Построение аналитики для корпоративных клиентов IT Resume

Цель и метрики анализа

Целью анализа является выявление узких мест в программе обучения, вызывающих наибольшее затруднение у корпоративных студентов и определение точек роста.
В ходе анализа будет проверены гипотеза:

- самостоятельно обучающиеся студенты более мотивированы.

Для решение данной задачи были использованы следующие метрики:

- степень удовлетворенности обучением,
- rolling retention,
- длительность решения задач

Анализ проводился для компании N (company_id = 1)

Сравнительный анализ обучения корпоративных студентов компании N с обучением остальных студентов

Сравнение метрик обучения студентов компании N с остальными студентами (удовлетворенность и rolling retention)

Прежде чем приступить к сравнению метрик подготовим дневник обучения, таблицу прогресса и представим основные модули задач компании N.
Код запроса SQL для дневника студента

```
select
	coalesce(u.first_name || ' ' || u.last_name,
	'Unknown') as full_name,
	p.name,
	max(case when c.is_false = 0 then 1 when c.is_false = 1 then 0 end) is_done
from
	codesubmit c
join problem p
on
	c.problem_id = p.id
join users u
on
```

```
        c.user_id = u.id
join problem_to_company ptc
on
        u.company_id = ptc.company_id
        and p.id = ptc.problem_id
where
        u.company_id = 1
group by
        u.id,
        p.name
```

Код запроса SQL прогресса обучения студентов

```
with
 users_company as (
--сначала установим количество попыток решения задач
select
        u.id as user_id,
        u.date_joined::date as dt,
        coalesce(u.first_name || ' ' || u.last_name,
        'Unknown') as full_name,
        u.email,
        c."name" as company_name,
        p.id as problem_id
from
        users u
join company c
  on
        u.company_id = c.id
join codesubmit c2
  on
        c2.user_id = u.id
join coderun c3
  on
        c2.user_id = c3.user_id
        and c2.problem_id = c3.problem_id
join problem p
  on
        c2.problem_id = p.id
join problem_to_company ptc on
        p.id = ptc.problem_id
        and u.company_id = ptc.company_id
where
        c.id = 1
),
 num_attemps as(
select
        distinct
                user_id,
        full_name,
        email,
        dt,
```

```
        COUNT(problem_id) over(partition by user_id,
        problem_id) as cnt_attemps,
        company_name
from
        users_company
 ),
 all_success as(
select
        distinct
        u.id as user_id,
        p.id as problem_id,
        u.company_id
from
        users u
join codesubmit c
  on
        c.user_id = u.id
join problem p
  on
        c.problem_id = p.id
join problem_to_company ptc
  on
        p.id = ptc.problem_id
        and u.company_id = ptc.company_id
where
        u.company_id = 1
        and c.is_false = 0
 ),
 all_problems as(
select
        distinct
                ptc.company_id,
        count(ptc.problem_id) over (partition by ptc.company_id) as total_problems
from
        problem_to_company ptc
where
        ptc.company_id = 1
        or ptc.company_id = 7
 ),
 num_success as(
select
        distinct
                user_id,
        problem_id,
        company_id,
        COUNT(problem_id) over(partition by user_id) as cnt_success
from
        all_success
),
total_actions as(
select
        na.user_id,
```

```
        full_name,
        email,
        dt,
        cnt_success as num_success,
        company_name,
        total_problems,
        sum(cnt_attemps) over(partition by na.user_id,
        problem_id) as num_attemps
from
        num_attemps na
join num_success ns
        on
        na.user_id = ns.user_id
join all_problems ap
        on
        ns.company_id = ap.company_id
),
grouped as(
select
        full_name,
        email,
        dt,
        num_attemps,
        num_success,
        total_problems
from
        total_actions
group by
        full_name,
        email,
        dt,
        num_attemps,
        num_success,
        total_problems
)
select
        *,
        round(num_success / total_problems::numeric,
        2) as progress
from
        grouped
order by
        dt
```

По каким модулям подготовлены задачи для компании N

Код запроса SQL

```
with full_name_problem as (
select
                case
                        when p.name like 'NEW! %' then replace(p.name,
                        'NEW! ',
```

```
                        ")
                else p.name
        end problem
from
                problem_to_company ptc
join problem p
        on
                ptc.problem_id = p.id
where
                ptc.company_id = 1
)
select
        split_part(problem,
        '.',
        1) as modul,
        count(*) as cnt_prblems
from
        full_name_problem
group by
        split_part(problem,
        '.',
        1)
```

Из круговой диаграммы видно, что большинство задач относится к модулю Python. Далее посмотрим удовлетворенность студентов обучением (какой процент решенных задач) и сравним этот показатель с аналогичным показателем студентов, не относящихся к компании N. Сравнивать будем по одним и тем же модулям.
Запрос SQL

```
select
        cast(count(case when c.is_false = 0 then 1 end) as numeric)/ count(*) as share
from
        codesubmit c
join problem p
on
        c.problem_id = p.id
join users u
on
        c.user_id = u.id
join problem_to_company ptc
on
        u.company_id = ptc.company_id
        and p.id = ptc.problem_id
where
        u.company_id = 1
```

А теперь такой же показатель для остальных студентов, напомним, что показатель необходимо отразить по тем же модулям, по которым занимаются студенты компании N
Код запроса SQL

```
with full_name_problem as (
select
	p.id as problem_id,
		case
			when p.name like 'NEW! %' then replace(p.name,
			'NEW! ',
			'')
		else p.name
	end problem
from
	problem p
),
moduls as(
select
	problem_id,
	split_part(problem,
	'.',
	1) as modul
from
	full_name_problem
)
select
	cast(count(case when c.is_false = 0 then 1 end) as numeric)/ count(*) as share
from
	codesubmit c
join moduls m
on
	c.problem_id = m.problem_id
join users u
on
	c.user_id = u.id
where
	u.company_id != 1
	and (modul = 'SQL'
		or modul = 'Pythom'
		or modul = 'Numpy')
```

Сравнив два графика видим, что, хотя удовлетворенность обучением остальных студентов несколько выше, при отборе из данного сегмента студентов, обучающихся самостоятельно, уровень удовлетворенности еще больше увеличился, разрыв составил 0.06.

Прежде чем вывести заключительную метрику - rolling retention, - посмотрим динамику притока студентов компании N.

Код запроса SQL

```
select
	to_char(date_joined,
	'YYYY/mm'),
	count(case when u.is_active = 1 then 1 end) as active_students
from
	users u
where
```

```
        u.company_id = 1
group by
        to_char(date_joined,
        'YYYY/mm')
order by
        to_char(date_joined,
        'YYYY/mm')
```

Из графика видим, что основном приток приходится на ноябрь 2021г., дальше он постепенно уменьшался вплоть до периода апрель 2022г., общий период составил 5 месяцев.
Построим аналогичный график для остальных студентов.
Здесь код запроса не приводим, т.к. он такой же, только меняется условие на

```
where
        coalesce(users.company_id, 0) !=1
```

Из графика видно, что хотя и основной приток остальных студентов приходится на период февраль 2022г., в ноябре 2021г. количество студентов достаточное для сопоставления rolling retention.
Rolling retention будем смотреть за этот период с разбивкой на периоды:

- 1 день,
- 3 дня,
- 7 дней,
- 2 недели,
- месяц,
- 2 месяца,
- 3 месяца,
- 4 месяца,
- 5 месяцев

Код запроса SQL

```
with rolling_retentions as(
select
        u2.user_id,
        u.date_joined as date_joined,
        u2.entry_at as entry_at,
        extract(days
from
        u2.entry_at - u.date_joined) as diff,
        to_char(u.date_joined::date,
        'YYYY-MM') as cohort
from
        users u
join userentry u2
on
        u.id = u2.user_id
where
        u.date_joined::date > '2021-10-31'
        and u.company_id = 1
)
```

```
select
	cohort,
	round(count(distinct case when diff >= 0 then user_id end) * 100.0 / count(distinct case when diff >= 0 then user_id end),
	2) as "0 (%)",
	round(count(distinct case when diff >= 1 then user_id end) * 100.0 / count(distinct case when diff >= 0 then user_id end),
	2) as "1 (%)",
	round(count(distinct case when diff >= 3 then user_id end) * 100.0 / count(distinct case when diff >= 0 then user_id end),
	2) as "3 (%)",
	round(count(distinct case when diff >= 7 then user_id end) * 100.0 / count(distinct case when diff >= 0 then user_id end),
	2) as "7 (%)",
	round(count(distinct case when diff >= 14 then user_id end) * 100.0 / count(distinct case when diff >= 0 then user_id end),
	2) as "14 (%)",
	round(count(distinct case when diff >= 30 then user_id end) * 100.0 / count(distinct case when diff >= 0 then user_id end),
	2) as "30 (%)",
	round(count(distinct case when diff >= 60 then user_id end) * 100.0 / count(distinct case when diff >= 0 then user_id end),
	2) as "60 (%)",
	round(count(distinct case when diff >= 90 then user_id end) * 100.0 / count(distinct case when diff >= 0 then user_id end),
	2) as "90 (%)",
	round(count(distinct case when diff >= 120 then user_id end) * 100.0 / count(distinct case when diff >= 0 then user_id end),
	2) as "120 (%)",
	round(count(distinct case when diff >= 150 then user_id end) * 100.0 / count(distinct case when diff >= 0 then user_id end),
	2) as "150 (%)"
from
	rolling_retentions
group by
	cohort
order by
	cohort
```

Код запроса SQL для остальных студентов такой же, с поправкой на условие по id компании.

Видим, что студенты компании N активнее заходят на платформу, при этом удовлетворенность обучением у некорпоративных студентов выше.

Сравнительный анализ метрик верхнего уровня студентов компании N и остальных студентов показал, что, в целом, метрики схожи, однако, студенты компании N чуть больше вовлечены в процесс.

Далее посмотрим, как тщательно решали задачи студенты компании N по сравнению с остальными студентами.

Сравнительный анализ попыток решения задач корпоративными студентами и студентами, не входящими в состав компании

Здесь сделаем оговорку, что при решении задачи сразу были созданы несколько таблиц для вывода результатов. Поскольку использовался CTE, а не подзапросы, ниже приведен полный код формирования таблиц для решения задач, далее по каждому промежуточному выводу будет приводится только результирующая таблица.
Код глобального запроса SQL

```
with
 attempts_company as (
--сначала установим количество попыток решения задач по модулям
select
	u.date_joined::date,
	p.name as name_problem,
	c.user_id as user_id,
	coalesce(u.first_name || ' ' || u.last_name,
	'Unknown') as full_name
from
	codesubmit c
join users u on
	c.user_id = u.id
join problem p on
	c.problem_id = p.id
join problem_to_company ptc on
	p.id = ptc.problem_id
	and u.company_id = ptc.company_id
where
	u.company_id = 1
 ),
 attempts_others as (
select
	u.date_joined::date,
	p.name as name_problem,
	c.user_id as user_id,
	coalesce(u.first_name || ' ' || u.last_name,
	'Unknown') as full_name
from
	codesubmit c
join users u on
	c.user_id = u.id
join problem p on
	c.problem_id = p.id
join problem_to_company ptc on
	p.id = ptc.problem_id
	and u.company_id = ptc.company_id
where
	u.company_id != 1
 ),
 grouped_by_date as (
select
```

```
	user_id,
	case
		when name_problem like 'NEW! %' then replace(name_problem,
		'NEW! ',
		'')
		else name_problem
	end name_prb
from
	attempts_company
group by
	user_id,
	name_prb
 ),
--количество попыток решенных задач
moduls_company as (
select
	--date_joined,
	split_part(name_prb,
	'.',
	1) as modul,
	count(*) as cnt_attempts_company,
	count(distinct user_id) as cnt_users_company
from
	grouped_by_date
group by
	modul
 ),
--количество попыток решенных задач с детализацией по студентам
grouped_by_date_and_users as (
select
	user_id,
	full_name,
	date_joined,
	case
		when name_problem like 'NEW! %' then replace(name_problem,
		'NEW! ',
		'')
		else name_problem
	end name_prb
from
	attempts_company
group by
	user_id,
	full_name,
	date_joined,
	name_prb
 ),
 moduls_by_student as (
select
	user_id,
	full_name,
	date_joined,
```

```
        split_part(name_prb,
        '.',
        1) as modul,
        count(*)
from
        grouped_by_date_and_users
group by
        user_id,
        full_name,
        date_joined,
        modul
 ),
 grouped_by_problems_others as (
select
        user_id,
        case
                when name_problem like 'NEW! %' then replace(name_problem,
                'NEW! ',
                '')
                else name_problem
        end name_prb
from
        attempts_others
group by
        user_id,
        name_prb
 ),
--количество попыток решенных задач остальными студентами
moduls_others as (
select
        split_part(name_prb,
        '.',
        1) as modul,
        count(name_prb) as cnt_attempts_other,
        count(distinct user_id) as cnt_users_others
from
        grouped_by_problems_others
group by
        modul
 ),
--количество попыток решенных задач студентами компании N и остальными студентами по
модулям
moduls_all as (
select
        mc.modul,
        mc.cnt_attempts_company,
        mo.cnt_attempts_other,
        mc.cnt_users_company,
        mo.cnt_users_others,
        round(
    cnt_attempts_company * 100.0 / (cnt_attempts_company + cnt_attempts_other) / 100,
        2
```

```
    ) as share_attempts_company,
        round(
      cnt_attempts_other * 100.0 / (cnt_attempts_company + cnt_attempts_other) / 100,
        2
    ) as share_attempts_other,
        round(
      cnt_users_company * 100.0 / (cnt_users_company + cnt_users_others) / 100,
        2
    ) as share_users_company,
        round(
      cnt_users_others * 100.0 / (cnt_users_company + cnt_users_others) / 100,
        2
    ) as share_users_others
from
        moduls_company mc
left join moduls_others mo on
        mc.modul = mo.modul
 )
select
        *
from
        moduls_all
```

Соотношение сегментов студентов (студенты компании N и остальные студенты).
Заметим, что здесь приведено соотношение студентов, решавших одни и те же задачи
Код финального запроса SQL из глобального запроса

```
select
        modul,
        cnt_users_company,
        cnt_users_others
from
        moduls_all
```

Определим долю попыток решения задач в разрезе модулей среди студентов компании N
Финальный запрос SQL для студентов компании N из глобального запроса

```
select
        modul,
        cnt_attempts_company
from
        moduls_all
```

и для остальных студентов

```
select
        modul,
        cnt_attempts_other
from
        moduls_all
```

Сравнив с круговой диаграммой количества задач, видим что, хотя задач по Python больше, чем задач по SQL, сложность у студентов вызвали именно задачи по SQL.

Опять же у обоих сегментов студентов, но у сегмента студентов компании N задачи модуля SQL вызвали даже несколько больше сложностей, чем у студентов компании N.

Сравнительный анализ выполнения задач

Далее нашей промежуточной целью была цель установить самые долго решаемые задачи.
При решении задачи в лоб выяснилось, что некоторые задачи решались более ста дней, что выглядит неправдоподобно, при выборочном анализе кода, отправленного на проверку, выяснилось, что код не менялся, просто студент почему-то решил снова нажать на кнопку Отправить через несколько месяцев.
К сожалению, объема памяти приложения DBeaver не хватило, чтобы программно исключить такие записи из глобального запроса SQL по количеству попыток решения задач, поэтому пришлось анализировать код по каждому такому студенту визуально и фильтр устанавливать непосредственно при формировании таблиц для вычисления duration.
Ниже представлена таблица таких пользователей и задач (длительность между первой и последней попытками не менее 45 дней), сразу же сделано сравнение текущего и предыдущего кода а также сравнение кода студента с эталонным кодом.
Код запроса SQL

```
with problems as(
	select
		u.id as user_id,
		p.id as problem_id,
		p.name as name,
		p.company_id,
		c.created_at,
		replace(p.solution, '<inject-highlight>', '') as solution,
		c.code,
		c.is_false
	from codesubmit c
	join users u
	on c.user_id = u.id
	join problem p
	on c.problem_id = p.id
	join problem_to_company ptc
	on p.id = ptc.problem_id and u.company_id = ptc.company_id
	where u.company_id = 1 --and u.id = 63 --and (ptc.problem_id = 117 or ptc.problem_id = 123)
),
min_max_duration as(
	select
		user_id,
		created_at,
		problem_id,
		name,
		code,
		is_false,
		replace(solution, '</inject-highlight>', '') as solution,
		min(created_at) over (partition by user_id, problem_id) as min_duration,
```

```
                max(created_at) over (partition by user_id, problem_id) as max_duration
        from problems
),
most_long_problem as(
        --при определении самых долго решаемых задач в лоб выяснилось, что некоторые задачи решались более ста дней, что выглядит неправдоподобно,
        --при выборочном анализе кода, отправленного на проверку, выяснилось, что код не менялся, просто студент почему-то решил снова нажать на кнопку Отправить через несколько месяцев.
        --К сожалению, объема памяти приложения DBeaver не хватило, чтобы программно исключить такие записи, поэтому пришлось анализировать код по каждому такому студенту визуально
        --и фильтр устанавливать непосредственно при формировании таблиц для вычисления duration
        --ниже представлена таблица таких пользователей и задач,
        --а визуальное сравнение кодов отправленных последними с предыдущим позволило установить записи, в которых код не менялся, с течением достаточно долгого времени:
        --user_id 50, problem_id - 117,
        --user_id 263, problem_id - 117,
        --user_id 861, problem_id - 133,
        --user_id 1107, problem_id - 134,
        --user_id 348, problem_id - 48,
        select
                user_id,
                problem_id,
                extract(day from max_duration - min_duration) as duration
        from min_max_duration
        where extract(day from max_duration - min_duration) > 30
        group by user_id, problem_id, duration
        order by duration desc
),
duration as(
        select
                user_id,
                problem_id,
                created_at,
                name,
                code,
                is_false,
                solution,
                extract(day from max_duration - min_duration) as duration
        from min_max_duration
),
change_code as(
        select *,
                lag(code) over (partition by user_id, problem_id order by user_id, duration desc, created_at) as code_old,
                row_number() over(partition by user_id, problem_id order by user_id, created_at) as num_row
        from duration
)
select *,
```

```
        case
                when code_old = code then 'no'
                else 'yes'
        end code_changed,
        case
                when solution = code then 'yes'
                else 'no'
        end as_etalon,
        max(num_row) over (partition by user_id, problem_id) as num_attempts
from change_code
where duration > 45
```

Собственно, детальное сравнение последующего и предыдущего кода, а также сравнение кода с эталонным и натолкнуло на гипотезу, насколько студенты компании N мотивированы получить именно знания, а не просто отчитаться перед работодателем о прохождении курса.
Итак, возьмем студента из самой первой строки (самое долгое решение задачи), заодно посмотрим и задачу с id 117, эту задачу студент решал долго и посмотрим, изменился ли код.
Удивительно, на первой, самой долго решаемой задаче было целых 6 попыток, но код ни разу не изменился по сравнению с изначальным, видимо, студент не видел прогресс, важный для него, и в отчаянии нажимал на кнопку. Со второй долго решаемой им задачей еще интереснее (id 123), код, действительно изменился, но он стал эталонным, т.е. студент в итоге не решил задачу, а просто вставил копию эталонной задачи.
Таким образом, визуальное сравнение предыдущего и последующего кода выявило студентов компании N, которые через несколько месяцев/недель, не меняли код, а просто снова нажимали на кнопку Отправить

user_id	problem_id	duration_total	duration_lag
50	117	126	125
263	117	124	124
861	133	79	63
1107	134	65	53
348	134	48	48

Теперь посмотрим, ведут ли себя схожим образом остальные студенты? В результирующей таблице только один студент (user_id = 63).
Посмотрим, менял ли он свой код. Сравнив коды видим, только в одной задаче (problem_id = 121) код не изменен по сравнению с предыдущей записью, в остальные случаях, а количество попыток иногда было больше 10, код менялся, причем, он становился лучше. Необходимо также отметить, что эту задачу студент смог решить только скопировав решение. В остальных случаях, если он и копировал решение из эталона, но, затем возвращался к задаче и старался решить её самостоятельно.

Выведем теперь пузырьковую диаграмму длительности решения задач. Запрос SQL использовался прежний, только убран фильтр по длительности попыток и добавлена сортировка по убыванию.
Пять самый долгих решений не учитываем, т.к. студенты не меняли код, только нажимали кнопку Отправить. Видно, что большинство задач решалось в течение 3 дней, среди задач, вызвавших сложности, нельзя выделить какую-то одну задачу, у разных студентов сложности вызывали разные задачи.

Выводы

Как показал сравнительный анализ, в целом, поведение студентов компании N не отличается от поведения остальных студентов, студенты больше попыток совершают при решении задач модуля SQL, студенты компании N чуть чаще заходят на платформу.
Однако, более детальный анализ аномальных выбросов решения задач, которые некоторые студенты решали достаточно долго (несколько месяцев) выявил случаи, когда студенты не меняли код, а просто нажимали кнопку Отправить и даже отправляли на проверку эталонное решение, а не свой код.
Предложения для повышения мотивации студентов решать задачи собственными силами и стараться совершенствовать свой код.

1. Ввести рейтинговые очки, которые увеличиваются при отправке верного решения ДО просмотра эталонного решения и уменьшаются при просмотре эталонного решения до отправки первого своего решения. Причем, сам размер бонуса зависит от попыток решить задачу правильно самостоятельно. Эти рейтинговые очки отразить как kpi при отчете перед компанией
2. Настроить контроль решения студента, чтобы исключить копирование эталонного решения, ввести большой виртуальный штраф за подобные действия.

Также предложение по улучшению структуры базы - добавить данные по модулю в таблицы с задачами, что позволит сравнивать поведение корпоративных студентов и студентов, обучающихся самостоятельно, при решении задач одного и того же модуля, для понимания мотивации корпоративных студентов.
То, что мотивации могут различаться можно предположить хотя бы из-за того, что удовлетворенность обучением у самостоятельно обучающихся студентов выше, чем у студентов, проходящих обучение от компаний.
На далекую перспективу можно использовать ИИ для определения авторства кода, отправляемого на проверку, был ли код написан самостоятельно или просто переделан из эталонного решения.