EdNet - Optimize Student Performance: Data Wrangling

Project GitHub: https://github.com/premonish/EdNet (https://github.com/premonish/EdNet)

EdNet is composed of a total of 131,441,538 interactions collected from 784,309 students of Santa since 2017.

"Dataset: As we said, there are four datasets named KT1, KT2, KT3, and KT4 with different extents. Here's common features of these datasets:

The whole dataset is divided by students: {user_id}.csv only contains {user_id}'s interactions. The timestamps are different from the real values, which are modified (shifted by fixed values) due to security issues. Download links EdNet-KT1 : bit.ly/ednet_kt1 EdNet-KT2 : bit.ly/ednet-kt2 EdNet-KT3 : bit.ly/ednet-kt3 EdNet-KT4 : bit.ly/ednet-kt4 Contents : bit.ly/ednet-content KT1

The whole dataset is divided by students: {user_id}.csv only contains {user_id}'s interactions."

SOURCE: https://github.com/riiid/ednet.git (https://github.com/riiid/ednet.git)

In [4]:
```
# I get by with a little help from my friends
import matplotlib.pyplot as plt
import numpy as np
import pandas as pd
# import seaborn as sns
# import datetime
# import glob
```

In [5]:
```
cd ~/Desktop/SPRINGBOARD_MASTER/EdNet/data/raw
```

```
C:\Users\Prem\Desktop\SPRINGBOARD_MASTER\EdNet\data\raw
```

In [6]:
```
prefix = '~/Desktop/SPRINGBOARD_MASTER/EdNet/data/raw/EdNet-'
folder_1 = prefix + 'KT1/KT1/'
folder_2 = prefix + 'KT2/KT2/'
folder_3 = prefix + 'KT3/KT3/'
folder_4 = prefix + 'KT4/KT4/'
questions = prefix + 'Contents/contents/questions.csv'

filenames = ['u1.csv']

# create pathname
# create pandas dataframe
# extract user_id from filename and assign user_id as values to column 1
path = folder_1 + filenames[0]
one = pd.read_csv(path)
one.insert(0, 'user_id', filenames[0].split('.')[0])

one.head()
```

Out[6]:

	user_id	timestamp	solving_id	question_id	user_answer	elapsed_time
0	u1	1565096190868	1	q5012	b	38000
1	u1	1565096221062	2	q4706	c	24000
2	u1	1565096293432	3	q4366	b	68000
3	u1	1565096339668	4	q4829	a	42000
4	u1	1565096401774	5	q6528	b	59000

In [7]:
```
# 2
path = folder_2 + filenames[0]
two = pd.read_csv(path)
two.insert(0, 'user_id', filenames[0].split('.')[0])
```

In [8]:
```
# 3
path = folder_3 + filenames[0]
three = pd.read_csv(path)
three.insert(0, 'user_id', filenames[0].split('.')[0])
```

In [9]:
```
# 4
path = folder_4 + filenames[0]
four = pd.read_csv(path)
four.insert(0, 'user_id', filenames[0].split('.')[0])
```

In [10]:
```
# concatenate 4 dataframes into 1
df1 = pd.concat([one,two,three,four])
df1.head(3)
```

Out[10]:

	user_id	timestamp	solving_id	question_id	user_answer	elapsed_time	action_type	iten
0	u1	1565096190868	1.0	q5012	b	38000.0	NaN	
1	u1	1565096221062	2.0	q4706	c	24000.0	NaN	
2	u1	1565096293432	3.0	q4366	b	68000.0	NaN	

In [11]:
```
# questions
path = questions
questions_df = pd.read_csv(path)
questions_df.head(3)
```

Out[11]:

	question_id	bundle_id	explanation_id	correct_answer	part	tags	deployed_at
0	q1	b1	e1	b	1	1;2;179;181	1558093217098
1	q2	b2	e2	a	1	15;2;182	1558093219720
2	q3	b3	e3	b	1	14;2;179;183	1558093222784

In [12]:
```
# Column Names
df1.columns
```

Out[12]:
```
Index(['user_id', 'timestamp', 'solving_id', 'question_id', 'user_answer',
       'elapsed_time', 'action_type', 'item_id', 'source', 'platform',
       'cursor_time'],
      dtype='object')
```

Column Label Metadata

timestamp is the moment the question was given, represented as Unix timestamp in milliseconds.

solving_id represents each learning session of students corresponds to each bundle. It is a form of single integer, starting from 1.

question_id is the ID of the question that given to student, which is a form of q{integer}.

user_answer is the answer that the student submitted, recorded as a character between a and d inclusively.

elapsed_time is the time that the students spends on each question in milliseconds.

In [13]:
```
# Data Types
df1.dtypes
```

Out[13]:
```
user_id          object
timestamp         int64
solving_id      float64
question_id      object
user_answer      object
elapsed_time    float64
action_type      object
item_id          object
source           object
platform         object
cursor_time     float64
dtype: object
```

In [14]:
```
# Description of the Columns
df1.describe().T
```

Out[14]:

	count	mean	std	min	25%	50%	
timestamp	9467.0	1.568029e+12	1.622956e+09	1.565096e+12	1.566655e+12	1.568800e+12	1.
solving_id	1082.0	3.646201e+02	1.807275e+02	1.000000e+00	2.352500e+02	3.675000e+02	5.
elapsed_time	1082.0	4.957844e+04	5.839036e+04	6.660000e+02	2.425000e+04	3.675000e+04	5.
cursor_time	1220.0	1.846678e+04	4.188368e+04	0.000000e+00	0.000000e+00	1.186600e+04	1.

In [15]:
```
df1['user_id'].value_counts(normalize=True)
```

Out[15]:
```
u1    1.0
Name: user_id, dtype: float64
```

In [16]:
```
df1['timestamp'].value_counts(normalize=True)
```

Out[16]:
```
1567115277665    0.000634
1566219874725    0.000423
1568767003631    0.000317
1569392463396    0.000317
1569590584935    0.000317
                   ...
1569285315335    0.000106
1565361242900    0.000106
1565184506667    0.000106
1565310624544    0.000106
1566025932980    0.000106
Name: timestamp, Length: 5058, dtype: float64
```

In [17]:
```
df1['solving_id'].value_counts(normalize=True)
```

Out[17]:
```
219.0    0.007394
271.0    0.004621
367.0    0.004621
610.0    0.004621
270.0    0.004621
           ...
303.0    0.000924
451.0    0.000924
594.0    0.000924
599.0    0.000924
1.0      0.000924
Name: solving_id, Length: 671, dtype: float64
```

In [18]:
```
df1['question_id'].value_counts(normalize=True)
```

Out[18]:
```
q555      0.004621
q698      0.003697
q243      0.003697
q729      0.003697
q485      0.003697
            ...
q3559     0.000924
q10709    0.000924
q8573     0.000924
q2148     0.000924
q1766     0.000924
Name: question_id, Length: 926, dtype: float64
```

In [19]:
```
df1['user_answer'].value_counts(normalize=True)
```

Out[19]:
```
b    0.302399
a    0.259704
c    0.248765
d    0.189132
Name: user_answer, dtype: float64
```

In [20]:
```
df1['elapsed_time'].value_counts(normalize=True)
```

Out[20]:
```
26000.0     0.028651
19000.0     0.027726
23000.0     0.025878
20000.0     0.024954
21000.0     0.024030
              ...
69000.0     0.000924
181000.0    0.000924
107000.0    0.000924
324000.0    0.000924
188000.0    0.000924
Name: elapsed_time, Length: 206, dtype: float64
```

In [21]:
```
df1['action_type'].value_counts(normalize=True)
```

Out[21]:
```
enter          0.322719
respond        0.208945
submit         0.184615
quit           0.138104
pause_audio    0.070006
play_audio     0.067621
pause_video    0.003936
play_video     0.003936
pay            0.000119
Name: action_type, dtype: float64
```

In [22]:
```
df1['item_id'].value_counts(normalize=True)
```

Out[22]:
```
b668     0.004651
q6572    0.004293
b1149    0.004055
b1452    0.003339
b7       0.003339
           ...
q5279    0.000358
q6568    0.000358
q5978    0.000358
b56      0.000239
p157     0.000119
Name: item_id, Length: 1476, dtype: float64
```

In [23]:
```
df1['source'].value_counts(normalize=True)
```

Out[23]:
```
sprint            0.845897
tutor             0.041269
my_note           0.035305
adaptive_offer    0.030177
review_quiz       0.023616
archive           0.014074
diagnosis         0.007514
review            0.002147
Name: source, dtype: float64
```

In [24]:
```
df1['platform'].value_counts(normalize=True)
```

Out[24]:
```
mobile    1.0
Name: platform, dtype: float64
```

In [25]:
```
df1['cursor_time'].value_counts(normalize=True)
```

Out[25]:
```
0.0        0.409836
1045.0     0.040984
15768.0    0.004918
14738.0    0.003279
15257.0    0.003279
             ...
40908.0    0.000820
28979.0    0.000820
21812.0    0.000820
21813.0    0.000820
41982.0    0.000820
Name: cursor_time, Length: 612, dtype: float64
```

In [26]:
```
# range of values

print('timestamp: '); print(df1['timestamp'].min()); print(df1['timestamp'].ma
x()), print('---')

print('elapsed_time: '); print(df1['elapsed_time'].min()); print(df1['elapsed_
time'].max()); print('---')

print('cursor_time: '); print(df1['cursor_time'].min()); print(df1['cursor_tim
e'].max()); print('---')
```

```
timestamp:
1565096151269
1569647680211
---
elapsed_time:
666.0
1200000.0
---
cursor_time:
0.0
389816.0
---
```

In [27]:
```
df1.isna().all()
```

Out[27]:
```
user_id         False
timestamp       False
solving_id      False
question_id     False
user_answer     False
elapsed_time    False
action_type     False
item_id         False
source          False
platform        False
cursor_time     False
dtype: bool
```

Data Cleaning: Clean up the data in order to prepare it for the next steps of your project.

NA or missing values

Duplicates

In [28]:
```
"""from pandas_profiling import ProfileReport
prof = ProfileReport(df1)
prof.to_notebook_iframe()
prof.to_file(output_file='./df1.html')"""
```

Out[28]:
```
"from pandas_profiling import ProfileReport\nprof = ProfileReport(df1)\nprof.
to_notebook_iframe()\nprof.to_file(output_file='./df1.html')"
```

In [29]:
```
"""for ui, ur in df1.iterrows():
    for qi, qr in questions_df.iterrows():
        if (ur.user_answer == qr.correct_answer) and (ur.question_id == qr.que
stion_id):
            df1['correct'][ui] = 1
    break"""
```

Out[29]:
```
"for ui, ur in df1.iterrows():\n    for qi, qr in questions_df.iterrows():\n
if (ur.user_answer == qr.correct_answer) and (ur.question_id == qr.question_i
d):\n            df1['correct'][ui] = 1\n    break"
```