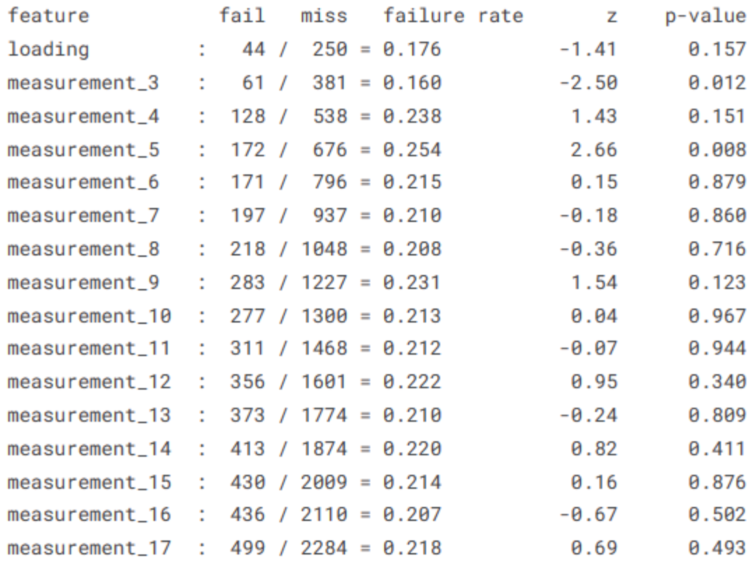
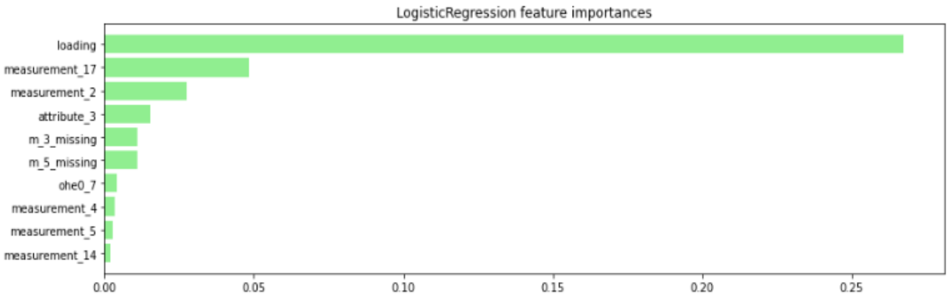
- GitHub link:
    - [2022_NYCU_ML/final_project at main · ycshan0415/2022_NYCU_ML_ (github.com)]
- Reference:
    - Hunting for missing values | Kaggle
    - TPSAUG22 EDA which makes sense ★ ★ ★ ★ ★ | Kaggle
- Brief introduction
    - In this project, I use logistic regression to as the model, and some special data pre-process to both pass the baseline and get a higher score.
- Methodology:
    - Data pre-process:
        - I use the steps of data pre-processing of the reference code. First, there are many missing values in the dataset, so we need to impute it to make our prediction more accurate. I use KNNimputer, which n_neighbors=3(based on other's advice online). Next, let's see this chart:



```
feature             fail   miss   failure rate          z    p-value
loading           :   44 /   250 = 0.176            -1.41      0.157
measurement_3     :   61 /   381 = 0.160            -2.50      0.012
measurement_4     :  128 /   538 = 0.238             1.43      0.151
measurement_5     :  172 /   676 = 0.254             2.66      0.008
measurement_6     :  171 /   796 = 0.215             0.15      0.879
measurement_7     :  197 /   937 = 0.210            -0.18      0.860
measurement_8     :  218 /  1048 = 0.208            -0.36      0.716
measurement_9     :  283 /  1227 = 0.231             1.54      0.123
measurement_10    :  277 /  1300 = 0.213             0.04      0.967
measurement_11    :  311 /  1468 = 0.212            -0.07      0.944
measurement_12    :  356 /  1601 = 0.222             0.95      0.340
measurement_13    :  373 /  1774 = 0.210            -0.24      0.809
measurement_14    :  413 /  1874 = 0.220             0.82      0.411
measurement_15    :  430 /  2009 = 0.214             0.16      0.876
measurement_16    :  436 /  2110 = 0.207            -0.67      0.502
measurement_17    :  499 /  2284 = 0.218             0.69      0.493
```

As we can see, when measurement_3 is missing, the failure rate is 0.160 (much lower than average; when measurement_5 is missing, the failure rate is 0.254 (much higher than average). As a result, I add "m3_missing" and "m5_missing" to the feature I select to train the model. Then, I check the correlation of different features with other feature and failure, and the model weights of different features:

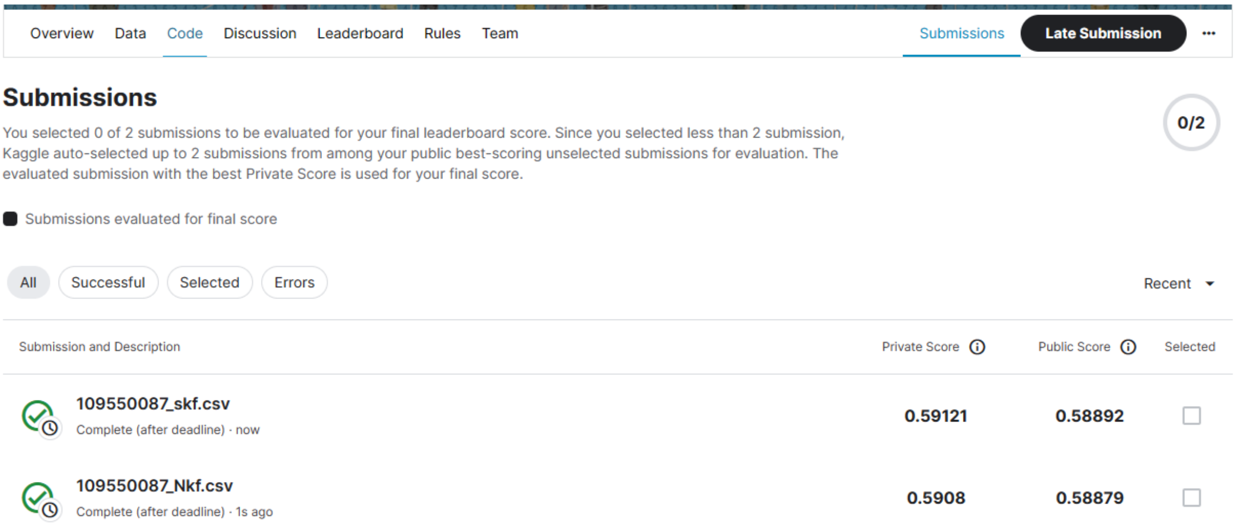


Thus I finished choosing features. Last, because the feature "attribute_0" is string, and the model I used doesn't allow a string as input, so I encoded that column. Last, I scaled the data. Finally, I finished pre-processing.

- Model architecture:
  - I use LogisticRegression in sklearn.linear_model, whose approach is relatively simple to implement than other model (such as NN). And I just fit the train data into the model, and finetune the hyperparameter of it. Here I use two ways to fit: the first is simply split the training data in to two set, x_train and x_valid, whose ration is 8:2; the second one, I use StratifiedKFold to split the training data, whose split will remain the ratio of original data's sample in different classes. Last, I choose the best model among all the models in the loop.
  - Nevertheless, if I predicted the result in every loop and average them at last, the performance will become better than simply choosing the best model which has the highest public score (0.59121 and 0.59112). I think this is because averaging the result can also average the bias of a single model. However, it is hard to save all the model state in every loop, or, if I did this, it would be similar to re-train a model in inference, so the most reasonable way is to save the best model.
- Hyperparameters:
  - Among all the Hyperparameters, I use n_neighbors=3 in KNNimputer, and epsilon=1.9 in HuberRegressor. As for LogisticRegression model, I have scrolled through many notebook and discussions to choose one.

- Summary:
  - In this final project, I think the most difficult part is the data pre-processing. Since the original data has many missing values, it is necessary for us to re-fill the data. Moreover, most features' correlation with the failure are very low, so we can imagine the model will have bad performance if we just use all the features to predict the result. In this part, I scrolled many notebooks

with EDA to help myself understand the data more. However, it was still very hard for me at the first place, and I have also try other approaches, but none of them had passed the baseline. Therefore, I refer to other people's approach, and found that many of them use logistic regression. After a week and a few days of hard-working, I got a satisfying score. In conclusion, I think the final project is a little bit too hard (in terms of data pre-processing), but fortunately, I got through this. Last, thanks for the professor and TAs for this semester.



Submissions

You selected 0 of 2 submissions to be evaluated for your final leaderboard score. Since you selected less than 2 submission, Kaggle auto-selected up to 2 submissions from among your public best-scoring unselected submissions for evaluation. The evaluated submission with the best Private Score is used for your final score.

■ Submissions evaluated for final score

Submission and Description	Private Score	Public Score	Selected
109550087_skf.csv Complete (after deadline) · now	0.59121	0.58892	□
109550087_Nkf.csv Complete (after deadline) · 1s ago	0.5908	0.58879	□