2EL1730: Machine Learning

CentraleSupélec - 2A

# Kaggle Project Report

27/09/2024

CentraleSupélec

*Autors:*
Emilie Bringer
Lydia Hammache
Mohamed Azahriou
Edward Lucyszyn

# Contents

# Introduction

The aim of this challenge is to classify given geographical areas into six classes (Demolition, Road, Residential, Commercial, Industrial, Mega Projects) using features already extracted from satellite images of the areas. Since the objective is to identify the eventual purpose of the buildings under construction, satellite images are captured at various stages of the building process. These extracted features consist of numerical data such as the dates when successive photos were taken, as well as the means and standard deviations of the primary colors in each photo. Additionally, categorical features include the urban and geographical neighborhood, as well as the construction status at different dates. The name of the team on Kaggle is **MJ**.

# Feature engineering

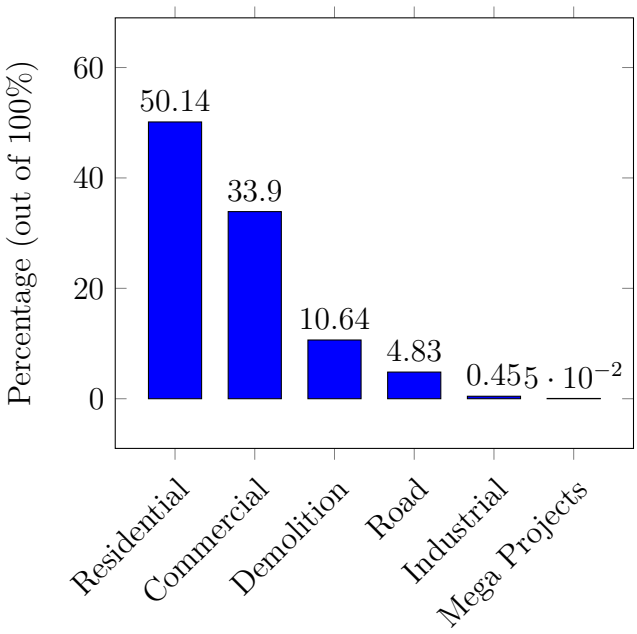
Before we can apply the algorithms covered in class to tackle this problem, we need to understand and organize the data being used.

## Description of the Data

Firstly, the training data consists of approximately 296,000 rows and exactly 45 columns. Among these 45 columns, they are:

- The **Polygon** that corresponds to the geographical area under study. Polygons are described as a list of points.
- The **Urban type**. It describes the urban characteristics of the photo. There are six types of urban landscapes: Dense Urban, Industrial, Rural, Sparse Urban, Urban Slum, and N/A. A location may belong to multiple categories simultaneously.
- The **Geography type**. It describes the characteristics of the environment of the photo. There are 12 types of geography type: Barren Land, Coastal, Dense Forest, Desert, Farms, Grass Land, Hills, Lakes, River, Snow, Sparse Forest, and N/A. A location may also belong to multiple categories.
- For each photo, the **Dates of the picture** in the 'dd-mm-yyyy' format.
- For each photo, the **Average and the Standard deviation of primary colors** (red, green, blue) in the picture.
- For each photo, the **Change status** for each photo. It describes the recent event that happened in the photo. There are ten types of them: Materials Introduced, Construction Done, Land Cleared, Construction Midway, Prior Construction, Construction Started, Materials Dumped, Greenland, Operational, and Excavation.

Moreover, only the training data contains labels in the **Change type** column. Below is the histogram showing the frequency of each category of labels. It is interesting to notice are not equally distributed.




## Dealing with missing data

Firstly, it was necessary to address the missing data. Indeed, the majority of algorithms studied in class and used in Machine Learning operate on complete data rather than columns with missing data. Some columns are unaffected by this phenomenon, such as Urban type, Geography type, and Geometry. However, all other columns have missing data, typically ranging from 1383 to 1954 missing entries.

Upon closer analysis of the missing data for each geographical type, it was noted that the number of missing entries was negligible compared to the total sample size. The initial instinct was to remove the rows with missing data. However, this approach was not feasible as we needed to be able to predict a geographical type for any input data format.

Therefore, we opted to replace the missing data with the mean for each numerical column and the mode for categorical data. This method of replacement with the mean and mode allows us to populate the table, albeit with potentially inaccurate information. Nevertheless, this approach is unlikely to significantly impact the results given the negligible frequency of missing data and the use of common replacement values.

## Sorting columns in chronological order

The dates were not sorted. For example, date 1 could correspond to November 28, 2019, and date 2 to December 13, 2013. Therefore, they needed to be arranged in chronological order. To achieve this, we utilized sorting with the Datetime format of the `Pandas` module. Consequently, for each feature associated with a date, it was necessary to reorder them chronologically according to their respective rows.

Rather than considering the dates themselves as data for our model, we chose to focus solely on the time interval between each date. This approach makes more sense because a date without reference is meaningless, and furthermore, the information in the Change Status column is very likely to depend on the time difference from the previous date.

We also included the mean and standard deviation of each primary color in our data. Of course, since these numerical data are always between 0 and 255, we standardized them.

### Encoding categorical columns

Afterwards, it was necessary to process the categorical data. Only the columns 'Geography Type', 'Urban type', and 'Change status' for each date were in this form. The initial idea was to apply one-hot encoding for these data. This is precisely what we did for 'Geography Type' and 'Urban type'. This approach is entirely suitable since these data can belong to multiple types, and these columns appear only once for each photo.

However, this idea is quite ineffective with the 'Change status' columns. Each 'Change Status' belongs to a single category among the 10 existing ones. Also, there are 5 'Change status' columns for the 5 existing dates. This means that if we had applied one-hot encoding, it would have resulted in 50 columns. For these two reasons, we preferred to establish a correspondence between each category and an integer between 0 and 9, inclusive. This saved us 45 columns and is much more efficient. Also, by analyzing the construction features, we realize that they follow a chronological order. Therefore, we classified the numbers from 0 to 9 according to the chronological order.

### Creating new features

We also created a new feature. For example, for the polygons, we considered the area of each polygon, but we also took into account the convexity of the shape. To achieve this, the selected feature is the ratio between the area of the polygon and the area of its convex hull. This results in a number between 0 and 1. The closer this number is to 1, the more convex the polygon is. After analysis, it was found that 17,069 (or 15%) of the surfaces are non-convex. This is coherent because a road is likely to be convex, but a building is not necessarily so. Therefore, this indicator is relevant for our problem.

### Treating large-scale Data

As we have large-scale data, we chose to save this data in multiple files to avoid recalculating them for each test of our model. For this purpose, we used the 'Save' function of the `NumPy` module.

Next, the last thing we did was to reduce the dimensionality of the data to reduce computation time and potentially increase the efficiency of our model. For this, we performed Principal Component Analysis (PCA). We retained the minimum number of dimensions to preserve 85% of the energy of our original data.

## Model tuning and comparison

After studying and formatting the data, we tested different models for this problem, starting with a basic model and then moving on to more complex approaches.

### k-Nearest Neighboors

The simplest algorithm is arguably the k-Nearest Neighbors (kNN). It was the example algorithm for the project. It achieved an accuracy of 33% on the test data. We used it very little

because we immediately had many features, and with a high number of dimensions, the concept of distance becomes redundant.

### SVM

Another algorithm we tried was the Support Vector Machine (SVM). We used a Gaussian kernel for which we found the theta by performing cross-validation with the `GridSearch` function. However, we did not initially achieve good results, even worse than those of the k-NN. This can be explained either by the fact that the model was not suitable for the problem, or that our training data formatting was not as good at the time, which skewed the results.

### Random Forests

One of the more complex models we tested was Random Forest. To do this, we used the `RandomForest` and `GridSearch` functions from `ScikitLearn`. This model is well-suited to our problem because it does not necessarily require normalizing the data, which is perfect since we have continuous and discrete numerical information. Additionally, it is easy to parallelize with Random Forests, which significantly reduces computation time. After reformatting the initial data, we achieved a maximum score of approximately 0.97.

### Chosen solution

We therefore chose the model with Random Forest. We also attempted to build a neural network, but the code was complicated to understand and did not yield good results, leading us to believe that we made a mistake. Since Random Forest already provided satisfactory results, we did not delve further into the issue.

| Model | Score |
|---|---|
| kNN | Between 0,30 and 0,40 |
| SVM | Less than 0,20 |
| RandomForest | Around 0,97 |

## Summary

This project has been a great opportunity to apply the principles covered in class. It allowed us to understand in more detail the various algorithms, but also to gain a professional perspective as data scientists. Indeed, we had to go through all the steps of the project, including preprocessing the data, which we had not seen much in class, especially dealing with missing data. Then, we had to test several models and find the right model with the right set of hyperparameters to achieve the best possible result. We are satisfied with this project, which serves as a good conclusion to the Machine Learning course.