach drew upon a range of techniques and algorithms covered in our machine learning course this semester, complemented by practical insights gleaned from our lab sessions.

2 Data preprocessing

Our initial task involved preprocessing the data for thorough analysis, including converting the original geojson dataset into a CSV format. The dataset contained both training and testing data, with 296,146 samples in the training set and 120,526 in the testing set. It included key features such as observation dates, status change indicators for each date, neighborhood urban and geographic types, and polygon vector representations. Additionally, mean and standard deviation values of color images derived from satellite imagery were provided.
We commenced the preprocessing phase by purging the data of any duplicates and conducting an examination of the distribution of classes within the training dataset .(see Figure 1).

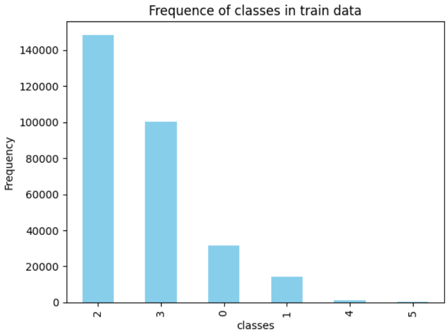

FIGURE 1 – Distribution of classes within the training dataset

Certain classes were more prevalent than others, requiring focus during model tuning. Additionally, we found missing values in both training and testing datasets, necessitating further preprocessing. Furthermore, the absence of sorting in dates and associated features highlighted the need for sorting procedures to improve data coherence.

2.1 Feature engineering

Given the diverse feature types in our dataset, some unrelated to our target labels, we needed to perform transformations on both training and testing datasets to achieve the following objectives :

1. Simplify and streamline feature types for improved usability.
2. Introduce new features with stronger connections to the labels while removing noise-inducing or complex features.
3. Address missing features to enhance robustness against data gaps.

2.2 Transforming features

Our aim was to convert all features into numerical form to support machine learning algorithms. The primary categories requiring treatment included multivalued categorical features ('urban type' and 'geography type'), dates, and geometric data (polygon vertex coordinates).

1. **'Urban type' and 'geography type' :** These columns represent comma-separated multiple values indicating neighborhood urban/geographical types. We applied one-hot encoding, transforming them into binary features where each category is represented by a binary

vector. For instance, 'urban type' was converted into five binary features : Dense urban, Industrial, Rural, Sparse urban, and Urban Slum. We handled instances with values 'N' and 'A' by creating additional binary columns and subsequently eliminating them to maintain consistency. Similar treatment was applied to 'geography type'.

2. **Status :** The status of images for each date is a unique categorical feature, enabling straightforward label encoding.
3. **Dates :** The conversion of dates into numerical format was necessary. Due to inconsistent sorting, we sorted dates for each data point, adjusting the corresponding columns accordingly.New date features were generated, with 'date0' transformed into numerical values representing the difference from the chosen reference date, set as the year 2000, in terms of days. Other date differences were similarly converted to reflect time steps from 'date0'.

2.2.1 Creating new features

We introduced new features related to dates and geometry :

— **Dates :** Recognizing that individual dates may not contain pertinent information, we focused on the differences between successive dates, creating new columns (e.g., delta1, delta2) to capture these variations, in terms of days. We retained 'date0' as it potentially held significant information, adjusting it by measuring its difference from the year 2000 to yield smaller numerical values.

— **Geometry :** To better characterize changes in specific regions, we generated new columns for area, perimeter, and edge count, which offer insights beyond polygon coordinates (e.g., larger projects correspond to larger areas).

2.2.2 Treating Null values

In both the train and test sets, we filled null values in each column with the mean of that column from the training set. This approach creates a recognizable pattern for the model to learn about the importance of features, as they assume the mean value of training data if missing.

2.3 Dimensionality reduction

After processing and adding new features, our dataset expanded to 62 columns. Intuitively, we aimed for dimensionality reduction and initially attempted Linear Discriminant Analysis (LDA), reducing the features to 5. However, this resulted in a disappointing outcome : with our best-performing model, random forest, the accuracy dropped by approximately 30%.

3 Model tuning and comparison

3.1 K-NN

we tried K-nearest neighbors using different attributes and different values of k. In the course we saw that as a rule of thumb it's better to choose $k = \sqrt{m}$ where m is the number of training instances, so we tested it out against other small values chosen randomly. We also The results were mediocre at best :

Values of k	3	12	35	50	80	$\sqrt{m} = 486$
Accuracy	60.35 %	63.78 %	63.22 %	62.73 %	62.01 %	58.73 %

It turned out that the rule of thumb doesn't work out in this case, and the best value is actually $k = 12$.

We disregarded this model since it didn't produce promising results and the fact that we have 60-ish features makes us lose the meaning of *distance* anyway.

3.2 Neural Networks

We experimented with various neural network architectures and discovered that having a large number of layers doesn't necessarily enhance learning. Surprisingly, a model with just two hidden layers, each containing a relatively modest number of neurons (50 and 30), achieved a 71% accuracy on a test set comprising 20% of the official training data. However, when submitting the results of the required test set, the accuracy did not surpass 40%.

3.3 Random Forest

Random forest was the sole model that showed promising results from the outset, yielding an initial accuracy of 90%,even without ordering dates.Our data processing pushed the accuracy beyond 97%. However, attempts to reduce feature count based on the most predictive features in our random forest classifier were futile ; accuracy plummeted to 50%.
The presence of unbalanced classes led us to explore three methods : oversampling, undersampling, and overfitting. Unfortunately, both oversampling and undersampling resulted in significant accuracy drops (21% and 61%, respectively). Surprisingly, tuning the model's parameters to increase complexity and address less represented classes proved ineffective, as default parameters from the sklearn library demonstrated superior predictivity.

4 Conclusion

In conclusion, throughout this challenge, we extensively explored various machine learning algorithms to assess their effectiveness in fulfilling the prediction task at hand. Surprisingly, only Random Forest emerged as a standout performer. Its success can be attributed to its inherent complexity and remarkable capability to handle categorical variables adeptly, eliminating the need for extensive preprocessing—a feature particularly beneficial in our dataset. Moreover, the ensemble learning approach of Random Forest, amalgamating multiple decision trees, often results in superior generalization compared to standalone models, especially when these models offer diverse and complementary perspectives. Additionally, Random Forest's resilience to overfitting and its capacity to navigate noisy data with minimal preprocessing likely contributed to its exceptional performance in this specific challenge.
This challenge also underscored the pivotal role of preprocessing and feature engineering, highlighting their significance in laying the groundwork for effective model development.