Disclaimer of Liability

The material and information contained on this website is for general information, reference, and self-learning purposes only. You should not rely upon the material or information on the website as a basis for making any academic, business, legal or any other decisions. You should not copy any material or information on the website into any of your academic, business, legal or any other non-private usages. ZHANG Wengyu will not be responsible for any consequences due to your violations.

Whilst ZHANG Wengyu endeavours to keep the information up to date and correct, ZHANG Wengyu makes no representations or warranties of any kind, express or implied about the completeness, accuracy, reliability, suitability or availability with respect to the website or the information, products, services or related graphics contained on the website for any purpose. Any reliance you place on such material is therefore strictly at your own risk.

ZHANG Wengyu will not be liable for any false, inaccurate, inappropriate or incomplete information presented on the website.

Although every effort is made to keep the website up and running smoothly, due to the nature of the Internet and the technology involved, ZHANG Wengyu takes no responsibility for and will not be liable for the website being temporarily unavailable due to technical issues (or otherwise) beyond its control or for any loss or damage suffered as a result of the use of or access to, or inability to use or access this website whatsoever.

Certain links in this website will lead to websites which are not under the control of ZHANG Wengyu. When you activate these you will leave ZHANG Wengyu's website. ZHANG Wengyu has no control over and accepts no liability in respect of materials, products or services available on any website which is not under the control of ZHANG Wengyu.

To the extent not prohibited by law, in no circumstances shall ZHANG Wengyu be liable to you or any other third parties for any loss or damage (including, without limitation, damage for loss of business or loss of profits) arising directly or indirectly from your use of or inability to use, this site or any of the material contained in it.

Table of Content

Introduction

This report includes the participation in a data science competition, the Kaggle's Don't Overfit II Competition[1], which is a supervised learning competition, emphasising the importance of overfitting problem. To this end, **the objective of this report is to introduce various methods to address the overfitting problem** in the competition, by analyzing the dataset, choosing several models, attempting different training strategies, evaluating the model performance, and further improvement methods and analysis. Finally, a best "anti-overfitting" method is built with the leadboard score improved from 0.849 to 0.867.

1. Exploratory Data Analysis

1.1 Dataset Overview

Statistical Analysis

The Don't Overfit! II dataset consists (1) training dataset with 250 samples, and each sample has 300 continuous attributes, 1 binary target variable `target` and 1 sample index `id`, and (2) testing dataset with 19750 samples, and each sample has the same structure as the training dataset except the `target` variable. The final prediction on the testing dataset is evaluated by Kaggle's Submissions Scoring System.
In the training dataset, there are no missing data, no duplicate data. And it consists 160 samples with target value `1` and 90 samples with target value `0`.

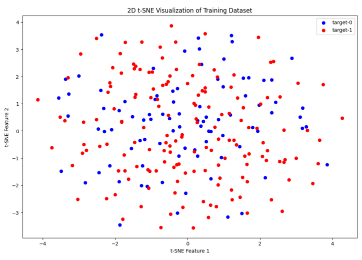


Figure 1: Distribution of Training Dataset

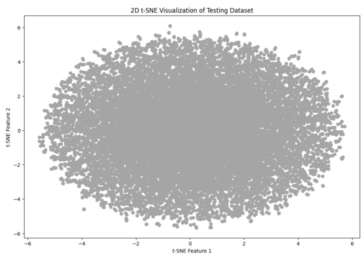


Figure 2: Distribution of Testing Dataset

By further observing Figure 1 and Figure 2 t-SNE visualizations of the trainng and testing dataset, it is found that:

1. In the training dataset, the number of attributes is greater than the number of the samples instants, which may lead to the overfitting problem.
2. In the training dataset, the number of the sample with target value `1` is greater than the number of the sample with target value `0`, which may potentially lead to the high bias in the unseen testing dataset.
3. The number of the testing sample is far greater than the number of the training sample, which may lead to a serious overfitting problem.

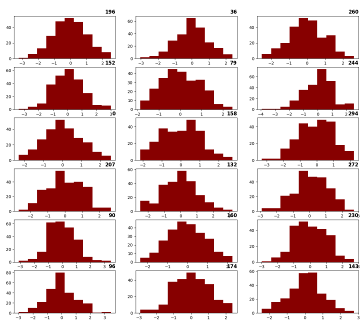


Figure 3: Distributions of Random 18 Columns

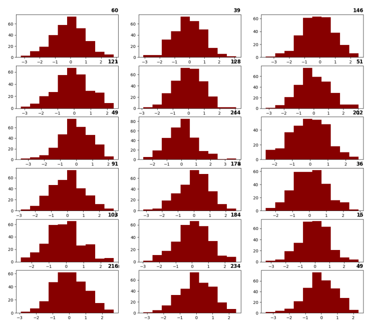


Figure 4: Distributions of Random 18 Rows

Feature Distribution Analysis

By further observing distributions of the randomly selected 18 columns and 18 rows from training dataset as shonw in Figure 3 and Figure 4, it is found that:

- For each attribute (column), the value among samples is almost normally distributed;
- And for each sample (row), the value among attributes is almost normally distributed.

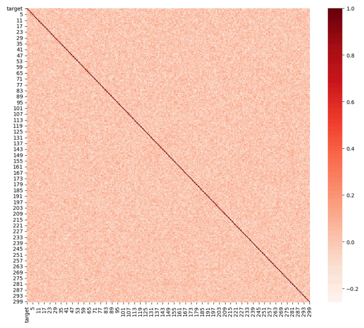


Figure 5: Correlation between Features and Target

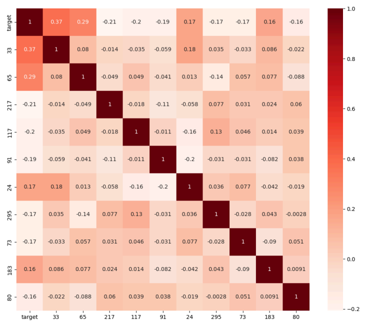


Figure 6: Correlation between Top 10 Features and Target

Feature Correlation Analysis on `target`

By further observing heatmaps of correlations between features and target in Figure 5 and Figure 6, it is found that:

- There is no strong correlation among features, which means there is no strong relation between any feature.
- There is no obvious correlation between features and the target, which means the importance of each feature is not so clear.

In summary, by EDA, the **High Entropy** (imbalance) is found in (1) the number of attributes and samples instants in training dataset, (2) the number of the samples between training and testing dataset, and (3) the number of the samples with target value `0` and `1`. In contrast, the **Low Entropy** is found in (1) the distribution among samples and features, and (2) the correlation among features as well as between features and target.

2. Basic Feature Engineering

In this section, the basic feature engineering is introduced, where we are tring to overcome the overfitting by manipulate the training dataset.

2.1 Data Cleaning and Preparation

The training dataset is relative clean, with no missing data and no duplicate data. The column `id` is removed from the training dataset.

2.3 Normalization

Feature normalization is performed through scaling by using `StandardScaler` [2] function from scikit-learn python library. Normalization is important for the following chosen base models, since it can enable feature value to be scaled, and fall within a small and controllable range, which mitigates the impact from extreme values by transforming the feature value into a standard normal distribution. The transformation is shown as follows:

$$z = \text{StandardScaler}(x) = \frac{x - \mu}{\sigma}$$

where x is the original feature, z is the normalized feature, μ is the mean of the feature, and σ is the standard deviation of the feature.

3. Basic Model Training

3.1 Evaluation Metrics and Training Objectives

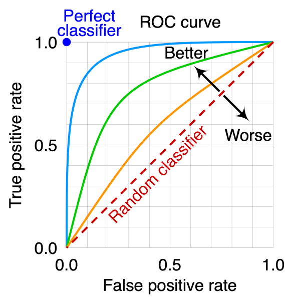


Figure 7: The ROC space for a "better" and "worse" classifier.[3]

According to the competition's overview[1:1], the submission is evaluated by the AUCROC (Area Under the Curve of Receiver Operating Characteristic) metrics between predicted value and the ground truth label. According to Wikipedia[3], ROC curve is the plot with True Positive Rate (TPR) vs. False Positive Rate (FPR) at **each prediction result as the threshold**, and each of them represents one point on the ROC curve. The AUCROC denotes the area under the ROC curve, and the larger the AUCROC, the better the classifier. The best performance is achieved when the AUCROC is 1, where the TPR is 1 and the FPR is 0, as shown in Figure 7.
Therefore, it is too strict to evaluate the model only results in 0 or 1 with AUCROC, since the thresholds are only 0 and 1, which may lead to a low AUCROC. Instead, it is more kind to evaluate the model results in continuous probability value with AUCROC, since the thresholds are various, which may give the model a more flexible evaluation.
In the following implementation, we use `predict_proba()` function to get the prediction result in continuous probability value, instead of `predict()` function to get the prediction result in 0 or 1 only.

3.2 Base Models

In this section, various base models are introduced for the competition together with different training and evaluation strategies, the performance of the models is evaluated, which is treated as the **baseline** for comparison with the further improvement methods. The purpose of choosing various base models is **(1) finding more potential models with high performance on testing dataset**, and **(2) build a baseline for evaluating and analyzing the further improvement methods on various models**. In total, 9 base models are chosen[4], including:

- **Logistic Regression** (LR): A linear model for binary classification tasks, predicting the probability that a sample belongs to a particular class.
- **Lasso Regression** (Lasso): A special case of linear regression, using L1 regularization, which may lead to a more robust model in "anti-overfitting" task.
- **Support Vector Machine Classifier** (SVC): A powerful classifier, using hyperplane to classify samples in a high-dimensional space.
- **k-Nearest Neighbors Classifier** (KNC): An instance-based classifier, storing training examples and delay the processing ("lazy evaluation") until a new instance must be classified, where instances are represented as points in a Euclidean space.
- **Decision Tree Classifier** (DT): A tree structure classifier, internal node for an attribute, edge represents a selection of attribute, leaf nodes for labels or distribution.
- **Random Forest Classifier** (RFC): An ensemble of decision tree classifiers, using bagging on samples and features, and prediction by voting or averaging.
- **Gaussian Naive Bayes Classifier** (GNB): A probabilistic classifier based on Bayes' theorem and the "Naive" assumption that features are independent.
- **AdaBoost Classifier** (ABC): Using boosting algorithm, turn weak classifiers to strong classifier by weighting training examples based on the classification error.
- **Linear classifiers with SGD training** (SGD): Optimizing a linear classifier by using Stochastic Gradient Descent algorithm.

3.3 Cross-Validation

In order to comprehensively evaluate the performance of our models and reduce the variance of the test prediction results, we used K-Folds Cross-Validation by using the `StratifiedKFold` function. By utilizing 20-fold cross-validation strategy, the dataset is randomly divided into 20 equal parts (folds). In each training iteration, 19 folds are used for training the model, and the remaining one fold is used for validation, and each fold used exactly once for validation during the whole process. The validation is performed by the AUCROC score as same as the Competition evaluation metrics. An ablation study is conducted in Section 5.

3.4 Bagging

Benefit from the K-Folds Cross-Validation, the training dataset has been divided into multiple subsets, and the model is trained on different subsets, which is a kind of bagging technique. The final prediction is the average of the predictions from all folds, as a optimal and reliable estimation of the model's performance on validation set. Bagging is an efficient way to reduce the variance of the model, and prevent the model from overfitting to the training dataset, which may lead to a better generalization capability on the unseen testing dataset. An ablation study is conducted in Section 5.

3.5 Regularization

Regularization is a discourages complex models, typically through constrained optimization, which is able to prevent overfitting by penalizing the model by cost function. In this competition, two types of regularization are introduced:

- **L1 Regularization**: Using the sum of absolute weights to penalize small values more;
- **L2 Regularization**: Using the sum of squared weights to penalize large values more, to discourage large weights.

The regularization is also based on the regularization parameter λ, which will be tuned by using GridSearchCV in the next section. An ablation study is conducted in Section 5.

3.6 GridSearchCV

Each model requires parameters to be set before training, which highly affects the model's performance. In order to identify the best parameters for models, we employed `GridSearchCV` function, a powerful tool that automates the process of tuning parameters by cross-validating the model with different parameter combinations. For example, **the optimal number of clusters for the kNN model**, **the optimal splits in samples and features for the Random Forest model**, and **the optimal regularization parameter for the regularization models**. The AUCROC score is used as the evaluation metric as well. After the best parameters are found for a model, the model is trained and evaluated with the best parameters, and produce the prediction results on the testing dataset with the best parameters. We applied `GridSearchCV` on each model by giving a set of parameters to be tuned, and the found best parameters are using for resulting the final prediction.

3.7 Baseline Model Performance

After applying the above strategies, the prediction results of the 9 base models are produced, and submitted to the Kaggle's competition platform. The Leaderboard (LB) score of the baseline models is shown in the Figure 8. We can find that the best model are Lasso and LR with the LB score of 0.849.

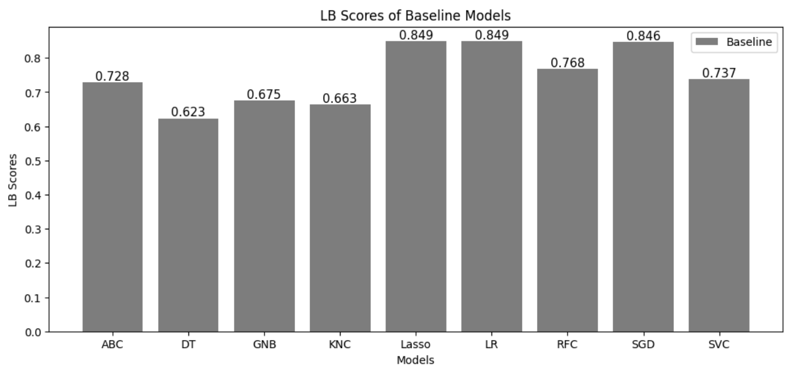


Figure 8: LB Scores of Baseline Models

4. Improvement Methods and Analysis

Based on the observation in the EDA, the distribution of the feature of the training dataset is quite normal distributed, and it is not easy to spot any pattern or relation between the features and the target. Therefore, I believe that the improvement of the model performance is not only based on the model selection and training strategy, but also based on the further feature engineering. Specifically, **Balancing Class Samples**, **Feature Creation** and **Feature Selection** are introduced in this section, followed by the analysis of the improvement methods.

4.1 Balancing Class Samples

An imbalance of samples in two classes can be found in Figure 1, we try to use SMOTE to mitigate this problem.
SMOTE: For Synthetic Minority Oversampling Technique[5], it is an oversampling technique to generate samples for the minority class. This method is to balance the number of samples between the two classes, and try to prevent the model from overfitting to the majority class.

4.2 Feature Creation

Another possible way is to create interesting features and add them into original feature space. We have tried following 3 method:

- **ADD_MEAN**: For each sample, calculate the mean of the feature values, and add it as a new feature, resulting in $300 + 1 = 301$ features.
- **ADD_STD**: For each sample, calculate the standard deviation of the feature values, and add it as a new feature, resulting in $300 + 1 = 301$ features.

- **ADD_MEAN_STD**: For each sample, calculate the mean and standard deviation of the feature values, and add them as new features, resulting in $300 + 2 = 302$ features.

After apply above four Improvement Methods to 9 baseline model, the LB scores are shown in the Figure 9. We can find that their improvements are **not so significant** compared with **Baseline**, but **ADD_MEAN_STD** method has a relatively larger improvement, while the SMOTE introduces more negative effect on the performance.

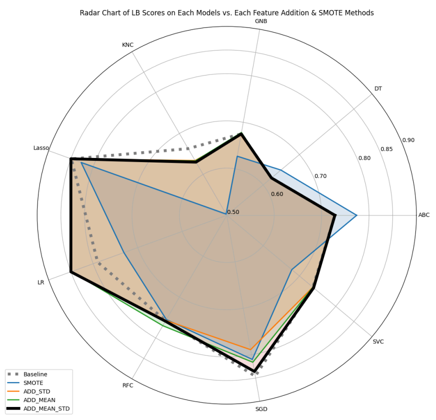
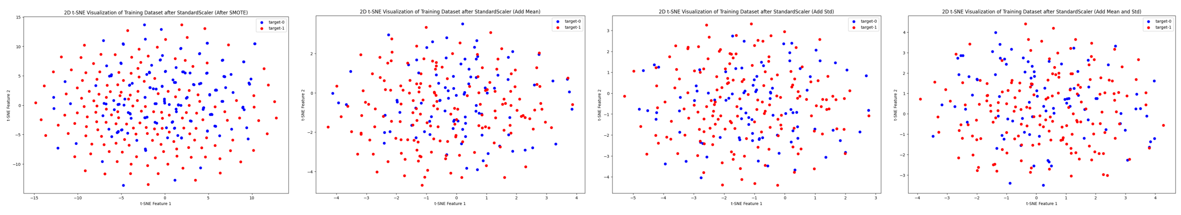


Figure 9: Radar Chart of LB Scores on 9 Models with 3 Feature Creation Methods and SMOTE

Discussion: We can analyse the these four method by observing the t-SNE visualization of the training dataset after applying the methods. Four methods do not introduce obvious patterns or clusters in the distribution of the samples features, which may not help the model to overcome the overfitting problem, although we can spot some small patterns after applying **ADD_MEAN_STD** as shown in the Figure 10. We believe that the created features enlarge the feature space even more, which may aggravate the overfitting problem. SMOTE method may introduce more noise samples, which increase the variance of the model in prediction.



SMOTE ADD_MEAN ADD_STD ADD_MEAN_STD

Figure 10: Distribution of Training Dataset

4.3 Feature Selection

Since the number of attributes (300) is greater than the number of the samples instants (250) in training dataset, it is possible to reduce the dimensionality of the feature space by selecting the most "important" features, which may help to prevent the model from overfitting to the training dataset. The importance is measured by the correlation between the feature and the target when training the model. We have tried following 3 method by utilizing several Python libraries:

- **K_BEST**: Use `SelectKBest` function from scikit-learn to select the top K important features [6].
- **RFE**: Use `RFE` (Recursive Feature Elimination) function from scikit-learn. It select features by recursively considering smaller and smaller sets of features until the specified number of features is achieved [7].
- **TOP9**: Use the `ELI5` library, which is able to to explain weights and predictions results of linear classifiers, and finally **top 9** features are selected (so we name it as **TOP9**). The iterative selection is shown in Figure 11, where selecting top 9 features gives hightest CV scores [8].

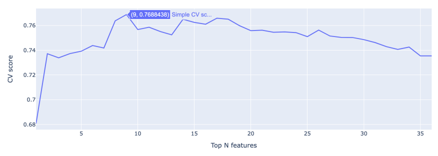


Figure 11: ELI5 Selected Top N Features vs. CV Scores

After apply above three Improvement Methods to 9 baseline model, the LB scores are shown in the Figure 12. We can find that **most of the feature selection methods have a positive effect on the overall performance**, and the **TOP9** method has the largest improvement on the LB score.

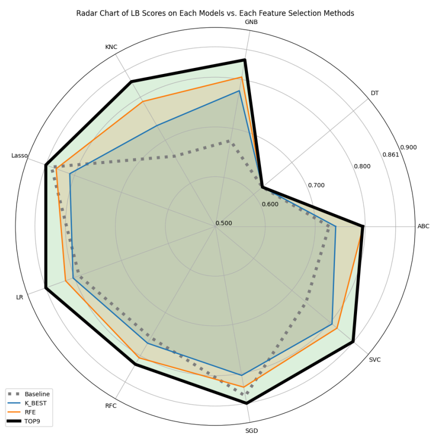
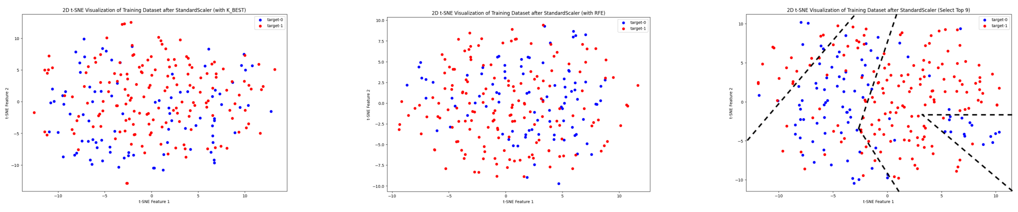


Figure 12: Radar Chart of LB Scores on 9 Models with 3 Feature Selection Methods

Discussion: We ara able to analyse the above three method by observing the t-SNE visualization of the training dataset after applying the methods. In this case, we are able to spot some **more clear patterns, boundaries, or even clusters** in the distribution of the samples features, which may help the model to overcome the overfitting problem. Especially, the **TOP9** method has the most obvious effect on the distribution of the samples features, some potential decision boundaries are annotated in the Figure 13. Therefore, we believe the feature selection method is more effective on the "anti-overfitting", since it is able to reduce the dimensionality of the feature space, as well as reduce the model's complexity for better generalization.



K_BEST RFE TOP9

Figure 13: Distribution of Training Dataset

4.4 Feature Fusion

So far, we have tested some improvement methods, where **TOP9** and **ADD_MEAN_STD** achieve the best improvement on the LB score. It is also possible to try to fuse the features named **TOP9_MEAN_STD**: Combine the top 9 features selected by the **TOP9** method, as well as the mean and standard deviation of the feature values, resulting in $9 + 2 = 11$ features.
After apply this method to 9 baseline model, the LB scores are shown in the Figure 14. The overall performance **is not further improved** compared to the **TOP9** method. However, the **LR** and **Lasso** models achieve the **Best LB score** of 0.867 after applying the **TOP9_MEAN_STD** method.

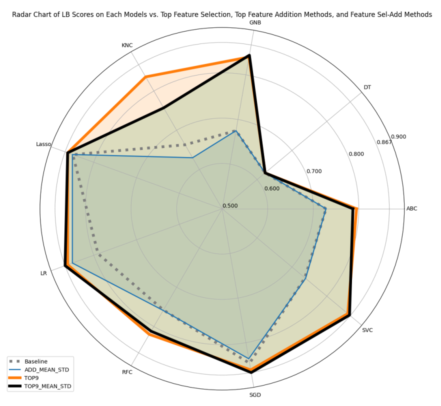
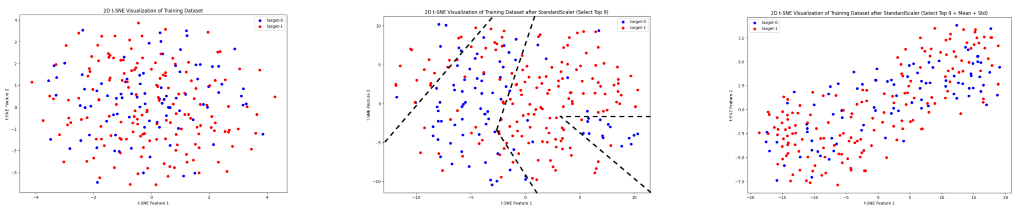


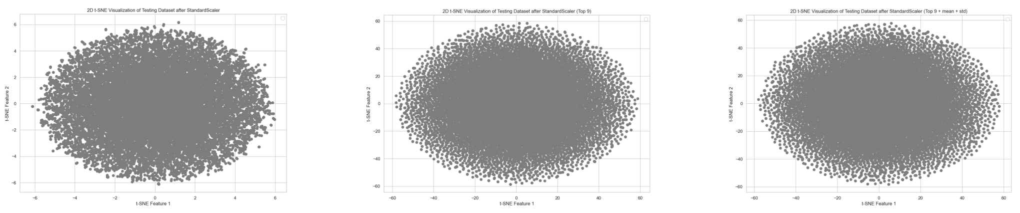
Figure 14: Radar Chart of LB Scores on 9 Models with ADD_MEAN_STD and TOP9

Discussion: After observing the t-SNE visualization of the training dataset, it is interesting to find that the shape of the distribution of the **TOP9_MEAN_STD** samples features is changed from a "disc" to a "stick", but the distribution pattern and potential decision boundaries become more **unclear** compared to the **TOP9** method, as shown in the Figure 15, where the best LB score achieved may be considered as a special edge case.



Baseline (Base F.E.) TOP9 TOP9_MEAN_STD

Figure 15: Distribution of Training Dataset



Baseline (Base F.E.) TOP9 TOP9_MEAN_STD

Figure 16: Distribution of Testing Dataset

Discussion: By further observing the distribution of the **Testing Dataset** in Figure 16, it is difficult to spot the potential decision boundaries due to the large number of samples, however, we can tell that, after applying **TOP9** and **TOP9_MEAN_STD** method, the testing dataset becomes more **clear** (less outliers compared to the Baseline), and the distribution is more **sparse** (notice the axis scale in Figure 16). And we believe models may produce a better decision boundary on the unseen testing dataset.

4.5 Top Improvement Method and Further Analysis

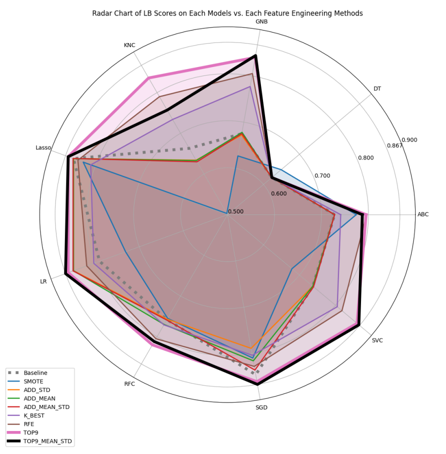


Figure 17: Radar Chart of LB Scores on 9 Models and 8 Feature Engineering Methods

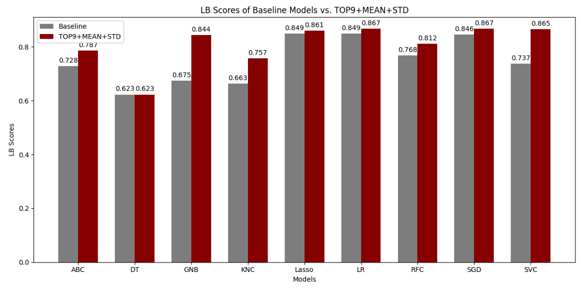


Figure 18: LB Scores of Baseline vs. Best Improvement Method

In total, we have experimented 8 improvement methods on 9 base models, as shown in the Figure 17. The KNC model has the largest overall improvement, and the DT base model has the smallest overall improvement. Most of Feature Creation methods have less or negative effect on the performance, and most of Feature Selection methods have a positive effect, where **TOP9** method contributes the most, as shown in the Figure 17. While **LR** and **Lasso** model with **TOP9_MEAN_STD** improvement method achieves the best LB score of 0.867 as shown in the Figure 18.

4.5.1 Exploratory Analysis with Association Rule Mining

The top 9 features selected by the **TOP9** method are `33, 65, 217, 91, 73, 199, 43, 117, 16`, and we are trying to find the possible reasons behind the best improvement we achieved, and the transformation of the distribution in Training and Testing Dataset as shown in Figure 15 and Figure 16. We try to use the Association Rule Mining[9], a technique from Data Mining domain, to find the possible relations between features and the target. Each of the continuous feature is discretized into 3 bins, and the `target` is transformed in a binary string format, such `target-0` and `target-1`. After applying the Association Rule Mining, we are able to find some important rules, as shown in the Figure 19.

Discussion: Some of the top 9 features, such as `117` and `65`, have a high association with the `target-1` with a notable support and confidence, which may be the reason behind the best improvement of the **TOP9** method, as well as other Feature Selection methods.

Rules	Support	Confidence
117(-1.698, 0.593] ⇒ target-1	0.460	0.701
65(-0.83, 1.193], 103(-1.17, 1.119] ⇒ target-1	0.380	0.725
217(-0.9, 0.882], 126(-1.444, 1.003] ⇒ target-1	0.352	0.727
43(-1.103, 0.709], 266(-1.144, 0.727] ⇒ target-1	0.308	0.720
33(-0.912, 0.917], 157(-0.913, 1.018] ⇒ target-1	0.288	0.706
73(-1.177, 0.464], 219(-0.552, 1.483] ⇒ target-1	0.264	0.733
91(-1.041, 0.76], 147(-1.067, 0.687] ⇒ target-1	0.264	0.702
199(0.349, 2.183] ⇒ target-1	0.264	0.717
16(-0.559, 1.317], 117(-1.698, 0.593] ⇒ target-1	0.256	0.711

Figure 19: Association Rule Mining Result on Training Dataset

5. Ablation Study

In addition, we have performed a simple ablation study on the **LR** model with the **ADD_MEAN_STD**, **TOP9** and **TOP9_MEAN_STD** improvement method. Specifically, we ran the ablations on 2, 10, 20 folds in the cross-validation (using bagging to get the prediction results), $L1$, $L2$ type of regularization, and the $C = 0.01, 0.1, 1$ regularization parameter (the larger the C, the smaller the regularization parameter λ [10]). The LB scores of each combination are shown in the Figure 20.

Discussion: We can tell that different improvement methods requires different parameter setting, which need to be tuned for the optimal performance, for example, the **ADD_MEAN_STD** prefers the $L1$ with $C = 0.1$ regularization, while the **TOP9_MEAN_STD** prefers the $L2$ with $C = 1$ regularization. However, our proposed **TOP9** and **TOP9_MEAN_STD** method are consistently giving the overall improvement almost regardless of the parameter settings.

	2 Fold CV			10 Fold CV			20 Fold CV		
		L1	L2		L1	L2		L1	L2
ADD_MEAN_STD	C = 0.01	0.500	0.744	C = 0.01	0.500	0.749	C = 0.01	0.500	0.749
	C = 0.1	**0.802**	0.735	C = 0.1	**0.850**	0.744	C = 0.1	**0.850**	0.743
	C = 1	0.791	0.728	C = 1	0.825	0.740	C = 1	0.825	0.737

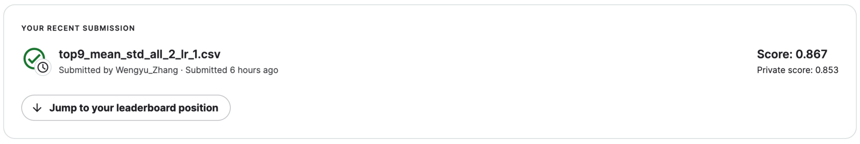
	2 Fold CV			10 Fold CV			20 Fold CV		
		L1	L2		L1	L2		L1	L2
	C = 0.01	0.500	0.797	C = 0.01	0.500	0.799	C = 0.01	0.500	0.800
TOP9	C = 0.1	0.812	0.813	C = 0.1	**0.848**	0.816	C = 0.1	**0.849**	0.817
	C = 1	**0.832**	0.819	C = 1	0.830	0.825	C = 1	0.831	0.826

	2 Fold CV			10 Fold CV			20 Fold CV		
		L1	L2		L1	L2		L1	L2
TOP9_MEAN_STD	C = 0.01	0.500	0.861	C = 0.01	0.500	0.861	C = 0.01	0.500	0.861
	C = 0.1	0.816	0.861	C = 0.1	0.853	0.862	C = 0.1	0.855	0.862
	C = 1	0.861	**0.862**	C = 1	0.862	**0.863**	C = 1	0.862	**0.867**

Figure 20: Ablation Study on Logistic Regression Model

7. Conclusion

Team name: **Wengyu_Zhang**


YOUR RECENT SUBMISSION

top9_mean_std_all_2_lr_1.csv
Submitted by Wengyu_Zhang · Submitted 6 hours ago

Score: 0.867
Private score: 0.853

Jump to your leaderboard position

Figure 21: Final LB Score

In this competition, we have performed the EDA, basic feature engineering, basic model training with some strategies, as well as proposed several improvement methods, and performed comprehensive experiments on each method with each model, followed by the analysis with t-SNE and Association Rule Mining, and a simple ablation study. We have found that the Feature Selection method contributes the most to the "anti-overfitting" task, where the **TOP9_MEAN_STD** method achieves the best LB score of 0.867 with **LR** and **Lasso** base models. Although the improvement is not so large, it is still a good practice to address the overfitting problem. The keys to the success are not only the model selection and training and optimization strategy, but also the data analysis and feature engineering.

References

1. "Don’t Overfit! II," *kaggle.com*. https://www.kaggle.com/c/dont-overfit-ii/. ↩ ↩
2. Scikit-Learn, "sklearn.preprocessing.StandardScaler — scikit-learn 0.21.2 documentation," *Scikit-learn.org*, 2019. https://scikit-learn.org/stable/modules/generated/sklearn.preprocessing.StandardScaler.html ↩
3. Wikipedia Contributors, "Receiver operating characteristic," *Wikipedia*, Mar. 20, 2019. https://en.wikipedia.org/wiki/Receiver_operating_characteristic ↩
4. Artgor, "How to not overfit?," *Kaggle*, Apr. 24, 2019. https://www.kaggle.com/code/artgor/how-to-not-overfit/ ↩
5. S. Satpathy, "SMOTE for Imbalanced Classification with Python," *Analytics Vidhya*, Nov. 17, 2023. https://www.analyticsvidhya.com/blog/2020/10/overcoming-class-imbalance-using-smote-techniques/ ↩
6. "sklearn.feature_selection.SelectKBest," *Scikit-learn*. https://scikit-learn.org/stable/modules/generated/sklearn.feature_selection.SelectKBest.html ↩
7. "Sklearn.feature_selection.RFE," *Scikit-learn*. https://scikit-learn.org/stable/modules/generated/sklearn.feature_selection.RFE.html ↩
8. "Overview — ELI5 0.11.0 documentation." https://eli5.readthedocs.io/en/latest/overview.html ↩
9. "Association Rules with Python," *kaggle.com*. https://www.kaggle.com/code/mervetorkan/association-rules-with-python ↩
10. "sklearn.linear_model.LogisticRegression," *Scikit-learn*. https://scikit-learn.org/stable/modules/generated/sklearn.linear_model.LogisticRegression.html ↩