

```
In [1]: from datetime import datetime

import numpy as np
import pandas as pd
from IPython.display import display

class FeatureGenerator:
    """This class provides the feature generation functionality, including clearing, merging and feature creation."""

    def __init__(self, df_train_path, df_questions_path, df_lectures_path, skip_blank_lines=True, nrows=None):
        """Constructs the FeatureGenerator instance.

        :param df_train_path: string, url or path to the train dataframe
        :param df_questions_path: string, url or path to the questions dataframe
        :param df_lectures_path: string, url or path to the lectures dataframe
        :param skip_blank_lines: boolean, True is the empty lines is to be skipped
        :param nrows: int, number of rows to read, None be default (read all lines)
        """
        pd.options.display.max_columns = 100
        self.df_train = pd.read_csv(df_train_path, skip_blank_lines=skip_blank_lines, low_memory = True, nrows=nrows)
        self.df_lectures = pd.read_csv(df_lectures_path, skip_blank_lines=skip_blank_lines, low_memory = True, nrows=nrows)
        self.df_questions = pd.read_csv(df_questions_path, skip_blank_lines=skip_blank_lines, low_memory = True, nrows=nrows)
        self.df_merged = None

    def generate(self):
        """Filters, merges dataframes and analyses the features."""
        self.clear_data()
        self.merge_atomise_dataframes()
        self.analyze_create_features()

    def clear_data(self):
        """Filters dataframes.
        Drops useless columns, NaN containing lines when needed and empty string containing lines."""
        print(datetime.utcnow(), ": <-- Очистка данных -->")

        """По условию значения df_train prior_question_elapsed_time, user_answer, answered_correctly и
           prior_question_elapsed_time отсутствуют для лекций, поэтому очистим лишь прочие строки df_train"""
        self.df_train = self.df_train[self.df_train['row_id'].notna()]
                           & self.df_train['timestamp'].notna()
                           & self.df_train['user_id'].notna()
                           & self.df_train['content_id'].notna()
                           & self.df_train['content_type_id'].notna()
```

```

        & self.df_train['task_container_id'].notna()]

self.df_train = self.clear_data_auxiliary(self.df_train, "df_train", False)
self.df_questions = self.clear_data_auxiliary(self.df_questions, "df_questions")
self.df_lectures = self.clear_data_auxiliary(self.df_lectures, "df_lectures")

print("Выведем первые три строки и посмотрим, насколько полезны столбцы:")
print("-----")
display(self.df_train.head(3))
display(self.df_questions.head(3))
display(self.df_lectures.head(3))
print("-----")

"""удалим столбец type_of из df_lectures, краткое описание лекции,
поскольку качественная обработка данных значений
может потребовать глубокого использования инструментов NLP,
например лемматизации или стемматизации"""
self.df_lectures = self.df_lectures.drop('type_of', axis=1)

"""удалим столбец user_answer из df_train,
поскольку нам интересны лишь результаты ответа, а тип интерактивности
можно вывести df_train content_type_id"""
self.df_train = self.df_train.drop('user_answer', axis=1)

"""удалим столбец correct_answer из df_questions,
поскольку верный ответ или нет уже известно из df_train answered_correctly"""
self.df_questions = self.df_questions.drop('correct_answer', axis=1)

print(datetime.utcnow(), ": <-- Очистка данных завершена -->")

def merge_atomise_dataframes(self):
    """Merges dataframes to one for the further analysis."""
    print(datetime.utcnow(), ": <-- Объединение датафреймов в один -->")

    print("Выведем типы для train dataframe: ", self.df_train.dtypes)

    """Объединим датафреймы, добавив к train, преобразуем к атомарности значения столбцов."""
    self.df_merged = self.df_train.merge(self.df_questions, how="left", left_on='content_id',
                                         right_on="question_id")
    self.df_merged = self.df_merged.drop('question_id', axis=1)
    self.df_merged = self.df_merged.rename(columns={'tags': 'tag_question'})
    self.df_train = None
    self.df_questions = None

```

```

        self.df_merged = self.df_merged.merge(self.df_lectures, how="left", left_on='content_id',
                                              right_on="lecture_id", suffixes=("","_lecture"))
        self.df_merged = self.df_merged.drop('lecture_id', axis=1)
        self.df_merged = self.df_merged.rename(columns={'tag': 'tag_lecture'})

    self.df_lectures = None

    print(datetime.utcnow(),
          ": <-- Объединение датафреймов в один -->")

def analyze_create_features(self):
    """Analyses merged dataframe, suggest the new features."""
    print(datetime.utcnow(), ": <-- Анализ объединенного датафрейма и создание фичей -->")
    print("Выведем типы для объединенного dataframe: ", self.df_merged.dtypes)
    print("Выведем размерность для объединенного dataframe: ", self.df_merged.shape)
    print("Выведем описательную статистику для объединенного dataframe: ", self.df_merged.describe())

    print("Выведем первых трех пользователей по числу верных ответов:")
    print(self.df_merged[self.df_merged['content_type_id'] == 0]
          .groupby('user_id')['answered_correctly'].sum().sort_values(ascending=False).head(3))
    print("Выведем первые три пары пользователь - task_container_id по числу верных ответов")
    print(self.df_merged[self.df_merged['content_type_id'] == 0]
          .groupby(['user_id', 'task_container_id'])['answered_correctly'].sum().sort_values(
          ascending=False).head(3))
    print("Выведем первые три пары пользователь - part по числу верных ответов")
    print(self.df_merged[self.df_merged['content_type_id'] == 0]
          .groupby(['user_id', 'part'])['answered_correctly'].sum().sort_values(ascending=False).head(3))
    print("Выведем первые три пары пользователь - tag_question по числу верных ответов")
    print(self.df_merged[self.df_merged['content_type_id'] == 0]
          .groupby(['user_id', 'tag_question'])['answered_correctly'].sum()
          .sort_values(ascending=False).head(3))
    print("Выведем первые три пары пользователь - bundle_id по числу верных ответов")
    print(self.df_merged[self.df_merged['content_type_id'] == 0]
          .groupby(['user_id', 'bundle_id'])['answered_correctly'].sum().sort_values(ascending=False).head(3))

    print("Выведем первых трех пользователей по времени, данному на верные ответы:")
    print(self.df_merged[(self.df_merged['content_type_id'] == 0)
                         & (self.df_merged['answered_correctly'] == 1)]
          .groupby('user_id')['timestamp'].sum().sort_values(ascending=False).head(3))
    print("Выведем первые три пары пользователь - task_container_id по времени, данному на верные ответы")
    print(self.df_merged[(self.df_merged['content_type_id'] == 0)
                         & (self.df_merged['answered_correctly'] == 1)]
          .groupby(['user_id', 'task_container_id'])['timestamp'].sum().sort_values(
          ascending=False).head(3))

```

```

print("Выведем первые три пары пользователь - part по времени, данному на верные ответы")
print(self.df_merged[(self.df_merged['content_type_id'] == 0)
                     & (self.df_merged['answered_correctly'] == 1)]
      .groupby(['user_id', 'part'])['timestamp'].sum().sort_values(ascending=False).head(3))
print("Выведем первые три пары пользователь - tag_question по времени, данному на верные ответы")
print(self.df_merged[(self.df_merged['content_type_id'] == 0)
                     & (self.df_merged['answered_correctly'] == 1)]
      .groupby(['user_id', 'tag_question'])['timestamp'].sum()
      .sort_values(ascending=False).head(3))
print("Выведем первые три пары пользователь - bundle_id по времени, данному на верные ответы")
print(self.df_merged[(self.df_merged['content_type_id'] == 0)
                     & (self.df_merged['answered_correctly'] == 1)]
      .groupby(['user_id', 'bundle_id'])['timestamp'].sum().sort_values(ascending=False).head(3))

print("Выведем первых трех пользователей по времени, затраченному на лекции:")
print(self.df_merged[self.df_merged['content_type_id'] == 1]
      .groupby('user_id')['timestamp'].sum().sort_values(ascending=False).head(3))
print("Выведем первые три пары пользователь - tag_lecture по времени, затраченному на лекции")
print(self.df_merged[self.df_merged['content_type_id'] == 1]
      .groupby(['user_id', 'tag_lecture'])['timestamp'].sum()
      .sort_values(ascending=False).head(3))
print("Выведем первые три пары пользователь - part_lecture по времени, затраченому на лекции")
print(self.df_merged[self.df_merged['content_type_id'] == 1]
      .groupby(['user_id', 'part_lecture'])['timestamp'].sum().sort_values(ascending=False).head(3))

print("Выведем первых трех пользователей по числу просмотра правильных ответов (подсказок):")
print(self.df_merged[self.df_merged['content_type_id'] == 0]
      .groupby('user_id')['prior_question_had_explanation'].count().sort_values(ascending=False).head(3))

print("Выведем первых трех пользователей по среднему времени ответов на предыдущие серии вопросов:")
sum_prior_question_elapset_time_by_user = \
    self.df_merged[self.df_merged['content_type_id'] == 0].groupby('user_id')[

        'prior_question_elapsed_time'].sum().sort_values(ascending=False)
count_prior_question_elapset_time_by_user = \
    self.df_merged[self.df_merged['content_type_id'] == 0].groupby('user_id')[

        'prior_question_elapsed_time'].count().sort_values(ascending=False)
sum_prior_question_elapset_time_by_user_count_prior_question_elapset_time_by_user_merged = \
    pd.DataFrame(sum_prior_question_elapset_time_by_user).merge(
        pd.DataFrame(count_prior_question_elapset_time_by_user), how="left", on='user_id')
sum_prior_question_elapset_time_by_user_count_prior_question_elapset_time_by_user_merged[
    'prior_question_elapsed_time_average'] = \
    sum_prior_question_elapset_time_by_user_count_prior_question_elapset_time_by_user_merged[
        "prior_question_elapsed_time_x"] / \
    sum_prior_question_elapset_time_by_user_count_prior_question_elapset_time_by_user_merged[

```

```

        "prior_question_elapsed_time_y"]

print(sum_prior_question_elapset_time_by_user_count_prior_question_elapset_time_by_user_merged.head(3))

self.df_merged['tag_question'] = self.df_merged['tag_question'].str.split(" ")
self.df_merged = self.df_merged.explode('tag_question') # приведем значения к атомарным

print("Выведем первые три пары пользователь - tag_question по количеству верных ответов")
print(self.df_merged[(self.df_merged['content_type_id'] == 0)
                     & (self.df_merged['answered_correctly'] == 1)]
      .groupby(['user_id', 'tag_question'])['tag_question'].count().sort_values(ascending=False).head(3))

print("Вышеприведенные показатели могут быть использованы как отдельные фичи в рамках статистического "
      "анализа (например, с использованием матрицы парных корреляций), что за пределами домашней задачи.")

print(datetime.utcnow(), ": <-- Анализ объединенного датафрейма и создание фичей завершены -->")

@staticmethod
def clear_data_auxiliary(data_frame, data_frame_name, dropna=True):
    """Drops the NaN containing lines if needed, drops empty trimmed string containing lines.

    :param data_frame: data_frame to clear
    :param data_frame_name: str, data_frame_name to clear
    :param dropna: boolean, is we need to drop NaN containing lines
    :return: data_frame filtered
    """

    print(datetime.utcnow(), ": Очищаем данные %s .." % data_frame_name)
    if (dropna):
        print("измерения %s до dropna():" % data_frame_name, data_frame.shape)
        data_frame = data_frame.dropna()
    print("измерения %s до удаления строк с NaN:" % data_frame_name, data_frame.shape)
    data_frame = data_frame.replace(r'^\s*$', np.nan, regex=True)
    print("измерения %s после очистки:" % data_frame_name, data_frame.shape)
    return data_frame

```

In [2]:

```

"""Записаны полные наборы данных."""
train_full_url = r'C:\\Users\\nM\\Downloads\\python\\train.csv'
questions_full_url = r'C:\\Users\\nM\\Downloads\\python\\questions.csv'
lectures_full_url = r'C:\\Users\\nM\\Downloads\\python\\lectures.csv'

generator = FeatureGenerator(train_full_url,
                             questions_full_url,

```

```
lectures_full_url)
generator.generate()
```

```
2022-10-10 18:18:12.103877 : <-- Очистка данных --
2022-10-10 18:18:18.719038 : Очищаем данные df_train ..
измерения df_train до удаления строк с NaN: (101230332, 10)
измерения df_train после очистки: (101230332, 10)
2022-10-10 18:19:40.926144 : Очищаем данные df_questions ..
измерения df_questions до dropna(): (13523, 5)
измерения df_questions до удаления строк с NaN: (13522, 5)
измерения df_questions после очистки: (13522, 5)
2022-10-10 18:19:40.946132 : Очищаем данные df_lectures ..
измерения df_lectures до dropna(): (418, 4)
измерения df_lectures до удаления строк с NaN: (418, 4)
измерения df_lectures после очистки: (418, 4)
Выведем первые три строки и посмотрим, насколько полезны столбцы:
```

	row_id	timestamp	user_id	content_id	content_type_id	task_container_id	user_answer	answered_correctly	prior_question_elapsed_time	prior_quest
0	0	0	115	5692		0	1	3	1	NaN
1	1	56943	115	5716		0	2	2	1	37000.0
2	2	118363	115	128		0	0	0	1	55000.0

	question_id	bundle_id	correct_answer	part	tags
0	0	0	0	1	51 131 162 38
1	1	1	1	1	131 36 81
2	2	2	0	1	131 101 162 92

	lecture_id	tag	part	type_of
0	89	159	5	concept
1	100	70	1	concept
2	185	45	6	concept

```
-----  
2022-10-10 18:19:44.576131 : <-- Очистка данных завершена -->  
2022-10-10 18:19:44.577129 : <-- Объединение датафреймов в один -->  
Выведем типы для train dataframe: row_id int64  
timestamp int64  
user_id int64  
content_id int64  
content_type_id int64  
task_container_id int64  
answered_correctly int64  
prior_question_elapsed_time float64  
prior_question_had_explanation object  
dtype: object  
2022-10-10 18:20:52.014889 : <-- Объединение датафреймов в один -->  
2022-10-10 18:20:52.014889 : <-- Анализ объединенного датафрейма и создание фичей -->  
Выведем типы для объединенного dataframe: row_id int64  
timestamp int64  
user_id int64  
content_id int64  
content_type_id int64  
task_container_id int64  
answered_correctly int64  
prior_question_elapsed_time float64  
prior_question_had_explanation object  
bundle_id float64  
part float64  
tag_question object  
tag_lecture float64  
part_lecture float64  
dtype: object  
Выведем размерность для объединенного dataframe: (101230332, 14)  
Выведем описательную статистику для объединенного dataframe:  
count 1.012303e+08 1.012303e+08 1.012303e+08 1.012303e+08 row_id timestamp user_id content_id \  
mean 5.061517e+07 7.703644e+09 1.076732e+09 5.219605e+03  
std 2.922268e+07 1.159266e+10 6.197163e+08 3.866359e+03  
min 0.000000e+00 0.000000e+00 1.150000e+02 0.000000e+00  
25% 2.530758e+07 5.243436e+08 5.408116e+08 2.063000e+03  
50% 5.061517e+07 2.674234e+09 1.071781e+09 5.026000e+03  
75% 7.592275e+07 9.924551e+09 1.615742e+09 7.425000e+03  
max 1.012303e+08 8.742577e+10 2.147483e+09 3.273600e+04  
  
content_type_id task_container_id answered_correctly \  
count 1.012303e+08 1.012303e+08 1.012303e+08  
mean 1.935222e-02 9.040624e+02 6.251644e-01
```

```
std      1.377596e-01      1.358302e+03      5.225307e-01
min     0.000000e+00      0.000000e+00      -1.000000e+00
25%    0.000000e+00      1.040000e+02      0.000000e+00
50%    0.000000e+00      3.820000e+02      1.000000e+00
75%    0.000000e+00      1.094000e+03      1.000000e+00
max     1.000000e+00      9.999000e+03      1.000000e+00
```

```
        prior_question_elapsed_time      bundle_id      part      tag_lecture \
count          9.887879e+07      1.000385e+08      1.000385e+08      3.262751e+06
mean         2.542381e+04      5.006305e+03      4.083069e+00      8.988080e+01
std          1.994815e+04      3.296905e+03      1.671998e+00      5.421660e+01
min     0.000000e+00      0.000000e+00      1.000000e+00      0.000000e+00
25%    1.600000e+04      2.027000e+03      2.000000e+00      4.400000e+01
50%    2.100000e+04      4.987000e+03      5.000000e+00      8.500000e+01
75%    2.966600e+04      7.224000e+03      5.000000e+00      1.360000e+02
max     3.000000e+05      1.352200e+04      7.000000e+00      1.870000e+02
```

```
        part_lecture
count  3.262751e+06
mean   4.204516e+00
std    1.838849e+00
min    1.000000e+00
25%   2.000000e+00
50%   5.000000e+00
75%   5.000000e+00
max    7.000000e+00
```

Выведем первых трех пользователей по числу верных ответов:

```
user_id
2139561972      14300
1615528747      13678
338684437       13416
```

```
Name: answered_correctly, dtype: int64
```

Выведем первые три пары пользователь - task_container_id по числу верных ответов

```
user_id      task_container_id
950779312      1079            10
1718256287      900             6
1662021225      702             5
```

```
Name: answered_correctly, dtype: int64
```

Выведем первые три пары пользователь - part по числу верных ответов

```
user_id      part
1219296290      5.0           7125
553683754       5.0           6874
497231435       5.0           6398
```

```
Name: answered_correctly, dtype: int64
```

Выведем первые три пары пользователь - tag_question по числу верных ответов

```
user_id      tag_question
1615528747    27          994
1219296290    8           884
553683754     8           853
Name: answered_correctly, dtype: int64
```

Выведем первые три пары пользователь - bundle_id по числу верных ответов

```
user_id      bundle_id
2053461581   7386.0      168
1617330386   1796.0      122
499347415    3180.0      122
Name: answered_correctly, dtype: int64
```

Выведем первых трех пользователей по времени, данному на верные ответы:

```
user_id
817562598    643721628763107
1057531137   606439622759213
758801035    598742766699655
Name: timestamp, dtype: int64
```

Выведем первые три пары пользователь - task_container_id по времени, данному на верные ответы

```
user_id      task_container_id
124443281   7959         420686940395
851832978   666          404811820195
937249824   1111         403034103315
Name: timestamp, dtype: int64
```

Выведем первые три пары пользователь - part по времени, данному на верные ответы

```
user_id      part
1057531137  5.0          288971025388884
1613056739  5.0          234619521713790
1686819041  5.0          226135340448276
Name: timestamp, dtype: int64
```

Выведем первые три пары пользователь - tag_question по времени, данному на верные ответы

```
user_id      tag_question
792038627    8           34413968676454
2101866594   8           30461341327906
1613056739   8           30368708462026
Name: timestamp, dtype: int64
```

Выведем первые три пары пользователь - bundle_id по времени, данному на верные ответы

```
user_id      bundle_id
2053461581   7386.0      5690615497949
1822049140   1796.0      4614326320204
1754.0       3787168345479
Name: timestamp, dtype: int64
```

Выведем первых трех пользователей по времени, затраченному на лекции:

```
user_id
```

```
817562598    18241402744824
```

```
1057531137   17603025682113
```

```
1451132808   15384576593989
```

```
Name: timestamp, dtype: int64
```

```
Выведем первые три пары пользователь - tag_lecture по времени, затраченному на лекции
```

```
user_id      tag_lecture
```

```
383318667   48.0        766482871810
```

```
153187243   113.0      503328016201
```

```
           161.0        502233814773
```

```
Name: timestamp, dtype: int64
```

```
Выведем первые три пары пользователь - part_lecture по времени, затраченному на лекции
```

```
user_id      part_lecture
```

```
817562598   5.0         8853565902916
```

```
383318667   5.0         7628028025062
```

```
1216339429  5.0         6354521873820
```

```
Name: timestamp, dtype: int64
```

```
Выведем первых трех пользователей по числу просмотра правильных ответов (подсказок):
```

```
user_id
```

```
801103753   17608
```

```
1478712595  16841
```

```
455973631   16639
```

```
Name: prior_question_had_explanation, dtype: int64
```

```
Выведем первых трех пользователей по среднему времени ответов на предыдущие серии вопросов:
```

```
          prior_question_elapsed_time_x  prior_question_elapsed_time_y  \
```

```
user_id
```

```
1660941992    1.042999e+09            16408
```

```
1743444187    6.839660e+08            16423
```

```
801103753    6.612836e+08            17608
```

```
          prior_question_elapsed_time_average
```

```
user_id
```

```
1660941992    63566.464956
```

```
1743444187    41646.835048
```

```
801103753    37555.862108
```