Model link: https://drive.google.com/drive/folders/1CR-N9encIodLJ7SmQiexBP9y4-LZohhr?usp=sharing
Github link: https://github.com/n9maple/final_project

1.Introduction

This project is aim to predict the failure rate of the main product of a fictional company with provided experimental statistic from a lab. Participants can join the participation on “Tabular Playground Series - Aug 2022” website, and the performance of the prediction is evaluated by area under the ROC curve (AUC). Therefore, this question can be treated as a regression problem rather than classification one.

This project is mainly inspired by a post on Kaggle named” TPS-Aug22 9th solution” (the link will be provided in reference section). This project using a simple logistic regression with data preprocessing and feature selection, which will be described in detail below.

2. Methodology

2.1. Data preprocessing

First, we can observe that there are some missing values in the training data. To make the data integral, the data which don’t have missing values are selected to implement KNNImputer algorithm, and the missing values can be calculated by the neighboring data points (In my model, 5 nearest neighboring data points are selected).

Second, I add two extra columns named ‘m3_missing’ and ‘m5_missing’ to record if 'measurement_3’ and 'measurement_5' values miss. If it misses, ‘missing’ column is set to 1, otherwise 0(mainly referenced from the post on Kaggle).

Third, the whole data are standardized with average=0 and variance=1.

Finally, only a few features are selected to fit the model based on their importance and overall performance of the model.

2.2. Feature importance

Not all the data should be chosen to train the model. Instead, only the important ones are selected. The importance is computed by permutation importance algorithm with AUC as evaluation method, and the result is as below:

```
measurement_16 : 0.0008808707302416786
m3_missing : 0.0009683591753644193
m5_missing : 0.0020612930993952894
measurement_10 : 0.002335083471667332
measurement_3 : 0.0024931250438329227
measurement_5 : 0.002788799276186804
measurement_15 : 0.0032243850060331702
measurement_13 : 0.0036128174564338264
measurement_14 : 0.005133944319852591
measurement_12 : 0.0060279426298508465
measurement_11 : 0.011284289257020852
measurement_0 : 0.011325576161837825
measurement_8 : 0.012112944838045703
measurement_9 : 0.01251641315951435
measurement_1 : 0.017915110759564312
measurement_7 : 0.030424975331840853
measurement_6 : 0.03217477682821057
measurement_4 : 0.04852614032563518
measurement_2 : 0.05890779547069913
measurement_17 : 0.08996162435631416
loading : 3.919965380459794
```

The number has been scaled, and the feature with larger number means it's more important.

After testing the model and reference to the referenced post, I found that the model can achieve baseline if we only use loading, measurement_2, and measurement_17 as selected feature. However, it performs pretty bad when measurement_4 is additional selected. The similar result can be observed if measurement_4 is replaced with measurement_6. But the model performs quite well if measurement_4 is replaced with measurement_1.

On the other hand, if I add m3_missing and m5_missing into the model, the model can get better performance even though these two seems unimportant based on permutation importance algorithm.

feature	private	public
loading, m2, m17	0.58992	0.5842
loading, m1, m2, m17	0.58997	0.5862
loading, m4, m2, m17	0.58965	0.58248
loading, m6, m2, m17	0.58855	0.58377
loading, m2, m17, miss3, miss5	0.58997	0.58423
loading, m1, m2, m17, miss3, miss5	0.59004	58625

(Red color means the score achieve baseline)

2.3. Model architecture

In this project, a simple logistic regression model is applied with feature selection. The hyperparameter is as following:

```
model = LogisticRegression(max_iter=1000, C=0.01, penalty='l2',
                           solver='newton-cg')
```

I have tried three kinds of C, and 0.01 is the best choice I find. As for max iteration, it just needs to be large enough to ensure that the model achieve the best point in the end, and 1000 is large enough in this model.

C (feature=[loading, m1, m2, m17, miss3, miss5])	private	public
1	0.59006	0.58645
0.01	0.59041	0.58636
0.0001	0.59004	58625

3. Something interesting

In general, this part should be the summary, but I found something interesting when adjusting the model. I accidentally selected the wrong feature columns when giving the test data for prediction. The original model should accept the data with features as [loading, m1, m2, m17], but I mistakenly fed [loading, m6, m2, m17] as the features of test data. Surprisingly, the private score is very high while the public one is pretty low. After this discovery, I applied the enhancement method mentioned above on this base, including adding miss3 and miss5 as feature, and setting the C (hyperparameter of the model) as 0.01. Finally, the private score is very close to the top 1 score in the original competition. More discussion will be made in the summary section below.

	private	public
loading, m1, m2, m17 (mistakenly replace m1 with m6), C=0.0001	0.59058	0.58402
add miss3, miss5	0.59065	0.58404
C=1	0.59078	0.58435
C=0.01	0.59109	0.58418

4. Summary

In this project, it shows that merely a simple logistic regression model with data preprocessing and feature selection can already achieve pretty great performance. On the other hands, it can be observed that the prediction of the model is almost between 0.1 and 0.3, which perhaps due to imbalance in the training data (the number of 0 in failure

is much more than 1). Therefore, the model tends to give the number closer to 0 (the evaluation method, namely AUC, may be one of the reasons as well).

Another issue is that part3 shows a strange result –a totally wrong data can give a fabulous prediction to this competition on the private score even though its public score is really bad. Here are several possible explanations. First, perhaps measurement_6 and measurement_1 are highly related, so the prediction can luckily get a pretty high score. Second, the selection of private data and public data may not be well chosen, so the difference between public score and private score may be quite large. Finally, Maybe the top 100 scores (or even top 500) are actually very close, so every model performs similarly well but the luck determine the final rank. We can also see the leaderboard on Kaggle that many participants who are the top on public score finally become out of the top on private score. Therefore, I think the final possible reason is convinced. Also, based on the result I get from part3, I also think the second one is convinced.

In sum, this project inspires me to solve the real-world problem, and I find many questions and solutions during this project (and something interesting perhaps). With these take-a-way, I have more confidence to solve further problems in the future.

5. Result

The highest public score comes from the logistic regression model with feature selection as [loading, measurement_1, measurement_2, measurement_17, m3_missing, m5_missing] and hyperparameter C=0.01, max_iter=1000, solver='newton-cg', penalty='l2'. The private score is 0.59041.

6. Reference

The idea of this project is mainly inspired from the post on Kaggle named “TPS-Aug22 9th solution”, including the value missing dealing, feature selection, and logistic regression model. Also, I reference KNNImputer method from the article “KNNImputer：一种可靠的缺失值插补方法” and reference permutation importance algorithm from the article “（機器學習）可解釋性(2) Permutation Importance”. Without the help from these excellent articles. The project may not be finished.

[1] ZJY27, “TPS-Aug22 9th solution” September, 2022. [Online]. Available: https://www.kaggle.com/code/takanashihumbert/tps-aug22-9th-solution/notebook [Accessed Jan. 8, 2023].

[2] KAUSHIK as writer, VK as translator, “KNNImputer：一种可靠的缺失值插补方法” July 28, 2020. [Online].

Available: https://www.cnblogs.com/panchuangai/p/13390354.html [Accessed Jan. 8, 2023].

[3] Ben Hu,” （機器學習）可解釋性(2) Permutation Importance”

February 2,2020. [Online].
Available:
https://medium.com/@hupinwei/%E6%A9%9F%E5%99%A8%E5%AD%B8%E7%BF%92-%E5%8F%AF%E8%A7%A3%E9%87%8B%E6%80%A7-machine-learning-explainability-%E7%AC%AC%E4%BA%8C%E8%AC%9B-c090149f0772 [Accessed Jan. 8, 2023].