Kaggle Competition Report - Machine Learning Course
Flex.ai Team

Louis De Oliveira, Alexandre Sajus, Clement Wang, Antoine Debouchage

January 2022

Introduction

I. Feature Engineering

"Feature engineering refers to the process of using domain knowledge to select and transform the most relevant variables from raw data when creating a predictive model using machine learning or statistical modelling." In this competition, this is particularly the case as we are dealing with dates, urban and geographical types and the geometry of each data which all have a deep meaning on the way to classify them. We managed to create more than 400 features using the fact that our final model was performing feature selection. There are too many individual features to list, but the general categories that they fall into are detailed below.

Dates

In order to exploit the given dates, we first split all the dates into new features : day,month, year. Then we compute the gap between each date and add these as new features as well.

Categorical features: Urban, Geographical, Change Status

We replaced the Urban types and geopgraphy types columns by many columns encoding their contents with binary attributes.

Geometry

A good way to differentiate roads from other projects would be to extract the road's length and width as a feature. A road will have a high length/width ratio, while other projects won't. These features (length, width, and their ratio) could also be useful features in general.

Closest Neighbours discovery

Running out of ideas, we decided to investigate in further detail the test and train dataset provided. We discovered that the datasets were not shuffled and that it was blocks of buildings from the same satellite view. Taking advantage of this, we added lots of features corresponding to some major characteristics (area, perimeter, ratio, representative points...) of the k closest neighbour being in a certain radius intending to make our model understand the links between high-density residential districts and industrial areas. Adding the density of buildings to a distance also helps the model.

This is the idea that generates most of our features, because we can harvest information from all the original features of each neighbor, and then add means.

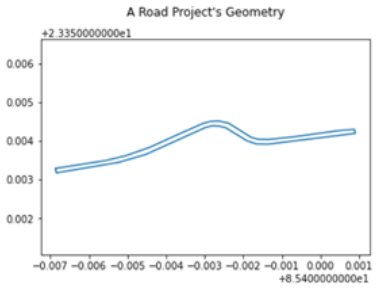

Figure 1: Road visualization.

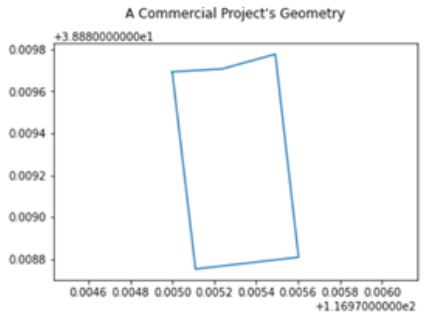

Figure 2: Commercial visualization.

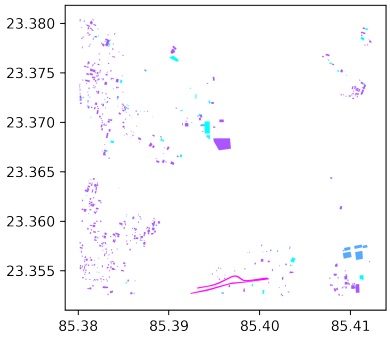

Figure 3: Neighborhood visualization.

II. Model tuning and comparison

Select the best model

Even though experience tells us that using Gradient Boosting models such as CatBoost, LightGBM or XGBoost are must-use techniques for winning tabular competitions in most of the cases, we are going to investigate thoroughly every models that we encountered in the course to stick with the instructions. Using very basic features engineering (that allowed us to reach $> 68\%$ on the public leaderboard) in order not to overload the workload and the time spent on the competition as our final feature engineered dataset takes a very long time to train (30min-1h), we got the results summed up in the table below:

	Model	Accuracy	AUC	Recall	Prec.	F1	Kappa	MCC	TT (Sec)
catboost	CatBoost Classifier	0.7756	0.8958	0.5291	0.7725	0.7724	0.6290	0.6294	31.9680
xgboost	Extreme Gradient Boosting	0.7723	0.8933	0.5292	0.7686	0.7686	0.6230	0.6237	9.4000
lightgbm	Light Gradient Boosting Machine	0.7502	0.8593	0.5152	0.7488	0.7483	0.5883	0.5889	18.3960
lda	Linear Discriminant Analysis	0.6925	0.8291	0.5156	0.6948	0.6896	0.4912	0.4927	12.8100
rbfsvm	SVM - Radial Kernel	0.6918	0.8217	0.3777	0.6869	0.6793	0.4723	0.4743	571.0380
dummy	Dummy Classifier	0.5281	0.5000	0.1667	0.2789	0.3650	0.0000	0.0000	0.2800
rf	Random Forest Classifier	0.7576	0.8860	0.4721	0.7573	0.7532	0.5942	0.5947	4.5640
knn	K Neighbors Classifier	0.7161	0.8321	0.4349	0.7078	0.7086	0.5210	0.5231	1.5000
dt	Decision Tree Classifier	0.6850	0.7356	0.4551	0.6854	0.6852	0.4793	0.4793	12.4740
ada	Ada Boost Classifier	0.6669	0.6471	0.4052	0.6506	0.6422	0.4196	0.4352	32.8940
lr	Logistic Regression	0.6183	0.7610	0.2728	0.6175	0.5858	0.3033	0.3144	3.4000
ridge	Ridge Classifier	0.5634	0.0000	0.3320	0.6113	0.5645	0.3172	0.3253	1.3230
nb	Naive Bayes	0.2804	0.6595	0.4332	0.6451	0.3653	0.1498	0.1966	1.0650
qda	Quadratic Discriminant Analysis	0.2074	0.4765	0.1967	0.3099	0.2269	-0.0289	-0.0407	7.2440
svm	SVM - Linear Kernel	0.1050	0.0000	0.2370	0.0213	0.0344	0.0412	0.0725	41.0850

Figure 4: Benchmark of different models

Without surprise, gradient boosting methods reveal to be the most cutting-edge models for this competition. Even though F1 score is used for the competition here we did not use Mean F1, hence it is more interpretable and coherent to evaluate our models using the MCC as it takes into consideration the imbalanced of the data to classify.

Remark: SVM using a Linear Kernel is absolutely a "no-no" as the features do not fit into any linear classifier solution.

Let's switch to deep models. Indeed, for tabular data using simple machine learning algorithms often do the trick but some networks perform relatively well and can even outperform gradient boosting ones. We test two different kinds of networks:

1. Tabular Networks such as TabTransformer or TabMLP in an attempt for our model to understand correlation and hidden relation between features that our feature engineering might not have captured. (≈ 0.61 on kaggle)

2. Simple Deep Networks such as Self-Normalizing Neural Network (SNN) and Gated Recurrent Network (GRN) showed great results in some Kaggle tabular competitions. (≈ 0.63 on kaggle)

The conclusion is that SNN surpasses other Neural Network tested but it did not manage to reach the level of LightGBM, CatBoost and XGBoost. Thus, we have chosen to use XGBoost as our final model to fine-tune.

Fine tuning and baseline

We trained all of our models using 10-fold cross-validation in order to keep track of the changes of hyperparameters that led to the best results.

XGBoost has many hyperparameters. The most important ones are :

- The learning rate η
- λ and α two regularizer parameters
- The number of samples per tree
- The number of features per tree
- The number of trees. This one will be replaced by a callback function to early stop the training if there is no gain in terms of loss.

All the other hyperparameters are tuned by hand to have a better understanding of the influence of each parameter.

Conclusion

Using only tabular data is a big restriction for better results as well as the highly imbalanced datasets provided for training the model. Indeed, under-represented classes (4,5) were abandoned automatically during training and class 2 and 3 are too highly correlated as a commercial building is just from above a residential one in terms of geometry and often located among residential residences resulting in lots of misclassification between both classes. We plotted the number of misclassified instances between to training results and it appears that almost 10% of the predicted train data are confused about those two classes.

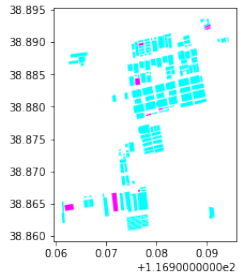

Figure 5: District with residential and commercial building.

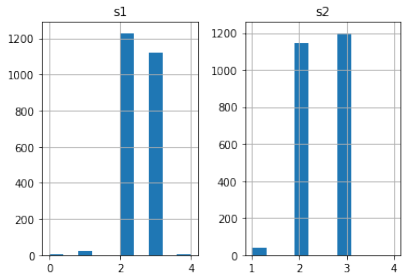

Figure 6: Misclassified classes between two runs.