Multi-Stage Ensemble and Feature Engineering for MOOC Dropout Prediction

Anonymous
Anonymous
anonymous@anonymous.com

Anonymous
Anonymous
anonymous@anonymous.com

Anonymous
Anonymous
anonymous@anonymous.com

***Abstract*—In this paper, we present the winning solution of KDD Cup 2015, where participants are asked to predict dropouts in a Massive Open Online Course (MOOC) platform. Our approach demonstrates best practices in feature engineering dealing with complex real world data, and pushes forward state-of-the-art for ensemble methods. We began with feature engineering. We extracted the hand-crafted and autoencoder features from raw student activity logs, course enrollment, and course material data. Then, we trained 64 classifiers with 8 different algorithms and different subsets of extracted features. Lastly, we blended predictions of classifiers with the multi-stage ensemble framework. Our final solution achieved AUC scores of 0.90918 and 0.90744 on the public and private leaderboards respectively, and put us to the 1st place out of 821 teams.**

I. Introduction

Since 1997, KDD Cup has been one of the most prestigious competitions in knowledge discovery and data mining, where experts around the world from both industry and academia compete with each other with best modeling practices to solve real world challenges in complex data sets.

The task of KDD Cup 2015 was to predict dropouts of students in a Massive Open Online Course (MOOC) platform. MOOC platforms aim at providing the mass population with open access to quality education. Despite of their initial success in some courses, MOOC platforms have struggled with extremely high dropout rates. Perna et al. reported that the average completion rate is 4% among 1 million students across 16 Coursera courses offered by the University of Pennsylvania from June 2012 to June 2013 [**?**]. If we identify those who are likely to drop out, we can engage with and help them complete courses successfully. For this task, XuetangX, one of the largest MOOC platforms in China provides the student activity logs, course enrollment, and course material data.

Student dropout in MOOC can be viewed as customer churn, which is a prevailing problem in publishing, financial services, insurance, electric utilities, health care, banking, Internet, telephone, and cable service industries [**?**]. There are previous competitions related to churn prediction. The task of Teradata Center for CRM (TCC)-Duke Competition was to predict churn with 171 variables provided by a major wireless telecommunication company [**?**]. The winning solution of TCC-Duke Competition was the ensemble of TreeNet (or MART [**?**]) models [**?**]. The task of KDD Cup 2009 was to predict churn, appetency and up-selling with 15,000 variables provided by the French telecommunication company Orange [**?**]. The winning solution of KDD Cup 2009 was the ensemble of single models trained with 10 base algorithms [**?**].

Both competitions focused on predictive modeling rather than feature engineering by providing preprocessed variables instead of raw data as in KDD Cup 2015. However, in practice, feature engineering is inevitable and crucial in predictive modeling. For example, Morik and Köpcke showed significant improvements in churn prediction performance of 4 different algorithms when time intervals for the states of variables are added to original variables [**?**]. Furthermore, winning solutions of both competitions used simple ensemble methods: an average of single model predictions at TCC-Duke Competition [**?**] and ensemble selection [**?**], which averages single model predictions with stepwise greedy forward selection, at KDD Cup 2009 [**?**].

Our final solution is a joint work from 9 data scientists, distributed around the world. The pipeline from raw data to final solution is as follows:

- Feature engineering - both hand crafted and automated.
- Single model training with feature sets.
- Stage-1 ensemble with single model predictions.
- Stage-2 ensemble with stage-1 ensemble model predictions.
- Stage-3 ensemble with all model predictions.

The rest of the paper is organized as follows. Section 2 describes our feature engineering approach. Section 3 introduces various classification algorithms used. Section 4 presents our multi-stage ensemble framework. Section 5 shows our final solution. Section 6 concludes the paper.

II. Feature Engineering

Our team members extracted 7 feature sets, namely F1, F2, F3, F4, F5, F6, and F7 from raw data independently.

A. Data Sets

Activity logs of 200,906 enrollments from 112,448 students across 39 courses are provided. Each activity is described by 6 fields of the username, course ID, timestamp, source, event, and object. For each object, 3 additional fields of the category, children, and start date are provided. The training set consists of 8,157,278 logs from 120,543 enrollments with the target variable indicating if a student dropped out. The test set consists of 5,387,848 logs from 80,363 enrollments. The full description of the data sets is available in [**?**]. In general,

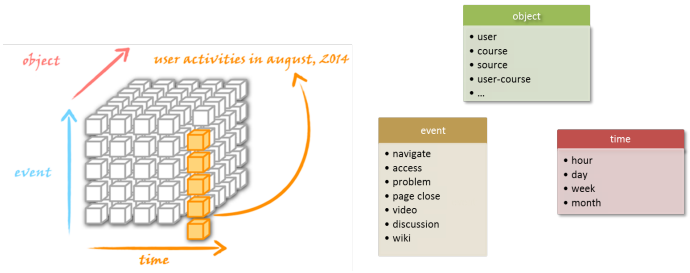

Fig. 1. Data cube.

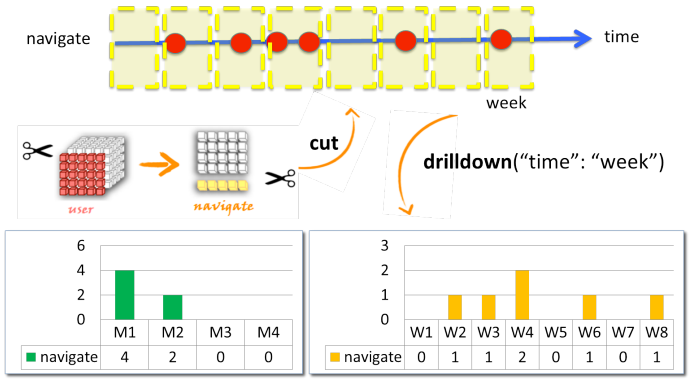

Fig. 2. Data slice and dice.

this data can be organized in 3 dimensional space of object, time, and event as shown in Figure 1. Feature engineering tasks were carried out based on these views.

B. Common Features

There are common features across 7 feature sets, F1 through F7 as follows:

- Number of objects
- Number of events
- Aggregation features of features above

The common features can be generated using cube operations. Figure 2 shows an example how weekly and monthly count features are calculated. First, the data is cut along with the object dimension. In this case, we choose to generate feature for users. Next, we select an event "navigate" in the event space to generate a time series presenting "navigate" event over the time. Finally, the drill-down operation is used to generate monthly or weekly count features.

C. F1

The feature set F1 is generated by Chen and Ozaki and includes features as follows:

- Enrollment-based features
- Username-based features
- Username-based features for each courses
- Features based on 10 days after the end date of course
- Features based on 1 day after the end date of a course
- Day-level features

Table A1 in Appendix shows the full list of features in F1.

D. F2

The feature set F2 is generated by Yan and Zhou and includes features as follows:

- Visit time(hour, day) set features (including time span and max absent days)
- Act(event, object) counting features (some uses missed content counts)
- Course drop rate
- Number of courses the user enrolled
- Minimum time interval between time points(first visit, last visit, course begin, course end, 10 days after course end) of current course and another enrolled course
- Active days between course end and 10 days after course end
- Active days between last visit and course end
- Number of courses ended after current course end

Table A2 in Appendix shows the full list of features in F2.

E. F3

The feature set F3 is generated by Nguyen, and are categorized into 3 groups of the count, aggregation, and date features.

1) Count Features: There are a few entities such as user, course, and object in the training dataset. Combining these entities together, we have user activities or events. The simplest way to generate features from these events is to count the number of times an entity engaging in the event. The motivation is that the more a user participate in course, the less likely she drop out the course. Table 1 shows the list of count features in F3.

TABLE I
LIST OF COUNT FEATURES IN F3.

No.	Description
1	The log count of each user
2	The log count of each course
3	The log count of each event
4	The log count of each user per week
5	The log count of each user per two weeks
6	The log count of each user per weekday
7	The log count of each user per month
8	The log count of each course per week
9	The log count of each course per two weeks
10	The log count of each course per weekday
11	The log count of each course per month
12	The log count of each event per week
13	The log count of each event per two weeks
14	The log count of each event per weekday
15	The log count of each event per month

2) Aggregation Features: Aggregation features are calculated based on count features. Usually, each course would have a fixed schedule for users to study. Therefore, students enrolled in the course must have consistent activity patterns. Aggregation features would measure the stability of course engagement. These features are minimum, mean, median, maximum, and standard deviation of count features on date basis such as weekly, monthly, etc.

3) Date Features: To capture how often users participate in a certain course, we generated date features. Date features can be time span among user activities as well as time span from last activity and last course date.

F. F4

The feature set F4 is generated by Jahrer, and includes features as follows:

- User ID
- Course ID
- User ID count
- Enrollment ID count
- Enrollment ID $\rightarrow$ Source ID count
- Enrollment ID $\rightarrow$ Event ID
- Enrollment ID $\rightarrow$ Object ID count
- Enrollment ID $\rightarrow$ Timestamp count
- User ID: floor(log(timespan^2+1))
- User ID $\rightarrow$ log(timespan to obj start + 1)
- Enrollment ID $\rightarrow$ timespan statistics

G. F5

The feature set F5 is generated by Bay, and includes features as follows:

- Course ID - One-hot-encoded course_id
- Source time counts by enrollment - The log count of each source type per day for each enrollment
- Source time counts by course id - The log count of each source type per day for each course id
- Event time counts by enrollment - The log count of each event type per day for each enrollment
- Event time counts by course id - The log count of each event type per day for each course id

H. F6

The feature set F6 is generated by Lee, and includes features as follows:

- User ID - One-hot-encoded username. Usernames appearing less than 100 times in training log data are grouped together as one user ID.
- Course ID - One-hot-encoded course_id.
- Source Event - One-hot-encoded combination of source and event.
- Object ID - One-hot-encoded object. Objects appearing less than 100 times in training log data are grouped together as one object ID.
- Count - Number of log entries for an hour_id.
- Object Category - Number of log entries with an object category for an enrollment_id.
- Number of Children Objects - One-hot-encoded total number of object's children for an enrollment_id.
- Object Timespan - One-hot-encoded timespan in days between object's start date and last day of the class
- Day of Class - One-hot-encoded day of the class
- Week of Class - One-hot-encoded week of the class
- End Month of Class - One-hot-encoded end month of the class
- Object Started in Dropout Period - Binary variable that is 1 if object started after but before 10 days after last day of the class and 0 otherwise.

I. F7

The feature set F7 is generated by Ozaki, and encodes target variables for each days. By adding F7 features, the best CV and public leaderboard score of our single model improved by 0.000846 and 0.001404 respectively.

- For 10 days after the end date of each course, number of active enrollment_id, which target variables are 1 in the training set, enrolled by a username.
- For 10 days after the end date of each course, number of active enrollment_id, which target variables are 0 in the training set, enrolled by a username.
- For 10 days after the end date of each course, number of active enrollment_id (in this case, days between last access and the end date of the course are also counted for active days), which target variables are 1 in the training set, enrolled by a username.
- For 10 days after the end date of each course, number of active enrollment_id (in this case, days between last access and the end date of the course are also counted for active days), which target variables are 0 in the training set, enrolled by a username.
- For 14 days before the end date of each course, number of active enrollment_id, which target variables are 1 in the training set, enrolled by a username.
- For 14 days before the end date of each course, number of active enrollment_id, which target variables are 0 in the training set, enrolled by a username.
- For 14 days before the end date of each course, number of active enrollment_id (in this case, days between last access and the end date of the course are also counted for active days), which target variables are 1 in the training set, enrolled by a username.
- For 14 days before the end date of each course, number of active enrollment_id (in this case, days between last access and the end date of the course are also counted for active days), which target variables are 0 in the training set, enrolled by a username.

J. Denoising Autoencoder Features

We generate denoising autoencoder (DAE) [**?**] features from feature sets above and use those as additional features. We experiment with two autoencoder networks:

- **Deep Stack** [**?**]: It is an architecture with input-x-x-x-input, where $x = 1,000$ for example. The resulting

feature dimensionality is $3 \times x$. We extract outputs from all layers as new features.

- **Bottleneck** [?] : It is an architecture with one layer with significantly fewer neurons such as input-1000-1000-30-1000-1000-input for example, which results in 30 features.

Both variants are trained with stochastic mini-batch SGD on the original training and test feature set. We use the rectified linear unit (ReLU) [?] transfer function in hidden layers and the linear function in the output and bottleneck layers.

III. Classification Algorithms

We selected algorithms that achieve good predictive performance, process large sparse data sets efficiently (with exception of K-Nearest Neighbors) and differ from other algorithms. The 8 classification algorithms selected are as follows:

- **Gradient Boosting Machine (GBM)**: We trained GBM classifiers using the Scikit-Learn Python package [?] and XGBoost [?]. We used various tree structures of 4 to 10 maximum depths and 0.004 to 0.05 shrinkage rates.
- **Neural Networks (NN)**: We trained NN classifiers with the dropout [?], ReLU transfer function and sigmoid activation function. We used various network architectures of 1 to 3 hidden layers and 16 to 500 hidden units per layer. We wrote our own C++ NN implementation optimized for sparse data sets.
- **Factorization Machine (FM)**: We trained FM classifiers using libFM [?] and libFFM [?]. We used 2-way interaction dimensions of 4 to 20. We transformed count variables x into $\log(1+x)$.
- **Logistic Regression (LR)**: We trained LR classifiers using the Scikit-Learn Python package [?] and Vowpal Wabbit [?]. We used the regularization parameter $C = 0.01$. We transformed count variables x into $\log(1+x)$.
- **Kernel Ridge Regression (KRR)**: We trained KRR classifiers using our own C++ implementation. We used the ridge regression constant $\lambda = 1.5e-3$ with the Gaussian kernel. We transformed count variables x into $\log(1+x)$.
- **Extremely Randomized Trees (ET)**: We trained ET classifiers using the Scikit-Learn Python package [?].
- **Random Forests (RF)**: We trained RF classifiers using the Scikit-Learn Python package [?]. Besides training classifiers, we used RF for feature selection. We trained an RF model and selected features with high variable importance.
- **K-Nearest Neighbors (KNN)**: We trained KNN classifiers using our own C++ implementation. We used $k = 124$ with the Euclidean distance. We transformed count variables x into $\log(1+x)$.

IV. Learning Framework

At previous KDD Cups, winning solutions either combined only single models without further combining ensemble models [?], [?], [?] or combined ensemble models based on public leaderboard scores, which are not available in practice [?], [?].

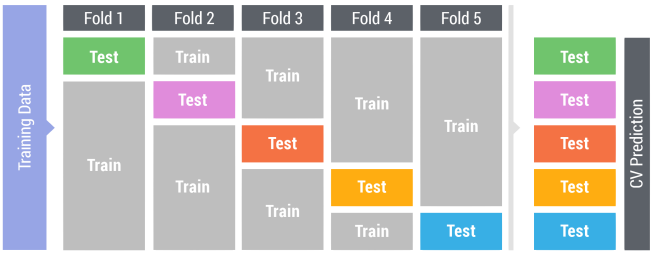

Fig. 3. 5-fold CV.

However, we were able to combine ensemble models in multiple stages without overfitting to training data or using public leaderboard scores by using our learning framework. Our learning framework consists of the stratified cross validation (CV) and multi-stage ensemble.

A. Model Validation

We used stratified 5-fold CV for model validation and ensemble. As shown in Figure 3, training data were split into 5 folds while the sample size and dropout rate were preserved across the folds.

For validation, each of single and ensemble models was trained 5 times. Each time, 1 fold was held out and the remaining 4 folds were used for training. Then, predictions for the hold-out folds were combined and formed the model's CV prediction. CV predictions were used as inputs for ensemble model training as well as the model's CV score calculation.

For test, each of single and ensemble models was retrained with whole training data. Then predictions for test data were used as inputs for ensemble prediction as well as for submission.

B. Multi-Stage Ensemble

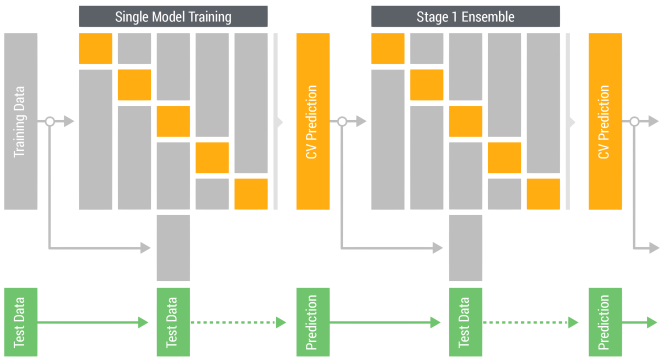

Fig. 4. 5-fold CV stacked generalization ensemble.

We used the multi-stage ensemble with stacked generalization [?] to blend predictions of various models. At each stage, we train ensemble models with 5-fold CV, and use the CV and test predictions of models in the previous stage as inputs. Then, we pass the CV and test predictions of the ensemble models

to the next stage as inputs. Figure 4 illustrates the process of multi-stage ensemble with 5-fold CV stacked generalization.

We stopped adding an additional ensemble stage when we saw no improvement in CV.

V. Final Solution

Our final solution is a stage-3 ensemble model trained with the multi-stage ensemble method described in Section 4.2 as follows:

- **Single Model Training**: First, we trained 64 single models with the 8 different algorithms and different subsets of 7 feature sets and DAE features. Out of 64 models, there were 26 GBM, 14 NN, 12 FM, 6 LR, 2 KRR, 2 ET, 1 RF, and 1 KNN models. Some of single models used RF feature selection, where we trained an RF model and selected features with high variable importances.
- **Stage-1 Ensemble**: Second, we trained 15 stage-1 ensemble models with different subsets of CV predictions of 64 single models. Out of 15 models, there were 7 GBM, 4 NN, 2 LR, 1 FM, and 1 ET models. Some of stage-1 ensemble models used rank orders between single models as additional inputs.
- **Stage-2 Ensemble**: Third, we trained 2 stage-1 ensemble models with different subsets of CV predictions of 15 stage-1 ensemble models. We used a LR with stepwise greedy forward selection and a GBM.
- **Stage-3 Ensemble**: Lastly, we trained a stage-3 ensemble model with CV predictions of all models. We used LR with stepwise greedy forward selection, and it selected 5 models out of total 81 models: 1 stage-2 ensemble models, 3 stage-1 ensemble models and 1 single model. Table 2 shows the list of models selected by the final stage-3 ensemble model.

Figure 6 shows the end-to-end pipeline for the final solution. Details of the single and ensemble models trained are available in Table A3 and A4 in Appendix.

Our final solution achieved AUC scores of 0.90918 and 0.90744 on the public and private leaderboards respectively, and put us to the 1st place out of 821 teams.

At KDD Cup 2015, we made some observations as follows:

- As shown in Figure 5, our CV scores were very consistent with public leaderboard scores. Therefore we used CV scores to determine (1) whether to add more ensemble stage or not and (2) whether to include a model for ensemble or not.
- GBM outperformed other algorithms. Our top 8 single models as well as top 2 stage-1 ensemble models are GBM models. NN and FM were next best algorithms. LR with stepwise greedy forward selection worked well in ensemble stages.
- Biggest performance improvement was from the stage-1 ensemble, and as we added more ensemble stages, we observed diminishing improvements. The stage-1, -2, and -3 ensembles improved the best CV score by 0.00967, 0.00028, and 0.000226 respectively. However, it was the improvement from the stage-2 and -3 ensemble that allowed us to finish the 1st.

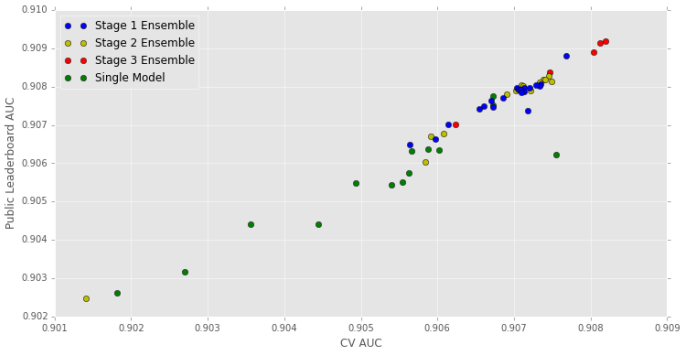

Fig. 5. CV vs. public leaderboard AUC scores.

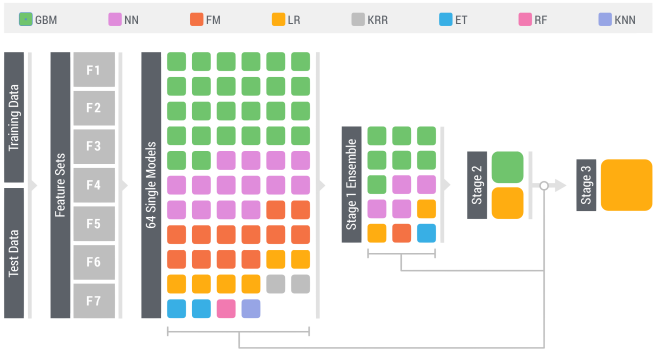

Fig. 6. End-to-end pipeline for the final solution

VI. Conclusions

In this paper, we demonstrated a comprehensive pipeline from the raw data to the final dropout prediction with best practices in predictive modeling. It started from feature engineering that extracts both hand crafted and automated features. At this step, discovering key features (such as F7 in Section 2.9) played a key role for us to proceed close to the top. Then, we trained 64 single models with 8 classification algorithms. Lastly, the multi-stage ensemble allows us to fully harness predictive signals in the extracted features and trained single models, and to finish the 1st at KDD Cup 2015.

Here we made 2 major contributions. First, our feature engineering approach can be useful for customer churn prediction in publishing, financial services, insurance, electric utilities,

TABLE II
Models selected in the stage-3 ensemble.

ID	Stage	Algorithm	5-CV	Weight
S1	Single	GBM	0.906721	1.1703
E4	1	GBM	0.907878	1.9626
E8	1	NN	0.907567	0.7871
E18	1	ET	0.906207	0.4580
E2	2	LR	0.907968	1.6146

health care, banking, Internet, telephone, and cable service industries, where similar customer log data are available. Second, we push forward current state-of-the-art for ensemble methods with our multi-stage ensemble framework.

ACKNOWLEDGMENT

The authors would like to thank...

REFERENCES

[1] L. Perna, A. Ruby, R. Boruch, N. Wang, J. Scull, C. Evans, and S. Ahmad, "The life cycle of a million MOOC users," in *Presentation at the MOOC Research Initiative Conference*, 2013.

[2] S. A. Neslin, S. Gupta, W. Kamakura, J. Lu, and C. H. Mason, "Defection detection: Measuring and understanding the predictive accuracy of customer churn models," *Journal of marketing research*, vol. 43, no. 2, pp. 204–211, 2006.

[3] J. H. Friedman and J. J. Meulman, "Multiple additive regression trees with application in epidemiology," *Statistics in medicine*, vol. 22, no. 9, pp. 1365–1381, 2003.

[4] https://www.salford-systems.com/resources/case-studies/119-teradata-center-for-crm-at-duke-competition.

[5] I. Guyon, V. Lemaire, M. Boullé, G. Dror, and D. Vogel, "Analysis of the KDD cup 2009: Fast scoring on a large orange customer database." *JMLR: Workshop and Conference Proceedings, Volume 7*, pp. 1–22, 2009.

[6] A. Niculescu-Mizil *et al.*, "Winning the KDD cup orange challenge with ensemble selection," *JMLR: Workshop and Conference Proceedings, Volume 7*, pp. 23–34, 2009.

[7] K. Morik and H. Köpcke, "Analysing customer churn in insurance data–a case study," in *Knowledge Discovery in Databases: PKDD 2004*. Springer, 2004, pp. 325–336.

[8] R. Caruana, A. Niculescu-Mizil, G. Crew, and A. Ksikes, "Ensemble selection from libraries of models," in *Proceedings of the twenty-first international conference on Machine learning*. ACM, 2004, p. 18.

[9] http://kddcup2015.com/submission-data.html.

[10] G. E. Hinton and R. R. Salakhutdinov, "Reducing the dimensionality of data with neural networks," *Science*, vol. 313, no. 5786, pp. 504–507, 2006.

[11] P. Vincent, H. Larochelle, I. Lajoie, Y. Bengio, and P.-A. Manzagol, "Stacked denoising autoencoders: Learning useful representations in a deep network with a local denoising criterion," *The Journal of Machine Learning Research*, vol. 11, pp. 3371–3408, 2010.

[12] T. N. Sainath, B. Kingsbury, and B. Ramabhadran, "Auto-encoder bottleneck features using deep belief networks," in *Acoustics, Speech and Signal Processing (ICASSP), 2012 IEEE International Conference on*. IEEE, 2012, pp. 4153–4156.

[13] G. E. Dahl, T. N. Sainath, and G. E. Hinton, "Improving deep neural networks for LVCSR using rectified linear units and dropout," in *Acoustics, Speech and Signal Processing (ICASSP), 2013 IEEE International Conference on*. IEEE, 2013, pp. 8609–8613.

[14] F. Pedregosa *et al.*, "Scikit-learn: Machine learning in Python," *Journal of Machine Learning Research*, vol. 12, pp. 2825–2830, 2011.

[15] T. Chen and T. He, "xgboost: eXtreme gradient boosting," *R package version 0.4-2*, 2015.

[16] N. Srivastava, G. Hinton, A. Krizhevsky, I. Sutskever, and R. Salakhutdinov, "Dropout: A simple way to prevent neural networks from overfitting," *The Journal of Machine Learning Research*, vol. 15, no. 1, pp. 1929–1958, 2014.

[17] S. Rendle, "Factorization machines with libFM," *ACM Transactions on Intelligent Systems and Technology (TIST)*, vol. 3, no. 3, p. 57, 2012.

[18] Y. Juan, Y. Zhuang, and W.-S. Chin, https://www.csie.ntu.edu.tw/∼cjlin/libffm/.

[19] J. Langford, L. Li, and A. Strehl, "Vowpal wabbit open source project," Technical Report, Yahoo, Tech. Rep., 2007.

[20] H.-F. Yu *et al.*, "Feature engineering and classifier ensemble for kdd cup 2010," in *Proceedings of the KDD Cup 2010 Workshop*, 2010, pp. 1–16.

[21] M. Jahrer, A. Toscher, J.-Y. Lee, J. Deng, H. Zhang, and J. Spoelstra, "Ensemble of collaborative filtering and feature engineered models for click through rate prediction," in *KDDCup Workshop*, 2012.

[22] P.-L. Chen *et al.*, "A linear ensemble of individual and blended models for music rating prediction," *JMLR: Workshop and Conference Proceedings, Volume 18*, pp. 21–60, 2011.

[23] K.-W. Wu *et al.*, "A two-stage ensemble of diverse models for advertisement ranking in KDD Cup 2012," in *ACM SIGKDD KDD-Cup WorkShop*, 2012.

[24] D. H. Wolpert, "Stacked generalization," *Neural networks*, vol. 5, no. 2, pp. 241–259, 1992.