Kaggle Competition

Written by

Hatim Mrabet
Abdennacer Badaoui
salahidine Lemaachi
Oussama Er-Rahmany

Students of CentraleSupélec (Université Paris-Saclay)

Kaggle team

ChatGPT v2025

January 30, 2023

Contents

Chapter 1

Feature engineering

1.1 First steps of preprocessing

1.1.1 Ordering Dates

First thing we observed in the data, is that the dates were not ordered, that means that $date_0$ was not always the smallest date, so we decide to order data, so as to obtain the correct order. After this, since many features depend on the date's feature, such as **change_status_date**'s features, **mean_colours** features (**img_green_mean_date1**,**img_red_mean_date1** ...) and **Coulour_std** features (**img_green_std_date1**,**img_red_std_date4** ...), we had to reorganize the data, so we wrote an algorithm that creates a new data set, where the dates are sorted and all the related features are organized.

1.1.2 Handling the NaN and infinity values

Since the data had many missing values, we decided to keep these data instead of getting rid of it. So we replaced each missing value, by the mean value of the corresponding column, that means that each missing value get the mean value of the corresponding feature. This task was done using the predefined function in Python **fillna()**, ad we give it as a parameter the list of the means values of all the columns. We did the same processing to get rid of the infinity values, but instead of replacing by the mean values, we replaced by the maximum of the corresponding feature.

1.2 New features

1.2.1 Dates

First, we noticed that the delay between the dates for each class was meaningful. Actually, for **Mega Projects**, for instance, we observed that the delay between dates was very big in comparison with other classes. As a consequence, we decided to add 4 new features, called **date_diff_i_i+1**, where **i** goes from **0 to 3**.

1.2.2 Urban and Geographical types

This features are categorical features, they are of type string, this type isn't supported by many Classification algorithms in Python, so we decided to turn it into a numerical feature. For urban type, there were exactly 6 classes : **'Dense Urban','Industrial','N,A','Sparse Urban','Rural','Urban Slum'**. We decided to turn these classes into new features, so then for each row if a class **A** is present in the **urban_type** feature, we put 1 for this row in our new feature **A**. The same thing was applied for **geography_type**, where we find 12 classes : **'N,A', 'Barren Land', 'Coastal', 'Dense Forest', 'Desert', 'Farms','Grass Land', 'Hills', 'Lakes', 'River', 'Snow','Sparse Forest'**.

1.2.3 Features related to the geometry

First, we added the feature **area** that gives the are of the Polygon. Then we observed that the characteristics of the Polygon may give additional information. This intuition comes from the idea that **Road** will be mainly regular Polygons, meanwhile Mega Projects will logically have more vertices. We decided then to add a new feature called **vertices_count** that refers to the number of vertices in each Polygon. Furthermore, we added other features, such as **Perimeter** that gives the perimeter of the polygon,**ratio A/P** that refers to the ratio of the area and the perimeter, **width** and **height** that are actually the width and the height corresponding to the minimum bounding box, and **aspect_ratio** the ratio of this height and width. We even added some more features by combining the geometric features, as the ratio of the area and the square of the length and the ration of the length and the width of the bounding box. These features appeared to be very important according to the graph of features importance that is based on MDI.

1.2.4 Change status features

For the **change_status_date** features, we observed that the labels were limited. There was exactly **10 classes** : **'Construction Done', 'Construction Midway','Construction Started','Excavation','Greenland', 'Land Cleared', 'Materials Dumped', 'Materials','Introduced', 'Operational','Prior Construction'**. We decided then to add for each date, 10 features that refers to these classes. For each date **i**, and for each row, if the class **A** is present in the **change_status_date_i** feature case, we assign 1 for this row in our new feature, that we will call **A_i**.We even added a new feature that calculate the score of each class change_status through the 5 dates.

1.3 Impact of features on predictive importance

After choosing the best model, we plotted the graph below that allows us to visualize the importance of each feature. This was done using the **plot_importance** function. We obtained the graph below.

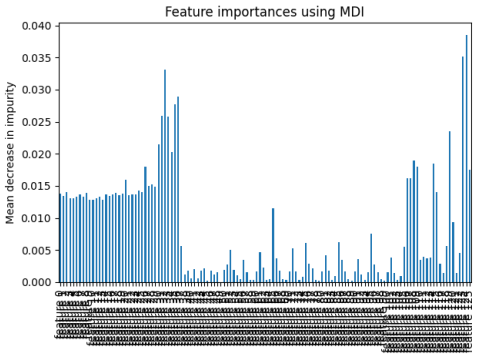

Figure 1.1: Feature Importance using MDI

This graph shows that the geometric characteristics of the Polygon have a huge impact, and are the most important features. We know that the labels can't be read, but we know the order of the labels by knowing the order of the features. You'll find in this code the list of all the features.

Chapter 2

Model tuning and comparison

To choose the best model, we trained different models on our data. You'll find below the different models we trained :

2.1 KNN :

We tried to improve the model that was given to us and we tested values of $k \in [1, 30]$ but it doesn't give us best results than the others models that we tried. We actually obtained a lower accuracy.

2.2 Multi-XGBoost :

We divide our training data into 4 sets; on each set we train 4 Xgboost models, in total 16 models. Each model takes a subset of random features among all our features. So we will have for each sample 16 predictions that will constitute our new data. Finally, we train Random Forest on this new data. That is to say that Random Forest takes 16 features which are the predictions given previously and the actual target.

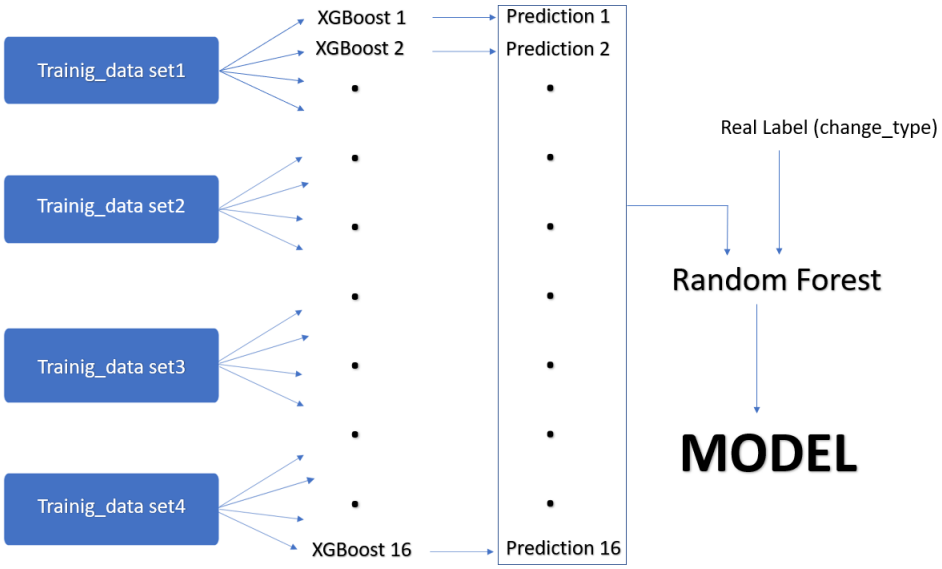

Figure 2.1: Our Ensemble Model

2.3 SVM with Text Categorization :

We also created another model that will allow us to manage categorical variables using Text Categorization with SVMs. The idea is to gather all the categories features: **geography_type , urban_type, change_status_0, change_status_1, change_status_2, change_status_3, change_status_4**, for each sample in a single sentence and use the method we have seen in TD that uses the TF-IDF matrix and combines it with the other numeric features

2.4 XGBoost :

XGBoost uses a tree-based ensemble learning approach, where multiple decision trees are combined to make a final prediction. The algorithm also includes regularization techniques to reduce overfitting, and advanced techniques such as parallel processing and cache optimization to improve its training speed. We used it with **params = {'objective': 'multi:softmax', 'num_class':6, 'max_depth':50 }** that we find them with cross-validation. You find its performances in the table below.

2.5 Random Forest

Our best model was the one using Random Forest classifier, in which which we used 600 estimators, the number of estimators is justified by the fact that the data is large, so we took approximately the square root of the number of data. We tried to add weights on our model, but we observed that it decreases the accuracy. This probably that we didn't pick the right weights. Our choice was based on the confusion matrix, that gives us an idea about the most misclassified labels. In addition, we worked with the **Gini criterion** to choose the best split. By using this model, we obtained an accuracy equal to **0.92466**, which was our highest accuracy on the test set.

2.6 Accuracy of the different models

Models	KNN	Multi-XGBoost	SVM	XGBoost	Random Forest
cross validated performance on the training data	0.67	0.8792	0.8121	0.7524	0.9134
Score on test set given by kaggle	0.72	0.8941	0.8231	0.7871	0.92601

We observe that **K_NN** doesn't perform well on the data, this is related to the fact in high dimensionality, the distance function doesn't have a true meaning (**curse of dimensionality**), and that what leads to a lot of misclassifications. Concerning the *Multi_Xgboost* model, it was so computationally very expensive, and the result weren't too good. The **SVM** model's performance was mainly good, but the accuracy didn't outpass 0.86. For our final model, that uses the **Random Forest classifier**, it outperformered all the other models, the reason behind this is that this classifier performs well on imbalanced data due to its bagging and random feature selection strategies. Bagging helps to reduce overfitting by combining predictions from multiple trees. Out-of-bag error estimation provides performance evaluation on minority classes.

Our score on Kaggle, with Random Forest, is 0.92601