Data Science Laboratory
In-class Kaggle Report

Quinito Baula
qdb94
qdbaula@utexas.edu

ABSTRACT

A three-level model was implemented to achieve a public LB score of 0.86771. First, seventeen models were trained and their predictions on the training dataset were used as the features for the second level. Second, XGBoost and RandomForest models were trained on the first-level output and predictions were made on the testing set. Third, and finally, the harmonic mean of the outputs of the second level were taken to produce the final predictions. In this report, I'll describe my attempts in improving my LB score and my final pipeline regarding feature engineering and ensembling.

1. PIPELINE

The pipeline consists of three modules where each module is allowed to run independently: data processing, model training, and ensembling. A side effect of the modularity means we can modify the dataset without affecting other models and we can easily add new models to the ensemble. Furthermore, we do not have to retrain every model when creating the final predictions. This also means it is possible to create the ensemble even if some of the first-level models have not finished training yet. Lastly, design considerations were made to make it easy to add new extracted features into the training set at any level.

1.1 First Stage: Data Processing

The data processing module first reads the raw training and testing data and performs various preprocessing (fill, log, scale) and feature extraction (interaction terms, summations, count) to create different forms of the dataset. It is important to create different types of datasets in order to tailor to the algorithms used by the variety of models used. Some models can handle missing values while others cannot. Some models assume zero-mean unit-variance data while others are more robust. Once we have created the datasets, we can begin move on to the second stage of training.

Table 1: Models Used for Ensembling

Model	Dataset	Comments
RandomForest	Raw	
XGBoost	Raw	
XGBoost	Filled In	With the mean
XGBoost	30x Bagged	
XGBoost	Interaction Terms Added	2nd-degree polynomial terms
Adaboost	Raw	
ExtraTrees	Raw	
Logistic Regression	Log(X+1)	
KNearestNeighbors	Raw	2 Neighbors
KNearestNeighbors	Raw	4 Neighbors
KNearestNeighbors	Raw	8 Neighbors
KNearestNeighbors	Raw	16 Neighbors
KNearestNeighbors	Raw	32 Neighbors
KNearestNeighbors	Raw	64 Neighbors
KNearestNeighbors	Raw	128 Neighbors
KNearestNeighbors	Raw	256 Neighbors
KNearestNeighbors	Raw	512 Neighbors

1.2 Second Stage: Model Training

Table 1 lists the models that were used for the first level training and the datasets that correspond with each model.[1] The models are all trained using the same 5-fold stratified sampling of the training set. Particularly, a model is trained on only 4 folds and its predictions are gathered for the 1 holdout fold. This is repeated so that each fold becomes a holdout fold once. Each of the 5 times the model is trained, predictions are made on the testing set and their arithmetic mean is taken as the model's final predictions. The reason for using the same folds for all models is to decrease variance when training models and avoid overfitting at the second level.

The RandomForest, XGBoost, Adaboost, ExtraTrees, and LogisticRegression models were tuned using GridSearchCV to find the optimal parameters that would fit for their dataset. The parameter values chosen to search over was taken either online, by intuition, or by hand-tuning. The cross-validation scores unfortunately did not perfectly match the LB scores. However, once we ensemble the models together, the CV and LB scores of the final model line up and are very correlated.

[1]Ideas here were taken from the winner of the Otto product classification Kaggle competition.

1.3 Ensembling

Finally, we can run the third stage of my pipeline, which is ensembling. A RandomForest and XGBoost model are both trained by combining all the predictions on the training set from the 17 first-level models. Once both ensemble models are trained, their predictions on the combined testing set are made and the harmonic mean is taken as the final predictions produced by my pipeline. I chose to use the harmonic mean instead of weighted arithmetic or weighted geometric mean because it produced a better LB score.

2. ATTEMPTS

Interestingly enough, a simple RandomForest (i.e. 10 lines of code) performs well enough to place me on #16 of the public LB as of time of writing. It took me many attempts of feature engineering and ensembling before I could defeat my simple RandomForest model. Feature engineering methods I tried consisted of taking interaction terms of the most important features according to the simple RandomForest model. Trial and error was also useful here, so the five attemps a day becomes a very important (and scarce) resource. Bagging was another useful tool of feature engineering to control overfitting. Additionally, one-hot encoding does not work here since each feature has different "categories" between the training set and the testing set.

Ensembling was very important in increasing my LB score and making my CV scores valid and correlated. Using my offline CV scores as a guide became a great tool in choosing which submissions to use.

3. ONLINE PRESENCE

First, many forum posts were read about what works and what does not work in a Kaggle competition. Models and model parameters were taken from other people's work and the ideas for which metafeatures to use were also found online. General strategies and tactics were also outlined in blog posts by famous Kagglers. In short, many winning solutions are shown online and their techniques are further explained by authors in forum posts. They are extremely helpful and can provide a lot of inspiration and guidance for performing data science. The next step for this project is to clean up the documentation and publish my work, report, and thoughts to give back to the online community.

4. CONCLUSIONS

A three-level model was created for the purpose of the In-class Kaggle competition of EE 379K Data Science Laboratory in Fall 2016. The model performed to achieve the third spot in the public leaderboard of the class. Furthermore, the three-stage pipeline presented is general enough to apply for other datasets with modifications needed only in the frontend data processing stage. Additionally, he modularity provides benefits in cutting down development time and training time.