Dig out information from models—an ensemble technique

Di Tian
Electrical and Computer Engineering Department
Texas A&M University
College Station, TX, US
tiandi@tamu.edu

***Abstract*—In this paper, we introduce a novel ensemble technique to improve the prediction performance of machine learning models. By stacking multiple random forest regressors and create as well as correct a new "pre-prediction" feature during the training process, the prediction performance of random forest regressor was improved. This technique reveal that the information from previous models can be used to improve the overall model performance.**

Keywords—ensemble, model stack, pre-prediction, create and correct

1. INTRODUCTION

A. Simple Ensemble technique

The ensemble technique raised in this paper was motivated and experimented on a dataset comes from Kaggle competition "PUBG". When we explored the dataset, we found out that very similar records may have a very different target value which implies that there exists multiple "pattern" that "produce" the target value from the given features. However, it is usually very hard to recognize these patterns by human insights or simple clustering models. So we introduce a technique which stacks multiple models and tells the new model to make a new prediction based on the same records plus a new "pre-prediciton" feature, ie. the prediction made by the previous model. In practice, this process is usually iterative which means that we need to train many same models, and each model takes input as the same record as well as the output of the last model. By this simple iteratively ensemble technique, we have seen that the prediction accuracy was improved continuously as the iteration goes on. This technique helps machine learning model to dig out more information from their predecessors which compensates the lack of to some extent.

B. The game and dataset

Battle Royale-style video games have taken the world by storm. 100 players are dropped onto an island empty-handed and must explore, scavenge, and eliminate other players until only one is left standing, all while the play zone continues to shrink.

PlayerUnknown's BattleGrounds (PUBG) has enjoyed massive popularity. With over 50 million copies sold, it's the fifth best selling game of all time, and has millions of active monthly players."

In this project, competitors are given over 65,000 games' worth of anonymized player data, split into training and testing sets, and asked to predict final placement from final in-game stats and initial player ratings.

By exploring and modeling the given dataset, challengers are able to provide some very important insights about the best strategy to win the game. This is crucial to game development because a good understanding about the game itself helps developers to keep the game playable and balance.

In a PUBG game, up to 100 players start in each match (matchId). Players can be on teams (groupId) which get ranked at the end of the game (winPlacePerc) based on how many other teams are still alive when they are eliminated. In game, players can pick up different munitions, revive downed-but-not-out (knocked) teammates, drive vehicles, swim, run, shoot, and experience all of the consequences -- such as falling too far or running themselves over and eliminating themselves.

Challengers are provided with a large number of anonymized PUBG game stats, formatted so that each row contains one player's post-game stats. The data comes from matches of all types: solos, duos, squads, and custom; there is no guarantee of there being 100 players per match, nor at most 4 player per group.

Challengers must create a model which predicts players' finishing placement based on their final stats, on a scale from 1 (first place) to 0 (last place).

II. METHODOLOGY

A. Learning task

Notice that although this competition seems to be a classification problem, it is actually a regression problem.

In this competition, players were given some game data and were asked to predict the final placement of each player on a scale from 1 (first place) to 0 (last place). The final score will be evaluated by mean absolute value between target and prediction.

B. Basic model

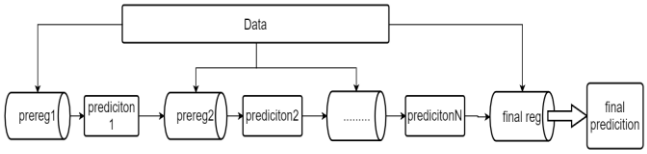
The original dataset contains 28 features each record. Most useful features are discrete and continuous numerical features. And different features have completely different scale and distribution.

Also, considering the ranking nature of this problem (although the target value is a float number scaled from 0 to 1, it does appear the nature of classification problem. For example, for a certain match with 100 players, the placements will be scaled from 0 to 1, with interval of 0.01. For example, the player who places 21th will have target

value of 1-21/100=0.79), random forest model is suitable for this problem. Since random forest does not need very careful and elegant feature engineering to handle the discrete and continuous features. Also, random forest regressor has the nature of classifier.

C. *Model stack*

The stack process can be shown clearly by the diagram below:



In this diagram, prereg1, prereg2…, preregN are all random forest regressors with exactly the same parameter configuration. We can clearly see that each random forest model take input as original dataset and the prediction of the last model. Once they successfully make a prediction, the old prediction will be replaced by the new prediction and sent to the next model along with original data. After iterative N time, we obtain N pre-regressors which form a regressor pipeline. And the Nth pre-prediction and the original data will be sent to the final regressor to make the final prediction.

III. RESULT

A. *Data resources*

The dataset comes from the Kaggle competition which can be fetched on https://www.kaggle.com/c/pubg-finish-placement-prediction/data.

B. *Experimental results*

a) *Comparison of models*

We have tried several basic models for this problem, here is the performance comparison of their baseline:

Model	MAE
Linear regression	0.0679
Feedforward neural network	0.0532
Random Forest	0.0514

Although neural network has a close performance to random forest, it requires very careful and elegant feature engineering as well as parameters tune. Stacking several NN will make the whole process much more time-consuming and tedious. Thus, we choose random forest as the basic model.

b) *Feature engineering*

Although random forest model does not require too much feature engineering to get a result. Some data preprocessing does help the model a little bit. The main feature engineering we have done in this project including:

① Data cleaning. Drop the records with “NaN” and “inf” values.

② Drop non-informative features. Some features such as “Id”, “matchId”, “matchType” provide little information. Other feature such as “DBNOs” has the same value in all records, which provide no information at all.

③ Feature creation. We have tried many ways to combine different features, such as divide, multiply and addition. The result shows that these new features did improve the performance a little bit.

④ Feature scaling. The result shows that normalization and standardization have no influence on random forest model at all.

Here is the improvement after applied certain feature engineering on the dataset.

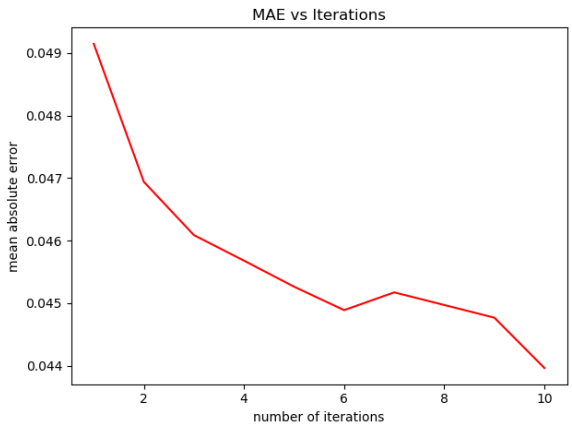
Feature engineering	MAE
No	0.0514
Yes	0.0487

c) *Model stack*

Here we show the influence of our ensemble technique. In our experiment, we used 11 pre-regressor and 1 final regressor. We can see that by applying this technique, 11 extra basic models provide about 0.006 decrease on mae. Which means the average prediction error is reduced by 0.6 placements. Considering the original average prediction error is about 4.8 placements, this is actually a decent improvement on performance.

Regressor	MAE
Prereg1	0.0487
Prereg2	0.0463
Prereg3	0.0456
Prereg4	0.0450
Prereg5	0.0445
Prereg6	0.0442
Prereg7	0.0444
Prereg8	0.0443
Prereg9	0.0442
Prereg10	0.04411
Prereg11	0.0439
Final reg	0.0429

Here is the amount of decrease on mae with respect to the number of stack iterations.



IV. CONCLUSIONS

In this paper, we proposed an ensemble technique to helps machine learning model dig out more useful information not from given dataset but from previous models. The result on "PUBG" dataset shows that as the stack iteration continued, the performance of a single model is enhanced by more specified models. These latter models are able to make a prediction based on original dataset and the output of the previous model.

This technique reveals that information can be not only obtained from dataset, but also dug out from machine learning models themselves.

In this paper, we simply replace the old prediction with the new prediction. There are many more ways to take advantage of the old prediction. For example, instead of replacing, we can apply moving average technique to it.

REFERENCES

[1] Shai Shalev-Shwarts and Shai Ben-David, "Understanding machie learning, from theory to algorithms". Cambrifge University press, 2014.

[2] Aurélien Géron, "Hands-On Machinle Learning with Scikit-Learn and TensorFlow". O'Reilly Media, Inc, 2017.

[3] Zhi-Hua Zhou, Ji Feng, "Deep Forest", IJCAI, 2017