Kaggle Competition Report

CentraleSupélec

Authors:
Mohammed Koucha
Mohammed Bouhamed
Ahmed Dargouthy
Hamza Jarraf

CentraleSupélec
Artificial Intelligence and Machine Learning Department
February 17, 2025

1 Data Processing

1.1 Introduction

Data preprocessing is a crucial step in any machine learning project. High-quality data ensures optimal performance for predictive models and provides more reliable results. Noisy, inconsistent, or incomplete data can lead to significant biases and distort the conclusions drawn from analyses.

In this study [nou25] [VPG21], we applied several transformation and cleaning techniques to our dataset to ensure better utilization of the available information. The primary goal was to maximize data quality and relevance while reducing complexity to facilitate model training.

1.2 Handling Missing Values

Missing values pose a significant challenge in data analysis. A large number of absent values can disrupt model learning and impact their generalization capabilities. To address this, we adopted an approach tailored to the nature of the data:

- For **numeric** columns, missing values were replaced with the **median** of each column. Using the median rather than the mean is advantageous as it is less sensitive to extreme values, ensuring better robustness.
- For **categorical** variables, we replaced missing values with specific categories such as "Unknown" for `urban_type` and "Unknown_geo" for `geography_type`. This allows us to retain these observations rather than discarding them, while explicitly marking the absence of information.
- Label normalization was performed to avoid inconsistencies, particularly by replacing "N,A" entries with more explicit null values.

1.3 Encoding Categorical Variables

Encoding categorical variables is essential for converting qualitative data into values usable by machine learning models. In this context, we applied **One-Hot Encoding** to the variables `urban_type` and `geography_type`. This involves creating a separate column for each unique category and assigning a value of 1 if the observation belongs to that category, otherwise 0.

This transformation allows models to correctly interpret the different classes without introducing artificial ordering.

1.4 Processing Dates and Statuses

Temporal analysis plays a fundamental role in predicting the evolution of the observed phenomenon. To leverage this information effectively, we performed several transformations on the date columns.

First, all dates were converted to the **datetime** format, ensuring consistent and homogeneous manipulation of temporal values. Next, we sorted these dates to ensure they were chronologically organized.

The statuses associated with these dates were also numerically encoded according to their progression order. For example, the status "Greenland" corresponds to the initial stage (0), while "Operational" corresponds to the final stage (9).

Finally, we introduced new features based on the time difference between each stage to capture trends and optimize model learning.

1.5 Cleaning and Analyzing Geometries

Geospatial data require special treatment to avoid analysis errors. In this study, we applied several filters to ensure the validity of geometries:

- Removal of **null or invalid** objects,
- Filtering entities to retain only **Polygons** and **MultiPolygons**,

- Conversion of coordinates to a homogeneous metric system (EPSG:3857) to standardize distances and areas.

Additionally, we calculated key indicators such as **area**, **perimeter**, and a **compactness index** measuring the shape of geographical objects.

1.6 Calculating Additional Attributes

To further enrich our dataset, we integrated additional features by calculating the **days elapsed between each construction stage**. This allows us to observe the progression speed and identify temporal trends that may influence model predictions.

1.7 Conclusion

In summary, the preprocessing we performed improved data quality by removing inconsistencies, enriching features, and ensuring better information coherence. These transformations are essential prerequisites for ensuring the relevance of analyses and optimizing the performance of machine learning models. Rigor in data cleaning is therefore crucial for obtaining robust and actionable results.

2 Development of a Classifier

2.1 Feature Selection

To optimize the performance of the classification model, we applied a **feature selection** process using the `SelectKBest` method from `sklearn.feature_selection`, combined with the **f_classif** statistical test based on ANOVA. This approach identifies the variables most correlated with the target variable, `change_type`, which represents the type of change occurring in the studied area. After encoding `change_type` into numerical values using a mapping dictionary (`change_type_map`), we extracted the most relevant features from the set of numerical variables. The selection was performed based on each feature's **F-score**, and the **35 most significant variables** were retained for model training. This process aims to **reduce data dimensionality** while preserving the most relevant information, thereby improving both the model's performance and generalization capability.

2.2 Random Forest Model and Hyperparameter Optimization

To improve the performance of our classification model, we implemented a **Random Forest Classifier** with iterative adjustments to its hyperparameters. Initially, we experimented with different numbers of trees (`n_estimators`) and progressively increased them to 120 to enhance model robustness. Similarly, we fine-tuned the `max_depth` parameter, setting it to 38 to balance between model complexity and overfitting prevention.

Moreover, we refined our feature selection process by reducing the number of input features from 35 to **25**, focusing on the most informative variables, gaining thus 1%. in the test accuracy This adjustment allowed the model to capitalize on the most significant predictors while reducing noise and computational cost. The model was trained on the selected features using the full dataset, leveraging

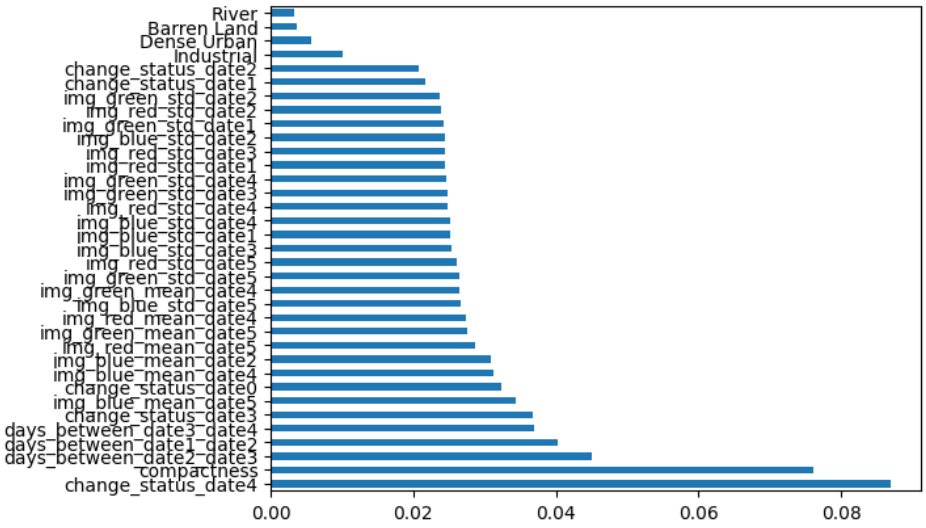

Figure 1: Feature importance in Random Forest

all available processing cores (`n_jobs=-1`) for efficiency. The accuracy results obtained from multiple submissions indicate the consistency of our approach, with scores stabilizing around 0.968, confirming the effectiveness of our hyperparameter tuning and feature selection strategy.

Model	Accuracy
Random Forest (final version)	0.96851
Random Forest (previous iterations)	0.96842, 0.96835, 0.96786, 0.96761
XGBoost (various configurations)	0.96580, 0.96691, 0.96681, 0.96304
XGBoost + Random Forest (ensemble)	0.63310

Table 1: Model performance comparison across different configurations

The results demonstrate that our optimized Random Forest model outperformed various XGBoost configurations. Additionally, an ensemble approach combining XGBoost and Random Forest yielded significantly lower performance, indicating that the models may not be complementary in this context.

2.3 AdaBoost Model and Performance Analysis

In addition to the Random Forest model, we experimented with **AdaBoost**, an ensemble learning algorithm that combines multiple weak classifiers to improve prediction accuracy. For this implementation, we used a **Decision Tree Classifier** with a maximum depth of 1 as the base learner, ensuring that each individual model remained simple and avoided overfitting.

The model was trained with `n_estimators=120` to allow for sufficient iterations of boosting, and a `learning_rate` of 0.2 to control the contribution of each weak learner to the final model. The training was performed using the selected 25 features, ensuring consistency with the previous experiments.

Despite tuning the hyperparameters, the AdaBoost model achieved an accuracy of only 79%, significantly lower than the optimized Random Forest model, which consistently reached around 96.8%. This discrepancy suggests that while AdaBoost can be effective in certain scenarios, it may not be the best suited algorithm for this specific classification problem. Possible reasons include:

- The dataset may be too complex for weak learners (depth-limited decision trees) to capture meaningful patterns.
- AdaBoost is more sensitive to noisy data, which might have influenced its learning process.
- The chosen learning rate and number of estimators may not be optimal for maximizing performance.

Given these findings, we decided to focus on further optimizing the **Random Forest model**, as it demonstrated superior predictive power for our dataset.

2.4 Logistic Regression: A Simpler but Less Effective Approach

In our exploration of classification models, we also tested a **Logistic Regression** model as a benchmark. While logistic regression is a well-established method for linear classification tasks, its performance depends heavily on the dataset's complexity and the separability of classes.

We configured the model with `multi_class='multinomial'` to handle the multi-class classification problem directly, using the `lbfgs` solver, which is efficient for small to medium-sized datasets. Since our dataset exhibited class imbalances, we set `class_weight='balanced'` to adjust for this automatically. Additionally, we fine-tuned the regularization strength by increasing `C` to 100.0, thereby reducing the effect of regularization, and we allowed up to 500 iterations to ensure convergence.

Despite these optimizations, the logistic regression model achieved an accuracy of only 42% (to be completed), significantly lower than the **Random Forest** model, which consistently reached around 96.8%. This performance gap suggests that:

- The dataset is likely non-linearly separable, limiting the effectiveness of logistic regression.
- More complex relationships between features may require non-linear models such as ensemble methods or neural networks.
- The model may be underfitting, as logistic regression lacks the ability to capture intricate interactions between variables.

While logistic regression serves as a useful baseline, these results confirm that more sophisticated models, like **Random Forest**, are better suited for our classification task. Consequently, we prioritized ensemble methods in further experiments to enhance predictive performance.

2.5 Neural Networks: A Complex but Less Promising Approach

Given the recent success of deep learning models in various classification tasks, we explored the potential of a **Multi-Layer Perceptron (MLP)** to improve performance. Our architecture consisted

of four fully connected layers with decreasing sizes: `(1024, 512, 256, 128)`, using the `ReLU` activation function to introduce non-linearity. The model was trained with the `adam` optimizer, an adaptive gradient-based optimization method, and an initial learning rate of 0.005 with an `adaptive` schedule, allowing the learning rate to adjust dynamically.

To mitigate overfitting, we introduced a small amount of L_2 regularization (`alpha` $= 10^{-2}$) but deliberately disabled `early_stopping` to let the model fully learn from the data. We also increased `max_iter` to 2000 to allow sufficient convergence time.

Despite these efforts, the accuracy obtained was below expectations, reaching only 76.5% (to be completed), significantly lower than our optimized **Random Forest** model. Several factors contributed to this outcome:

- The dataset might not have been large enough to fully leverage deep learning capabilities.
- The feature space might have been too limited for a deep neural network to extract meaningful hierarchical patterns.
- The tuning of hyperparameters, including layer sizes, learning rate, and batch size, added significant complexity without clear improvements.

Given the computational cost and the challenges of fine-tuning deep learning models, we decided not to further pursue this approach. Instead, we focused on **ensemble methods** like Random Forest, which provided both higher accuracy and interpretability with less effort.

References

[nou25] nouzir. 2el1730 machine learning project - jan. 2025. https://kaggle.com/competitions/2-el-1730-machine-learning-project-jan-2025, 2025. Kaggle.

[VPG21] Sagar Verma, Akash Panigrahi, and Siddharth Gupta. Qfabric: Multi-task change detection dataset. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR) Workshops*, pages 1052–1061, June 2021.