Abstract

This project is an approach to solving the IEE-CIS-Fraud-detection binary classification problem in Kaggle competition using deep neural network and Random forrest identify a fraud or not.

The link to the competition is given below:

https://www.kaggle.com/c/ieee-fraud-detection/overview. The data is separated into train transaction and test transaction, train and test identity with a common merger on Transaction Id.

Methodology

Let's take a look at the training data and discover some important information about it, like the number of columns in the identity and transaction data, the number of frauds, and how much missing information there is in each column.

We have a very large dataset, with many missing values, and the columns does not have a clear meaning. The data description only tells us what are the categorical features and states that the column "isFraud" is the target variable, "TransactionDT" is a timedelta, and "TransactionID" is the column that joins the identity and transaction files.

There is a way to reduce the memory of a dataset, which consists of converting every categorical column data type to category and the numerical columns to the type that ocuppies the minimum amount of memory possible while also preserving all the information in that column.

Gradient boosting models like XGBoost and LightGBM are actually able to handle missing data, and since we already discussed that imputing nulls in this dataset is not easy we will not bother with that now. Since LightGBM is faster than XGBoost,

we will prefer this model here (like the majority of public kernels in this competition). The steps for modelling are the following:

Encode the categorical features - This is a tricky part. One-hot-encoding the categorical features allows us to to handle unseen categories, but it is not optimal for LightGBM, which performs better when categories are encoded as integers. LabelEncoder is able to encode categorical data in integers, however, it is meant to encode the target variable, not the features. This means that it only accepts a single column as the input and it cannot handle unseen categories. Since in this competition we only care about the training and test sets, one way to surpass this problem is to use LabelEncoder to encode the categorical features of the training and test set together, as done in https://www.kaggle.com/artgor/eda-and-models.

Search for the best hyperparameters - this is a necessary step for achieving better scores. However, since the code takes some minutes to run and this is only a simple model made to gain insights over the dataset, we will not bother with that now. Instead, we shall use the hyperparameters that were already tuned in other notebooks.

Kfold cross validate the model - we must do cross validation to see how well our model is able to perform on unseen data that is in the training set and to be able to obtain a better estimation of the feature importances. We evaluate the model using the AUC metric.

Predict the target variable on the test set - we shall submit our result and see how well our model can perform on unseen data outside the training set. It will be good to compare this score with the mean score on the cross validation sets to draw important conclusions that will guide our future work.

Data manipulation and feature engineering

Now it's time to do some data cleaning and manipulation, and some feature engineering to try to improve the model. One important part of the workflow that helps finding good ways of doing this is data visualization. I will not focus into it

very much here since there are some great kernels out there with many nice visualizations. Let's start by some basic data cleaning. The first thing that we shall do is to get rid of highly correlated features (most V features are correlated). Chris Deotte, one of the winners of the competition, excluded many V features in a very interesting way, which is presented in this notebook. Here I will prefer a rather simple approach, based on computing the correlation matrix and recording the correlated columns. Now we shall do some feature engineering. Feature engineering in this problem seemed very difficult to me, and I took a while reading discussion posts and notebooks trying to understand what techniques could be used in this problem. As already mentioned, we do not have much information about the dataset, which makes things even more difficult.

Result Analysis

Nonetheless, there are many ways to do feature engineering that does not require a deep understanding of the dataset, just a try and error approach. The many ways to do that are very well explained in this other topic written by Chris Deotte, and one good approach to this specific competition can be found in this notebook by Konstantin Yakovlev. Indeed, these guys did an amazing (and hard) work on this problem, and reading their notebooks and discussion posts were by far the moments when I learned the most. The model performance on the test set was about 0.91 (private score, 80% of the data), while the mean score on the cross-validation sets is 0.964. This means that our model is probably overfitting the training set. However, it is still surprising that with no work on the data LightGBM was able to achieve this score.