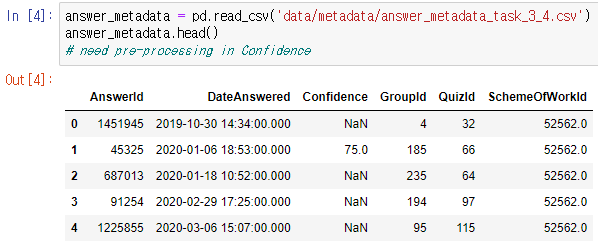
**1. Data exploration and preprocessing.**

When looking at the metadata, some values are missing in the ‘confidence; column, Therefore, preprocessing for this item is necessary.



**answer\_metadata['Confidence'].fillna(value=answer\_metadata['Confidence'].mean(),inplace=True)**: This line fills the missing values in the "Confidence" column with the mean value of that column. The **fillna** function is used to replace the missing values. The **value** parameter is set to **answer\_metadata['Confidence'].mean()** which calculates the mean of the "Confidence" column. The **inplace=True** parameter ensures that the changes are made directly to the DataFrame without creating a new copy.

텍스트, 스크린샷, 폰트, 번호이(가) 표시된 사진

자동 생성된 설명

**2. Calculate quality by merging and grouping the data**

**data = data.merge(answer\_metadata, on="AnswerId", how="left")**: This line merges the "data" DataFrame with the "answer\_metadata" DataFrame based on a common column named "AnswerId". The **merge()** function is used to perform the join operation. The **on** parameter specifies the column used for matching the data, which in this case is "AnswerId". The **how** parameter is set to "left", indicating that all the rows from the left DataFrame ("data") will be included in the merged DataFrame, while matching rows from the right DataFrame ("answer\_metadata") will be added if available. If there are no matching rows in "answer\_metadata" for a particular "AnswerId" in "data", the columns from "answer\_metadata" will have missing values in the merged DataFrame.

텍스트, 스크린샷, 소프트웨어, 번호이(가) 표시된 사진

자동 생성된 설명

**data['IsCorrect']**: This retrieves the values from the "IsCorrect" column of the "data" DataFrame. Assuming the "IsCorrect" column contains binary values (0 or 1) indicating whether an answer is correct or not.

**data['Confidence']**: This retrieves the values from the "Confidence" column of the "data" DataFrame. Assuming the "Confidence" column contains numerical values representing the confidence level of the answer.

**data['IsCorrect'] \* data['Confidence']**: This performs element-wise multiplication between the values in the "IsCorrect" and "Confidence" columns. Since "IsCorrect" contains binary values (0 or 1), multiplying it by the "Confidence" values effectively gives higher weight to correct answers (where "IsCorrect" is 1) and lower weight to incorrect answers (where "IsCorrect" is 0). The result of the multiplication will be stored in the new "Quality" column.

**data["Quality"] = ...**: This assigns the resulting values from the multiplication to a new column called "Quality" in the "data" DataFrame.

텍스트, 전자제품, 스크린샷, 번호이(가) 표시된 사진

자동 생성된 설명

**3. Sorting Quality**

**data = data.sort\_values("Quality", ascending=True)**: This line sorts the DataFrame "data" based on the values in the "Quality" column. The **sort\_values()** function is used to perform the sorting operation. The column to sort by is specified with the **by** parameter, which in this case is "Quality". The **ascending** parameter is set to **True**, indicating that the values should be sorted in ascending order.

**.reset\_index()**: This line resets the index of the DataFrame "data" after sorting. By default, the index values are reassigned to a new range starting from 0. The **reset\_index()** function is used to reset the index.

텍스트, 스크린샷, 번호, 폰트이(가) 표시된 사진

자동 생성된 설명

Reads a template CSV file into a DataFrame called "submission", then performs a loop over the rows of another DataFrame called "data". For each row in "data", it finds the index of the first occurrence where the "QuestionId" column matches the current loop index "i", and assigns this index value to the "ranking" column in the "submission" DataFrame. Finally, it converts the "ranking" column to integers, and saves the "submission" DataFrame to a new CSV file. Here is a breakdown of what each line does:

**submission = pd.read\_csv('data/submission/template.csv')**: This line reads a template CSV file named 'template.csv' from the 'data/submission/' directory into a DataFrame called "submission". The **pd.read\_csv()** function from the pandas library is used to read the CSV file.

**for i in range(len(data)):**: This line initiates a loop over the range of values from 0 to the length of the DataFrame "data".

**index = data[data['QuestionId'] == i].index**: This line filters the rows of the DataFrame "data" based on the condition where the value in the "QuestionId" column equals the current loop index "i". It then retrieves the index of the filtered rows and assigns it to the variable "index".

**submission.loc[i, 'ranking'] = index[0]**: This line assigns the value of the first element in the "index" variable (the index of the first occurrence where the condition is satisfied) to the "ranking" column in the "submission" DataFrame at the current loop index "i".

**submission['ranking'] = submission['ranking'].astype('int')**: This line converts the "ranking" column in the "submission" DataFrame to integer data type using the **astype()** function. This step ensures that the column contains integer values.

**submission.to\_csv('data/submission/20172656.csv')**: This line saves the "submission" DataFrame to a new CSV file named '20172656.csv' in the 'data/submission/' directory. The **to\_csv()** function from the pandas library is used to write the DataFrame to a CSV file.

I will attach the csv file separately.

Github: https://github.com/blueeye09/Machine\_Learning