

Система скоринга обучающихся 1Т

#4EduTech Pipeline обработки данных

Документ для понимания обработки данных в проекте

Разработчик документа: Яковлев А.Л. DO

Дата создания: 31.10.2024

Оглавление

Концепция проекта «Система скоринга обучающихся 1Т»	2
Анализ существующей ситуации, параметры, по которым можно определять эти статусы	3
Общая схема обработки данных	1
Получение данных, преобразование из формата CSV в формат SQL	1
Работа с данными в базе PostgreSQL	4
Работа DS над данными	10
Рассматриваемые гипотезы	17

Название проекта: Система скоринга обучающихся 1Т.

Цель проекта: Создать систему скоринга на основе искусственного интеллекта, которая позволит автоматически оценивать уровень вовлеченности и вероятность успешного завершения курса каждым учеником. Основная цель системы — предупреждать отток студентов.

Видение:

Система должна в автоматическом режиме собирать и обрабатывать с помощью искусственного интеллекта полный набор доступных характеристик учеников и их прогресса в обучении и предсказывать:

- Вероятность того, что студент бросит курс.
- Вероятность успешного завершения курса каждым учеником.
- Автоматически распределять студентов по категориям активности (например, "активные", "засыпающие", "уснувшие").
- “Критические точки” перехода в категории “засыпающих” и “уснувших”.

Классификация активности учеников, дополнительно к вероятности закончит/не закончит, поможет на более ранних этапах заметить снижение вовлеченности. Например, даже если вероятность завершения курса все еще высока, ученик может попасть в категорию "засыпающих", что сигнализирует о возможных будущих проблемах. Это дает возможность вмешаться до того, как ситуация станет критической, а также понять, на каких этапах студенты испытывают наибольшие затруднения, и когда целесообразна дополнительная мотивация их к обучению.

Анализ существующей ситуации, параметры, по которым можно определять эти статусы

Предлагаемые статусы

1. Спящие:

- * Не активны в течение определенного периода времени (например, 20+ дней).
- * Не взаимодействуют с учебными материалами
- * Не выполняют задания
- * Отсутствует прогресс по курсу

2. Засыпающие:

- * Не активны в течение 7-19 дней.
- *

Проявляют минимальную активность (например, просматривают определенные разделы курса).

- * Выполняют часть доступных заданий
- * Прогресс по курсу минимальный

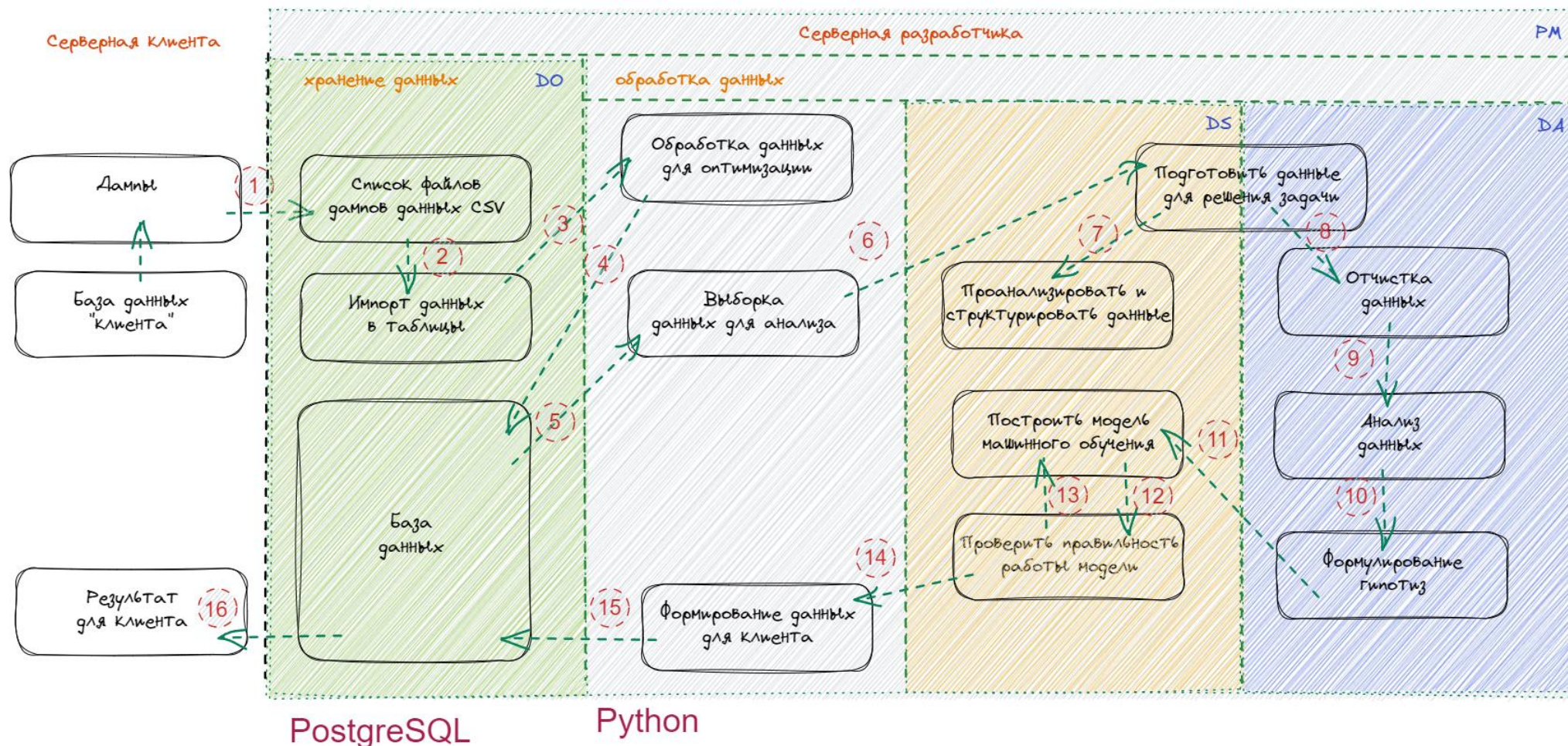
3. Активные:

- * Регулярно заходят на платформу электронного обучения.
- * Взаимодействуют с учебными материалами (например, просматривают видео, читают тексты, отвечают на тесты)
- * Завершают задания в срок.
- * Прогресс по курсу заметен

Параметры для определения статуса

- Активность на платформе: частота захода на платформу, время просмотра материалов, количество просмотренных страниц.
- Взаимодействие с учебными материалами: количество просмотренных видео, прочитанных текстов, пройденных тестов.
- Взаимодействие с преподавателем и студентами: количество заданных вопросов, ответов на вопросы преподавателя и других студентов, участие в обсуждениях.
- Сдача заданий: своевременность сдачи заданий, качество выполнения.
- Успеваемость: результаты тестирования, оценка работ.
- Посещение вебинаров

Общая схема обработки данных



1 – на данный момент данные в виде CSV файлов представлены, в будущем подключение к БД клиента, 2 – Импорт данных в базу данных, 3 – чистка и обработка данных, 4 – сохранение в базу данных, 5 – получение данных для дальнейшего анализа, 6 – формирование датасетов для DA и DS, 7 – этап работы DS с данными для понимания метрик, 8,9,10 – работа с данными для создания датасета, 11 – получение готового датасета для DS, 12,13 – обучение модели на датасете и проверка гипотез, 14 – формирование результирующих данных для клиента, 15 – сохранение данных в базе данных, 16 – передача клиенту результата. В будущем данные будут приходить, формироваться датасет, передаваться модели, возвращаться клиенту

Получение данных, преобразование из формата CSV в формат SQL

Данные были предоставлены файлами CSV, т.к. в будущем проект будет получать данные непосредственно из базы данных клиента, делать автоматическую загрузку не было смысла. В процессе работы с данными были сделаны предварительные изменения с названиями полей

CSV: users_age_timezone.csv				Table: users_age_timezone		
	A	B	C	Название	#	Тип данных
1	userID	982	2566	123 user_id	1	int4
2	timeZone	+03:00	+03:00	ABC time_zone	2	varchar(50)
3	age	16	16	123 age	3	int4

CSV: authorization.csv				Table: authorization		
	A	B	C	Название	#	Тип данных
1	user_id	7722	7722	123 user_id	1	int4
2	created_at	2022-12-07 17:43:14	2022-12-07 19:51:34	🕒 created_at	2	timestamp
3	user_agent	Mozilla/5.0 (Windows	Mozilla/5.0 (Window	ABC user_agent	3	varchar(256)
4	window size	1872x932	1872x918	ABC window_size	4	varchar(50)

CSV: schedule.csv				Table: schedule		
	A	B	C	Название	#	Тип данных
1	course_id	3	3	123 course_id	1	int4
2	type	активность	активность	ABC type	2	varchar(50)
3	taskID	81	81	123 task_id	3	int4
4	activivtyID	8719	8720	123 activivty_id	4	int4
5	activityType	slide	slide	ABC activity_type	5	varchar(50)
6	isAttestation	0	0	123 is_attestation	6	int4
7	visibility	{}	{}	ABC visibility	7	varchar(50)
8	flows	1	1	123 flows	8	int4
9	dateShown	2023-12-05 11:00:00	2023-12-05 11:00:00	🕒 date_shown	9	timestamp

CSV: activity_history_viewed.csv				Table: activity_history_viewed		
	A	B	C	Название	#	Тип данных
1	user_id	982	982	123 user_id	1	int4
2	created_at	2024-01-24 19:14:11	2024-01-24 19:34:09	🕒 created_at	2	timestamp
3	page_type	занятие	занятие	ABC page_type	3	varchar(50)
4	page_id	3015	3015	123 page_id	4	int4
5	module	2	2	123 module	5	int4
6	attestation	1	1	123 attestation	6	int4
7	activity_type			ABC activity_type	7	varchar(50)

CSV: webinars_logs.csv

	A	B	C
1	userId	74952	74952
2	dateTime	23.12.2023 12:10	23.12.2023 12:15
3	eventName	Подключение	Отключение
4	webinarId	13378	13378
5	формат подключения	офлайн	офлайн
6	вводный вебинар	1	1
7	module	2	2

Table: webinars_logs

Название	#	Тип данных
123 user_id	1	int4
🕒 datetime	2	timestamp
ABC event_name	3	varchar(50)
123 webinar_id	4	int4
ABC conn_format	5	varchar(50)
123 webinar_vvod	6	int4
123 module	7	int4

CSV: exercise_results.csv

	A	B	C
1	module	2	2
2	activityID	13278	13278
3	userId	72002	66421
4	createdAt	2023-12-01 08:53:00	2023-12-01 09:18:55
5	result	100	100
6	success	1	1

Table: exercise_results

Название	#	Тип данных
123 module	1	int4
123 activity_id	2	int4
123 user_id	3	int4
🕒 created_at	4	timestamp
ABC result	5	varchar(50)
123 success	6	int4

CSV: users_logs.csv

	A	B	C
1	user_id	2566	2614
2	created_a	14.11.2023 15:07	19.02.2024 17:20
3	event	tags-changed	tags-changed
4	comment	Были хештеги: #entry-email #офлайн, стали: #entry-email #онлайн	Были хештеги: #мотив #онлайн #M1-завершил #сверка-дат #M1-оплачен #M2-завершил, стали: #мотив #онлайн #M1-завершил #M1-оплачен #M2-завершил

Table: users_logs

Название	#	Тип данных
123 user_id	1	int4
🕒 created_at	2	timestamp
ABC event	3	varchar(50)
ABC comment	4	varchar(256)

CSV: users.csv

	A	B	C
1	unti_id	1051004	1118021
2	userID	982	2566
3	course_id	3	3
4	flow_num	1.00	1.00
5	tgBot	—	—
6	M2_progress	13	0
7	M2_attestation	Не сдана	Не сдана
8	M2_attestation_date		

Table: users

Название	#	Тип данных
123 unti_id	1	int4
123 user_id	2	int4
123 course_id	3	int4
123 flow_num	4	float4
ABC tg_bot	5	varchar(50)
ABC m2_progress	6	varchar(50)
ABC m2_attestation	7	varchar(50)
🕒 m2_attestation_date	8	timestamp

CSV: activities_guide.csv				Table: activities_guide		
	A	B	C	Название	#	Тип данных
1	courseID	3		123 course_id	1	int4
2	Курс	Нейро.PY	Нейро	ABC course	2	varchar
3	Провайдер	1T		ABC provider	3	varchar
4	Модуль	2		123 modul	4	int4
5	themeID	13		123 theme_id	5	int4
6	Тема	Функции и классы	Функции и к	ABC theme	6	varchar
7	taskID	81		123 task_id	7	int4
8	Занятие	Функции и запроса данных	Функции и запроса д	ABC exercise	8	varchar
9	taskPosition	1		123 task_position	9	int4
10	Признак Аттестации	0		123 att_priznak	10	int4
11	activityID	14823		123 activity_id	11	int4
12	Тип Активности	CodeExercise	CodeEx	ABC activity_type	12	varchar
13	Активность	Тренажер 1	Трена	ABC activity	13	varchar
14	Признак Обязательного	0		123 obyaz_priznak	14	int4
15	Видимость	{}		ABC visibility	15	varchar

Работа с данными в базе PostgreSQL

Фильтр выборки обучающихся по данным был задан заказчиком, брать тех кто имеет статус **онлайн**

```
with
t1 as(
select distinct user_id, created_at, comment
from public.users_logs logs
where comment like 'Были хештеги:%онлайн%стали:%онлайн%'
or comment like 'Установлен хештег "#онлайн" --'%Установлен%онлайн%'
or comment like 'Были хештеги: , стали:%онлайн%'
or comment like 'Были хештеги%оулайн%стали:%онлайн%'
or comment like 'Были хештеги%оглайн%стали:%онлайн%')

select *
from users
where "userID" in(select distinct user_id from t1)
```

@TatianaGlu

В процессе работы с данными аналитиками были сформированы представления, по которым проверялись гипотезы

Коллеги, позволил себе нескромность сделать новые вьюшки из датасетов, которые собрали мы с Таней. По крайней мере мне так удобнее пользоваться, если работать напрямую с базой.

dataset_v - 1й датасет, где все даты расположены по вертикали (V)

dataset_h - 2й датасет, где часть событий сопоставлена по горизонтали (H)

Важное уточнение! Убрал в 1м датасете фильтрацию только успешных студентов (строк стало больше), но сохранил фильтрацию тех студентов, по которым есть след во всех таблицах.

2й датасет сразу был собран по такому принципу.

@ryurikovich_37

dataset_h	dataset_h_v3	dataset_v	glu_dataset
123 user_id	123 user_id	123 user_id	123 user_id
123 course_id	123 course_id	123 course_id	123 age
ABC type	ABC type	ABC event	ABC time_zone
ABC activity_type	ABC activity_type	ABC type	123 course_id
123 task_id	123 task_id	ABC activity_type	ABC event
123 activity_id	123 activity_id	123 task_id	ABC time
date_shown	date_shown	123 activity_id	123 task_id
123 is_attestation	123 is_attestation	123 result	ABC type
created_at	created_at	123 success	123 activity_id
123 module	123 module	123 m2_progress	ABC activity_type
123 attestation	123 attestation	123 m2_attestation	123 obyzar_priznak
result_time	result_time	123 age	123 result
123 result	123 result	ABC time_zone	123 success
123 success	123 success	123 age	123 m2_progress
ABC tg_bot	ABC tg_bot		123 m2_attestation
123 m2_progress	123 m2_progress		
m2_attestation_date	m2_attestation_date		
123 m2_attestation	123 m2_attestation		
123 age	123 age		
ABC time_zone	ABC time_zone		

Новую версию сохранил как `dataset_h_v3`

Изменения:

- 1) Данные на основе датасета `students_v3` (см. выше)
 - 2) Не стал выкидывать студентов, которые засветились не во всех таблицах
 - 3) Добавил столбец обязательного признака задания
- Датасет стал тяжелым (1 217 959 строк), выполняется долго.

@ryurikovich_37

```
with
cross_schedule as (
select sv2.user_id, sv.course_id, sv."type", sv.activity_type, sv.task_id, sv.activity_id, sv.date_shown, sv.is_attestation
from schedule_v2 sv
full outer join students_v3 sv2
on sv2.course_id = sv.course_id
),
cross_history as (
select
cross_schedule.user_id, cross_schedule.course_id, cross_schedule."type", cross_schedule.activity_type, cross_schedule.task_id,
cross_schedule.activity_id, cross_schedule.date_shown, cross_schedule.is_attestation,
ahvv.created_at, ahvv."module", ahvv.attestation
from cross_schedule
left join activity_history_viewed_v2 ahvv
on (ahvv.user_id = cross_schedule.user_id) and (ahvv.page_id = cross_schedule.task_id)
where type = 'занятие'
union
select cross_schedule.user_id, cross_schedule.course_id, cross_schedule."type", cross_schedule.activity_type, cross_schedule.task_id,
cross_schedule.activity_id, cross_schedule.date_shown, cross_schedule.is_attestation,
ahvv.created_at, ahvv."module", ahvv.attestation
from cross_schedule
left join activity_history_viewed_v2 ahvv
on (ahvv.user_id = cross_schedule.user_id) and (ahvv.page_id = cross_schedule.activity_id)
where type = 'активность'),
history_results as (
select cross_history.*, erv.created_at as result_time,
case when erv.result = 'Пропуск' then -1 else cast(erv.result as int4) end as result,
erv.success
from cross_history
left join exercise_results_v2 erv
on (erv.user_id = cross_history.user_id) and (erv.activity_id = cross_history.activity_id)
)
select history_results.*, agv.obyzar_priznak, sv.tg_bot,
case when sv.m2_progress = 'Нет данных' then -1 else cast(sv.m2_progress as int4) end as m2_progress,
sv.m2_attestation_date,
case when sv.m2_attestation = 'Не сдана' then -1 else cast(sv.m2_attestation as int4) end as m2_attestation,
uatv.age, uatv.time_zone
from history_results
left join students_v3 sv
on sv.user_id = history_results.user_id
left join users_age_timezone_v2 uatv
on history_results.user_id = uatv.user_id
left join activities_guide_v2 agv
on agv.activity_id = history_results.activity_id
order by history_results.user_id, history_results.date_shown
```

m_course_params 123 course_id ABC name 🕒 min_date_shown 🕒 max_date_shown 123 num_tasks 123 num_activity 123 num_attestation_activity ABC attestation_activity_types 123 num_webinar_v2 123 num_questions 123 num_exercise 123 num_interactive 123 num_codeexercise 123 num_slide 123 num_empty_activity	m_online_users_exercise 123 user_id 123 course_id 123 task_id 123 m 🕒 activity_datetime_shown 🕒 result_datetime_upd 123 success 123 num_success 123 num_activities_all 123 max_num_repeat 123 is_attestation ABC m2_progress ABC m2_attestation 🕒 m2_attestation_date	m_user_tags_history 123 user_id 🕒 from_datetime 🕒 to_datetime ABC last_tag ABC tag <input checked="" type="checkbox"/> from_users_logs ABC comment	m_exercise_lag_time_series 123 user_id 123 course_id 123 day_num 🕒 date_column 123 sum_lag ABC m2_progress ABC m2_attestation 🕒 m2_attestation_date
students_v2 123 unti_id 123 user_id 123 course_id 123 flow_num ABC tg_bot ABC m2_progress ABC m2_attestation 🕒 m2_attestation_date	students_v3 123 unti_id 123 user_id 123 course_id 123 flow_num ABC tg_bot ABC m2_progress ABC m2_attestation 🕒 m2_attestation_date	m_online_students 123 unti_id 123 user_id 123 course_id 123 flow_num ABC tg_bot ABC m2_progress ABC m2_attestation 🕒 m2_attestation_date ABC learning_format	m_date_interval 123 day_num 🕒 date_column

По представлениям много было испробовано и отброшено общим решением ДА, после долгих споров, анализов и уточнений у заказчиков оставили текущие версии.

запросы, которыми формировали стартовые датасеты?

интересно, как считали sum_m2_progress и остальные значения

with

events as(

select user_id, created_at as time, 'authorization' as event, '' as type, '' as activity_type, 0 as task_id, 0 as activity_id, 0 as result, 0 as success, 0 as m2_progress, 0 as

m2_attestation

from authorization_v2 av

where user_id in (select user_id from public.students_v3)

union

select user_id, created_at as time, event, '' as type, '' as activity_type, 0 as task_id, 0 as activity_id, 0 as result, 0 as success, 0 as m2_progress, 0 as m2_attestation

from users_logs_v2 ulv

where user_id in (select user_id from public.students_v3)

union

select user_id, m2_attestation_date as time, 'attestation' as event, '' as type, '' as activity_type, 0 as task_id, 0 as activity_id, 0 as result, 0 as success,

case when m2_progress = 'Нет данных' then -1 else cast(m2_progress as int4) end as m2_progress,

case when m2_attestation = 'Не сдана' then -1 else cast(m2_attestation as int4) end as m2_attestation

from students_v2

where user_id in (select user_id from public.students_v3)

union

select user_id, created_at as time, 'history' as event, page_type as type, activity_type, 0 as task_id, null as activity_id, 0 as result, 0 as success, 0 as m2_progress, 0 as

m2_attestation

from public.activity_history_viewed_v2

where (page_type = 'занятие') and (user_id in (select user_id from public.students_v3))

```

union
select ahvv.user_id, ahvv.created_at as time, 'history' as event, page_type as type, ahvv.activity_type
, sv.task_id as task_id, ahvv.page_id as activity_id, 0 as result, 0 as success, 0 as m2_progress, 0 as m2_attestation
from public.activity_history_viewed_v2 ahvv
left join schedule_v2 sv on ahvv.page_id = sv.activity_id
where (ahvv.page_type = 'активность') and (ahvv.user_id in (select user_id from public.students_v3))
union
select user_id, created_at as time, 'results' as event, '' as type, '' as activity_type, 0 as task_id, activity_id,
case when result = 'Пропуск' then -1 else cast(result as int4) end as result, success as success, 0 as m2_progress, 0 as m2_attestation
from public.exercise_results_v2 erv
where user_id in (select user_id from public.students_v3)
union
select user_id, datetime as time, event_name as event, 'Вебинар' as type, '' as activity_type, 0 as task_id, webinar_id as activity_id, 0 as result, 0 as success, 0 as
m2_progress, 0 as m2_attestation
from public.webinars_logs_v2
where user_id in (select user_id from public.students_v3)
)
,--выбираем юзеров сдавших аттестацию по модулю вовремя
success_intime_users as(
select user_id from public.users_v2 --where m2_attestation_date between '2023-12-01 00:00:00.000' and '2024-02-01 00:00:00.000' закомментировали для неуспешных
)
,--собрали user_id, кто сдал аттестацию
success_users as(
select user_id from events where m2_attestation >= 50 and m2_progress >=50 and user_id in (select user_id from success_intime_users)
)
,--подтягиваем номер группы, часовой пояси возраст
events_with_personal_info as(
select ev.user_id, users.course_id, ev.time, ev.event, ev.type, ev.activity_type, ev.task_id, ev.activity_id, ev.result, ev.success, ev.m2_progress, ev.m2_attestation
, uat.time_zone, uat.age
from events ev
left join public.users_v2 users
on ev.user_id=users.user_id
left join public.users_age_timezone_v2 uat
on ev.user_id=uat.user_id
union
select 0 as user_id, course_id, date_shown as time, 'schedule' as event, type, activity_type, task_id, activity_id, 0 as result, 0 as success, 0 as m2_progress, 0 as
m2_attestation, '' as time_zone, 0 as age
from public.schedule_v2 sv2
order by user_id, time
)
,--выбрали юзеров из 49 группы успешно сдавших модуль и разбили их по неделям
dataset_vsev_suc as(
select user_id, age
, date_part('year', time) as year, date_part('week', time) week
-- , count(event) filter (where event = 'authorization') authorization_count
, count(event) filter (where event = 'history') all_activity_count
, count(event) filter (where event = 'attestation') attestation_event_count
, count(event) filter (where event = 'results') results_event_count
-- , count(activity_type) filter (where activity_type = 'CodeExercise') "CodeExercise_count"
-- , count(activity_type) filter (where activity_type = 'interactive') interactive_count
-- , count(activity_type) filter (where activity_type = 'slide') slide_count
-- , count(activity_type) filter (where activity_type = 'exercise') exercise_count
, sum(result) total_score
, sum(success) success_attempts
, count(success) total_attempts
, round((1.0*sum(success)/count(success)), 2) success_rate --удачные попытки / все попытки
, case when sum(success)!= 0 then 1.0*sum(result)/sum(success) else 0 end as avg_success_score --ср.балл удачных попыток сдачи

```

```

, case when count(success)!= 0 then 1.0*sum(result)/count(success) else 0 end as avg_score --средний балл всех попыток сдачи заданий
, sum(m2_progress) sum_m2_progress
from events_with_personal_info
where user_id in(select user_id from success_users) and user_id != 0
and course_id=49
and date_part('week', time) in(1, 2, 3, 4, 5, 48, 49, 50, 51, 52)
group by user_id, year, week, age
order by user_id, year, week
)

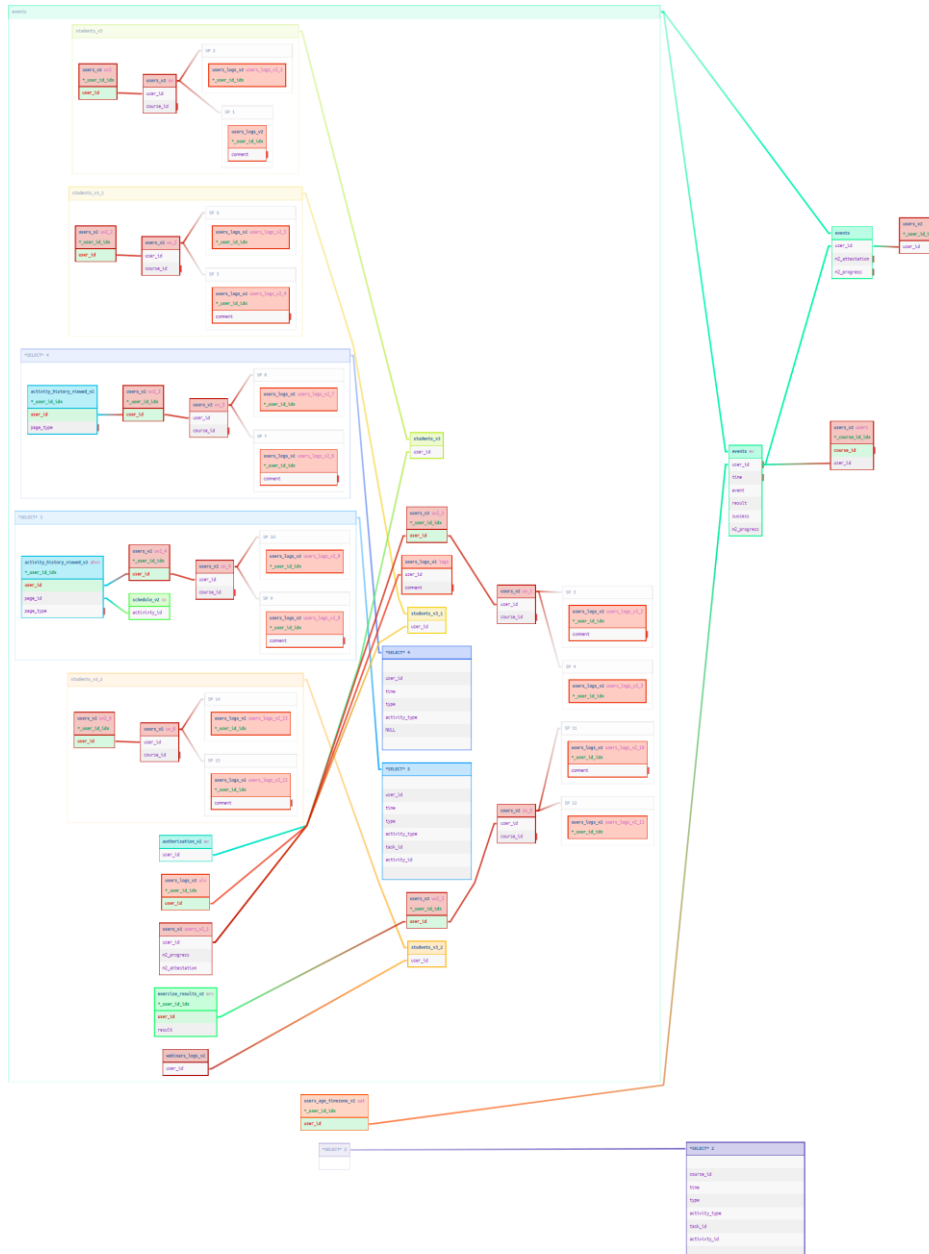
```

```

select * from dataset_vsev_suc

```

@TatianaGlu



20241024_dataset_timeseries_with_metrics_v_gavrilova
123 id
123 user_id
123 payment
123 time_zone
123 age
123 unti_id
123 course_id
123 flow_num
123 tg_bot
123 m2_progress
123 m2_attestation
123 module
123 m2_delay
123 sum_auth
123 sum_schedule_activities
123 sum_required_activity
123 sum_attestation_activity
123 view_delay_first
123 view_delay_sum
123 sum_activity_viewed
123 sum_required_activity_viewed
123 sum_attestation_activity_viewed
123 w_view_hours
123 sum_exercise
123 sum_required_exercises
123 sum_attestation_exercises
123 exercise_delay_first
123 exercise_delay_sum
123 result_delay_mean
123 result_delay_sum
123 sum_exercise_attempts_mean
123 mean_required_result
123 mean_non_req_result
123 mean_attestation_result
123 sum_result
123 conn
123 online_rate
123 mean_result
123 progress
ABC cur_date

Одно из представлений сделали таблицей и вывели в общее использование.

Работа DS над данными

Описание датасета с временными рядами и метриками

Датасет содержит 39 столбцов и 574140 строк.

Датасет содержит информацию о 2734 уникальных юзерах и для каждого юзера исследования на 210 дат, начиная с 2023-12-01 (начало обучения на курсе) с интервалом в 1 день. Если необходим другой интервал, лишние строки просто удаляются с нужными промежутками.

В датасете содержится большинство метрик, описанных аналитиками в документе <https://docs.google.com/document/d/1gvpjHjb-05bBbcDs7jgaP4128XxgwRwHuT8wo0OWQVk/edit?tab=t.0>

Информация по столбцам:

0. *user_id* - id юзера
1. *payment* - была ли оплата
2. *time_zone* - временная зона юзера в цифровом формате
3. *age* - возраст юзера
4. *unti_id* - id юзера по УНТИ
5. *course_id* - номер курса
6. *flow_num* - номер потока
7. *tg_bot* – подключен ли tg бот (0 – не подключен, 1 – подключен, 2 – остановлен)
8. *m2_progress* - итоговый прогресс по 2 модулю
9. *m2_attestation* - результат сдачи аттестации
10. *module* - модуль
11. *m2_delay* – задержка в сдаче аттестации в днях на дату *cur_date*
12. *sum_auth* – кол-во авторизаций на дату *cur_date*
13. *sum_schedule_activities* - кол-во открытых активностей на дату *cur_date*
14. *sum_required_activity* - кол-во обязательных открытых активностей на дату *cur_date*
15. *sum_attestation_activity* - кол-во аттестационных открытых активностей на дату *cur_date*
16. *view_delay_first* - среднее кол-во дней до первого просмотра активности на дату *cur_date*
17. *view_delay_sum* - суммарная задержка непросмотренных занятий в днях на дату *cur_date*
18. *sum_activity_viewed* - кол-во просмотров активностей на дату *cur_date*
19. *sum_required_activity_viewed* - кол-во просмотров обязательных активностей на дату *cur_date*
20. *sum_attestation_activity_viewed* - кол-во просмотров аттестационных активностей на дату *cur_date*
21. *w_view_hours* - кол-во часов просмотра вебинаров на дату *cur_date*
22. *sum_exercise* - кол-во УСПЕШНО сданных заданий на дату *cur_date*
23. *sum_required_exercises* - кол-во УСПЕШНО обязательных сданных заданий на дату *cur_date*
24. *sum_attestation_exercises* - кол-во УСПЕШНО аттестационных сданных заданий на дату *cur_date*
25. *exercise_delay_first* - среднее отклонение в сроках ПЕРВОЙ сдачи задания на дату *cur_date*
26. *exercise_delay_sum* - суммарное отклонение в сроках ПЕРВОЙ сдачи задания на дату *cur_date*
27. *result_delay_mean* - среднее отклонение в сроках УСПЕШНОЙ сдачи задания на дату *cur_date*
28. *result_delay_sum* - суммарная задержка в сроках УСПЕШНОЙ сдачи задания на дату *cur_date*
29. *sum_exercise_attempts_mean* - среднее кол-во попыток сдачи задания на дату *cur_date*
30. *mean_required_result* - средний результат сдачи обязательных заданий на дату *cur_date*
31. *mean_non_req_result* - средний результат необязательных заданий на дату *cur_date*
32. *mean_attestation_result* - средний результат сдачи аттестационных заданий на дату *cur_date*
33. *sum_result* - суммарный результат выполненных заданий на дату *cur_date*
34. *conn* - кол-во подключений к вебинарам на дату *cur_date*
35. *online_rate* - доля онлайн подключений к вебинарам на дату *cur_date*
36. *mean_result* - средний результат по выполненным заданиям на дату *cur_date*
37. *progress* - прогресс выполнения заданий на дату *cur_date*
38. *cur_date* – дата исследования

Примечание:

При загрузке датасета в pandas следует указывать параметр *index_col=0*.

Прогнозирование показателя результативности обучения на курсах

Код разработан пользователем с телеграмм username @zloy

Прогнозирование показателя результативности обучения на курсах

Исходные данные: выборки, содержащие показатели прохождения курса обучающимися.

1. course_49_lubov_dataset.csv - датасет с данными учеников 49 группы.

Задача: используя модели машинного обучения, спрогнозировать показатель результативности прохождения курса.

	user_id	age	m2_delay_1_week	m2_delay_2_week	m2_delay_3_week	m2_delay_4_week	m2_delay_5_week	m2_delay_6_week	m2_delay_7_week
1	23052	15	-58.458332	-58.458332	-58.458332	-58.458332	-58.458332	-58.458332	-58.458332
2	29079	15	-58.458332	-58.458332	-58.458332	-58.458332	-58.458332	-58.458332	-58.458332

	m2_delay_8_week	m2_delay_9_week	m2_delay_10_week	view_delay_first_1_w...	view_delay_first_2_w...	view_delay_first_3_w...	view_delay_first_4_w...	view_delay_first_5_w...	view_delay_first_6_w...
1	-58.458332	-58.458332	-58.458332	0.0	0.0	0.0	0.0	0.0	0.0
2	-58.458332	-58.458332	-58.458332	0.0	0.0	0.0	0.0	0.0	0.0

	view_delay_first_7_w...	view_delay_first_8_w...	view_delay_first_9_w...	view_delay_first_10_...	view_delay_sum_1_w...	view_delay_sum_2_w...	view_delay_sum_3_w...	view_delay_sum_4_w...	view_delay_sum_5_w...
1	-0.18401042	-0.17836145	-0.17836145	-0.17836145	0.0	0.0	0.0	0.0	0.0
2	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0

	view_delay_sum_6_w...	view_delay_sum_7_w...	view_delay_sum_8_w...	view_delay_sum_9_w...	view_delay_sum_10_...	sum_activity_viewed_...	sum_activity_viewed_...	sum_activity_viewed_...	sum_activity_viewed_...
1	0.0	-15.484445	-24.888681	-24.888681	-24.888681	0.0	0.0	0.0	0.0
2	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0

	sum_activity_viewed_...	sum_activity_viewed_...	sum_activity_viewed_...	sum_activity_viewed_...	sum_activity_viewed_...	sum_activity_viewed_...	sum_required_activity...	sum_required_activity...	sum_required_activity...
1	0.0	0.0	70.0	111.0	111.0	111.0	0.0	0.0	0.0
2	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0

	sum_required_activity...	sum_required_activity...	sum_required_activity...	sum_required_activity...	sum_required_activity...	sum_required_activity...	sum_required_activity...	sum_required_activity...	sum_exercise_1_week
1	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
2	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0

	sum_exercise_2_week	sum_exercise_3_week	sum_exercise_4_week	sum_exercise_5_week	sum_exercise_6_week	sum_exercise_7_week	sum_exercise_8_week	sum_exercise_9_week	sum_exercise_10_week
1	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
2	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0

	sum_required_exerci...	sum_required_exerci...	sum_required_exerci...	sum_required_exerci...	sum_required_exerci...	sum_required_exerci...	sum_required_exerci...	sum_required_exerci...	sum_required_exerci...
1	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
2	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0

	sum_required_exerci...	exercise_delay_first...	exercise_delay_first...	exercise_delay_first...	exercise_delay_first...	exercise_delay_first...	exercise_delay_first...	exercise_delay_first...	exercise_delay_first...
1	0.0	0.0	0.0	0.0	0.0	0.0	0.0	4.7916665	8.827381
2	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0

	exercise_delay_first...	exercise_delay_first...	exercise_delay_sum_...	exercise_delay_sum_...	exercise_delay_sum_...	exercise_delay_sum_...	exercise_delay_sum_...	exercise_delay_sum_...	exercise_delay_sum_...
1	15.827381	22.827381	0.0	0.0	0.0	0.0	0.0	0.0	19.166666
2	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0

	exercise_delay_sum_...	exercise_delay_sum_...	result_delay_sum_1_...	result_delay_sum_2_...	result_delay_sum_3_...	result_delay_sum_4_...	result_delay_sum_5_...	result_delay_sum_6_...	result_delay_sum_7_...
1	110.791664	159.79167	0.0	0.0	0.0	0.0	0.0	0.0	0.0
2	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0

	result_delay_sum_8_...	result_delay_sum_9_...	result_delay_sum_10_...	sum_exercise_attemp...	sum_exercise_attemp...	sum_exercise_attemp...	sum_exercise_attemp...	sum_exercise_attemp...	sum_exercise_attemp...
1	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
2	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0

	sum_exercise_attemp...	sum_exercise_attemp...	sum_exercise_attemp...	sum_exercise_attemp...	sum_result_1_week	sum_result_2_week	sum_result_3_week	sum_result_4_week	sum_result_5_week
1	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
2	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0

	sum_result_6_week	sum_result_7_week	sum_result_8_week	sum_result_9_week	sum_result_10_week	mean_result_1_week	mean_result_2_week	mean_result_3_week	mean_result_4_week
1	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
2	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0

	mean_result_5_week	mean_result_6_week	mean_result_7_week	mean_result_8_week	mean_result_9_week	mean_result_10_week	progress_1_week	progress_2_week	progress_3_week
1	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
2	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0

```

# Формируем стратифицированную обучающую выборку

import pandas as pd

# Предполагаем, что df уже определен

# Шаг 1: Выбор 30 записей из группы course_id=3
group_3_sample = df[df['course_id'] == 3].sample(n=30, replace=True)

# Шаг 2: Размножение выбранных записей до 500
group_3_expanded = group_3_sample.sample(n=500, replace=True)

# Шаг 3: Выбор 15 записей из группы course_id=77
group_77_sample = df[df['course_id'] == 77].sample(n=15, replace=True)

# Шаг 4: Размножение выбранных записей до 500
group_77_expanded = group_77_sample.sample(n=500, replace=True)

# Шаг 5: Получение стратифицированной выборки по 500 образцов из остальных групп
other_groups = df[df['course_id'] != 77]
other_groups = other_groups[other_groups['course_id'] != 3] # Исключаем группу course_id=3

# Получаем уникальные значения course_id для остальных групп
other_course_ids = other_groups['course_id'].unique()

# Проверяем, что у нас есть как минимум две другие группы
if len(other_course_ids) < 2:
    raise ValueError("Ожидается как минимум две другие группы.")

# Шаг 6: Выбор по 500 образцов из каждой другой группы
stratified_samples = []
for course_id in other_course_ids:
    samples = other_groups[other_groups['course_id'] == course_id].sample(n=500, replace=True)
    stratified_samples.append(samples)

# Объединение всех выборок
train_sample = pd.concat([group_3_expanded, group_77_expanded] + stratified_samples)

# Проверка результата
print("\nРазмер итоговой выборки:", len(train_sample))
print("\nКоличество записей из course_id=3:", len(train_sample[train_sample['course_id'] == 3]))
print("\nКоличество записей из course_id=77:", len(train_sample[train_sample['course_id'] == 77]))
print("\nКоличество записей из других групп:")
for course_id in other_course_ids:
    print(f"course_id={course_id}: {len(train_sample[train_sample['course_id'] == course_id])}")

```

Размер итоговой выборки: 2000

Количество записей из course_id=3: 500

Количество записей из course_id=77: 500

Количество записей из других групп:

course_id=71: 500

course_id=49: 500

```
# Создаем валидационную выборку методом исключения из датасета обучающих примеров
val_sample = df.drop(train_sample.index)
```

```
# Проверка результатов
print("Обучающая стратифицированная выборка:")
print(len(train_sample))
print("\nВалидационная выборка:")
print(len(val_sample))
```

Обучающая стратифицированная выборка:
2000

Валидационная выборка:
1880

```
# остатки 77 группы для тестирования
df_71_test = val_sample[val_sample['course_id'] == 77]
df_71_test = val_sample[val_sample['course_id'] == 71]
df_49_test = val_sample[val_sample['course_id'] == 49]
df_3_test = val_sample[val_sample['course_id'] == 3]
```

df_3_test

	user_id	course_id	age	payment	tg_bot	m2_delay	view_delay_first_1_week	view_delay_first_2_week	view_delay_first_3_week
50	31678	3	15	0	1	-58.458332	-4.496157	-2.752473	-2.886936
73	33026	3	19	0	1	-45.245810	-3.194155	-2.963096	-3.352257
238	51059	3	19	0	2	0.725949	266.864600	25.554272	25.554272
341	55218	3	15	1	1	0.671539	0.000000	0.000000	0.000000
494	66184	3	18	0	1	-58.458332	12.727256	12.727256	12.727256
508	66405	3	19	0	1	8.577789	13.070785	12.799987	12.799987
1335	69383	3	19	0	2	-0.379537	0.000000	-4.597754	-4.212048

Выводы:

- в датасете нет пропусков, он хорошо подходит для обучения моделей;
- для прогнозирования целевого признака m2_progress в различные периоды необходимо сформировать датасеты для каждой недели, что позволит исключить "подглядывание в будущее".

1.3. Создание отдельных датасетов для каждой недели

1.3. Создание отдельных датасетов для каждой недели

```
# train_sample.columns.tolist()
```

```
from itertools import chain
```

```
# Определяем названия столбцов для каждой недели
```

```
sum_exercise_week_10 = [f'sum_exercise_{i}_week' for i in range(1, 11)]
sum_required_exercises_week_10 = [f'sum_required_exercises_{i}_week' for i in range(1, 11)]
result_delay_sum_week_10 = [f'result_delay_sum_{i}_week' for i in range(1, 11)]
sum_exercise_attempts_mean_week_10 = [f'sum_exercise_attempts_mean_{i}_week' for i in range(1, 11)]
sum_result_week_10 = [f'sum_result_{i}_week' for i in range(1, 11)]
mean_result_week_10 = [f'mean_result_{i}_week' for i in range(1, 11)]
progress_week_10 = [f'progress_{i}_week' for i in range(1, 11)]

view_delay_first_week_10 = [f'view_delay_first_{i}_week' for i in range(1, 11)]
view_delay_sum_week_10 = [f'view_delay_sum_{i}_week' for i in range(1, 11)]
# sum_activity_viewed_week_10 = [f'sum_activity_viewed_{i}_week' for i in range(1, 11)]
# sum_required_activity_viewed_week_10 = [f'sum_required_activity_viewed_{i}_week' for i in range(1, 11)]
# exercise_delay_first_week_10 = [f'exercise_delay_first_{i}_week' for i in range(1, 11)]
exercise_delay_sum_week_10 = [f'exercise_delay_sum_{i}_week' for i in range(1, 11)]
```

```
# Уменьшаем количество элементов в списке для каждой последующей недели
```

```
def generate_weekly_columns(base_list, weeks):
    return [base_list[:i] for i in range(1, weeks + 1)]
```

```
# Генерируем названия столбцов по неделям
```

```
sum_exercise_weeks = generate_weekly_columns(sum_exercise_week_10, 10)
sum_required_exercises_weeks = generate_weekly_columns(sum_required_exercises_week_10, 10)
result_delay_sum_weeks = generate_weekly_columns(result_delay_sum_week_10, 10)
sum_exercise_attempts_mean_weeks = generate_weekly_columns(sum_exercise_attempts_mean_week_10, 10)
sum_result_weeks = generate_weekly_columns(sum_result_week_10, 10)
mean_result_weeks = generate_weekly_columns(mean_result_week_10, 10)
progress_weeks = generate_weekly_columns(progress_week_10, 10)

view_delay_first_weeks = generate_weekly_columns(view_delay_first_week_10, 10)
view_delay_sum_weeks = generate_weekly_columns(view_delay_sum_week_10, 10)
# sum_activity_viewed_weeks = generate_weekly_columns(sum_activity_viewed_week_10, 10)
# sum_required_activity_viewed_weeks = generate_weekly_columns(sum_required_activity_viewed_week_10, 10)
# exercise_delay_first_weeks = generate_weekly_columns(exercise_delay_first_week_10, 10)
exercise_delay_sum_weeks = generate_weekly_columns(exercise_delay_sum_week_10, 10)
```



```
# Формируем столбцы для каждой недели
columns = []
for i in range(10):
    columns.append(list(chain(
        ['user_id', 'age', 'payment', 'tg_bot', 'm2_delay'],
        view_delay_first_weeks[i],
        view_delay_sum_weeks[i],
        sum_exercise_weeks[i],
        sum_required_exercises_weeks[i],
        # exercise_delay_first_weeks[i],
        result_delay_sum_weeks[i],
        sum_exercise_attempts_mean_weeks[i],
        sum_result_weeks[i],
        mean_result_weeks[i],
        progress_weeks[i],
        # view_delay_sum_weeks[i],
        # sum_activity_viewed_weeks[i],
        # sum_required_activity_viewed_weeks[i],
        exercise_delay_sum_weeks[i],
        ['m2_progress']
    )))

# Теперь на позиции `columns[0]` у нас находятся столбцы для 1-й недели,
# на `columns[1]` для 2-й и так далее до 10-й недели.
```

```
Week 1 - Mean Absolute Error: 7.9924794391680525
Week 2 - Mean Absolute Error: 6.924550653267929
Week 3 - Mean Absolute Error: 6.700902867699872
Week 4 - Mean Absolute Error: 6.300852912703257
Week 5 - Mean Absolute Error: 5.999639314988462
Week 6 - Mean Absolute Error: 5.704828925945423
Week 7 - Mean Absolute Error: 4.9113169553779565
Week 8 - Mean Absolute Error: 3.818703643099225
Week 9 - Mean Absolute Error: 2.934018585007375
Week 10 - Mean Absolute Error: 2.8606435373430092
```

rf_models

```
# Сериализация моделей!!!

import joblib # Импортируем библиотеку для сериализации
import os

# Укажите директорию для сохранения моделей
model_dir = '/home/shared_notebooks/zloy/saved_models'












# Создаем директорию, если она не существует
os.makedirs(model_dir, exist_ok=True)

{'rf_model_week_1': RandomForestRegressor(max_features='log2', n_estimators=50),
 'rf_model_week_2': RandomForestRegressor(max_features='log2', n_estimators=50),
 'rf_model_week_3': RandomForestRegressor(max_features='log2', n_estimators=50),
 'rf_model_week_4': RandomForestRegressor(max_depth=30, max_features='log2', n_estimators=200),
 'rf_model_week_5': RandomForestRegressor(max_features='log2', n_estimators=50),
 'rf_model_week_6': RandomForestRegressor(max_features='log2', n_estimators=200),
 'rf_model_week_7': RandomForestRegressor(max_features='sqrt', n_estimators=200),
 'rf_model_week_8': RandomForestRegressor(max_features='sqrt', n_estimators=200),
 'rf_model_week_9': RandomForestRegressor(max_features='sqrt', n_estimators=200),
 'rf_model_week_10': RandomForestRegressor(max_features='sqrt', n_estimators=200)}

# Сериализация моделей на диск
for model_name, model in rf_models.items():
    model_filename = os.path.join(model_dir, f'{model_name}.joblib') # Создаем имя файла для модели
    joblib.dump(model, model_filename) # Сохраняем модель в файл
    print(f"Модель '{model_name}' успешно сохранена в '{model_filename}'")

print(f"\nВсе модели успешно сохранены в директорию: {model_dir}")
```

В результате получены модели

 rf_model_week_1.joblib	8 минут назад
 rf_model_week_10.joblib	8 минут назад
 rf_model_week_2.joblib	8 минут назад
 rf_model_week_3.joblib	8 минут назад
 rf_model_week_4.joblib	8 минут назад
 rf_model_week_5.joblib	8 минут назад
 rf_model_week_6.joblib	8 минут назад
 rf_model_week_7.joblib	8 минут назад
 rf_model_week_8.joblib	8 минут назад
 rf_model_week_9.joblib	8 минут назад
 Untitled.ipynb	24 минуты назад

На данный момент идет работа и проверка гипотез

Рассматриваемые гипотезы



Илья Трубников
17:32 27 окт.

Первые три гипотезы нравятся: они выглядят прямолинейными в хорошем смысле и довольно твердыми.. Единственное, есть смысл, наверное, у Карины уточнить, не трекается ли уже что-то из этого.

Гипотеза о прогрессе:

- **Гипотеза:** Студенты, которые не успевают завершить определенный процент task (тем) и activities (заданий) модуля, включая просмотр учебных материалов, к дате среза, с большей вероятностью бросят его.
- **Основание:** Недостижение определенных вех (промежуточные дедлайны) может указывать на проблемы с дисциплиной или трудности с усвоением материала.
- **Метрики для проверки:** процент завершенных activities к дате среза, процент выполненных activities, обязательных к проверке, процент выполненных необязательных activities, индекс успешности выполнения заданий, средний балл удачных попыток сдачи, средний балл всех попыток сдачи заданий.

Гипотеза о регулярности обучения:

- **Гипотеза:** Студенты, которые имеют нерегулярное расписание обучения (большие перерывы между сессиями на платформе), с большей вероятностью бросят курс.
 - **Основание:** Нерегулярная активность может указывать на отсутствие дисциплины или сниженный интерес к обучению.
 - **Метрики для проверки:** среднее время между логинами, количество завершенных activities в неделю

Гипотеза о выполнении работ вовремя:

- **Гипотеза:** Студенты, которые часто пропускают сроки сдачи работ - сделанное задание по task до открытия нового task, с большей вероятностью бросят курс.
 - **Основание:** Пропуск сроков сдачи часто является индикатором недостатка времени, организации или мотивации.
 - **Метрики для проверки:** количество пропущенных сроков сдачи, среднее время сдачи заданий относительно срока сдачи.

Гипотеза о ранних успехах:

- **Гипотеза:** Студенты, которые успешно выполняют первые задания и тесты, с большей вероятностью завершат курс.
 - **Основание:** Успехи на раннем этапе могут укрепить уверенность студента и мотивацию для дальнейшего обучения.
 - **Метрики для проверки:** результаты первых нескольких activities, скорость выполнения заданий.



Илья Трубников
17:31 27 окт.

Мне очень нравятся все гипотезы, но вот эта вызывает переживание. Дело в том, что поначалу большинство людей как раз-таки очень активны. Они вдохновлены, они видят будущие перспективы. Но со временем вдохновение и мотивация сменяются рутинной, и человеку уже нужна дисциплина -- а с этим как раз у большинства проблемы.

В то же время мне кажется, что эта интуиция у вас в правильном направлении -- нужно просто перевернуть оптику. Я бы сказал, что

! отсутствие ранних успехов увеличивает вероятность, что студент бросит. !

Здесь как раз-таки понятный механизм работает: втягиваться сложно, когда отстаешь. Начинаешь оправдывать себя и бессознательно обесценивать происходящее. А тут человек отстаёт прямо сходу: то есть он ценность особо увидеть и не успел, поэтому и обесценить проще. Вот в это я уже верю сильно.