

Machine Learning Project 2026

Équipe Kaggle : For Sure

Angélique PENTENG, Titouan GAY-CORTIJO, Ely Cheikh Abass, Joseph LÉGER

17 février 2026

Introduction et Organisation

Pour la réalisation de ce projet, notre collaboration s'est articulée autour d'un dépôt GitLab centralisé. L'exécution des modèles a nécessité un environnement hybride, combinant nos machines personnelles avec les ressources cloud de Google Colab et des Notebooks Kaggle, afin de tirer parti de leur puissance de calcul GPU indispensable pour l'optimisation des hyperparamètres.

Présentation du Jeu de Données

Le jeu de données d'entraînement est constitué de 296 146 échantillons géospatiaux. L'analyse exploratoire de la variable cible (*change_type*) révèle une distribution extrêmement déséquilibrée qui pose un défi majeur de classification supervisé.

1 Section 1 : Feature Engineering

1.1 Motivation et Intuition Globale

L'objectif a été de convertir des données hétérogènes (séries temporelles, polygones et texte) en vecteurs structurés maximisant l'information discriminante. Notre stratégie repose sur quatre axes : la **morphologie** (formes types), la **chronologie** (rythme des chantiers), la **signature spectrale** (matériaux) et l'**environnement**.

1.2 Features Spatiales (Géométrie)

La morphologie des polygones constitue le premier levier de discrimination pour identifier la nature des projets (ex : structures linéaires pour les routes vs parcelles compactes pour le résidentiel). À l'aide de **GeoPandas** et **Shapely**, nous avons extrait des descripteurs géométriques avancés après avoir assuré la validité topologique des données via `make_valid`.

Projection : Les coordonnées WGS84 initiales ont été reprojetées en système métrique (UTM) via `estimate_utm_crs()` pour calculer l'aire et le périmètre pour éviter la distorsion à cause de la latitude.

Descripteurs de Forme : Pour capturer la "signature" visuelle des objets, nous avons extrait des indices morphologiques adimensionnels. Les dimensions du *Minimum Rotated Rectangle* (longueur, largeur) et l'**Aspect Ratio** permettent d'isoler les formes allongées typiques des infrastructures routières. Nous avons également calculé la **compacité** (Polsby-Popper) et des indicateurs de **complexité** (aire convexe, nombre de sommets, rayon maximal) pour distinguer les structures massives et irrégulières (*Mega Projects*) des parcelles simples comme les routes. Enfin, les coordonnées des **centroïdes** ont été intégrées pour capturer les biais géographiques régionaux.

1.3 Dynamique Temporelle et Cinétique de Chantier

L'exploitation de la dimension temporelle impose une reconstruction rigoureuse de la chronologie ($t_0 \rightarrow t_4$) pour pallier les incohérences d'indexation, les dates désordonnées et les valeurs manquantes du dataset brut. Cette étape inclut un *forward-fill* des statuts afin de garantir la continuité sémantique du cycle de vie des projets et d'éviter des ruptures artificielles lors du calcul des variations. À partir de cette timeline synchronisée, nous avons dérivé des descripteurs de rythme capturant l'amplitude globale (`total_duration_days`), les intervalles locaux (`delta_days_k`), la fréquence de changement réel et les délais conditionnels liés aux transitions d'état (`days_if_change_*`).

L'analyse est enrichie par l'agrégation du temps passé par phase métier (`phase_days_<X>`), les délais d'apparition des statuts et le décompte des transitions spécifiques (`tr_<A>_to_`). Pour stabiliser l'apprentissage, des indicateurs de synthèse ont été intégrés, notamment le niveau ordinal maximal atteint (`status_rank_max`), la progression nette (`status_progression_delta`), le nombre de statuts uniques et la durée active du chantier

(`work_duration_days`). Cette approche transforme une suite d'événements en une véritable cinétique de changement : elle permet au modèle de discriminer les signatures rapides et brutales des démolitions des évolutions pluriannuelles ou de la stabilité prolongée caractérisant le tissu résidentiel.

1.4 Encodage du Contexte Sémantique et Dynamique

L'environnement d'un projet étant déterminant pour sa classification, les tags multi-valués des colonnes `urban_type` et `geography_type` ont été transformés en variables binaires (`urban_*`, `geo_*`) via un *one-hot encoding* multi-label. Cette approche préserve l'intégralité du contexte contrairement à un encodage mono-catégoriel. Pour condenser ce signal, nous avons dérivé un `urban_density_score` (échelle ordinaire d'intensité urbaine) et des indicateurs géographiques ciblés (`has_water`, `has_dense_forest`, `has_farms`, `is_rural`) capturant les contraintes du terrain à faible coût dimensionnel. Enfin, l'état sémantique précis à chaque mesure est intégré via 50 indicateurs binaires `status_<STATUS>_date<i>`, permettant au modèle d'identifier des configurations temporelles complexes sans imposer d'hypothèse linéaire sur l'évolution du chantier.

1.5 Pipeline de Sélection et Analyse de Pertinence

Une fois l'ensemble des features créées, nous avons plus de 300 variables., nous avons mis en place un protocole de sélection en deux étapes visant à maximiser la parcimonie du modèle sans sacrifier sa précision.

1.5.1 Filtrage de la Redondance (Corrélation)

La première phase a consisté à éliminer la colinéarité excessive. Nous avons calculé une matrice de corrélation de Pearson pour l'ensemble des descripteurs.

- **Seuil de coupure** : Toutes les paires de features présentant un coefficient $|r| > 0.95$ ont été isolées. Pour chaque paire, seule la variable la plus corrélée à la cible a été conservée.
- **Résultat** : Cette étape a permis de supprimer plus de 100 variables redondantes, notamment des indices spectraux qui n'apportaient aucune information supplémentaire par rapport aux canaux RGB bruts (voir Fig. 1).

1.5.2 Classement par Importance de Gini

Le set de données filtré a ensuite été soumis à un modèle de *Random Forest* pour évaluer l'importance relative de chaque feature via le *Mean Decrease in Gini*. Ce code renvoie une liste de toutes les features triées par importance.

Nous avons ensuite utilisé un certain pourcentage de ces features en partant des meilleurs. Nous n'avons pas eu la puissance de calcul pour faire une élimination récursive des features mais nous avons environ évalué que XGBoost performait mieux avec une centaine de feature contre une 50aine pour Random Forest.

2 Réglage des modèles et comparaison

2.1 Méthodologie d'évaluation

Afin d'estimer la capacité de généralisation de nos modèles et de prévenir le surapprentissage, nous avons opté pour une validation croisée stratifiée (*Stratified K-Fold Cross-Validation*).

Cette approche repose sur deux piliers :

- **Validation Croisée ($K = 3$ à 5)** : La stratification assure que la distribution des classes est préservée dans chaque pli, ce qui est crucial compte tenu du déséquilibre des données. Le choix d'un K entre 3 et 5 résulte d'un compromis entre la robustesse statistique et le coût computationnel.
- **Métrique** : La performance a été évaluée via le **F1-score pondéré** (*weighted*).

2.2 Comparaison des classifieurs et sélection

2.2.1 Modèles rejetés

- **Support Vector Machines (SVM)** : Nous avons testé un SVM linéaire. Les performances se sont révélées médiocres ($\approx 40\%$). L'utilisation de noyaux non-linéaires (RBF) s'est avérée prohibitive en temps de calcul du fait de la haute dimensionnalité de notre espace de features après ingénierie.
- **Réseaux de Neurones (NN)** : Différentes architectures de perceptrons multicouches ont été testées en variant la profondeur et le nombre de neurones par couche. Le modèle a peiné à converger, plafonnant autour de 62% de F1-score, suggérant un manque de données pour entraîner efficacement une architecture profonde ou une architecture inadaptée aux données tabulaires structurées de ce projet.

- **K-Nearest Neighbors (KNN)** : Initialement testé sur la base du code fourni, ce modèle n’a donné de bonne performance. Avec l’ajout massif de features temporelles et spatiales, et donc de dimensions nous n’avons pas jugé pertinent de le réévaluer en fin de projet.
- **Régression Logistique** : Sa faible performance confirme la non-linéarité complexe des frontières de décision entre les types de changements urbains.

2.2.2 Modèles performants : Méthodes d’ensemble

Les méthodes basées sur les arbres de décision se sont rapidement imposées.

- **CatBoost** : Bien qu’avantageux pour sa gestion native des variables catégorielles (sans encodage préalable), il a montré des performances inférieures à XGBoost dans nos expérimentations.
- **Random Forest** : Avec un score moyen de 76% sur une CV à 5plis, ce modèle est notre deuxième meilleur modèle. Notre meilleur submission kaggle avec un RF est de 96.235%.
- **XGBoost** : Notre meilleur modèle individuel obtient un F1 score de 78% sur une CV à 5 plis. Sa capacité à optimiser une fonction de perte spécifique et à gérer les valeurs manquantes s’est révélée décisive. Notre meilleur submission kaggle avec un RF est de 97.345%.

2.3 Optimisation des hyperparamètres

Pour les modèles retenus (RF et XGBoost), nous avons procédé à une recherche par grille (*Grid Search*).

- **Infrastructure** : Face à la demande en ressources, nous avons exploité les GPU fournis par Kaggle et Google Colab.
- **Constat** : Il est apparu que le gain de performance lié au réglage fin des hyperparamètres (profondeur des arbres, *learning rate*, sous-échantillonnage) était marginal comparé au gain apporté par le Feature Engineering. Tant que les hyperparamètres restaient dans des plages raisonnables, la qualité et la quantité des features prédominaient.

2.4 Stratégies de Stacking et Blending

Dans une tentative d’améliorer nos résultats, nous avons combiné nos deux meilleurs modèles (XGBoost et Random Forest) via deux approches :

1. **Vote Pondéré** : Nous avons appliqué une moyenne pondérée des probabilités prédites :

$$P_{final} = 0.6 \times P_{XGB} + 0.4 \times P_{RF}$$

Malgré la prépondérance donnée au meilleur modèle, cette approche n’a pas surpassé le XGBoost seul, produisant un score très proche mais légèrement inférieur.

2. **Stacking** : Nous avons entraîné un méta-modèle (Random Forest : 100 arbres, profondeur max 4) prenant en entrée les prédictions des modèles de base. Le résultat fut décevant (72% en CV). Cette sous-performance suggère un problème probable d’implémentation (potentielle fuite de données ou *data leakage* dans la génération des prédictions hors-sac) ou un surapprentissage du méta-modèle sur le bruit des prédictions de base.

Conclusion

Ce projet nous a permis d’explorer en profondeur la chaîne de traitement d’un problème de classification géospatiale complexe. À travers l’expérimentation de divers algorithmes et une sélection de features rigoureuse, nous avons atteint un score préliminaire sur le *Leaderboard* Kaggle de **0.97345**, nous classant temporairement 8^{ème} sur 75 participants.

Cependant, une analyse fine de nos matrices de confusion révèle des disparités notables :

- **Points forts** : La classe *Demolition* est identifiée avec une grande précision, probablement grâce à sa signature visuelle et temporelle très distincte.
- **Points faibles** : Nos modèles peinent considérablement sur les classes minoritaires telles que *Industrial* et surtout *Mega Projects*.

Des techniques de rééquilibrage de données, telles que SMOTE (*Synthetic Minority Over-sampling Technique*) et l’*undersampling*, ont été tentées pour corriger ce biais, mais n’ont pas permis d’améliorer les métriques globales.

3 Annexe

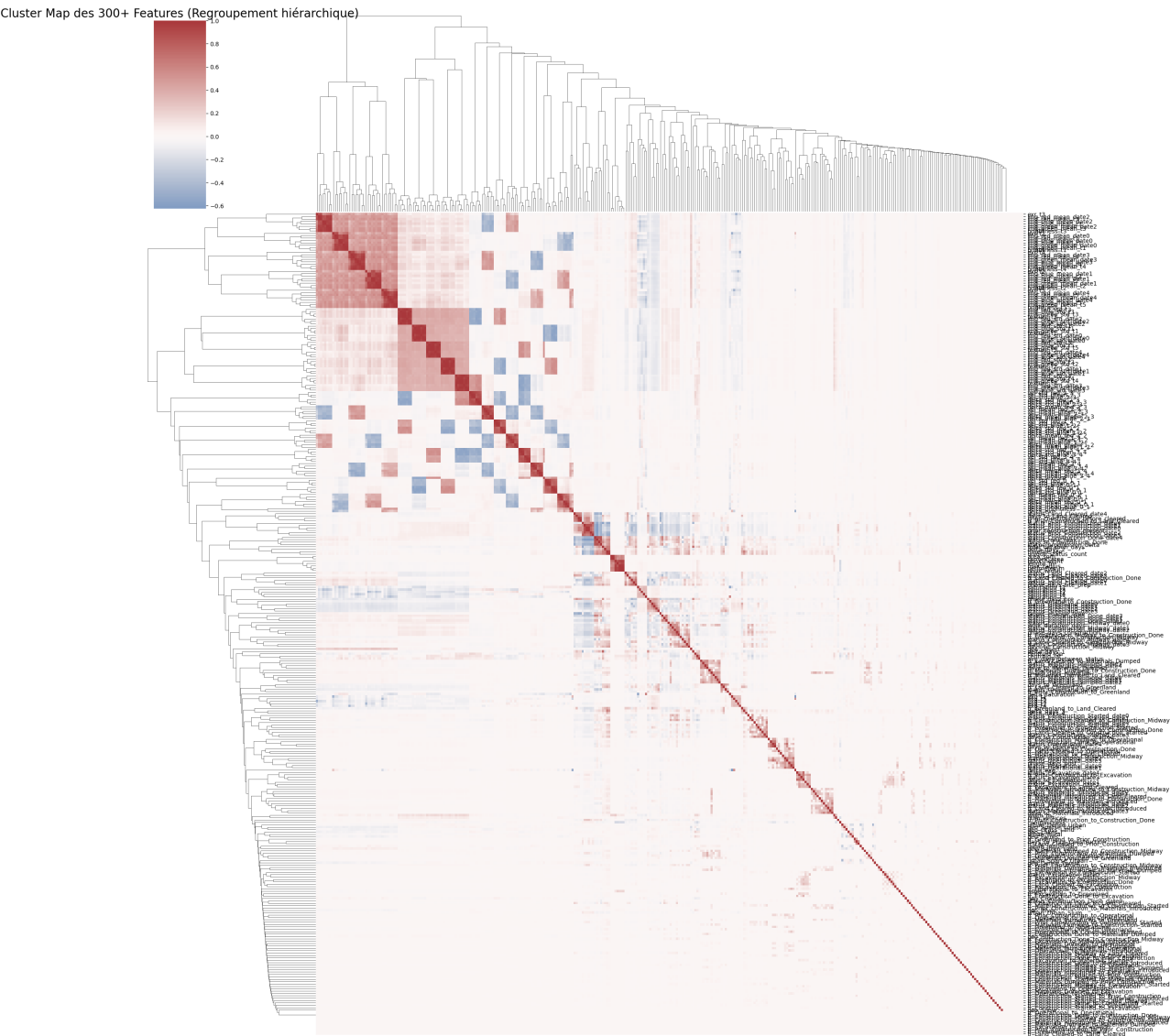


FIGURE 1 – Matrice de corrélation globale montrant la forte redondance entre les features.

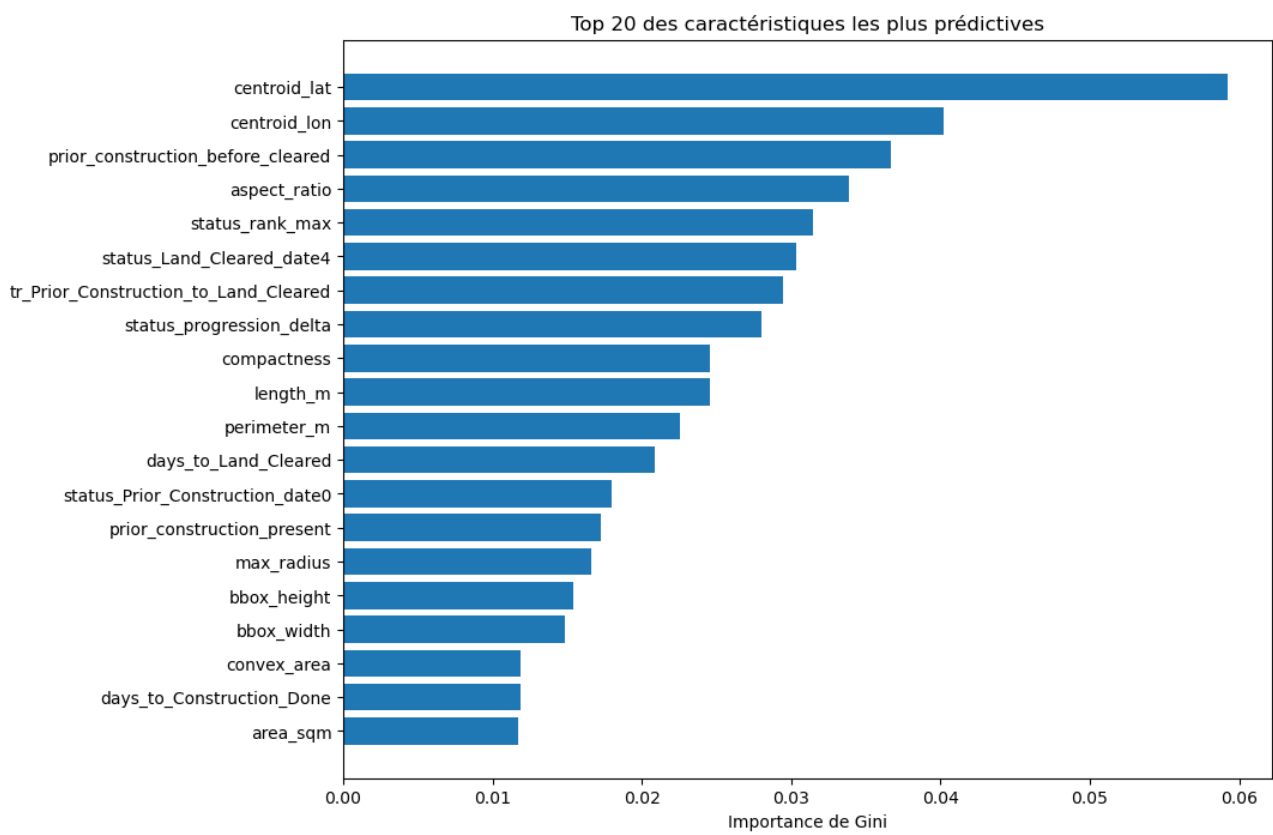


FIGURE 2 – Top 20 des features les plus discriminantes selon l'importance de Gini (Modèle final).