2 Задание

У следующих 5 асессоров очень большой разрыв по процентам ошибок от других асессоров. Асессоры с id 3, 56, 118, 234, 390 имеют более чем по 50% ошибочных ответов в отличии от других. Показатели ошибок остальных асессоров не поднимались выше 27.343750%. При том важно понимать, что так же было 4 задания, в которых все допустили ошибки. Их error rate = 100%. Но асессоры с рейтингом ошибок более 50% не выполняли тяжелые задания. Все задания, которые они провалили, хоть один другой асессор смог верно выполнить.

uid	error	not_error	error_rate
234	51	48	**51.515152**
390	214	198	**51.941748**
118	205	186	**52.429668**
3	230	196	**53.990610**
56	236	175	**57.420925**

Для решения этой задачи я использовал метрики точности (precision) и полноты (recall)
Считаю, что данного подхода достаточно для решения качества асессоров. Но при усложнении задачи могут потребоваться новые метрики.

Этапы решения:

1. Прочитал датасет и проверил количество строк и размер.
2. Добавил новую колонку, в которой отобразил есть ли ошибка или нет. Сравнивая показатели jud — оценка асессора и cjud — правильная оценка.
3. Дальше создал два новых датафрейма. Первый - для подсчета ошибок асессоров. Второй (не обязательный, просто решил проверить) - для проверки сложности задания.
4. В датафрейме сложности задания были пустые ячейки, пришлось заменять пустые значения на ноль.
5. Дальше в каждом датафрейме (подсчета ошибок асессоров и проверка сложности задания) создал рейтинг ошибок. Он в себя включал отношение ошибок к общему количеству выполненных заданий.
   100 / (ошибки + правильные ответы) * количество ошибок

6. Дальше отсортировал и увидел разрыв по рейтингу ошибок, решил перепроверить весь датафрейм ошибок асессоров и увидел, что только 5 асессоров имеют больше 50% ошибок.
7. Необязательно, но решил проверить сложные задания, в которых все допустили ошибки. И выполняли ли их асессоры с показателем ошибок более 50%. Выяснил, что они не выполняли сложных заданий.

In [1]:
```
import pandas as pd
import numpy as np
```

In [2]:
```
data = pd.read_csv('data_test_task/file2.csv', sep="\t")
```

In [3]:
```
data.head()
```

Out[3]:

	login	uid	docid	jud	cjud
0	assessor158	158	0	0	0
1	assessor238	238	0	0	0
2	assessor488	488	0	0	0
3	assessor136	136	0	0	0
4	assessor300	300	0	0	0

In [4]:
```
data.sort_values(['docid']).head(12)
```

Out[4]:

	login	uid	docid	jud	cjud
0	assessor158	158	0	0	0
1	assessor238	238	0	0	0
2	assessor488	488	0	0	0
3	assessor136	136	0	0	0
4	assessor300	300	0	0	0
9	assessor295	295	1	1	1
7	assessor409	409	1	1	1
8	assessor396	396	1	1	1
5	assessor123	123	1	1	1
6	assessor491	491	1	0	1
10	assessor37	37	2	0	0
11	assessor291	291	2	0	0

In [5]:
```
data.docid.nunique()
```

Out[5]:
```
50000
```

In [6]:
```
len(data[data.jud == 1])
```

Out[6]:
```
59530
```

In [7]:
```
len(data[data.cjud == 1])
```

Out[7]:
```
29980
```

In [8]:
```
data.uid.nunique()
```

Out[8]:
```
600
```

250 000 строк

600 Асессоров

Получается уникальных docid — id оцениваемого документа (document id) 50 000

jud — оценка асессора (judgement) 59 530

cjud — правильная оценка (correct judgement) 29 980

Ошибочных строк 39678

In [9]:
```
data['fails'] = np.where(data.jud != data.cjud, 'error', 'not_error')
```

In [10]:
```
data.head(8)
```

Out[10]:

	login	uid	docid	jud	cjud	fails
0	assessor158	158	0	0	0	not_error
1	assessor238	238	0	0	0	not_error
2	assessor488	488	0	0	0	not_error
3	assessor136	136	0	0	0	not_error
4	assessor300	300	0	0	0	not_error

	login	uid	docid	jud	cjud	fails
5	assessor123	123	1	1	1	not_error
6	assessor491	491	1	0	1	error
7	assessor409	409	1	1	1	not_error

Дальше создаем два датафрейма, чтобы понять сколько ошибок допустили исполнители и на сколько сложные были задания

In [11]:
```
new_data_doc = data.groupby(['docid','fails']).size()
new_data_doc
```

Out[11]:
```
docid  fails
0      not_error    5
1      error        1
       not_error    4
2      error        1
       not_error    4
                   ..
49997  error        1
       not_error    4
49998  error        1
       not_error    4
49999  not_error    5
Length: 78910, dtype: int64
```

In [12]:
```
new_data_fails = data.groupby(['uid','fails']).size()
new_data_fails
```

Out[12]:
```
uid  fails
0    error         65
     not_error    336
1    error         82
     not_error    330
2    error         76
                 ...
597  not_error    383
598  error         94
     not_error    324
599  error         40
     not_error    384
Length: 1200, dtype: int64
```

In [13]:
```
new_data_doc = new_data_doc.unstack(level=1)
new_data_doc = new_data_doc.fillna(0)    # заменили NaN на 0
new_data_doc
```

Out[13]:

fails	error	not_error
docid		
0	0.0	5.0
1	1.0	4.0
2	1.0	4.0
3	1.0	4.0
4	1.0	4.0
...	...	...
49995	0.0	5.0
49996	0.0	5.0
49997	1.0	4.0
49998	1.0	4.0
49999	0.0	5.0

50000 rows × 2 columns

In [14]:
```
new_data_fails = new_data_fails.unstack(level=1)
new_data_fails
```

Out[14]:

fails	error	not_error
uid		
0	65	336
1	82	330
2	76	303
3	230	196
4	72	346
...	...	...
595	67	377
596	33	365

uid	fails	error	not_error
597		46	383
598		94	324
599		40	384

600 rows × 2 columns

In [15]:
```
new_data_doc['error_rate'] = 100 / (new_data_doc.error + new_data_doc.not_error) * new_data_doc.error
new_data_doc
```

Out[15]:

docid	fails	error	not_error	error_rate
0		0.0	5.0	0.0
1		1.0	4.0	20.0
2		1.0	4.0	20.0
3		1.0	4.0	20.0
4		1.0	4.0	20.0
...		...	...	...
49995		0.0	5.0	0.0
49996		0.0	5.0	0.0
49997		1.0	4.0	20.0
49998		1.0	4.0	20.0
49999		0.0	5.0	0.0

50000 rows × 3 columns

In [16]:
```
new_data_fails['error_rate'] = 100 / (new_data_fails.error + new_data_fails.not_error) * new_data_fails.error
new_data_fails
```

Out[16]:

fails	error	not_error	error_rate

fails	error	not_error	error_rate
uid			
0	65	336	16.209476
1	82	330	19.902913
2	76	303	20.052770
3	230	196	53.990610
4	72	346	17.224880
...	...	...	...
595	67	377	15.090090
596	33	365	8.291457
597	46	383	10.722611
598	94	324	22.488038
599	40	384	9.433962

600 rows × 3 columns

In [17]:
```
new_data_doc.sort_values(['error_rate'])
```

Out[17]:

fails	error	not_error	error_rate
docid			
0	0.0	5.0	0.0
26688	0.0	5.0	0.0
26686	0.0	5.0	0.0
26685	0.0	5.0	0.0
26678	0.0	5.0	0.0
...	...	...	...
25274	4.0	1.0	80.0
41326	5.0	0.0	100.0

docid	fails	error	not_error	error_rate
34709		5.0	0.0	100.0
9457		5.0	0.0	100.0
2906		5.0	0.0	100.0

50000 rows × 3 columns

Увидеть самые плохие показатели по количеству ошибок.

In [18]:
```
new_data_fails.sort_values(['error_rate']).tail(12)
```

Out[18]:

uid	fails	error	not_error	error_rate
219		94	300	23.857868
221		96	304	24.000000
71		105	328	24.249423
154		99	306	24.444444
335		108	311	25.775656
9		110	313	26.004728
550		35	93	27.343750
234		51	48	51.515152
390		214	198	51.941748
118		205	186	52.429668
3		230	196	53.990610
56		236	175	57.420925

In [19]:
```
new_data_fails[new_data_fails.error_rate > 50].sort_values(['error_rate'])
```

Out[19]:

uid	fails	error	not_error	error_rate

uid	fails	error	not_error	error_rate
234		51	48	51.515152
390		214	198	51.941748
118		205	186	52.429668
3		230	196	53.990610
56		236	175	57.420925

Проверка заданий. В каких заданиях высокий рейтинг ошибок.

In [20]:
```
new_data_doc.sort_values(['error_rate']).tail(7)
```

Out[20]:

docid	fails	error	not_error	error_rate
47323		4.0	1.0	80.0
36739		4.0	1.0	80.0
25274		4.0	1.0	80.0
41326		5.0	0.0	100.0
34709		5.0	0.0	100.0
9457		5.0	0.0	100.0
2906		5.0	0.0	100.0

Выполняли ли задания с 100% error_rate ассесоры с большим рейтингом ошибок?

In [21]:
```
data[(data.docid == 41326) | (data.docid == 34709) \
     | (data.docid == 9457) | (data.docid == 2906)].sort_values(['uid'])
```

Out[21]:

	login	uid	docid	jud	cjud	fails
206631	assessor58	58	41326	1	0	error
173545	assessor79	79	34709	1	0	error
14530	assessor91	91	2906	1	0	error

	login	uid	docid	jud	cjud	fails
47285	assessor99	99	9457	1	0	error
47288	assessor170	170	9457	1	0	error
14534	assessor199	199	2906	1	0	error
206634	assessor281	281	41326	1	0	error
206633	assessor300	300	41326	1	0	error
47289	assessor307	307	9457	1	0	error
206630	assessor323	323	41326	1	0	error
173547	assessor339	339	34709	1	0	error
14531	assessor386	386	2906	1	0	error
173548	assessor410	410	34709	1	0	error
173549	assessor427	427	34709	1	0	error
47286	assessor452	452	9457	1	0	error
14533	assessor497	497	2906	1	0	error
206632	assessor539	539	41326	1	0	error
47287	assessor540	540	9457	1	0	error
173546	assessor562	562	34709	1	0	error
14532	assessor570	570	2906	1	0	error

In [ ]: