JUNE 2020

TECHNICAL REPORT

HAENSEL AMS RECRUITMENT CHALLENGE

Prepared by: Amanda Ferreirar

//

TOOLS USED: JUPTER NOTEBOOKS, TABLEAU

ALL THE CODE CAN BE FOUND AT GITHUB/AMANDAJUNQUEIRA

CHANNEL ATTRIBUTION

The goal here is to find which channels and which paths are determinant for the conversions. From this, it is even possible to calculate all the probabilities of conversion using a Markov Model, for example.

	channels	attribution_shapley_size4_conv_rate_algorithmic
0	A	38383.659990
1	B	19572.915798
2	C	4648.762252
3	D	1679.750436
4	E	9524.804432
5	F	952.310734
6	G	33264.267639
7	H	24347.480912
8	I	21417.898374
9	J	1953.331773
10	K	4733.819424

The Shapley value provide a stable way to measure channel influence and fairly divide the credit for sales conversions between the channels, based on their individual contribution to the total payoff.

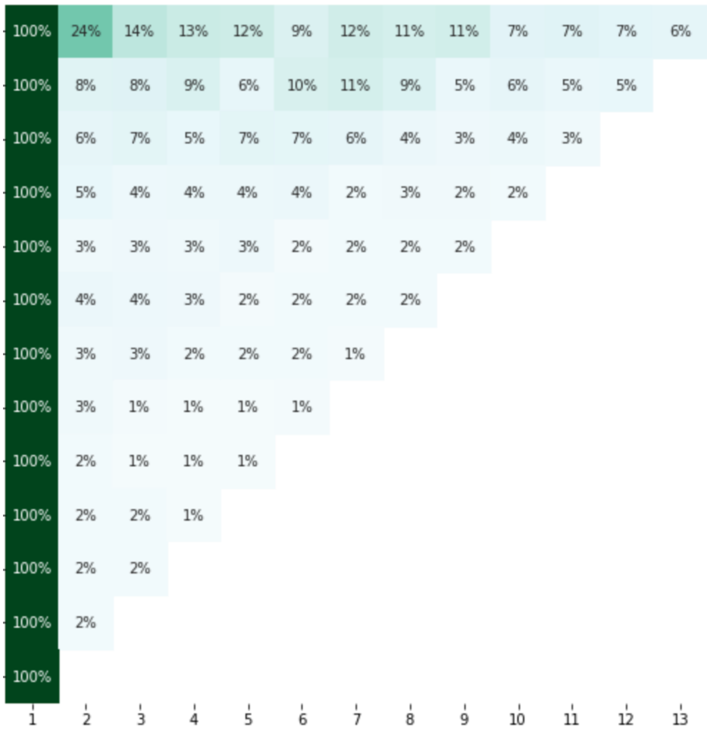
	journey_id	channels_agg	converted_agg	conversion_value
0	id:0_J:0	Direct > Youtube	True	1
1	id:0_J:1	Organic > Organic > Organic	True	1
2	id:0_J:10	Facebook	True	1
3	id:0_J:11	Facebook > Google Search	True	1
4	id:0_J:12	Google Search > Google Search	True	1
...	...	...	...	...

Understanding the customers journey might be a very complex task, for it involves many aspects. Understanding its path, however, might lead to interesting patterns discovery.

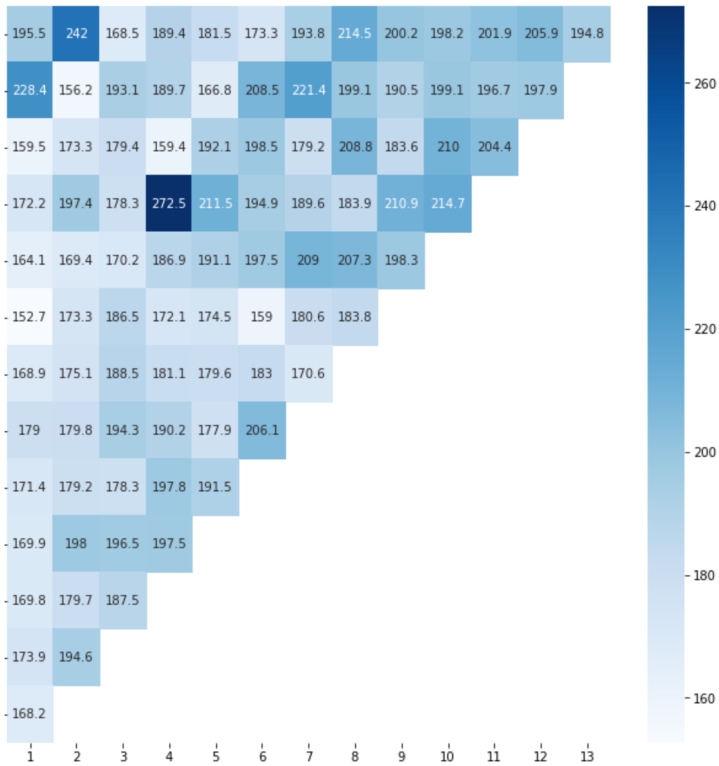
COHORT ANALYSIS

"Cohort analysis can be helpful when it comes to understanding your business' health and "stickiness" - the loyalty of your customers.'

RETENTION RATE



AVERAGE REVENUE



The cohorts are monthly grouped since their first purchase.

```python
def get_month(x):
    return dt.datetime(x.year, x.month, 1)
tablex['InvoiceMonth'] = tablex['Conv_Date'].apply(get_month)
tablex['CohortMonth'] = tablex.groupby('User_ID')['InvoiceMonth'].transforn
```

```python
def get_date(df, column):
    year = df[column].dt.year
    month = df[column].dt.month
    day = df[column].dt.day
    return year, month, day

invoice_year, invoice_month, _ = get_date(tablex, 'InvoiceMonth')
cohort_year, cohort_month, _ = get_date(tablex, 'CohortMonth')
year_diff = invoice_year - cohort_year
month_diff = invoice_month - cohort_month
tablex['CohortIndex'] = year_diff * 12 + month_diff + 1
```

```python
cohort_data = tablex.groupby(['CohortMonth', 'CohortIndex'])['User_ID'].apr
cohort_count = cohort_data.pivot_table(index = 'CohortMonth',
                                       columns = 'CohortIndex',
                                       values = 'User_ID')
cohort_count
```

Function to generate the heatmaps

RFMSCORE, CUSTOMER SEGMENTATION

"RFM stands for Recency, Frequency, and Monetary value, each corresponding to some key customer trait. These RFM metrics are important indicators of a customer's behavior because frequency and monetary value affects a customer's lifetime value, and recency affects retention, a measure of engagement."

User_ID	recency	frequency	monetary_value
00003ce67d6b73b2d49f4036f60cb73385a9c96e	769	4	615.360
0003509d64606735e66a3d32f2a1a084f613ee4b	712	5	700.864
00035f943a8a8e176fdd5a44059b38dcc0c73f5a	661	7	3146.624
0003f10010cd3dadcb7182ed7b0abf5166393e91	910	1	121.808
0003fc733e4ff3bfb295f2c10c7077fb0763ebcc	651	1	108.720

The RMFScore of the Users of the dataset provided was calculated, but how is this relevant? To make it more useful, it would be interesting to know more about the business. It is difficult to say which factor is more important without knowing the field of the company. One will buy a car (highly monetary) but in a very low frequency.

TOP CHAMPIONS: THESE ARE THE 10 BEST CUSTOMERS, in all possible aspects. They can be a good strategy, since they can promote your brand and be an early adopter of a new release:

User_ID	recency	frequency	monetary_value	r_
2c75940486d75040f269c9671ab746dffefe9692	627	720	184557.13812	
31e3c730764f2913e56fcae325f92a82bc94a4aa	629	175	82045.18148	
0ad05472146efb8b505f113c4cdc3a88b5a89f41	634	334	65669.35200	
72df33e2b3ccfebff04123e211ef07d5f39a2324	654	116	36141.88224	
37339f068a6e98afabcde8943c552254d67f49b2	625	91	34377.49360	
01b91ca588ca5072bbe879bd0bebf5f733ddf933	653	57	32743.38400	
4ac5eadb5c74e24d80aa806624b3de3ee5c0732a	641	65	30192.80400	
be7cd84e8b175933f5b86276b429d87d414b5f4a	653	146	29802.27924	
f3082ca1f6c34452b3e4048ae0de7810f0edc2d5	642	126	28118.73200	
d76ba1531664ec1e4baad0cb061df17048c77370	634	101	25399.02728	