**DILAN’S TRAVEL GUIDE ANALYSIS MEGOLDÁS**

**(BORSI-NAGY GÁBOR)**

A feladatot több környezetben valósítottam meg. A kezdeti lépéseket bash-ben végeztem el. Az adatokat elsősorban SQL-ben elemeztem, Pythonban csak a prediction rész készült. A vizualizáció Power BI-ban és Looker Studio-ban készült, a prediction Pythonban. Ehhez Power BI-ban is fel kellett építenem a projektet, illetve megírni a megfelelő measure-okat hozzá.

**ELSŐ LÉPÉSEK BASH-BEN**

Letöltöttem az adatot egy külön könyvtárba wget-el.

Date és time különválasztása:

sed 's/ /;/1' dilans-data.csv > dilans—data.csv

Külön fájlokba szétválasztottam a tartalmakat event\_type alapján:

grep 'read' dilans--data.csv > dilans-read.csv

grep 'buy' dilans--data.csv > dilans-buy.csv

grep 'subscribe' dilans--data.csv > dilans-subscribe.csv

Read file-ban ha egy oszloppal kevesebb van (visszatérő olvasó), akkor beszúr egy oszlopot:

#!/bin/bash

input\_file="dilans-read.csv"

output\_file="dilans--read.csv"

awk -F';' '

BEGIN { OFS=";" }

{

if (NF == 6) {

$6 = "Returning;" $6

}

print $1, $2, $3, $4, $5, $6, $7

}' "$input\_file" > "$output\_file"

Eltávolítja a végéről a ;-t, ha van:

sed 's/;$//' dilans--read.csv > dilans\_read.csv

**SQL kódok**

Létrehoztam a 3 event\_type táblázatot SQL-ben.

Bemásoltam az adatokat a szerverről:

COPY dilans\_subscribe FROM '/home/pivotpanda/dilan/dilans\_\_subscribe.csv' DELIMITER ';';

Összesítő adatok elkészítése:

SELECT EXTRACT(MONTH FROM r.my\_date) AS month,

COUNT(DISTINCT r.user\_id) AS reader,

COUNT(DISTINCT re\_read.user\_id) AS returning\_reader,

COUNT(DISTINCT d.user\_id) AS subscribe,

COUNT(DISTINCT b.user\_id) AS buyer,

COUNT(DISTINCT re\_buy.user\_id) AS returning\_buyer,

COUNT(DISTINCT r.user\_id) - COUNT(DISTINCT re\_read.user\_id) AS churn\_after\_read,

COUNT(DISTINCT re\_read.user\_id) - COUNT(DISTINCT d.user\_id) AS churn\_after\_re\_read,

COUNT(DISTINCT d.user\_id) - COUNT(DISTINCT b.user\_id) AS churn\_after\_subscribe,

COUNT(DISTINCT b.user\_id) - COUNT(DISTINCT re\_buy.user\_id) AS churn\_after\_buy

FROM dilans\_read r

LEFT JOIN dilans\_subscribe d ON r.user\_id = d.user\_id

LEFT JOIN dilans\_buy b ON r.user\_id = b.user\_id

LEFT JOIN

(SELECT user\_id

FROM dilans\_read

GROUP BY user\_id

HAVING COUNT(user\_id)>1) AS re\_read

ON r.user\_id = re\_read.user\_id

LEFT JOIN

(SELECT user\_id

FROM dilans\_buy

GROUP BY user\_id

HAVING COUNT(user\_id)>1) AS re\_buy

ON r.user\_id = re\_buy.user\_id

GROUP BY EXTRACT(MONTH FROM r.my\_date);

Ugyanez részletesebben (és ennek a különböző verziói, más-más lebontásokban, pl. napi, heti ):

SELECT r.my\_date, EXTRACT(MONTH FROM r.my\_date) AS month,

topic, country, source,

COUNT(DISTINCT r.user\_id) AS reader,

COUNT(DISTINCT re\_read.user\_id) AS returning\_reader,

COUNT(DISTINCT d.user\_id) AS subscribe,

COUNT(DISTINCT b.user\_id) AS buyer,

COUNT(DISTINCT re\_buy.user\_id) AS returning\_buyer,

COUNT(DISTINCT r.user\_id) - COUNT(DISTINCT re\_read.user\_id) AS churn\_after\_read,

COUNT(DISTINCT re\_read.user\_id) - COUNT(DISTINCT d.user\_id) AS churn\_after\_re\_read,

COUNT(DISTINCT d.user\_id) - COUNT(DISTINCT b.user\_id) AS churn\_after\_subscribe,

COUNT(DISTINCT b.user\_id) - COUNT(DISTINCT re\_buy.user\_id) AS churn\_after\_buy,

SUM(b.price) AS revenue

FROM dilans\_read r

LEFT JOIN dilans\_subscribe d ON r.user\_id = d.user\_id

LEFT JOIN dilans\_buy b ON r.user\_id = b.user\_id

LEFT JOIN

(SELECT user\_id

FROM dilans\_read

GROUP BY user\_id

HAVING COUNT(user\_id)>1) AS re\_read

ON r.user\_id = re\_read.user\_id

LEFT JOIN

(SELECT user\_id

FROM dilans\_buy

GROUP BY user\_id

HAVING COUNT(user\_id)>1) AS re\_buy

ON r.user\_id = re\_buy.user\_id

GROUP BY r.my\_date, EXTRACT(MONTH FROM r.my\_date), topic, country, source;

SELECT

COUNT(DISTINCT r.user\_id) AS reader,

COUNT(DISTINCT re\_read.user\_id) AS returning\_reader,

COUNT(DISTINCT d.user\_id) AS subscribe,

COUNT(DISTINCT b.user\_id) AS buyer,

COUNT(DISTINCT re\_buy.user\_id) AS returning\_buyer,

COUNT(DISTINCT r.user\_id) - COUNT(DISTINCT re\_read.user\_id) AS churn\_after\_read,

COUNT(DISTINCT re\_read.user\_id) - COUNT(DISTINCT d.user\_id) AS churn\_after\_re\_read,

COUNT(DISTINCT d.user\_id) - COUNT(DISTINCT b.user\_id) AS churn\_after\_subscribe,

COUNT(DISTINCT b.user\_id) - COUNT(DISTINCT re\_buy.user\_id) AS churn\_after\_buy

FROM dilans\_read r

LEFT JOIN dilans\_subscribe d ON r.user\_id = d.user\_id

LEFT JOIN dilans\_buy b ON r.user\_id = b.user\_id

LEFT JOIN

(SELECT user\_id

FROM dilans\_read

GROUP BY user\_id

HAVING COUNT(user\_id)>1) AS re\_read

ON r.user\_id = re\_read.user\_id

LEFT JOIN

(SELECT user\_id

FROM dilans\_buy

GROUP BY user\_id

HAVING COUNT(user\_id)>1) AS re\_buy

ON r.user\_id = re\_buy.user\_id;

Bevétel és profit számítások - havi (és ezek variációi, napit nem is írom):

WITH user\_first\_read AS (

SELECT

user\_id,

country,

source,

MIN(my\_date) AS first\_read\_date

FROM

dilans\_read

WHERE source != 'Direct'

GROUP BY

user\_id, country, source)

SELECT

EXTRACT(MONTH FROM b.my\_date) AS month,

country,

source,

SUM(b.price) AS revenue

FROM

dilans\_buy b

LEFT JOIN

UserFirstRead r ON b.user\_id = r.user\_id

GROUP BY

EXTRACT(MONTH FROM b.my\_date), country, source

ORDER BY

EXTRACT(MONTH FROM b.my\_date);

SELECT EXTRACT(MONTH FROM my\_date) AS month,

SUM(price) AS revenue,

SUM(price) - 1000 AS profit

FROM

dilans\_buy

GROUP BY EXTRACT(MONTH FROM my\_date);

Heti bevétel:

WITH UserFirstRead AS (

SELECT

user\_id, country, source,

MIN(my\_date) AS first\_read\_date

FROM

dilans\_read

WHERE source != 'Direct'

GROUP BY

user\_id, country, source)

SELECT

EXTRACT(MONTH FROM b.my\_date) AS month,

country, source,

SUM(b.price) AS revenue

FROM

dilans\_buy b

LEFT JOIN

UserFirstRead r ON b.user\_id = r.user\_id

GROUP BY

EXTRACT(MONTH FROM b.my\_date), country, source

ORDER BY EXTRACT(MONTH FROM b.my\_date);

BIN-ek (különböző bontásokban készültek, egy mintát másolok ide):

|  |
| --- |
| WITH UserReadCounts AS ( |
| SELECT |
| user\_id, country, |
| COUNT(\*) AS read\_count |
| FROM |
| dilans\_read |
| GROUP BY |
| user\_id, country |
| ), |
| BinnedData AS ( |
| SELECT |
| user\_id, country, |
| read\_count, |
| CASE |
| WHEN read\_count = 1 THEN '01' |
| WHEN read\_count = 2 THEN '02' |
| WHEN read\_count = 3 THEN '03' |
| WHEN read\_count = 4 THEN '04' |
| WHEN read\_count BETWEEN 5 AND 10 THEN '05-10' |
| WHEN read\_count BETWEEN 11 AND 20 THEN '11-20' |
| WHEN read\_count BETWEEN 21 AND 30 THEN '21-30' |
| WHEN read\_count BETWEEN 31 AND 40 THEN '31-40' |
| ELSE 'above 50' |
| END AS read\_bin |
| FROM |
| UserReadCounts |
| ), |
| PurchaseData AS ( |
| SELECT |
| b.user\_id, |
| SUM(b.price) AS total\_purchase |
| FROM |
| dilans\_buy b |
| GROUP BY |
| b.user\_id |
| ), |
| CombinedData AS ( |
| SELECT |
| bd.read\_bin, |
| bd.user\_id, |
| bd.country, |
| COALESCE(pd.total\_purchase, 0) AS total\_purchase |
| FROM |
| BinnedData bd |
| LEFT JOIN |
| PurchaseData pd ON bd.user\_id = pd.user\_id |
| ) |
| SELECT |
| read\_bin, country, |
| COUNT(DISTINCT user\_id) AS user\_count, |
| SUM(total\_purchase) AS total\_purchase |
| FROM |
| CombinedData |
| GROUP BY |
| read\_bin, country |
| ORDER BY |
| read\_bin, country |

Revenue megoszlás a termékek között:

|  |
| --- |
| WITH total\_count AS ( |
| SELECT COUNT(\*) AS total\_count |
| FROM dilans\_buy |
| ), |
| price\_8\_count AS ( |
| SELECT COUNT(\*) AS price\_8\_count |
| FROM dilans\_buy |
| WHERE price = 8 |
| ) |
| SELECT |
| (p.price\_8\_count \* 1.0 / t.total\_count) \* 100 AS ratio |
| FROM |
| price\_8\_count p, |
| total\_count t |

Funnel, rates, churn, group by country (ebből más variáció is lehetséges)

SELECT EXTRACT(MONTH FROM r.my\_date) AS month,

country,

COUNT(DISTINCT r.user\_id) AS reader,

COUNT(DISTINCT re\_read.user\_id) AS returning\_reader,

COUNT(DISTINCT d.user\_id) AS subscribe,

COUNT(DISTINCT b.user\_id) AS buyer,

COUNT(DISTINCT re\_buy.user\_id) AS returning\_buyer,

COUNT(DISTINCT r.user\_id) - COUNT(DISTINCT re\_read.user\_id) AS churn\_after\_read,

COUNT(DISTINCT re\_read.user\_id) - COUNT(DISTINCT d.user\_id) AS churn\_after\_re\_read,

COUNT(DISTINCT d.user\_id) - COUNT(DISTINCT b.user\_id) AS churn\_after\_subscribe,

COUNT(DISTINCT b.user\_id) - COUNT(DISTINCT re\_buy.user\_id) AS churn\_after\_buy,

(COUNT(DISTINCT r.user\_id) - COUNT(DISTINCT re\_read.user\_id)) \* 1.0 / COUNT(DISTINCT r.user\_id) \* 100 AS reader\_churn\_rate,

(COUNT(DISTINCT re\_read.user\_id) - COUNT(DISTINCT d.user\_id)) \* 1.0 / COUNT(DISTINCT re\_read.user\_id) \* 100 AS returning\_reader\_churn\_rate,

(COUNT(DISTINCT d.user\_id) - COUNT(DISTINCT b.user\_id)) \* 1.0 / COUNT(DISTINCT d.user\_id) \* 100 AS subscriber\_churn\_rate,

(COUNT(DISTINCT b.user\_id) - COUNT(DISTINCT re\_buy.user\_id)) \* 1.0 / COUNT(DISTINCT b.user\_id) \* 100 AS buyer\_churn\_rate

FROM dilans\_read r

LEFT JOIN dilans\_subscribe d ON r.user\_id = d.user\_id

LEFT JOIN dilans\_buy b ON r.user\_id = b.user\_id

LEFT JOIN

(SELECT user\_id

FROM dilans\_read

GROUP BY user\_id

HAVING COUNT(user\_id)>1) AS re\_read

ON r.user\_id = re\_read.user\_id

LEFT JOIN

(SELECT user\_id

FROM dilans\_buy

GROUP BY user\_id

HAVING COUNT(user\_id)>1) AS re\_buy

ON r.user\_id = re\_buy.user\_id

GROUP BY EXTRACT(MONTH FROM r.my\_date), country;

**POWER BI kódok:**

Funnel analízis:

returning buyers =

 CALCULATE( COUNTROWS( FILTER( SUMMARIZE('dilans-buy', 'dilans-buy'[user\_id], "BuyCount", COUNT('dilans-buy'[user\_id]) ), [BuyCount] > 1 ) ) )

returning readers count =

CALCULATE(DISTINCTCOUNT('dilans-read'[user\_id]),

'dilans-read'[source] = "Returning")

subscribers =

    CALCULATE(

        COUNT('dilans\_\_subscribe'[user\_id]),

        USERELATIONSHIP('dilans-read'[user\_id], dilans\_\_subscribe[user\_id]))

buyers =

    CALCULATE(

        DISTINCTCOUNT('dilans-buy'[user\_id]),

        USERELATIONSHIP('dilans-read'[user\_id], 'dilans-buy'[user\_id]))

ReturningBuyers2 =

CALCULATE(

    COUNTROWS(

        FILTER(

            SUMMARIZE(

                'dilans-buy',

                'dilans-buy'[user\_id],

                "BuyCount", COUNT('dilans-buy'[user\_id])),

            [BuyCount] > 1)),

    USERELATIONSHIP('dilans-read'[user\_id], 'dilans-buy'[user\_id]))

UserReadCount =

CALCULATE(

    COUNT('dilans-read'[user\_id]),

    ALLEXCEPT('dilans-read', 'dilans-read'[user\_id]))

Egyéb measure-ök:

revenue =

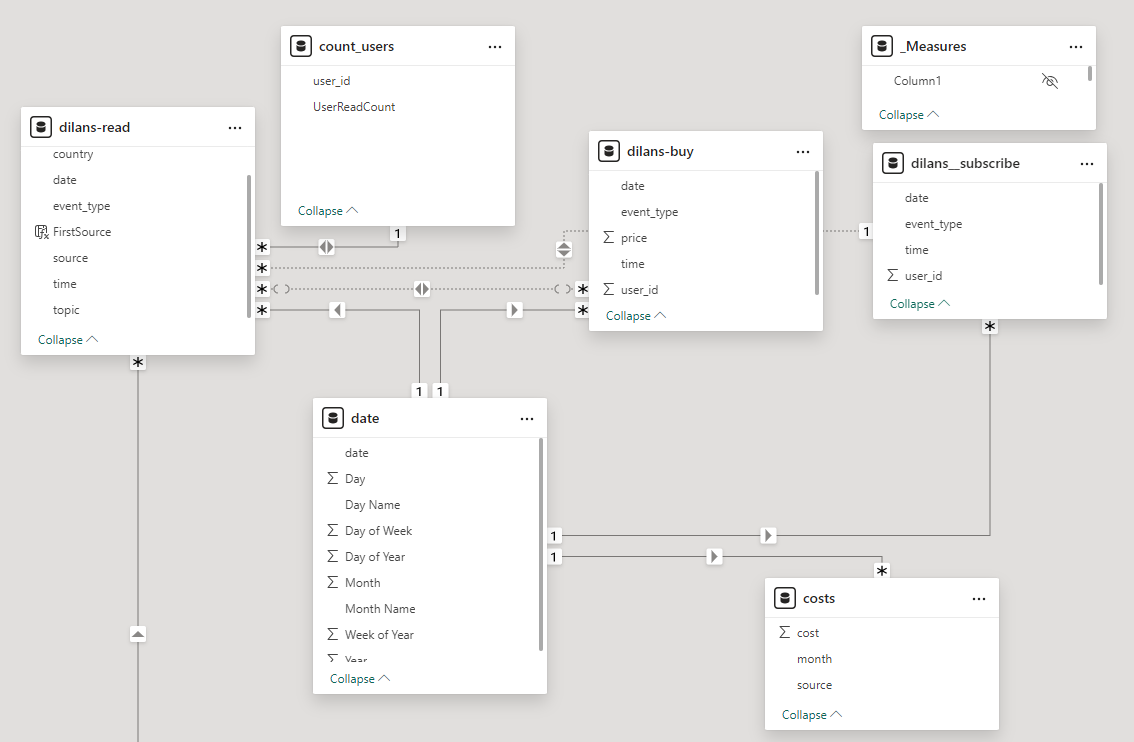
CALCULATE(

    SUM('dilans-buy'[price]),

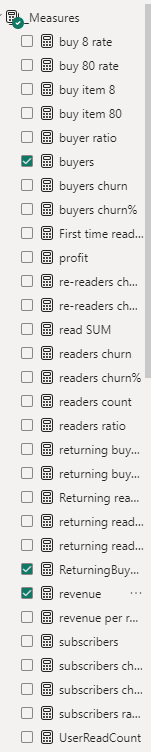
    USERELATIONSHIP('dilans-read'[user\_id], 'dilans-buy'[user\_id]))

revenue per reader = DIVIDE([revenue],[readers count],0)

Model view:



összes measure:



**PYTHON kódok:**

|  |
| --- |
| import numpy as np |
| import pandas as pd |
| import matplotlib |
| import matplotlib.pyplot as plt |
| %matplotlib inline |
|  |
| dilans\_read = pd.read\_csv('/home/pivotpanda/dilan/dilans--read.csv', delimiter = ';', names = ['my\_date', 'my\_time', 'event\_type', 'country', 'user\_id', 'source', 'topic']) |
| dilans\_buy = pd.read\_csv('/home/pivotpanda/dilan/dilans-buy.csv', delimiter = ';', names = ['my\_date', 'my\_time', 'event\_type', 'user\_id', 'price']) |
|  |
| dilans\_read['my\_date'] = pd.to\_datetime(dilans\_read['my\_date']) |
| last\_date = dilans\_read['my\_date'].max() |
| last\_date\_before = last\_date - pd.Timedelta(days=1) |
| dilans\_read\_filtered = dilans\_read[(dilans\_read['my\_date'] != last\_date) & (dilans\_read['my\_date'] != last\_date\_before)] |
|  |
| dilans\_read\_date = dilans\_read\_filtered.groupby('my\_date').count() |
| dilans\_read\_date = dilans\_read\_date.reset\_index(drop=1) |
| dilans\_read\_date = dilans\_read\_date[['my\_time']] |
| dilans\_read\_date = dilans\_read\_date.rename(columns={'my\_time': 'total\_read\_count'}) |
|  |
| x = dilans\_read\_date.index |
| y = dilans\_read\_date.values |
| coefs = np.polyfit(x,y,2).reshape(-1) |
| predict = np.poly1d(coefs) |
|  |
| from sklearn.metrics import r2\_score |
| r2\_score(y, predict(x)) |
| 0.74364 |
|  |
| x\_test = np.linspace(0, 136) |
| y\_pred = predict(x\_test[:, None]) |
| plt.scatter(x, y) |
| plt.plot(x\_test, y\_pred, c = 'r') |
| plt.show() |
|  |
| dilans\_buy['my\_date'] = pd.to\_datetime(dilans\_buy['my\_date']) |
| last\_date = dilans\_buy['my\_date'].max() |
| last\_date\_before = last\_date - pd.Timedelta(days=1) |
| dilans\_buy\_filtered = dilans\_buy[(dilans\_buy['my\_date'] != last\_date)] |
|  |
| dilans\_buy\_date = dilans\_buy\_filtered.groupby('my\_date').sum() |
| dilans\_buy\_date = dilans\_buy\_date.reset\_index(drop=1) |
| dilans\_buy\_date = dilans\_buy\_date[['price']] |
| dilans\_buy\_date = dilans\_buy\_date.rename(columns={'price': 'total\_revenue'}) |
|  |
| x = dilans\_buy\_date.index |
| y = dilans\_buy\_date.values |
| coefs = np.polyfit(x,y,3).reshape(-1) |
| predict = np.poly1d(coefs) |
|  |
| from sklearn.metrics import r2\_score |
| r2\_score(y, predict(x)) |
| 0.626434 |
|  |
| x\_test = np.linspace(0, 136) |
| y\_pred = predict(x\_test[:, None]) |
| plt.scatter(x, y) |
| plt.plot(x\_test, y\_pred, c = 'r') |
| plt.show() |