Group Assignment: PFG Bank (aka Capital One): Credit Card Design (4-8 hours)

Graded

Group

Gaetan Rieben
Albara Altoukhi
Mohannad Alsaegh
...and 1 more
View or edit group

Total Points

30 / 35 pts

Autograder Score

0.0 / 0.0

Question 2

1. Why does Customer Lifetime Value vary with BK score? Why does Customer Lifetime Value vary by product? **3** / 3 pts

✔ **– 0 pts** Correct

Question 3

2. Are predictive models estimated on historical data useful in this case? If so, why? If not, why not? **4** / 4 pts

✔ **– 0 pts** Correct

Question 4

3. Is there a "best product" that will likely be preferred by all customers? If so, what is it? **3** / 3 pts

✔ **– 0 pts** Correct

Question 5

4. Describe and justify your testing strategy **10** / 10 pts

✔ **– 0 pts** Correct

Question 6

5. Performance **7** / 10 pts

✔ **+ 7 pts** Performance score

Question 7

6. Usage of Gen AI **3** / 5 pts

✔ **+ 3 pts** Solid discussion of your teams Gen AI approach

Autograder Results

This assignment does not have an autograder configured.

Submitted Files

.github/workflows/run-code.yml

Download

```yaml
name: GitHub Case - PFG Bank (CI)
on: [push, pull_request, fork]
jobs:
  build:
    runs-on: self-hosted
    container:
      image: vnijs/rsm-msba-intel:2.9.2
      options: --user root
    steps:
      - uses: actions/checkout@v2
      - name: Upgrade pyrsm
        run: pip install --upgrade pyrsm
      - name: Evaluate notebook
        run: jupyter nbconvert --execute --to html pfg-bank.ipynb
```

.gitignore

Download

```
.Rproj.user
.Rhistory
.RData
.Ruserdata
.DS_Store
*.pyc
```

data/exhibits.xls

Download

Binary file hidden. You can download it using the button above.

data/test.xlsx

Download

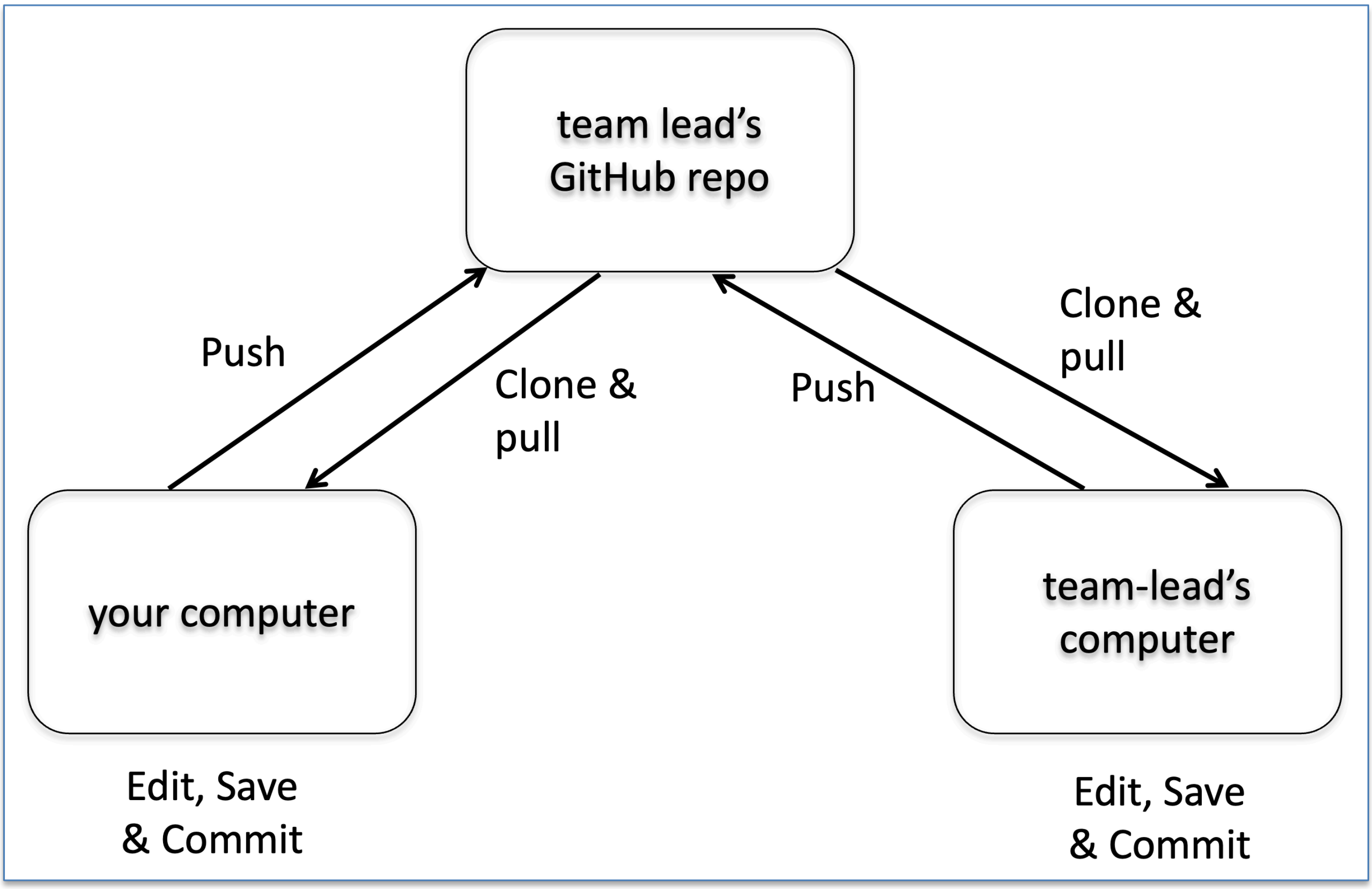
Binary file hidden. You can download it using the button above.

▾ images/full_design.png Download

```
Attributes and levels:
APR: 14.9, 16.8, 19.8
Fee: Yes, No
rate: fixed, variable

Design efficiency:
 Trials D-efficiency Balanced
      5        0.135    FALSE
      6        0.819     TRUE
      7        0.670    FALSE
      8        0.819    FALSE
      9        0.705    FALSE
     10        0.651    FALSE
     11        0.565    FALSE
     12        1.000     TRUE
```

▾ images/github-group-workflow.png

Download



▾ images/partial_design.png Download

```
Partial factorial design correlations:
** Note: Variables are assumed to be ordinal **
     APR  Fee rate
APR    1  0.0  0.0
Fee    0  1.0 -0.5
rate   0 -0.5  1.0

Partial factorial design:
 trial  APR Fee     rate
     2 14.9 Yes variable
     3 14.9  No    fixed
     5 16.8 Yes    fixed
     8 16.8  No variable
    10 19.8 Yes variable
    11 19.8  No    fixed
```

Summary

▾ images/results.png Download

Analyze | Final Results

Round 1 Results

Solicitation Development	$15,000	Round 1 Number Sent	66,312
Mailing Costs	$800	Round 1 Number of Responses	1,152
Cost of Pieces Mailed	$33,156	Total Response Value	$55,542
Total Cost of Mailing	$48,956		
		Total Profit	$6,586

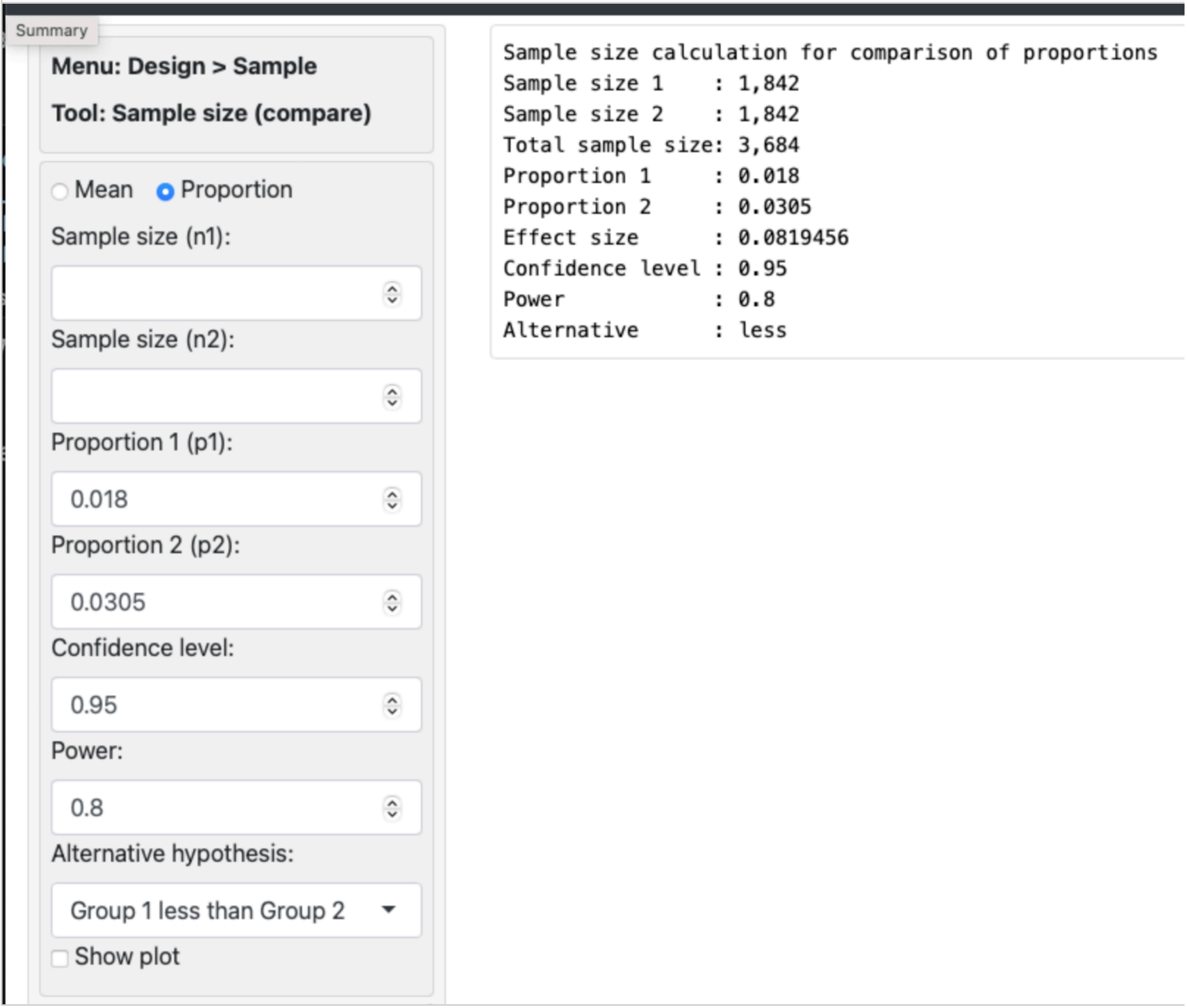
Round 2 Results

		Round 2 Number Sent	683,688
Mailing Costs	$800	Round 2 Number of Responses	23,989
Cost of Pieces Mailed	$341,844	Total Response Value	$1,011,378
Total Cost of Mailing	$342,644		
		Total Profit	$668,734

Cumulative Results

Solicitation Development	$15,000	Round 1 Number of Responses	1,152
Mailing Costs	$1,600	Round 2 Number of Responses	23,989
Cost of Pieces Mailed	$375,000	Total Response Value	$1,066,920
Total Cost of Mailing	$391,600		
		Total Profit	$675,320

▾ images/sample_size.png Download



pfg-bank-msba.pdf

Download

Your browser does not support PDF previews. You can download the file instead.

▾ pfg-bank.ipynb Download

PFG Bank: Credit Card Design

- Team-lead GitHub userid: rsm-xyz123
- Group name: Group 5
- Team member names:
  - Gaetan Rieben
  - Albara Altoukhi
  - Mohib Mohyuddin
  - Mohannad Alsaegh

Setup

Please complete this python notebook with your group by answering the questions in `pfg-bank-msba.pdf`.

Create a Notebook with all your results and comments and push the Notebook to GitHub when your team is done. Make sure to connect the GitHub repo to GradeScope before the due date. All results MUST be reproducible (i.e., the TA and I must be able to recreate your output from the Jupyter Notebook without changes or errors). This means that you should NOT use any python-packages that are not part of the RSM-MSBA docker container.

> Note: Please do not install any packages as part of your Jupyter Notebook submission

This is a group assignment and you will be using Git and GitHub. If two people edit the same file at the same time you could get what is called a "merge conflict". This is not something serious but you should realize that Git will not decide for you who's changes to accept. The team-lead will have to determine the edits to use. To avoid merge conflicts, **always** "pull" changes to the repo before you start working on any files. Then, when you are done, save and commit your changes, and then push them to GitHub. Make "pull first" a habit!

If multiple people are going to work on the assignment at the same time I recommend you work in different notebooks. You can then `%run ...` these "sub" notebooks from the main assignment file. You can seen an example of this in action below for the `model1.ipynb` notebook

Some group work-flow tips:

- Pull, edit, save, stage, commit, and push

- Schedule who does what and when
- Try to avoid working simultaneously on the same file
- If you are going to work simultaneously, do it in different notebooks, e.g.,
  - model1.ipynb, question1.ipynb, etc.
- Use the `%run ...` command to bring different pieces of code together into the main jupyter notebook
- Put python functions in modules that you can import from your notebooks. See the example below for the `example` function defined in `utils/functions.py`

A graphical depiction of the group work-flow is shown below:

In [1]:

```
import pandas as pd
import pyrsm as rsm
rsm.__version__ # should be 0.9.23 or newer
```

Out [1]:

```
'0.9.24'
```

Question answers

Why does Customer Lifetime Value vary with BK score? Why does Customer Lifetime Value vary by product? (See Exhibit 2 to help answer these questions)

Banks limit exposure to high bk/credit risk customers by offering them products with higher interest rates, lower credit limits, or more stringent repayment terms which impacts their CLV. Also, banks are typically conservative; minimize potential losses through management of the account rather than maximizing revenue through upselling or cross-selling additional products and services. Banks incur additional cost for defaulting customers which includes collections, account monitoring, and legal proceedings which can diminish the net CLV overtime. Finally, customers with lower scores may exhibit more cautious financial behavior, affecting their utilization of banking products and services. Conversely, those with better scores may be more financially active, using a broader range of products and engaging in more transactions, potentially increasing their CLV.

In [12]:

```
###Loading past data
import pandas as pd
import pyrsm as rsm

exh1 = pd.read_excel('data/exhibits.xls', sheet_name='exhibit1', dtype=
{'apr':'category','fixed_var':'category', 'annual_fee':'category', 'visamc':'category',
'nr_mailed':'int', 'non_resp':'int', 'resp':'int', 'bk_score':'category', 'average_bk':
'category'})
```

In [13]:

```
exh_melt = pd.melt(
    exh1,
    id_vars=['apr', 'fixed_var', 'annual_fee', 'bk_score'],
    value_vars=['resp', 'non_resp'],
    var_name='resp',
    value_name='freq'
)
```

In [14]:

```
exh_melt
```

Out [14]:

	apr	fixed_var	annual_fee	bk_score	resp	freq
0	16.8	Fixed	20	200	resp	1533
1	16.8	Fixed	0	200	resp	2896
2	19.8	Fixed	20	200	resp	590
3	19.8	Fixed	0	200	resp	2052
4	14.9	Fixed	20	250	resp	4329
5	14.9	Variable	20	250	resp	3004
6	16.8	Fixed	20	250	resp	2983
7	19.8	Fixed	20	250	resp	175
8	16.8	Fixed	0	250	resp	2516
9	19.8	Fixed	0	250	resp	2115
10	14.9	Fixed	20	150	resp	1761
11	14.9	Fixed	0	150	resp	2451
12	14.9	Variable	20	150	resp	708
13	16.8	Fixed	20	150	resp	373
14	16.8	Fixed	20	200	non_resp	165467
15	16.8	Fixed	0	200	non_resp	78104
16	19.8	Fixed	20	200	non_resp	142410
17	19.8	Fixed	0	200	non_resp	97948
18	14.9	Fixed	20	250	non_resp	172671
19	14.9	Variable	20	250	non_resp	166996
20	16.8	Fixed	20	250	non_resp	252017
21	19.8	Fixed	20	250	non_resp	34825
22	16.8	Fixed	0	250	non_resp	62484
23	19.8	Fixed	0	250	non_resp	92885
24	14.9	Fixed	20	150	non_resp	80239
25	14.9	Fixed	0	150	non_resp	47549

```
26  14.9  Variable     20     150  non_resp   49292
27  16.8     Fixed     20     150  non_resp   49627
```

In [24]:

```
lr = rsm.model.logistic(
    data={'exh_melt': exh_melt},
        rvar='resp',
        lev='resp',
        evar=['apr', 'fixed_var', 'annual_fee', 'bk_score'],
        weights='freq')
```

In [26]:

```
lr.summary()
```

```
Logistic regression (GLM)
Data                 : exh_melt
Response variable    : resp
Level                : resp
Explanatory variables: apr, fixed_var, annual_fee, bk_score
Weights used         : freq
Null hyp.: There is no effect of x on resp
Alt. hyp.: There is an effect of x on resp

                        OR     OR%  coefficient  std.error  z.value p.value
Intercept            0.060  -94.0%        -2.81      0.016 -175.303  < .001  ***
apr[16.8]            0.471  -52.9%        -0.75      0.019  -39.875  < .001  ***
apr[19.8]            0.257  -74.3%        -1.36      0.024  -57.254  < .001  ***
fixed_var[Variable]  0.741  -25.9%        -0.30      0.021  -14.336  < .001  ***
annual_fee[20]       0.290  -71.0%        -1.24      0.015  -84.443  < .001  ***
bk_score[200]        1.232   23.2%         0.21      0.024    8.519  < .001  ***
bk_score[250]        1.425   42.5%         0.35      0.019   18.388  < .001  ***

Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Pseudo R-squared (McFadden): 0.033
Pseudo R-squared (McFadden adjusted): 0.033
Area under the RO Curve (AUC): 0.662
Log-likelihood: -132938.55, AIC: 265891.101, BIC: 265976.74
Chi-squared: 9185.877, df(6), p.value < 0.001
Nr obs: 1,520,000
```

In [27]:

```
dct = rsm.levels_list(exh_melt[['apr', 'fixed_var', 'annual_fee', 'bk_score']])
dct

exh1_mult_expand = rsm.expand_grid(dct)
exh1_mult_expand
```

Out [27]:

```
    apr fixed_var  annual_fee  bk_score
0   16.8     Fixed         20       200
```

1	16.8	Fixed	20	250
2	16.8	Fixed	20	150
3	16.8	Fixed	0	200
4	16.8	Fixed	0	250
5	16.8	Fixed	0	150
6	16.8	Variable	20	200
7	16.8	Variable	20	250
8	16.8	Variable	20	150
9	16.8	Variable	0	200
10	16.8	Variable	0	250
11	16.8	Variable	0	150
12	19.8	Fixed	20	200
13	19.8	Fixed	20	250
14	19.8	Fixed	20	150
15	19.8	Fixed	0	200
16	19.8	Fixed	0	250
17	19.8	Fixed	0	150
18	19.8	Variable	20	200
19	19.8	Variable	20	250
20	19.8	Variable	20	150
21	19.8	Variable	0	200
22	19.8	Variable	0	250
23	19.8	Variable	0	150
24	14.9	Fixed	20	200
25	14.9	Fixed	20	250
26	14.9	Fixed	20	150
27	14.9	Fixed	0	200
28	14.9	Fixed	0	250
29	14.9	Fixed	0	150
30	14.9	Variable	20	200
31	14.9	Variable	20	250
32	14.9	Variable	20	150
33	14.9	Variable	0	200
34	14.9	Variable	0	250
35	14.9	Variable	0	150

In [28]:

```
exh1_mult_expand['pred'] = lr.predict(data=exh1_mult_expand)['prediction']
exh1_mult_expand
```

Out [28]:

	apr	fixed_var	annual_fee	bk_score	pred
0	16.8	Fixed	20	200	0.010029
1	16.8	Fixed	20	250	0.011581
2	16.8	Fixed	20	150	0.008156
3	16.8	Fixed	0	200	0.033799
4	16.8	Fixed	0	250	0.038884
5	16.8	Fixed	0	150	0.027609
6	16.8	Variable	20	200	0.007448
7	16.8	Variable	20	250	0.008604
8	16.8	Variable	20	150	0.006054
9	16.8	Variable	0	200	0.025257

```
10  16.8  Variable         0      250  0.029096
11  16.8  Variable         0      150  0.020598
12  19.8     Fixed        20      200  0.005496
13  19.8     Fixed        20      250  0.006351
14  19.8     Fixed        20      150  0.004466
15  19.8     Fixed         0      200  0.018725
16  19.8     Fixed         0      250  0.021593
17  19.8     Fixed         0      150  0.015252
18  19.8  Variable        20      200  0.004077
19  19.8  Variable        20      250  0.004712
20  19.8  Variable        20      150  0.003312
21  19.8  Variable         0      200  0.013938
22  19.8  Variable         0      250  0.016084
23  19.8  Variable         0      150  0.011342
24  14.9     Fixed        20      200  0.021044
25  14.9     Fixed        20      250  0.024258
26  14.9     Fixed        20      150  0.017148
27  14.9     Fixed         0      200  0.069096
28  14.9     Fixed         0      250  0.079057
29  14.9     Fixed         0      150  0.056822
30  14.9  Variable        20      200  0.015673
31  14.9  Variable        20      250  0.018083
32  14.9  Variable        20      150  0.012759
33  14.9  Variable         0      200  0.052115
34  14.9  Variable         0      250  0.059786
35  14.9  Variable         0      150  0.042719
```

In [34]:

```
# Convert the list of dictionaries to a pandas DataFrame
df = pd.DataFrame(exh1_mult_expand)

# Filter the DataFrame for bk_score = 200 and find the row with the maximum pred value
filtered_df = df[df['bk_score'] == 200]
max_pred_row_alternative = filtered_df.loc[filtered_df['pred'].idxmax()]
print(max_pred_row_alternative)
```

```
apr              14.9
fixed_var       Fixed
annual_fee          0
bk_score          200
pred         0.069096
Name: 27, dtype: object
```

In [35]:

```
# Filter the DataFrame for bk_score = 200 and find the row with the maximum pred value
filtered_df = df[df['bk_score'] == 250]
max_pred_row_alternative = filtered_df.loc[filtered_df['pred'].idxmax()]
print(max_pred_row_alternative)
```

```
apr            14.9
fixed_var      Fixed
annual_fee         0
bk_score         250
pred        0.079057
Name: 28, dtype: object
```

In [36]:

```
# Filter the DataFrame for bk_score = 200 and find the row with the maximum pred value
filtered_df = df[df['bk_score'] == 200]
max_pred_row_alternative = filtered_df.loc[filtered_df['pred'].idxmax()]
print(max_pred_row_alternative)
```

```
apr            14.9
fixed_var      Fixed
annual_fee         0
bk_score         150
pred        0.056822
Name: 29, dtype: object
```

Based on the logistic regression conducted on previous data, we can see that the offering should be the same for each BK_score customers. As far as we can see, the lowest APR, with a fixed interest rate and no annual fee is the best offering that maximize the likelihood of conversion. However, it might be useful to check if those offering are the one who maximize profitability (CLV * probability).

Although this product might not be the highest profitability, this product is preferred by all customers as per this past data analysis (*Answer to question 3*). This makes sense as this product has the lowest (and fixed) interest rate with no annual fee. This is by far the most affordable product for the clients. It might be interesting to compare this offering with the ones from competitors.

Our testing strategy

Given the fact that these mailing has been sent in the past but the situation has evolved, we believe that the customer responses might has changed given the market. Indeed, many factors beyond PFG's control had changed and could affect customers' responses. Several of PFG's competitors had launched major marketing campaigns during the holiday season. One competitor was aggressively marketing no-fee cards and a second was offering a substantial rebate program. (Answer question 2). Furthermore, the data are not representative of experiments/design nuances. The experiment design was not balanced. For instance, in September, they have sent offers only with a fixed rate. They haven't tired with variable rate. As such, we want to design the experiments properly.

Therefore, we are willing to conduct a test mailing and analyze the results from it. But first, as we want to design the most optimal one, we want to use a partial factor design that would allow us to select the most balanced offerings. To do so, we use radiant

From the result, we can see that only 2 solutions might be balanced. The one with 6 trials, or the one with 12 trials. As such, as we want to minimize cost, we decided to select the one with a D-Efficiency closer to 0.8. We will go with 6 trials. See below the product designs to be tested

Calculate sample size

In my opinion, we should check the response rate of previous mailings, and set a higher response rate that we are willing to achieve in order to compute the sample size for our tests

In [40]:

```
resp = exh_melt[exh_melt['resp']=='resp']
non_resp = exh_melt[exh_melt['resp']=='non_resp']
```

In [43]:

```
total_resp = sum(resp['freq'])
total_send = sum(resp['freq']) + sum(non_resp['freq'])
```

In [45]:

```
response_rate = total_resp / total_send
print(response_rate)
```

```
0.018082894736842107
```

The current response rate is 1.80%. We will check what response rate we would need considering the CLV's. As such, we plan to take the average CLV across all products and all BK scores and use this average to compute the breakeven

In [169]:

```
exh2 = pd.read_excel('data/exhibits.xls', sheet_name='exhibit2', dtype=
{'offer':'category','apr':'category', 'fixed_var':'category', 'annual_fee':'category',
'clv150':'int', 'clv200':'int', 'clv250':'int'})
clv150, clv200, clv250= exh2[['clv150', 'clv200', 'clv250']].mean()
clv150
```

Out [169]:

```
95.16666666666667
```

In [170]:

```
mean = (clv150 + clv200 + clv250) / 3
mean
```

Out [170]:

```
71.83333333333333
```

In [171]:

```
fixed = 10000+1600+6000
cost_per_customer = fixed/750000
breakeven = cost_per_customer + 0.50 / mean
breakeven
```

Out [171]:

```
0.03042722351121423
```

As per the analysis above, we should go from a response rate of 1.80% to 3.05%. Using this number, we use radiants to estimate the sample size for our test mailing

Now that we have estimated the sample size to launch our test, we have to decide if we send a test mailing to everyone or only a few product offerings. As per our idea, we want to produce the most efficient test campaign at the lowest cost. Hence, we will send a test campaign with our 6 product design to each of the BK_score, with a sample size as above.

Results from the test

In [127]:

```
test = pd.read_excel('data/test.xlsx')
```

In [128]:

```
test.head()
```

Out [128]:

```
   apr fixed_var annual_fee  sent_150  responses_150  sent_200  \
0  14.9  Variable       Yes      3684             37      3684
```

```
1  14.9     Fixed        No     3684          144     3684
2  16.8     Fixed       Yes     3684            9     3684
3  16.8  Variable        No     3684           50     3684
4  19.8  Variable       Yes     3684           14     3684

   responses_200  sent_250  responses_250
0             51      3684             63
1            170      3684            228
2             30      3684             23
3             74      3684             76
4             13      3684             12
```

In [129]:

```
# Melting the DataFrame
sent_melted = test.melt(id_vars=["apr", "fixed_var", "annual_fee"], value_vars=
["sent_150", "sent_200", "sent_250"], var_name="bk_score", value_name="sent")
responses_melted = test.melt(id_vars=["apr", "fixed_var", "annual_fee"],
value_vars=["responses_150", "responses_200", "responses_250"],
var_name="bk_score", value_name="responses")

# Adjusting the bk_score values to reflect just the score
sent_melted['bk_score'] = sent_melted['bk_score'].str.replace('sent_', '')
responses_melted['bk_score'] =
responses_melted['bk_score'].str.replace('responses_', '')

# Merging the melted DataFrames
test = pd.merge(sent_melted, responses_melted, on=["apr", "fixed_var",
"annual_fee", "bk_score"])

print(test)
```

```
     apr fixed_var annual_fee bk_score  sent  responses
0   14.9  Variable        Yes      150  3684         37
1   14.9     Fixed         No      150  3684        144
2   16.8     Fixed        Yes      150  3684          9
3   16.8  Variable         No      150  3684         50
4   19.8  Variable        Yes      150  3684         14
5   19.8     Fixed         No      150  3684         51
6   14.9  Variable        Yes      200  3684         51
7   14.9     Fixed         No      200  3684        170
8   16.8     Fixed        Yes      200  3684         30
9   16.8  Variable         No      200  3684         74
10  19.8  Variable        Yes      200  3684         13
11  19.8     Fixed         No      200  3684         48
12  14.9  Variable        Yes      250  3684         63
13  14.9     Fixed         No      250  3684        228
14  16.8     Fixed        Yes      250  3684         23
15  16.8  Variable         No      250  3684         76
16  19.8  Variable        Yes      250  3684         12
17  19.8     Fixed         No      250  3684         59
```

In [130]:

```
test['no_resp'] = test['sent'] - test['responses']
```

In [131]:

```
test
```

Out [131]:

	apr	fixed_var	annual_fee	bk_score	sent	responses	no_resp
0	14.9	Variable	Yes	150	3684	37	3647
1	14.9	Fixed	No	150	3684	144	3540
2	16.8	Fixed	Yes	150	3684	9	3675
3	16.8	Variable	No	150	3684	50	3634
4	19.8	Variable	Yes	150	3684	14	3670
5	19.8	Fixed	No	150	3684	51	3633
6	14.9	Variable	Yes	200	3684	51	3633
7	14.9	Fixed	No	200	3684	170	3514
8	16.8	Fixed	Yes	200	3684	30	3654
9	16.8	Variable	No	200	3684	74	3610
10	19.8	Variable	Yes	200	3684	13	3671
11	19.8	Fixed	No	200	3684	48	3636
12	14.9	Variable	Yes	250	3684	63	3621
13	14.9	Fixed	No	250	3684	228	3456
14	16.8	Fixed	Yes	250	3684	23	3661
15	16.8	Variable	No	250	3684	76	3608
16	19.8	Variable	Yes	250	3684	12	3672
17	19.8	Fixed	No	250	3684	59	3625

In [132]:

```
test = test.melt(id_vars=["apr", "fixed_var", "annual_fee", "bk_score"], value_vars=
["no_resp", "responses"], var_name="resp", value_name="freq")
```

In [133]:

```
test
```

Out [133]:

	apr	fixed_var	annual_fee	bk_score	resp	freq
0	14.9	Variable	Yes	150	no_resp	3647
1	14.9	Fixed	No	150	no_resp	3540
2	16.8	Fixed	Yes	150	no_resp	3675
3	16.8	Variable	No	150	no_resp	3634
4	19.8	Variable	Yes	150	no_resp	3670
5	19.8	Fixed	No	150	no_resp	3633
6	14.9	Variable	Yes	200	no_resp	3633
7	14.9	Fixed	No	200	no_resp	3514
8	16.8	Fixed	Yes	200	no_resp	3654
9	16.8	Variable	No	200	no_resp	3610
10	19.8	Variable	Yes	200	no_resp	3671
11	19.8	Fixed	No	200	no_resp	3636
12	14.9	Variable	Yes	250	no_resp	3621
13	14.9	Fixed	No	250	no_resp	3456
14	16.8	Fixed	Yes	250	no_resp	3661
15	16.8	Variable	No	250	no_resp	3608

```
16  19.8  Variable    Yes    250    no_resp  3672
17  19.8     Fixed     No    250    no_resp  3625
18  14.9  Variable    Yes    150  responses    37
19  14.9     Fixed     No    150  responses   144
20  16.8     Fixed    Yes    150  responses     9
21  16.8  Variable     No    150  responses    50
22  19.8  Variable    Yes    150  responses    14
23  19.8     Fixed     No    150  responses    51
24  14.9  Variable    Yes    200  responses    51
25  14.9     Fixed     No    200  responses   170
26  16.8     Fixed    Yes    200  responses    30
27  16.8  Variable     No    200  responses    74
28  19.8  Variable    Yes    200  responses    13
29  19.8     Fixed     No    200  responses    48
30  14.9  Variable    Yes    250  responses    63
31  14.9     Fixed     No    250  responses   228
32  16.8     Fixed    Yes    250  responses    23
33  16.8  Variable     No    250  responses    76
34  19.8  Variable    Yes    250  responses    12
35  19.8     Fixed     No    250  responses    59
```

Logistic regression on the latest test

In [134]:

```
test['apr'] = test['apr'].astype('category')
```

In [149]:

```
evar=['apr', 'fixed_var', 'annual_fee', 'bk_score']
```

In [151]:

```
ivar=[f'{e}:bk_score' for e in evar if e != 'bk_score']
ivar
```

Out [151]:

```
['apr:bk_score', 'fixed_var:bk_score', 'annual_fee:bk_score']
```

In [155]:

```
lr = rsm.model.logistic(
    data={'test': test},
        rvar='resp',
        lev='responses',
        evar=['apr', 'fixed_var', 'annual_fee', 'bk_score'],
        weights='freq')
```

In [156]:

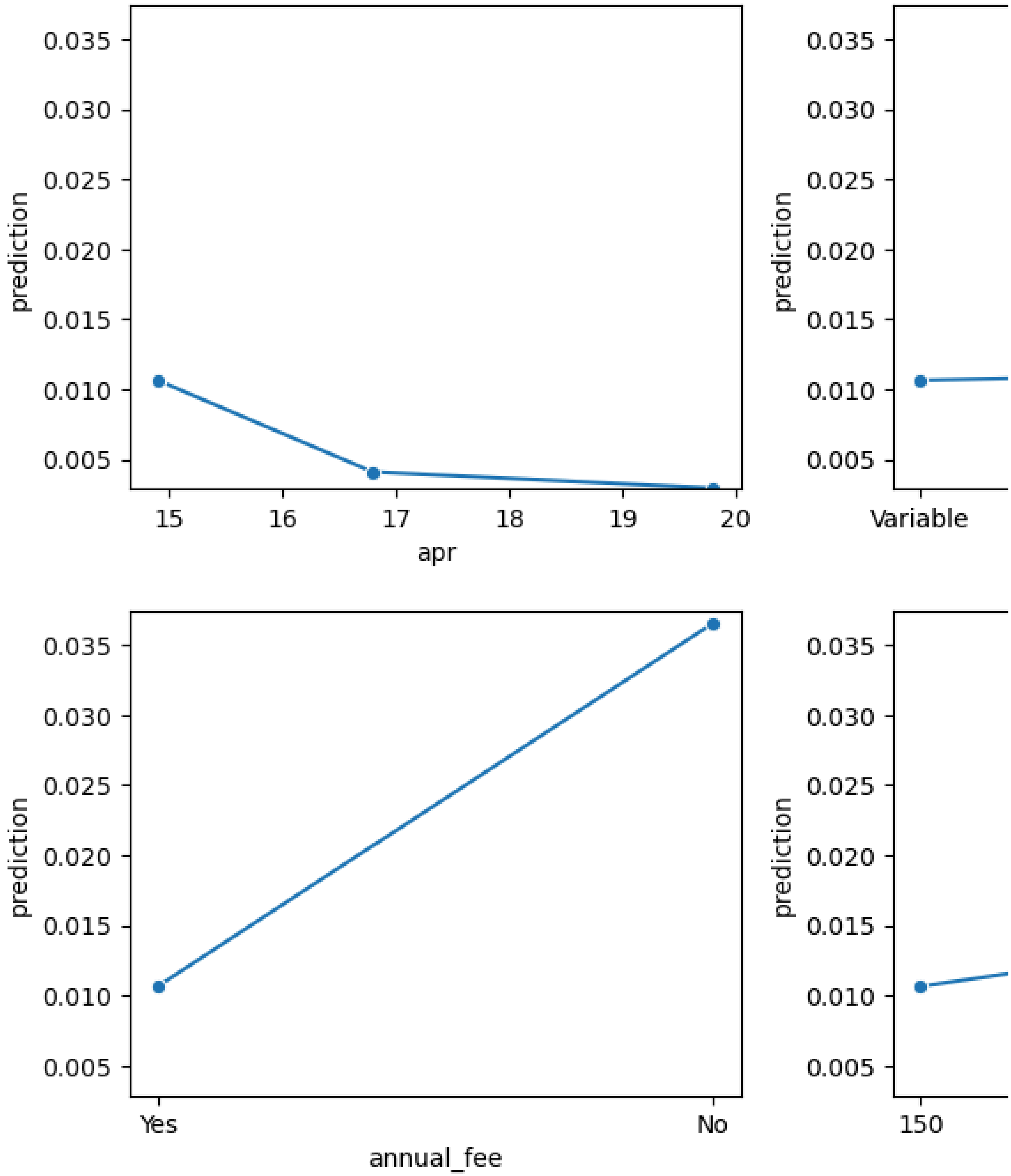
```
lr.summary()
```

```
Logistic regression (GLM)
Data                 : test
```

```
Response variable    : resp
Level                : responses
Explanatory variables: apr, fixed_var, annual_fee, bk_score
Weights used         : freq
Null hyp.: There is no effect of x on resp
Alt. hyp.: There is an effect of x on resp

                       OR     OR%  coefficient  std.error  z.value p.value
Intercept           0.041  -95.9%        -3.20      0.065  -48.807  < .001  ***
apr[16.8]           0.383  -61.7%        -0.96      0.086  -11.167  < .001  ***
apr[19.8]           0.275  -72.5%        -1.29      0.082  -15.815  < .001  ***
fixed_var[Variable] 0.927   -7.3%        -0.08      0.084   -0.906   0.365
annual_fee[Yes]     0.284  -71.6%        -1.26      0.084  -15.025  < .001  ***
bk_score[200]       1.274   27.4%         0.24      0.078    3.118   0.002   **
bk_score[250]       1.531   53.1%         0.43      0.075    5.685  < .001  ***

Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Pseudo R-squared (McFadden): 0.068
Pseudo R-squared (McFadden adjusted): 0.067
Area under the RO Curve (AUC): 0.726
Log-likelihood: -5415.446, AIC: 10844.892, BIC: 10908.607
Chi-squared: 790.793, df(6), p.value < 0.001
Nr obs: 66,312
```

In [159]:

```
lr.plot('pred')
```



We want to try interactions to see if it has an influence on the logistic regression

In [157]:

```
lr_ivar = rsm.model.logistic(
    data={'test': test},
        rvar='resp',
        lev='responses',
        evar=['apr', 'fixed_var', 'annual_fee', 'bk_score'],
        ivar=ivar,
        weights='freq')
```

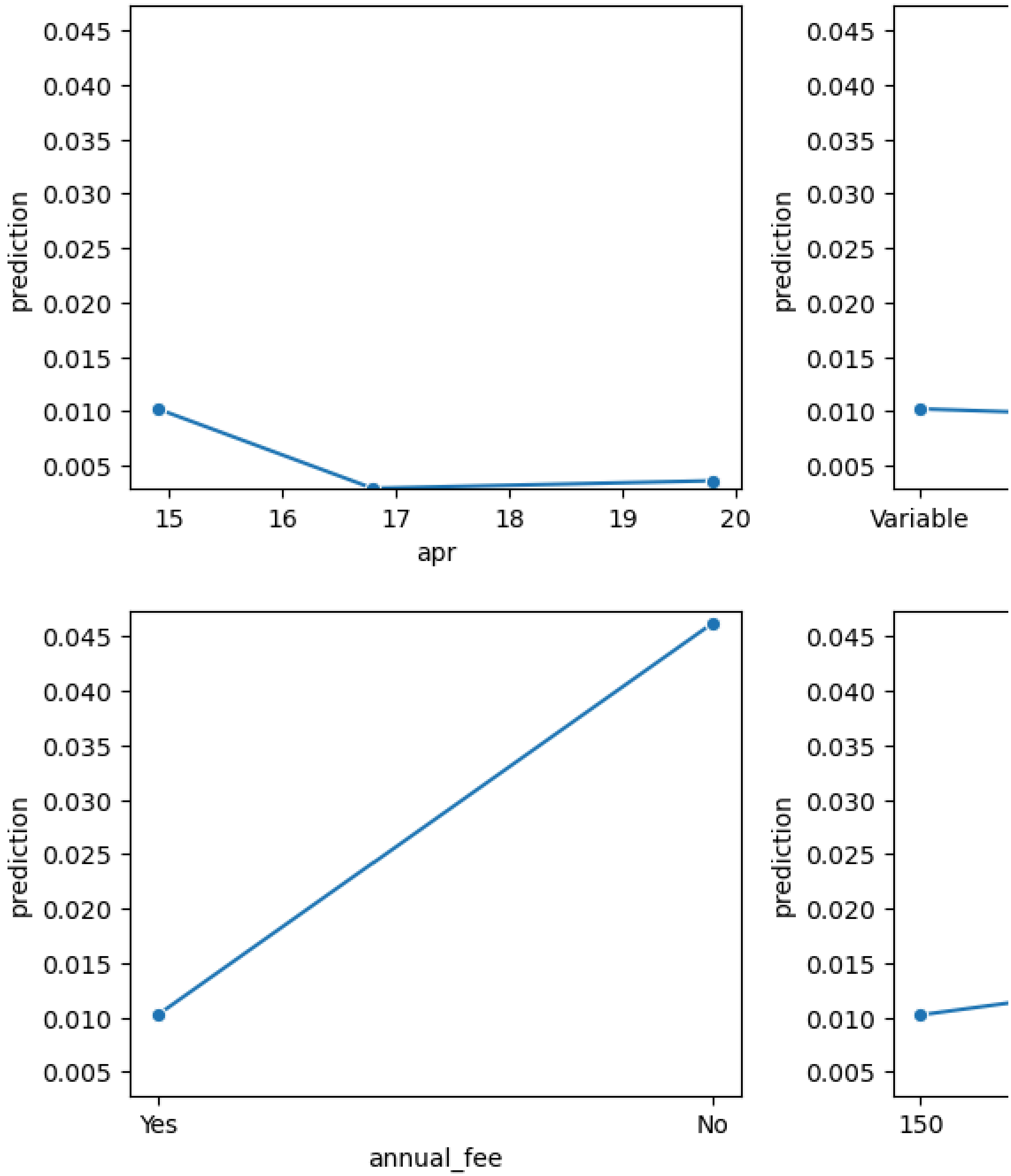
In [158]:

```
lr_ivar.summary()
```

```
Logistic regression (GLM)
Data                 : test
Response variable    : resp
Level                : responses
Explanatory variables: apr, fixed_var, annual_fee, bk_score
Weights used         : freq
Null hyp.: There is no effect of x on resp
Alt. hyp.: There is an effect of x on resp

                                  OR     OR%  coefficient  std.error  z.value p.value
Intercept                      0.040  -96.0%        -3.21      0.083  -38.745  < .001  ***
apr[16.8]                      0.284  -71.6%        -1.26      0.202   -6.243  < .001  ***
apr[19.8]                      0.351  -64.9%        -1.05      0.146   -7.168  < .001  ***
fixed_var[Variable]            1.197   19.7%         0.18      0.198    0.910   0.363
annual_fee[Yes]                0.213  -78.7%        -1.55      0.198   -7.809  < .001  ***
bk_score[200]                  1.200   20.0%         0.18      0.113    1.617   0.106
bk_score[250]                  1.648   64.8%         0.50      0.106    4.702  < .001  ***
apr[16.8]:bk_score[200]        1.762   76.2%         0.57      0.242    2.337   0.019    *
apr[19.8]:bk_score[200]        0.763  -23.7%        -0.27      0.206   -1.311    0.19
apr[16.8]:bk_score[250]        1.213   21.3%         0.19      0.245    0.789    0.43
apr[19.8]:bk_score[250]        0.666  -33.4%        -0.41      0.198   -2.053    0.04    *
fixed_var[Variable]:bk_score[200] 0.705  -29.5%     -0.35      0.237   -1.476    0.14
fixed_var[Variable]:bk_score[250] 0.766  -23.4%     -0.27      0.240   -1.110   0.267
annual_fee[Yes]:bk_score[200]  1.586   58.6%         0.46      0.237    1.946   0.052    .
annual_fee[Yes]:bk_score[250]  1.283   28.3%         0.25      0.240    1.038   0.299

Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Pseudo R-squared (McFadden): 0.069
Pseudo R-squared (McFadden adjusted): 0.067
Area under the RO Curve (AUC): 0.728
Log-likelihood: -5407.927, AIC: 10845.853, BIC: 10982.385
Chi-squared: 805.832, df(14), p.value < 0.001
Nr obs: 66,312
```

In [160]:

```
lr_ivar.plot('pred')
```



In [137]:

```
dct = rsm.levels_list(test[['apr', 'fixed_var', 'annual_fee', 'bk_score']])
dct

test_mult_expand = rsm.expand_grid(dct)
test_mult_expand
```

Out [137]:

	apr	fixed_var	annual_fee	bk_score
0	14.9	Variable	Yes	150
1	14.9	Variable	Yes	200
2	14.9	Variable	Yes	250
3	14.9	Variable	No	150
4	14.9	Variable	No	200
5	14.9	Variable	No	250
6	14.9	Fixed	Yes	150
7	14.9	Fixed	Yes	200

```
8   14.9    Fixed     Yes    250
9   14.9    Fixed     No     150
10  14.9    Fixed     No     200
11  14.9    Fixed     No     250
12  16.8  Variable    Yes    150
13  16.8  Variable    Yes    200
14  16.8  Variable    Yes    250
15  16.8  Variable    No     150
16  16.8  Variable    No     200
17  16.8  Variable    No     250
18  16.8    Fixed     Yes    150
19  16.8    Fixed     Yes    200
20  16.8    Fixed     Yes    250
21  16.8    Fixed     No     150
22  16.8    Fixed     No     200
23  16.8    Fixed     No     250
24  19.8  Variable    Yes    150
25  19.8  Variable    Yes    200
26  19.8  Variable    Yes    250
27  19.8  Variable    No     150
28  19.8  Variable    No     200
29  19.8  Variable    No     250
30  19.8    Fixed     Yes    150
31  19.8    Fixed     Yes    200
32  19.8    Fixed     Yes    250
33  19.8    Fixed     No     150
34  19.8    Fixed     No     200
35  19.8    Fixed     No     250
```

In [138]:

```
test_mult_expand['pred'] = lr.predict(data=test_mult_expand)['prediction']
df = test_mult_expand.copy()
```

Include CLV's value as per the exhibit 2 depending on the product offering

In [140]:

```
print(df.dtypes)
```

```
apr           float64
fixed_var      object
annual_fee     object
bk_score       object
pred          float64
dtype: object
```

In [141]:

```
df['apr'] = df['apr'].astype('float')
df['bk_score'] = df['bk_score'].astype('int')
```

In [142]:

```
import numpy as np
# Define the conditions and the respective CLV values
conditions = [
    (df['apr'] == 14.9) & (df['fixed_var'] == "Variable") & (df['annual_fee'] == "Yes") &
(df['bk_score'] == 150),
    (df['apr'] == 14.9) & (df['fixed_var'] == "Variable") & (df['annual_fee'] == "Yes") &
(df['bk_score'] == 200),
    (df['apr'] == 14.9) & (df['fixed_var'] == "Variable") & (df['annual_fee'] == "Yes") &
(df['bk_score'] == 250),
    (df['apr'] == 14.9) & (df['fixed_var'] == "Fixed") & (df['annual_fee'] == "No") &
(df['bk_score'] == 150),
    (df['apr'] == 14.9) & (df['fixed_var'] == "Fixed") & (df['annual_fee'] == "No") &
(df['bk_score'] == 200),
    (df['apr'] == 14.9) & (df['fixed_var'] == "Fixed") & (df['annual_fee'] == "No") &
(df['bk_score'] == 250),
    (df['apr'] == 16.8) & (df['fixed_var'] == "Fixed") & (df['annual_fee'] == "Yes") &
(df['bk_score'] == 150),
    (df['apr'] == 16.8) & (df['fixed_var'] == "Fixed") & (df['annual_fee'] == "Yes") &
(df['bk_score'] == 200),
    (df['apr'] == 16.8) & (df['fixed_var'] == "Fixed") & (df['annual_fee'] == "Yes") &
(df['bk_score'] == 250),
    (df['apr'] == 16.8) & (df['fixed_var'] == "Variable") & (df['annual_fee'] == "No") &
(df['bk_score'] == 150),
    (df['apr'] == 16.8) & (df['fixed_var'] == "Variable") & (df['annual_fee'] == "No") &
(df['bk_score'] == 200),
    (df['apr'] == 16.8) & (df['fixed_var'] == "Variable") & (df['annual_fee'] == "No") &
(df['bk_score'] == 250),
    (df['apr'] == 19.8) & (df['fixed_var'] == "Variable") & (df['annual_fee'] == "Yes") &
(df['bk_score'] == 150),
    (df['apr'] == 19.8) & (df['fixed_var'] == "Variable") & (df['annual_fee'] == "Yes") &
(df['bk_score'] == 200),
    (df['apr'] == 19.8) & (df['fixed_var'] == "Variable") & (df['annual_fee'] == "Yes") &
(df['bk_score'] == 250),
    (df['apr'] == 19.8) & (df['fixed_var'] == "Fixed") & (df['annual_fee'] == "No") &
(df['bk_score'] == 150),
    (df['apr'] == 19.8) & (df['fixed_var'] == "Fixed") & (df['annual_fee'] == "No") &
(df['bk_score'] == 200),
    (df['apr'] == 19.8) & (df['fixed_var'] == "Fixed") & (df['annual_fee'] == "No") &
(df['bk_score'] == 250),
]

choices = [93, 73, 43, 52, 32, 2, 103, 83, 53, 82, 62, 32, 141, 121, 91, 100, 80, 50]


# Applying the conditions to assign values to 'CLV'
df['CLV'] = np.select(conditions, choices)  # Using np.nan as default to clearly see
unmatched conditions
```

In [143]:

```
df['exp_profit'] = df['pred'] * df['CLV']
```

In [144]:

```
df
```

Out [144]:

	apr	fixed_var	annual_fee	bk_score	pred	CLV	exp_profit
0	14.9	Variable	Yes	150	0.010639	93	0.989396
1	14.9	Variable	Yes	200	0.013514	73	0.986538
2	14.9	Variable	Yes	250	0.016195	43	0.696406
3	14.9	Variable	No	150	0.036529	0	0.000000
4	14.9	Variable	No	200	0.046077	0	0.000000
5	14.9	Variable	No	250	0.054859	0	0.000000
6	14.9	Fixed	Yes	150	0.011468	0	0.000000
7	14.9	Fixed	Yes	200	0.014565	0	0.000000
8	14.9	Fixed	Yes	250	0.017451	0	0.000000
9	14.9	Fixed	No	150	0.039298	52	2.043491
10	14.9	Fixed	No	200	0.049532	32	1.585024
11	14.9	Fixed	No	250	0.058932	2	0.117865
12	16.8	Variable	Yes	150	0.004107	0	0.000000
13	16.8	Variable	Yes	200	0.005226	0	0.000000
14	16.8	Variable	Yes	250	0.006273	0	0.000000
15	16.8	Variable	No	150	0.014330	82	1.175090
16	16.8	Variable	No	200	0.018185	62	1.127496
17	16.8	Variable	No	250	0.021773	32	0.696737
18	16.8	Fixed	Yes	150	0.004429	103	0.456195
19	16.8	Fixed	Yes	200	0.005636	83	0.467770
20	16.8	Fixed	Yes	250	0.006765	53	0.358527
21	16.8	Fixed	No	150	0.015444	0	0.000000
22	16.8	Fixed	No	200	0.019592	0	0.000000
23	16.8	Fixed	No	250	0.023451	0	0.000000
24	19.8	Variable	Yes	150	0.002953	141	0.416381
25	19.8	Variable	Yes	200	0.003759	121	0.454855
26	19.8	Variable	Yes	250	0.004514	91	0.410757
27	19.8	Variable	No	150	0.010335	0	0.000000
28	19.8	Variable	No	200	0.013130	0	0.000000
29	19.8	Variable	No	250	0.015736	0	0.000000
30	19.8	Fixed	Yes	150	0.003185	0	0.000000
31	19.8	Fixed	Yes	200	0.004055	0	0.000000
32	19.8	Fixed	Yes	250	0.004868	0	0.000000
33	19.8	Fixed	No	150	0.011141	100	1.114137
34	19.8	Fixed	No	200	0.014151	80	1.132068
35	19.8	Fixed	No	250	0.016956	50	0.847813

In [162]:

```
# Filter the DataFrame for bk_score = 200 and find the row with the maximum pred value
filtered_df = df[df['bk_score'] == 200]
```

```
max_pred_row_alternative = filtered_df.loc[filtered_df['exp_profit'].idxmax()]
print(max_pred_row_alternative)
```

```
apr            14.9
fixed_var      Fixed
annual_fee        No
bk_score        200
pred        0.049532
CLV             32
exp_profit   1.585024
Name: 10, dtype: object
```

In [168]:

```
filtered_df = df[df['bk_score'] == 250]
max_pred_row_alternative = filtered_df.loc[filtered_df['exp_profit'].idxmax()]
print(max_pred_row_alternative)
```

```
apr            19.8
fixed_var      Fixed
annual_fee        No
bk_score        250
pred        0.016956
CLV             50
exp_profit   0.847813
Name: 35, dtype: object
```

In [165]:

```
# Filter the DataFrame for bk_score = 200 and find the row with the maximum
pred value
filtered_df = df[df['bk_score'] == 150]
max_pred_row_alternative = filtered_df.loc[filtered_df['exp_profit'].idxmax()]
print(max_pred_row_alternative)
```

```
apr            14.9
fixed_var      Fixed
annual_fee        No
bk_score        150
pred        0.039298
CLV             52
exp_profit   2.043491
Name: 9, dtype: object
```

Our strategy here is to multiply the predictions with the Customer LifeTime Value in order to compute the expected profit per email sent. Once computed, we have to look at the offering that yields the highest expected profit for each customer segment. Also, we made sure that the expected profit is higher than the cost of mailing per customer.

Based on these results, customers with BK scores 150 and 200 should receive the offering with the lowest APR and fixed rate, with no annual fee. The customers with a BK score of 250 should receive the offering with the highest APR, but still a fixed rate and no annual fee.

Please see below the results of the simluation.

CHATGPT use

Our chat used:

- https://chat.openai.com/share/f8eb7c1d-a0ec-4b56-bd83-a5f6311ea76a
- https://chat.openai.com/share/aef13a4c-ae5a-4435-8522-473e049171da

In tackling our case study, our team strategically leveraged Generative AI tools, notably ChatGPT, to enhance our research, analysis, and solution formulation processes. Our approach was multifaceted, aiming to capitalize on ChatGPT's capabilities to streamline our workflow, augment our creativity, and bolster our analytical depth.

- Research and Information Gathering:

We initiated our project by using ChatGPT to perform preliminary research. Given ChatGPT's expansive knowledge base, we queried it for background information on customer lifetime value and its relationship with bankruptcy scores in the banking sector. This step provided us with a foundational understanding and pointed us towards relevant financial models and analytical techniques. While ChatGPT offered broad overviews and insights, we noted its limitations in accessing the most current studies or data due to its training cut-off in April 2023. To mitigate this, we complemented AI-generated insights with current research from reputable sources, ensuring our project was informed by the most up-to-date information.

- Strategy Formulation:

Our strategy development greatly benefited from brainstorming sessions with ChatGPT. We discussed various strategic approaches, including predictive modeling and segmentation techniques, to address the case study's objectives. ChatGPT's instant responses to our queries about different strategies allowed us to quickly evaluate the pros and cons of each approach, significantly accelerating our decision-making process. However, we encountered challenges in assessing the feasibility of certain AI-suggested strategies, requiring us to critically evaluate each suggestion before incorporation into our plan.

- Drafting and Refinement:

For documentation and reporting, ChatGPT was instrumental in drafting sections of our report, generating well-articulated explanations of complex concepts, and suggesting layouts for presenting our findings. It enhanced our productivity by handling routine writing tasks, allowing us to focus on deeper analytical work. We did face challenges in ensuring the accuracy and relevance of AI-generated content, which necessitated thorough reviews and edits to align with our project's specific context.

- Analysis and Simulation:

We explored the potential of using AI for data analysis and simulation, discussing with ChatGPT various statistical techniques and models that could be applied to our data. ChatGPT provided code snippets and explanations for data manipulation and visualization, which we adapted for our analysis. However, the inability to execute code directly or access external databases through ChatGPT limited its utility for hands-on analysis, leading us to use other software tools for actual data processing.

- Maximizing Benefits and Mitigating Limitations:

Our thought process throughout the project was geared towards leveraging ChatGPT's strengths—speed, breadth of knowledge, and versatility—while being mindful of its limitations. We adopted a strategy of using ChatGPT for initial ideation, drafting, and knowledge checks, followed by a rigorous review process where we verified information, refined ideas, and applied professional judgment to ensure the accuracy and appropriateness of our final output.

In conclusion, our experience with using Generative AI tools like ChatGPT in this case study was predominantly positive, significantly enhancing our efficiency and creativity. The key to maximizing benefits from these tools was a balanced approach: leveraging AI for its strengths while critically evaluating its output and supplementing it with detailed research and expert analysis. This approach allowed us to navigate the limitations of AI and produce a comprehensive, informed, and nuanced case study solution.

rsm-mgta455-pfg-bank.code-workspace Download


```
{"folders": [{"path": "."}], "settings": {}}
```

sub-notebooks/model1.ipynb Download

In [ ]:

```
print("Add code as needed ...")
```

▾ using-r-and-python.ipynb   Download

In [1]:

```
import pandas as pd
```

In [2]:

```
%reload_ext rpy2.ipython
```

In [3]:

```
%%R
result <- radiant.design::sample_size_comp(
  type = "proportion",
  p1 = 0.51,
  p2 = 0.52,
  conf_lev = 0.95,
  power = 0.8
)
summary(result)
```

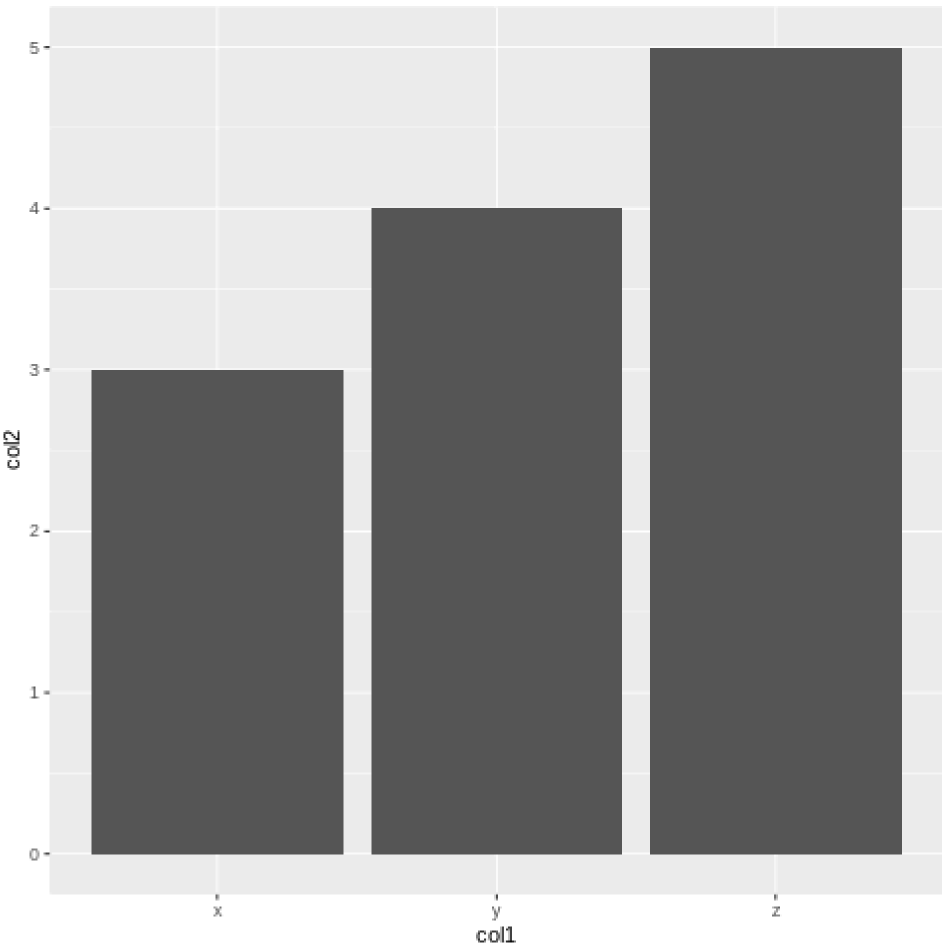
```
Sample size calculation for comparison of proportions
Sample size 1    : 39,208
Sample size 2    : 39,208
Total sample size: 78,416
Proportion 1     : 0.51
Proportion 2     : 0.52
Effect size      : 0.02000934
Confidence level : 0.95
Power            : 0.8
Alternative      : two.sided
```

In [4]:

```
dframe = pd.DataFrame({
    "col1": ["x", "y", "z"],
    "col2": [3, 4, 5]
})
```

In [5]:

```
%%R -i dframe
library(ggplot2)
p <- ggplot(dframe, aes(x=col1, y=col2)) + geom_bar(stat="identity")
print(p)
```



5
4
3
2
1
0
col2
x
y
z
col1

using-r-notebook.ipynb

Download

In [1]:

```
result <- radiant.design::sample_size_comp(
  type = "proportion",
  p1 = 0.51,
  p2 = 0.52,
  conf_lev = 0.95,
  power = 0.8
)
summary(result)
```

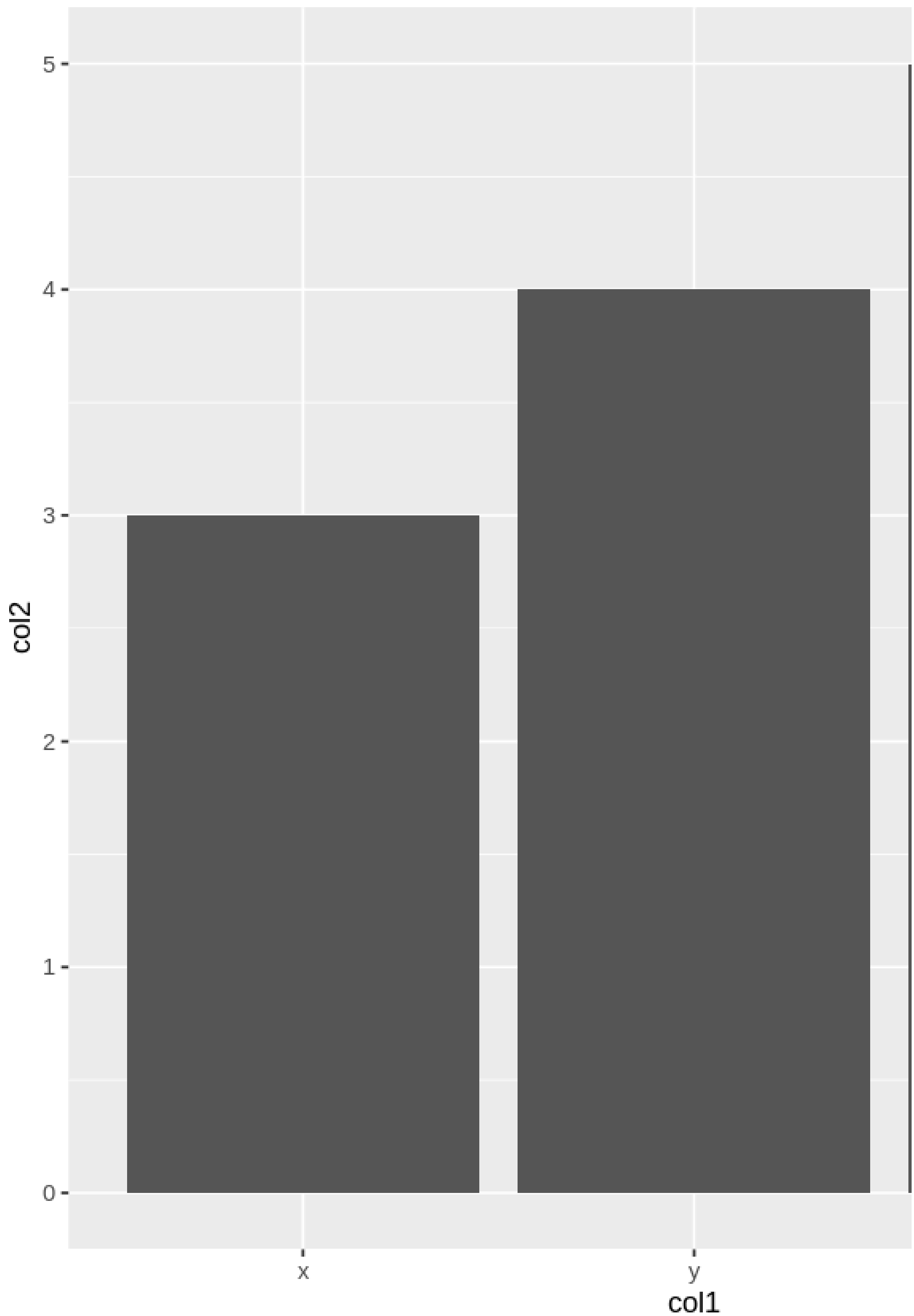
```
Sample size calculation for comparison of proportions
Sample size 1    : 39,208
Sample size 2    : 39,208
Total sample size: 78,416
Proportion 1     : 0.51
Proportion 2     : 0.52
Effect size      : 0.02000934
Confidence level : 0.95
Power            : 0.8
Alternative      : two.sided
```

In [2]:

```
dframe <- data.frame(
    col1 = c("x", "y", "z"),
    col2 = c(3, 4, 5)
)
```

In [3]:

```
library(ggplot2)
p <- ggplot(dframe, aes(x=col1, y=col2)) + geom_bar(stat="identity")
print(p)
```



5
4
3
2
1
0
col2
x
y
col1

▾ **utils/functions.py** Download

```python
import numpy as np
import pyrsm as rsm


def example():
    text = """
You just accessed a function from your first python packages!
Change the code in utils/function.py to whatever you need for this assignment
Use 'from utils import functions' to get access to your code
You can add modules to import from by adding additional .py files to the 'utils' directory
Note: If you make changes to the content of this file you will have to restart the notebook kernel to get the updates
"""
    print(text)

```