2EL1730 - MACHINE LEARNING
Felipe Pimentel and Nicolas Verras - Team Amigos do Ratón
Julio Incerti and Melanie Pacheco - Team Amigos del Ratón

1 Data Treatment

The first approach taken in this project was data treatment and can be divided in three steps.

- **Remove Duplicates :** Basic tests to check for this type of data were applied and resulted in no duplicates in the training data.
- **Dealing with 'NaN' values :** Samples in the train data with 'NaN' values were dropped, except for the one-hot encoded data were 'NaN' values were replaced with zero.
- **Mapping the labels :** The labels in *'change_ type'* are mapped into numerical values as presented in the skeleton code provided.

2 Feature Engineering

2.1 Features *'urban_ type'* and *'geography_ type'*

One-hot encoding was applied to these features. This created 17 new columns, that only have binary values and are very sparse. The *'urban_ type'* and *'geography_ type'* features are dropped.

2.2 Time series related features

The features related to the colors and construction status for each of the five dates are considered here. After sorting the dates in ascending order, the slope of the linear regression of the time series for each of feature is taken as a new feature. The construction status is also mapped into a sequence of ordered integers from one to nine indicating the progression of the construction.

The reasoning is to capture the evolution of the color palette and construction speed. On top of that, it is expected that the "Demolished" label is the only one with has a negative value in the slope of the *'construction_ status'*. The original time series features are dropped.

2.3 Geographical positioning feature

The latitude and longitude of the centroid of each sample's polygon is used to cluster the sample's by geographical position using the k-means method. Using the elbow-method, the best value of k is chosen as four. With this feature, the geographical positioning information is captured.

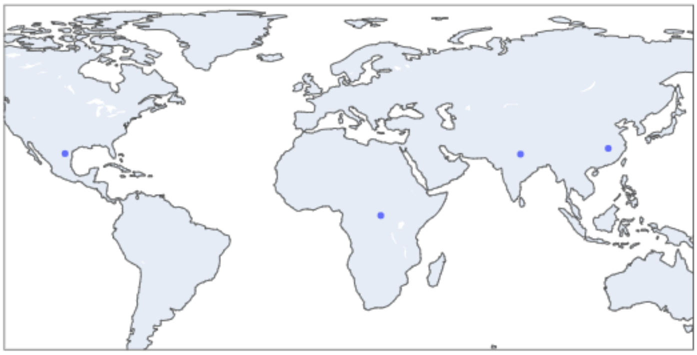

Figure 1 – Position of the four clusters using the K-means algorithm

2.4 Geometrical features

The geometrical attributes of the polygon are calculated through the area and perimeter. Therefore, the shape and size of the polygons is captured and expected to impact the classification, especially identifying "Mega Projects". The *’geometry’* feature is dropped.

2.5 Results

Following the proceedings described, the final features of the treated data are given in Table 1.

Feature	**Column(s)**	**Feature**	**Column(s)**
Geometry	***’area’***, ***’perimeter’***	Mean green color slope	*’green_ mean’*
Global position	***’geo_ cluster’***	Mean red color slope	*’red_ mean’*
Geographical type	*’geography_ Grass Land’*	Mean blue color slope	*’blue_ mean’*
	’geography_ River’	Green color STD slope	***’green_ std’***
	’geography_ Sparse Forest’	Red color STD slope	***’red_ std’***
	’geography_ Lakes’	Blue color STD slope	***’blue_ std’***
	’geography_ Hills’	Construction status slope	***’status’***
	’geography_ Barren Land’	Urban type	***’urban_ Industrial’***
	’geography_ Coastal’		*’urban_ Rural’*
	’geography_ Dense Forest’		***’urban_ Sparse Urban’***
	’geography_ Desert’		*’urban_ Urban Slum’*
	’geography_ Farms’		***’urban_ Dense Urban’***

TABLE 1 – Features present in the treated data. Highlights indicate chosen features.

Using both *f-classifi* and *f-regression* methods, the features performance is measured. In few words, the first method determines if is there a significant difference between the means of the feature and the target label, while the second tries to find a linear relation between these both groups. Both methods return a score value that indicates the relationship between the features and labels. In the analysis, all the features were considered and the 16 best were picked for the model, which are highlighted in Table 1.

3 Model tuning and comparison

3.1 Comparison of differents classifiers

To better compare the different methods, the train data must be divided into a pack that is used to model fitting and another that is used to estimate the final performance, therefore avoiding biased results. Choosing 75% of the training data, for all chosen models, the hyperparameters were optimized using Random Search, while the remaining 25% are used to validate the final tuned models performance. It is important to state that hyperparameter tuning can be computationally expensive and is a heavy limitation to model fitting.

As the training data is multi-classed, tree-based methods are evaluated because of their overall good performance with this type of data. Furthermore, the presence of one-hot encoded data along with numerical real data is another strong point in tree-based methods. As a comparison, the SVM method is expected to have worse results as it generally has a worse performance on data with the previously mentioned characteristics. Because of this, the method RandomForest is chosen and SVM also chosen for comparison.

Boosting methods are also expected to have good results since they put more weight into difficult to classify data and this could prove useful since the training data has a few instances of some classes, especially ’Mega Projects’ with 151 instances. Therefore, the method Extreme Gradient Boosting (XGBoost) is chosen.

Because of the data complexity, the Neural Network is also chosen, however due to the difficulty in tuning the model and limited computing power available, the hyperparameter analysis is not robust enough to provided a well tuned model. Therefore, the expectation is of a limited performance model.

Another method chosen is the KNN classifier, due to its capability in dealing with data similarity that is empowered especially by the geographical clusters feature. The expected performance is, therefore, optimistic.

3.2 Parameter tuning procedures

During the model tuning, several approaches were considered. All of them used the same set of features as described in the previous section. The part of the training data chosen to tune the parameters is further divided using cross-validation.

The optimization of the hyperparameters was done using the Random Search method. It is similar method to the Grid Search, but instead of testing all combinations in the neighborhood, the Random Search tests random combinations of hyperparameters, according to a specified number of samples to be drawn. It proves to be an interesting alternative when the data set is too large, or there are too many hyperparameters to optimize, favoring lower computational power.

3.3 Model performance and Kaggle result

Using the 25% of the training data previously reserved to validate the model and after the hyperparameters' tuning, the accuracy of the models is calculated and presented in Table 2. The accuracy of the Kaggle submission is added for comparison.

Model	Accuracy	Kaggle score
XGBoost	0.82	0.89
RandomForest	0.71	0.73
KNN	0.72	0.69

Table 2 – The best models accuracies

It can be seen that XGBoost achieved the best results for both accuracy and at the Kaggle score, following the previous presented reasoning. Along with it, good results were obtained with the KNN and RandomForest methods, as expected.

3.4 Additional models that did not perform well

The other types of models tested did not perform well to solve this problem, as expected for SVM and Neural Network. The accuracies of these models are presented in Table 3.

Model	Accuracy
SVM	0.62
Neural Network	0.59

Table 3 – Models that didn't perform well.

The poor performance of the Neural Network can be explained by its tuning complexity, proving challenging and computationally expensive, as expected. A better performance could be achieved with better tuning. This also applies to the SVM method, as it works best for linear data. Also, the high number of features makes the process very slow, which makes good optimization impractical.

4 Conclusion

Previous knowledge about the data set proved useful and extracting this knowledge using feature engineering is essential for a good prediction of the data. The estimation method is also chosen with the data characteristics in mind, especially since the computational limitation make it impossible to use brute force to find the methods, having to use rational choices taking into account the methods nature. A better model could have been achieved with better tuning and feature selection testing, both time consuming endeavors.