Machine Learning (2150534602) Term Project Report

20182691 Wonseok Do

Summary:

Introduction:

There were several issues with the Term Project submitted for the midterm assignment. In this Final Team project, we aim to address minor issues and explore a simpler approach than before. Ultimately, the quality of the problem will be determined by the evaluation panel. Some evaluators may prefer the right side, while others may prefer the left side. Although the evaluation criteria are somewhat disclosed, it is not easy to predict problems that are subject to individual judgment. Therefore, we determine factors that may influence the quality of the problem based on the available data and use them to predict the quality.

Here are the errors in the midterm assignment:

- The ranks were assigned using normalized values instead of the range from 1 to 948.
- The sorting was incorrectly done in descending order.
- There are too many instances of identical values.

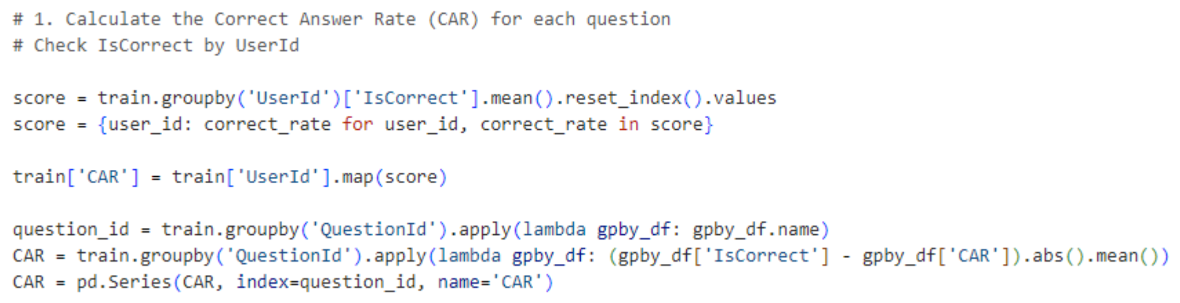
I will make the necessary corrections to the mentioned errors and attempt a different approach to improve the problem quality compared to the previous attempt.

Methods:

Firstly, I used three metrics to predict the quality, unlike the previous approach.

- Calculate the Correct Answer Rate (CAR).
- Clarity for each question.
- Entropy for each question.

Initially, I examined the "IsCorrect" based on the "UserID" to determine the answer accuracy rate for each problem. By analyzing the accuracy rate, we can infer the difficulty level of the problems to some extent. However, it is not necessarily true that problems with very high or very low difficulty are of high quality. I think that difficulty alone is not a sufficient indicator of problem quality.



```
# 1. Calculate the Correct Answer Rate (CAR) for each question
# Check IsCorrect by UserId

score = train.groupby('UserId')['IsCorrect'].mean().reset_index().values
score = {user_id: correct_rate for user_id, correct_rate in score}

train['CAR'] = train['UserId'].map(score)

question_id = train.groupby('QuestionId').apply(lambda gpby_df: gpby_df.name)
CAR = train.groupby('QuestionId').apply(lambda gpby_df: (gpby_df['IsCorrect'] - gpby_df['CAR']).abs().mean())
CAR = pd.Series(CAR, index=question_id, name='CAR')
```

The first step is to calculate the Correct Answer Rate for each user (UserId). Calculating the mean of column 'IsCorrect' based on user (UserId). The 'IsCorrect' column indicates whether each question is correct or not. After, saves each user's correct rate in the form of a dictionary. The key of the dictionary is the user (UserId) and the value is the percent correct.

The second step calculates the CAR for each question (QuestionId). Extracting the ID of the question for each group. This allows the question ID to be used as an index for grouped results. Each question group (correct or not - Calculate the average of the absolute values of CAR). It represents the difference between the percentage of correct answers to a given question and the actual number of correct answers.

Therefore, as a result of executing the code, the CAR values are returned as a series indexed by the question ID. This allows to measure the difference between the percentage of correct answers for each question and the actual number of correct answers, and calculate the average value.

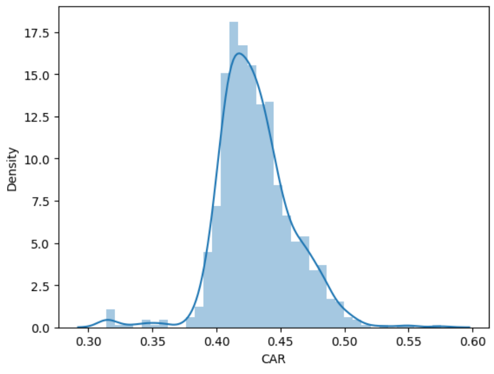
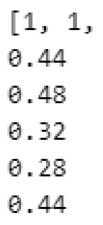


Figure 2. Distribution plot of CAR

Additionally, here is the data visualized using a Distplot, which shows the density. The result exhibits a normal distribution shape that is similar to a Gaussian distribution. However, there is a possibility that this may lead to inaccurate predictions.

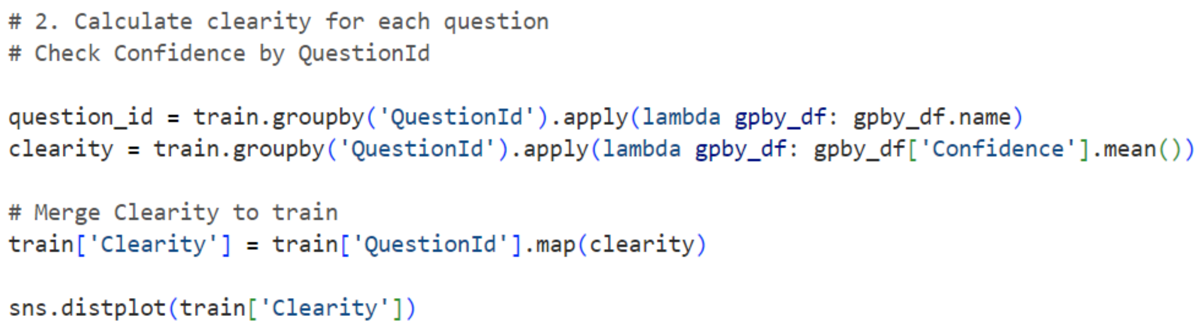
After conducting evaluation and testing using CAR data, the highest validation score achieved was 0.52, while the test score was 0.48. Essentially, it demonstrates a prediction performance of around 50%. This indicates that CAR metrics may not have a significant impact on quality prediction alone, and it might be necessary to consider using other metrics in conjunction with CAR.



```
[1, 1,
0.44
0.48
0.32
0.28
0.44
```

Figure 1. CAR Accuracy

Second, I pulled the Confidence number from the Metadata and used it. The hypothesis is When students solve a problem, if they are confident that they have solved it, it is likely that the problem is clear and easy to understand.



```
# 2. Calculate clearity for each question
# Check Confidence by QuestionId

question_id = train.groupby('QuestionId').apply(lambda gpby_df: gpby_df.name)
clearity = train.groupby('QuestionId').apply(lambda gpby_df: gpby_df['Confidence'].mean())

# Merge Clearity to train
train['Clearity'] = train['QuestionId'].map(clearity)

sns.distplot(train['Clearity'])
```

I extracted the ID of the question for each group. This way, I could use the question ID as an index for the grouped results. Then, I calculate the average of the 'Confidence' column in each group of questions. The 'Confidence' column is the column that represents the confidence level for each question. Therefore, this average value represents the clarity of that question.

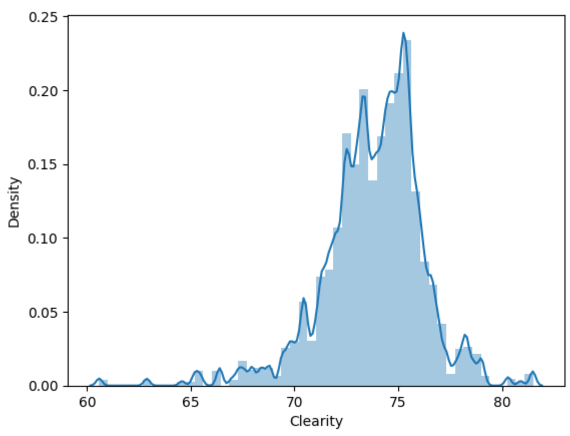


Figure 3. Distribution plot of Clarity

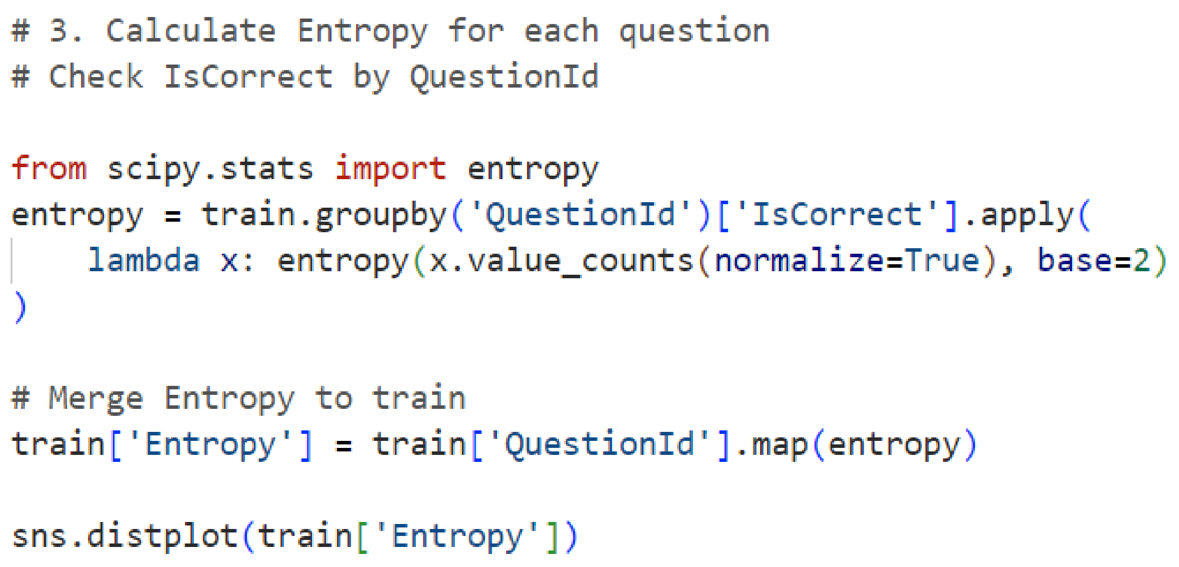
I ran validation and test beforehand as in the previous method. I noticed a different result from the previous CAR metric: predictive performance increased up to 0.76 in validation and up to 0.8 in test.

Unlike the previous metrics, this shows significant predictive power. I will first establish that this metric is excellent at predicting question quality before moving on.

- Validation: 0.72, 0.68, 0.76, 0.64, 0.64
- Test: 0.72, 0.6, 0.6, 0.6, 0.8

https://github.com/donasensei/QQA

Finally, I tried calculating the entropy of the question. It calculates the entropy for the values in the 'IsCorrect' column from the data grouped by 'QuestionId'. It's similar to the CAR I calculated before, but it only considers IsCorrect regardless of 'UserId'.



```python
# 3. Calculate Entropy for each question
# Check IsCorrect by QuestionId

from scipy.stats import entropy
entropy = train.groupby('QuestionId')['IsCorrect'].apply(
    lambda x: entropy(x.value_counts(normalize=True), base=2)
)

# Merge Entropy to train
train['Entropy'] = train['QuestionId'].map(entropy)

sns.distplot(train['Entropy'])
```

Calculates the entropy based on the calculated relative frequencies. base=2 indicates that the base of the entropy is 2. Entropy is used in information theory as an indicator of the degree of uncertainty. I used entropy to estimate the difficulty of a question. Questions with high entropy may have a wide range of possible answers, making it difficult to choose an answer, and therefore relatively more difficult. Conversely, questions with low entropy may be easier because the answers are limited or tend to be specific. Thus, entropy can be used to estimate the difficulty of a question.

The results showed some similarities and differences when compared to CAR. For CAR, both validation and test had average predictive performance, but for Entropy, validation performed significantly worse. To be precise, it was at least 0.16 lower.

However, both CAR and entropy performed around 0.5 in our tests. I do not know if this is because it's a binary problem of choosing left or right, so the performance scale is around 0.5, or if it's because it's more general.

Additionally, I created a new rate that incorporated all the rates and ran a test. The result was that it got closer to 0.5. I think this is because both metrics are giving results close to 0.5.

Discussion:

First, let's talk about CAR and Entropy. I mentioned earlier that I wanted to use the difficulty of the question as an indicator of the quality of the question. But there's a problem with that. It's not clear what constitutes good quality: a hard question is good quality, or an easy question is good quality?

Also, can a question be considered high quality if it is moderately challenging and not both? Difficulty is perceived differently by different people. This is also true for raters. And inside the metadata, there are categories for issues. For example, finding the solution to a cubic equation and finding the volume of two shapes. Can you compare two completely different categories of problems based on difficulty alone? I think it was weak as a hypothesis to see the aggregated difficulty in different ranges, but not in the same category.

For metadata, there were a lot of missing values. Filling them in may have caused the data to cluster in certain areas. Was this a good thing or a bad thing? I think that's something to consider and evaluate.

Additionally, I noticed that there was a missing value in the test data, which I didn't populate into the unmodifiable databases, but could this minor missing value have affected the accuracy?

Conclusion:

In the end, question quality assessment using Confidence performed the best. The confidence provided by the students who solved the questions was a better predictor of quality than the relative metrics. In preparing for this project, I was able to see what metrics the then top-ranked team on this topic used: they used question images and checked for readability. I thought it was an interesting approach and a metric to consider. If I hadn't seen it used by another team, I think I would have tried it. There's other metadata that I didn't use, but that's because I intentionally didn't use it and only pulled the data I thought I needed. However, this is just a personal judgment. I wonder what it would be like to have the data selected mechanically instead of personally.

https://github.com/donasensei/QQA