RealSelf_BA_test

April 30, 2021

1 RealSelf BA Test Question 1

[ ]:
```
-- Q1
SELECT question_id
FROM table_of_questions_by_created_date qc
LEFT JOIN table_of_questions_and_tag_ids qt ON qc.question_id = qt.question_id
WHERE qc.created LIKE 2018%
AND tag_ids.tag_id LIKE "pre-op"

-- Q2
SELECT tt.tag_name, MONTH(qc.created) AS months, SUM(qc.question_id) AS␣
 ↪total_questions
FROM table_of_questions_by_created_date qc
LEFT JOIN table_of_questions_and_tag_ids qt ON qt.question_id = qc.question_id
LEFT JOIN table_of_tags tt ON tt.tag_id = qt.tag_id
WHERE YEAR(qc.created) > 2018
ORDER BY months,tt.tag_name

-- Q3
SELECT tt.tag_name, AVG(COUNT(*)) AS total
FROM table_of_tags tt
JOIN table_of_questions_and_tag_ids qt ON qt.tag_id = tt.tag_id
JOIN (
  SELECT qc.question_id
  FROM table_of_questions_by_created_date qc
  WHERE (qc.created LIKE "2018%") OR (qc.created LIKE "2019%")
)
ON qc.question_id = qt.question_id
GROUP BY tt.tag_name
ORDER BY tt.tag_name DESC
```

2 RealSelf BA Test Question 2

2.1 Import Libraries

[7]:
```
import pandas as pd
import numpy as np
```

2.2 Read in Data

[2]:
```
df = pd.read_excel("./drive/MyDrive/data_sets/question2_data.xlsx")
```

2.3 General Summary Stats

[6]:
```
display(df.head(5))
display(df.describe())
```

```
  Created Date  ...    Cost
0   2019-05-31  ...     0.0
1   2019-02-13  ...     0.0
2   2019-01-13  ...     0.0
3   2019-06-30  ...     0.0
4   2019-04-08  ...  3000.0

[5 rows x 9 columns]
```

```
       Number of Photos  Number of Words  Provider Rating         Cost
count        662.000000       662.000000       644.000000   662.000000
mean           0.267372        87.483384         4.854037   206.856737
std            1.287479        75.813763         0.710157   395.072841
min            0.000000        28.000000         1.000000     0.000000
25%            0.000000        45.000000         5.000000     0.000000
50%            0.000000        61.000000         5.000000     0.000000
75%            0.000000       101.750000         5.000000   320.000000
max           18.000000       704.000000         5.000000  5000.000000
```

2.4 Null Value Analysis

We will check for null values first. Physician type has a substantial amount of missing entries which means limiting the data to a subset or abandoning the column should be considered.

[16]:
```
nulls = {x:(df[x].isna().sum()/len(df)*100)for x in df.columns}
(df.isna().sum() / len(df))*100
```

[16]:
```
Created Date          0.000000
Treatment Rating      0.000000
Number of Photos      0.000000
```

```
Number of Words      0.000000
Provider Rating      2.719033
Physician Type      17.371601
Treatment Name       0.000000
Title                0.000000
Cost                 0.000000
dtype: float64
```

The cost column has 60% of it's data imputed with 0 as a placeholder. This makes cost analysis somewhat precarious given the limited sample size, however, as cost analysis is a primary goal of this work it cannot be dropped from consideration. At this time I do not recommend a form of statistical imputation (k-means, fuzz-c, mean, median, mode) as they won't have a substantial impact on the summary statistics of the colum anyways.

[18]:
```
nulls['Cost'] = len(df['Cost'].loc[df['Cost'] == 0])/len(df)*100
nulls
```

[18]:
```
{'Cost': 59.516616314199396,
 'Created Date': 0.0,
 'Number of Photos': 0.0,
 'Number of Words': 0.0,
 'Physician Type': 17.371601208459214,
 'Provider Rating': 2.719033232628399,
 'Title': 0.0,
 'Treatment Name': 0.0,
 'Treatment Rating': 0.0}
```

2.4.1 Null Value Summary

For the satisfaction portion, I would leave the data as is and analyze everything. For the cost portion, I'm going to abadon all rows that are missing cost data, do some additional analysis and will consider additional data imputation at that point.

2.5 Outlier Detection

Given what I know about each of the proceedures, costs are highly correlated with the amount of product/surface area of the treatment. Because our sample sizes are small for the non-botoz proceedures, I'm going to use the interquartile range method of filtering outliers as the mean and standard deviation are highly unreliable under these conditions.

Costs will be broken down within the proceedural category.

[32]:
```
# break things into the proceedural categories
dfCost = df.loc[df['Cost'] != 0]
dfCostB = {'df':dfCost.loc[dfCost['Treatment Name'] == 'botox'], 'name':'Botox'}
dfCostCP = {'df':dfCost.loc[dfCost['Treatment Name'] == 'chemical peel'],'name':
 ↪'Chemical Peel'}
dfCostCM = {'df':dfCost.loc[dfCost['Treatment Name'] == 'coolmini'],'name':
 ↪'Coolmini'}
```

```
dfCostV = {'df':dfCost.loc[dfCost['Treatment Name'] == 'volbella'],'name':
 ↪'Volbella'}
proceedures = [dfCostB, dfCostCP, dfCostCM, dfCostV]
```

[33]:
```
# get summary stats
for x in proceedures:
  display(x['name'])
  display(x['df'].describe())
  print()
  print()
```

'Botox'

	Number of Photos	Number of Words	Provider Rating	Cost
count	240.000000	240.000000	231.000000	240.000000
mean	0.225000	87.908333	4.818182	464.016292
std	1.054677	72.154297	0.775617	342.334018
min	0.000000	28.000000	1.000000	13.000000
25%	0.000000	45.000000	5.000000	258.750000
50%	0.000000	63.000000	5.000000	383.500000
75%	0.000000	102.000000	5.000000	577.717500
max	10.000000	584.000000	5.000000	3000.000000

'Chemical Peel'

	Number of Photos	Number of Words	Provider Rating	Cost
count	10.000000	10.000000	8.000000	10.000000
mean	2.800000	106.700000	4.750000	879.013000
std	5.750362	99.897781	0.707107	1625.890274
min	0.000000	40.000000	3.000000	75.000000
25%	0.000000	46.250000	5.000000	108.052500
50%	0.000000	58.500000	5.000000	142.500000
75%	3.000000	105.500000	5.000000	325.000000
max	18.000000	292.000000	5.000000	5000.000000

'Coolmini'

	Number of Photos	Number of Words	Provider Rating	Cost
count	8.00000	8.000000	6.0	8.000000
mean	1.50000	219.875000	5.0	1193.750000
std	2.13809	238.194061	0.0	374.583102
min	0.00000	50.000000	5.0	750.000000
25%	0.00000	54.750000	5.0	950.000000
50%	0.00000	85.500000	5.0	1100.000000
75%	3.25000	353.250000	5.0	1500.000000
max	5.00000	704.000000	5.0	1800.000000

'Volbella'

	Number of Photos	Number of Words	Provider Rating	Cost
count	10.000000	10.000000	9.000000	10.000000
mean	2.000000	102.600000	4.111111	723.512000
std	3.091206	38.505988	1.763834	545.501897
min	0.000000	52.000000	1.000000	400.000000
25%	0.000000	81.000000	5.000000	449.985000
50%	1.000000	91.500000	5.000000	492.570000
75%	2.750000	114.500000	5.000000	737.500000
max	10.000000	172.000000	5.000000	2200.000000

This gives a much better sense of the average cost of each proceedure. With more time, it would be worth examining changing the standard, 25th% - 75th% implementation of the interquartile range, but it will suffice here.

2.6 Ratings

As there are no null values in the Treatment Rating column, I see no need to drop any rows for this portion of the analysis.

Looking at the imbalance in the number of samples accross the various treatments, a normalized metric will be needed to determine ranking. Ideally, I would only analyze a subsample of the data, stratifying each proceedure accross the available providers in case the provider column adds a stong bias. Because the provider information isn't available in this dataset, I will skip this.

My approach is going to add the number of "Worth it" treatment ratings per proceedure and divede said number by the total number of proceedures.

[53]:
```
# break things into the proceedural categories
dfB = {'df':df.loc[df['Treatment Name'] == 'botox'], 'name':'Botox'}
dfCP ={'df':df.loc[df['Treatment Name'] == 'chemical peel'],'name':'Chemical␣
 ↪Peel'}
```

```
dfCM ={'df':df.loc[df['Treatment Name'] == 'coolmini'],'name':'Coolmini'}
dfV = {'df':df.loc[df['Treatment Name'] == 'volbella'],'name':'Volbella'}
proceedures = [dfB, dfCP, dfCM, dfV]
```

[54]:
```
for x in proceedures:
  x['df']['worth_it'] = [1 if y == 'Worth it'  else 0 for y in␣
 ↪x['df']['Treatment Rating']]
  worthit = len(x['df'].loc[x['df']['Treatment Rating'] == 'Worth it'])
  display(x['name'])
  display(f"Score: {worthit/len(x['df'])}")
  display(f"error = {x['df']['worth_it'].std()}")
  print()
  print()
```

```
/usr/local/lib/python3.7/dist-packages/ipykernel_launcher.py:2:
SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-
docs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy


'Botox'


'Score: 0.9613445378151261'


'error = 0.19293464800353932'




'Chemical Peel'


'Score: 0.8928571428571429'


'error = 0.314970394174356'




'Coolmini'
```

```
'Score: 0.875'
```

```
'error = 0.3415650255319866'
```

```
'Volbella'
```

```
'Score: 0.8695652173913043'
```

```
'error = 0.34435022157509093'
```

I would like to note that data imputation would be worth another look using one fo the more advanced methods such as k-means or fuzzy-c. Such methods would help detect trends between the other columns and the cost column and make the most informed decision when imputing data. At this point, however, because this imputation involves training a machine learning model on the data I'm going to omit it for the sake of staying under the 2 hour window.

2.7 Answer to Question 1:

Based on the data and a brief analysis, I would suggest providing the cost information in the form of a range with the 25th percentile as the bottom and the 75th percentile as the top of the range. This is what that would look like for each proceedure.

- Volbella: 450 - 740
- Coolmini: 950 - 1500
- Chemical Peel: 100 - 325
- Botox: 260 - 380

This approach will give a good understanding as to what most customers will recieve as a bill. I would also like to point out that there are more cleaver marketing strategies that could be used without being misleading such as using the lowest non-discounted price and saying "Starting From [INSERT PRICE]." However, I stand by my analysis as providing the most realistic price expectation to customers.

2.8 Answer to Question 2

Normalizing our analysis to percentages, we have the following breakdown:

1. Botox: 96% approval / Error = 19%
2. Chemical Peel: 89% approval / Error = 31%

3. Coolmini: 88% approval / Error = 34%
4. Volbella: 87% approval / Error = 34%

Overall, I'm not confident in my ranking for two reasons.

1. Few number of samples for the non-botox proceedures makes them highly succeptible to random varience. Using the standard deviation as an error, there is overlap in all rankings, meaning no statistical conclusions can be reasonable drawn without more info.
2. The Provider name isn't given. There is a high degree of correlation between treatment rating and provider rating, meaning the treatment rating likely has more to do with the provider the patient saw than the proceedure itself. Thus, without being able to stratify the data to the provider, these conclusions are highly succeptible to sample bias.

As noted, provider rating is highly correlated with treatment ranking.

3 RealSelf Question 3:

3.1 Part A

[63]:
```
thing = [437,96,79,65,50,25,5]
for n in range(len(thing)-1):
  display(f'Step {n+1} drop-off: {round((thing[n] - thing[n+1])/thing[n],2)}␣
 ↪percent')
  print()
```

```
'Step 1 drop-off: 0.78 percent'
```

```

```

```
'Step 2 drop-off: 0.18 percent'
```

```

```

```
'Step 3 drop-off: 0.18 percent'
```

```

```

```
'Step 4 drop-off: 0.23 percent'
```

```

```

```
'Step 5 drop-off: 0.5 percent'
```

```

```

```
'Step 6 drop-off: 0.8 percent'
```

- Step 5 has the largest percentage drop-off and Step 1 has the largest magnitidue drop off

3.2 Part B

I begin by determining the revenue expecation per customer that clicks on an add. There are only 5 per 437 who are ultimately referred which implies only 1.14% of those who click are ultimately referred. With 1000 dollars per referral,

- the expected amount per customer is 1000*.00114 = $11.44

With a proportional examination this implies

- the advertizing cost per live transfer is $349.65

3.3 Part C

Using the same logic from part B

- the total cost per live transfer is $786.71

3.4 Part D

if 50 of the users submit forms and only 5 are ultimately refered, this mean * the conversion rate from form submission to live transfer is 10%

3.5 Part E

Removing step 4 from the process map, I would assume the best case senario is that you keep all customers from step 3 and they are now ported into step 5. Under this assumption, we add back those who drop off at step 4 which is 14.

- We should expect 64 users to submit the form

3.5.1 Part F

The break even point for conversion rate for the entire process is 0.9% from beginning to end. This would mean 3.9 users are referred per 437 users. * So the mimnimum conversion rate for submission to transfer is 7.8%

4 RealSelf Question 4:

4.1 Link to visual

Link to Viz

4.2 Summary

The visual is showing the contribution of CO2 emissions per county from 1990 until 2013. A key fearure of this graph is the color coding of countries by their "income bracket". There are three representations of this information that the visual provides. 1. The Geological representation showing the counties on a world map, with weighted dots representing the magnitude of their emmissions 2. A segmented line graph broken out by income bracket showing where the various

members of the brackets are on the scale of their contribution. 3. A stacked ling graph lumping all countries together, color coded by income brack on a scale of their emissions contribution.

Two bits that stick out to me are the following * Japan has the 5th highest contribution of emissions despite being a small country with the 11th highest population. * India has more than 5 times more contribution than the second largest contributer in the lower-middle income bracket.

4.3 Concerns

1. I'm not a fan of the logirithmic scale on emissions contribution. In my opinion, this opscures the true contributions and nature of the relationship with county income.
2. The visual should remove outliers prior to showing summary statistics for the income brakets, as there are few countries which heavily sckew these results.
3. The age of the data. The most recent year of study was 2013. It's not trivial to verify that the trends identifies are easily extendable through 2021. This would require more reserch of the users part, making the whole point of the visual somewhat moot.

4.4 Recommendations

Data about income and how it relates to emissions would be beneficial for the government, banking and activist sectors. Specifically, Chase Bank, which has a large global footprint, could use this money to inform some of its investment strategies in the modern world.

1. I recommend that Chase invest in green energy and technology companies in the countries with the hightest emission footprints. It's clear from the visual that emissions are increasing over time and from external research that the costs of emissions to banks are real. This means there is monetary insentive to invest in technologies that mitigate the emissions problem as the visual shows this has an effect on emissions. Moreover, there is peace of mind knowing this is a good investment as total emissions keep trending up.
2. I'm out of time but I'm sure there are other good recommendations!!!

[ ]: