CASE STUDY – Data Analyst

Part 1 – Clicked items investigation

In the resources section, you will find trivago_data_analysis_case_study_part_1.csv with information about items (hotels). Each row displays for each item, it's id, number of times it was displayed impressions), number of click-outs (clicks), set of clicked displayed positions (0=first position), average impressed position, number of users and sessions with at least one impression. Only clicks and impressions from the first page results are considered.

1. Calculate the CTR of each item. What is the overall avg CTR?

Ans: CTR of each item was calculated by taking the ratio of the number of clicks and number of impressions as shown below:

CTR of each item

```
In [118]: # CTR of each item
          df['CTR'] = round(df['clicks']*100/df['impressions'] , 2)
          df
```

Out[118]:

	item_id	impressions	clicks	clicked_displayed_positions	avg_impressed_position	num_users	num_sessions	CTR
0	5001	27	2	7;5	5.52	8	10	7.41
1	5003	21	1	3	2.81	8	13	4.76
2	5009	6	2	1;1	8.67	2	3	33.33
3	5014	72	8	0;-11;-11;-11;3;-11;0;2	6.69	40	42	11.11

Average CTR was calculated as 11.45 as shown below

Overall average CTR is 11.45

```
In [119]: # Overall average CTR

          df.CTR.mean()
```

```
Out[119]: 11.455641646675604
```

2. What is the distribution of clicks among the top 25 positions? What is the share of the first positions? On how many positions are approx. Half of the click-outs made?

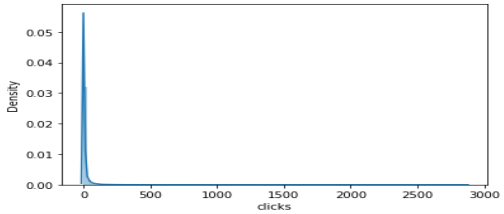
Ans: Distribution of clicks among the top 25 positions looked something like below(right skewed and peaked at '0'th position)

Distribution of clicks

```
In [122]: import seaborn as sns

In [123]: sns.distplot(df1.clicks, bins=100)
          warnings.warn(msg, FutureWarning)
```

```
Out[123]: <AxesSubplot:xlabel='clicks', ylabel='Density'>
```



For finding the share of clicks of the first positions, first the data was filtered out with records of avg_impressed_position == 0 and then ratio of sum of clicks of first positions and sum of clicks of all position was taken as shown below:

Share of first positions is 0.15 %

```
In [124]: df2 = df1[df1.avg_impressed_position == 0]
          share_of_first_positions = sum(df2.clicks) * 100 / sum(df1.clicks)
          print('Share of the first positions is : ',' ',round(share_of_first_positions,3),'%')

          Share of the first positions is :    0.155 %
```

To find the number of positions on which half of the click outs were made, first the index of the records was found upto which the cumulative sum of the clicks is approx. close to the sum of all the clicks in the data. Then mean of the avg_impressed_position was calculated to find the number of positions on which half of the total click outs were made. This is shown below:

Number of positions on which half of the click outs are made

```
In [129]: position_till_halfofclickouts = np.flatnonzero(np.isclose(df1.clicks.cumsum().values, sum(df1.clicks)/2, atol=2))[0]
          position_till_halfofclickouts

Out[129]: 4135

In [132]: df1['avg_impressed_position'].iloc[0:position_till_halfofclickouts + 1].mean()

Out[132]: 9.71692456479692
```

Thus, on approx first 10 positions out of 25 positions , half of the click outs are made

3. Describe the relationship between the average displayed position and the clicked displayed position. What are your thoughts about the variance between the two?

Ans: Relationship between the average displayed position and clicked display position can be found out by using the pearson correlation coefficient. Coefficient value obtained was 0.44 which represents a moderate positive correlation between the two entities. Such values which are below 0.8 are generally no considered as significant or useful for statistical analysis.

Also, the covariance value between the two columns was found to be 11.41 which is positive and hence tells us that both the variables increase and decrease in the same direction i.e. the relationship between the two entities is positive. However the covariance value does not give the measure of strength of positive relationship between the two variables.

4. In the dataset, we provided you with the average displayed position. What can be wrong with using averages?

Ans: Average value is influenced by the outliers. Suppose the frequency of the top positions is more and there is one position in the list which is the lowest position. This might shift the average from near the top positions towards the lower positions. This gives a false impression of the average position at which the ad was displayed maximum number of times.

Also as it is evident from the previous obtained value of correlation or covariance, the average displayed position does not seem to help much in further statistical analysis of the data.

Part 2 – Session investigation

For this task, you now have some session level information. Take a look at the data in trivago_data_analysis_case_study_part_2.csv and get familiar with the structure. Each row corresponds to a click-out from a user and you can find basic information on the structure at the end of this document.

1. Describe the data set that you have received. Calculate the 5 most frequent values per column (with frequency). Can you find any suspicious results? If so, what are they? And how would you fix these for the analysis?

Ans: Dataset contains 1.9 million records.

User_id: Each row represents the clicked information of users. Each user_id represents unique user. Dataset contains repeated records of every user. Records of '4509866267912745159' user occur the most.

Session_id: Represents id of the sessions assigned to each user.

Clicked_item_id: Represents the unique hotel Id. '2834772' seems to be the popular hotel with most clicks.

displayed_position: Represents the position of the hotel on the list. First position(0) seems to be repeated the most which means users click on the first position frequently.

page_num: Represents the page number of the item on which the user has clicked. Statistics shows that the first five pages have the maximum visits from users i.e. about 98.7% of the total clicks.

sort_order: Its a categorical variable. Sort_order = 12 seems to be the most popular one among the users.

search_type: Its a categorical variable. Search_type=2113 seems to be the popular one.

path_id: Represents the location of the hotel. '38715' location seems to be the most popular one among the users.

arrival_days: Represents the number of days from the day of search until check in date. On an average users search for the hotels 57 days in advance.

departure_days: Represents the number of days from the day of search until the check out date.

traffic_type: Its a categorical variable. traffic_type = 2 is the most used way by the users to access the trivago website.

impressed_items_id: Represents the list of all other items that were displayed along with clicked_item_id

Top 5 frequent values for each column is presented as below:

```
In [144]: for i in df2.columns:
              print("\n Top 5 frequent values of column ",i,"is \n" )
              print(df2[i].value_counts().head(5))
```

Top 5 frequent values of column user_id and their count is

4509866267912745159 549
3212529366726084995 373
9051529085815567256 332
8844433817516149751 272
6297403635493453063 219
Name: user_id, dtype: int64

Top 5 frequent values of column session_id and their count is

5.009065e+18 337
8.560570e+18 148
2.839951e+18 109
7.154740e+18 92
3.191225e+18 88
Name: session_id, dtype: int64

Top 5 frequent values of column clicked_item_id and their count is

2834772.0 2869
1321090.0 2828
12624.0 2575
32940.0 2324
1455251.0 2297
Name: clicked_item_id, dtype: int64

Top 5 frequent values of column displayed_position and their count is

0.0 495014
1.0 203090
-11.0 152437
2.0 138789
3.0 108490
Name: displayed_position, dtype: int64

Top 5 frequent values of column page_num and their count is

0.0 1677391
1.0 124109
2.0 44578
3.0 21226
4.0 11857
Name: page_num, dtype: int64

Top 5 frequent values of column sort_order and their count is

12.0 1162269
312.0 283516
212.0 229803
412.0 94343
41.0 54259
Name: sort_order, dtype: int64

Top 5 frequent values of column search_type and their count is

2113.0 928598

2116.0 375583
2111.0 230316
2115.0 194310
2114.0 174986
Name: search_type, dtype: int64

Top 5 frequent values of column path_id and their count is

38715.0 138706
46814.0 41379
41579.0 36498
38961.0 31634
31965.0 30928
Name: path_id, dtype: int64

Top 5 frequent values of column arrival_days and their count is

0.0 65863
3.0 63180
1.0 60910
4.0 54871
5.0 50279
Name: arrival_days, dtype: int64

Top 5 frequent values of column departure_days and their count is

4.0 59506
1.0 52780
-1000000.0 52530
5.0 51982
6.0 48879
Name: departure_days, dtype: int64

Top 5 frequent values of column traffic_type and their count is

2.0 1496629
3.0 301763
1.0 105403
Name: traffic_type, dtype: int64

Top 5 frequent values of column impressed_item_ids and their count is

1455251;40899;4995818;40887;15854;40928;2282460;40923;2805646;40917;15119;1332489;2292434;40921;40894;1234586;40935;40883;2727758;4775026;40880;15122;1105676;15114;40910 14
1
3439776
80
2092330;3390442;3439776
78
1455251;40887;4995818;40899;40928;15854;40923;2282460;2805646;40917;40921;15119;40909;1332489;1234586;4775026;40883;40894;40935;2292434;2065710;914555;150636;1833933;2727758
76
1330971;149339;624396;693916;927159;945921;1109455;1247036;1259540;2173240;2588352;3056364;4527530;3812986;909353;97917;900401;97940;153032;2396300;1929999;3492384;920327;1363627;449196 56
Name: impressed_item_ids , dtype: int64

Suspicious results and corresponding fixes for further analysis:

1. If we look at the top 5 frequent values of the column departure_days then you can see the abnormal valu e of -1000000 days. This is abnormal because first of all, this value is expected to be higher than arrival_da ys and number of days cannot be negative from the day of search. So to fix these values, we can replace th em with the following method:
Find the average stay of individual "clicked_item_id" in the main data set obtained by omitting records wit h "departure_days" as -1000000 days. Average stay can be calculated by subtracting the departure_days an d arrival_days. Then increase the corresponding "arrival_days" for all those particular "clicked_item_id" w hich have departure_days as -100000 by the average stay calculated and record it as "departure_days".

2. Also in the displayed_position column, we see that the -11 position is among the top 5 frequent values. This value can be replaced by '14' as -11 represents 14th position as per list principle
or
We can calculate the average display position for each clicked_item_id having -11 position from the datase t and replace -11 with the rounded average displayed position value.

2. Which search type has the lowest average displayed position? What is the best sorting order for this search type? Which search type should be excluded for a statistical reason?

Ans: For finding the search type having the lowest average displayed position, the data was grouped by search_type and then mean of the avg_displayed_position was calculated for each search_type and then the search_type with the lowest mean value was selected.

```
In [145]: ## Preparing a dictionary for each search_type and its corresponding average displayed position
          avg_disp_lst = dict(df2.groupby('search_type')['displayed_position'].mean())

In [146]: # min(avg_clicked_lst, key=avg_clicked_lst.get)
          min(avg_disp_lst.items(), key=lambda x: x[1])

Out[146]: (2116.0, 1.372423671997934)
```

Thus, search_type = 2116 has the lowest average display position

Once search_type = 2116 was found to have the lowest mean displayed position, then records of search_type = 2116 were filtered out and then this data was further grouped by sort_order and mean of the avg_displayed_position was calculated for each sort_order. Then the sort_order having the lowest mean value(sort_order = 312) was selected as the best value for the search_type = 2116.

```
In [12]: avg_sort_lst = dict(df3.groupby('sort_order')['displayed_position'].mean())

In [13]: avg_sort_lst

Out[13]: {0.0: 7.545454545454546,
          12.0: 1.232831422718942,
          21.0: 2.142704044221332,
          32.0: 5.098214285714286,
          41.0: 5.320428826682549,
          112.0: 4.995951417004049,
          212.0: 5.224325412807088,
          312.0: 0.7984770702326525,
          412.0: 5.613883299798792}
```

Best sort order for search type having lowest average display position (search_type =2116)

```
In [14]: best_sortorder = min(avg_sort_lst.items(), key=lambda x: x[1])[0]
         best_sortorder

Out[14]: 312.0
```

Now to find the search_type to be excluded for statistical analysis, the search_type having the maximum average_display_position should be selected as shown below.

```
In [15]: search_type_to_be_excluded = max(avg_disp_lst.items(), key=lambda x: x[1])[0]

In [16]: search_type_to_be_excluded

Out[16]: 2111.0
```

Thus, 2111 search_type is best to be excluded from statistical pov because it has the maximum average display position

3. Describe and visualize the relationship between the average displayed position and the CTR among the top 1000 most clicked items

Ans: To approach this problem, first we find the top 1000 clicked items by finding the counts of unique clicked_item_id and selecting the top 1000 records having the highest count. A list of top 1000 clicked_item_ids is prepared and the main data is filtered out according to this list(say df4).

Now, again counts of the unique item_ids is found from df4. Thus we have the top 1000 item_ids and their corresponding clicks. A separate dataframe called clicks_df is formed out of this.

```
clicks_df = df4.clicked_item_id.value_counts().reset_index()
clicks_df.columns = ['clicked_item_id', 'clicks']
clicks_df = clicks_df.sort_values(by='clicked_item_id', ascending=True).reset_index(drop=True)
clicks_df
```

Out[60]:

	clicked_item_id	clicks
0	5563.0	250
1	5742.0	369
2	6743.0	254
3	6753.0	282
4	6768.0	623
...	...	...
995	5209538.0	486
996	5592002.0	334
997	5599748.0	823
998	6692704.0	250
999	7028444.0	330

1000 rows × 2 columns

Now, the df4 is grouped by clicked_item_id and mean of the displayed_position is calculated for each item_id. Thus we have the top 1000 item_ids and their corresponding average displayed position. A dataframe called avg_disp_position_df is formed out of this.

```
In [61]: ## Finding average displayed position for each item id
avg_disp_position_df = df4.groupby('clicked_item_id', as_index=False)['displayed_position'].mean()
avg_disp_position_df
```

Out[61]:

	clicked_item_id	displayed_position
0	5563.0	1.580000
1	5742.0	1.262873
2	6743.0	4.181102
3	6753.0	3.865248
4	6768.0	1.385233
...	...	...
995	5209538.0	4.296296
996	5592002.0	1.658683
997	5599748.0	2.509113
998	6692704.0	4.492000
999	7028444.0	2.766667

1000 rows × 2 columns

Next challenge is to calculate the number of impressions of the top 1000 clicked items. For this, the impressed_item_ids column is used from the original data. The whole column is merged into a single large list and then count of unique item_ids is calculated from this list. Thus we have item_ids and their impressions. A dataframe is formed out of this. Note that this dataframe contains 'all' the item_ids from the original dataset.

The above obtained dataframe is now filtered to obtain only the records related to the top 1000 clicked items found previously. Thus, finally we have the impressions dataframe called impressions_df1 for the top 1000 clicked item_ids.

The 3 dataframes i.e. clicks_df, avg_disp_position_df and impressions_df1 are concatenated together to form a single dataframe containing the top 1000 clicked_item_ids , number of clicks, avg display position and corresponding impressions.

```
In [105]: ## Merging the clicks dataframe, average displayed position dataframe and number of impressions data frame into single dataframe
final_ctr_df =  pd.concat([clicks_df,avg_disp_position_df.displayed_position,impressions_df1.impressions], axis=1)
final_ctr_df
```

Out[105]:

	clicked_item_id	clicks	displayed_position	impressions
0	5563.0	250	1.580000	3040
1	5742.0	369	1.262873	2976
2	6743.0	254	4.181102	4708
3	6753.0	282	3.865248	6676
4	6768.0	623	1.385233	7466
...	...	...	...	...
995	5209538.0	486	4.296296	16457
996	5592002.0	334	1.658683	5807
997	5599748.0	823	2.509113	15345
998	6692704.0	250	4.492000	2988
999	7028444.0	330	2.766667	3859

1000 rows × 4 columns

Now CTR can be calculated by taking the ratio of 'clicks' and impressions and then relationship between CTR and avg_display_position can be found as shown below.

	clicked_item_id	clicks	displayed_position	impressions	CTR
0	5563.0	250	1.580000	3040	8.223684
1	5742.0	369	1.262873	2976	12.399194
2	6743.0	254	4.181102	4708	5.395072
3	6753.0	282	3.865248	6676	4.224086
4	6768.0	623	1.385233	7466	8.344495
...	...	...	...	...	...
995	5209538.0	486	4.296296	16457	2.953151
996	5592002.0	334	1.658683	5807	5.751679
997	5599748.0	823	2.509113	15345	5.363311
998	6692704.0	250	4.492000	2988	8.366801
999	7028444.0	330	2.766667	3859	8.551438

1000 rows × 5 columns

```
final_ctr_df['CTR'].corr(final_ctr_df['displayed_position'])
```

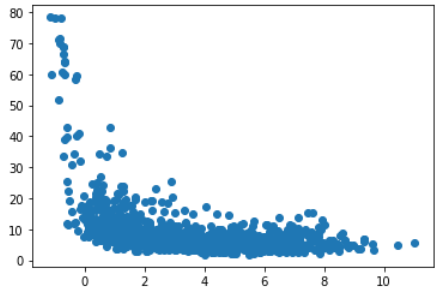
```
-0.45700969797534013
```

Pearson correlation coefficient value(-0.45) is between -0.30 to -0.49 and hence indicates weak negative linear correlation which means that if average display position increases, then the CTR decreases.

Visualization of CTR and average displayed position relationship of top 1000 clicked items is as follows:

Visualizing the relationship between CTR and displayed position

```
In [112]: from matplotlib import pyplot
          pyplot.scatter(final_ctr_df.displayed_position, final_ctr_df.CTR)
          pyplot.show()
```



While, covariance between CTR and average displayed position of top 1000 clicked items can be calculated as follows:

Finding the Relationship between average displayed position and CTR of the top 1000 clicked items

```
In [114]: from numpy import cov
          cov(final_ctr_df.displayed_position, final_ctr_df.CTR) ##Negative value indicates that variables increase in the opposite directi

Out[114]: array([[  5.84952668, -10.30945479],
                 [-10.30945479,  86.99611099]])
```

Negative value of covariance(-10.30) indicates that variables increase in the opposite directions

```
In [173]: from scipy.stats import pearsonr
          pearsonr(final_ctr_df.displayed_position, final_ctr_df.CTR) ##Value between -0.30 to -0.49 indicates weak negative linear correl

Out[173]: (-0.4570096979753403, 9.326399090293188e-53)
```

Pearson correlation coefficient value(-0.45) is between -0.30 to -0.49 and hence indicates weak negative linear correlation which means that if average display position increases, then the CTR decreases.