This docs contains the steps involved in all the python files given as a solution to Starbucks Promotional Strategy Problem:

Links:

<https://towardsdatascience.com/implementing-a-profitable-promotional-strategy-for-starbucks-with-machine-learning-part-1-2f25ec9ae00c>

<https://towardsdatascience.com/implementing-a-profitable-promotional-strategy-for-starbucks-with-machine-learning-part-2-8dd82b21577c>

<https://github.com/joshxinjie/Data_Scientist_Nanodegree/tree/master/capstone>

**input\_missing\_values.ipynb:**

In this file we are going to find the missing values for age, income and gender in profile.json dataframe using XGB models.

1. Import packages, datasets
2. Plot original distribution of demographics (Age, income, gender)
3. Check null values in profile df. You will find that customer is either missing income and gender both or nothing and where there is missing value, age is 118
4. Encode person id from profile and transcript df from hash to numeric value
5. Convert profile[‘became\_member\_on’] into 3 separate columns month, year & day
6. transcript df contains timestamps for offers (received, viewed, completed) as well as timestamps for transactions.
7. Create 2 new columns: value\_type (offer or transaction) and value\_id\_amt (offer\_id or trans\_amt). Separate transcript df into transcript\_offers and transcript\_trans based on these new columns
8. Encode offer\_id from portfolio and transcript\_offer from has to numeric value
9. Apply one hot encoding for channel in portfolio (email, phone, social, web)
10. Merge transcript\_offer with profile and then with portfolio
11. Split transcript\_offer into 3 based on event column (offer received, viewed, completed) and then merge them together to get the time\_received, time\_viewed, time\_completed
12. This merging will generate some false offers. Apply below conditions:
    1. Time\_viewed > time\_received & time\_completed > time\_viewed
    2. Time\_viewed > time\_received & time\_completed is null
    3. Both time\_viewed and time\_completed are null
13. Now find probable successful offers, probable tried offers and probable failed offers and add 3 columns: successful, tried and failed
14. Drop duplicates from all\_offers and transcript\_offers
15. Get subset of all\_offers which are successful: succ\_tried\_offers df
16. Merge succ\_tried\_offers with transcript\_trans. Then only keep those records where:
    1. Time\_spend > time\_received and time\_spend < time\_completed -> truly successful
    2. Time\_spend > time\_received and time\_spend < time\_expiry -> tried
17. Now we will find below statistics:
    1. No. of offers successfully completed by each person for each type
    2. No of offers tried by each person for each type
    3. Amount spent by each person on each type
    4. No. of transactions done by each person for each type
    5. Avg spending per transaction by each person for each type
    6. No. of offers of each type received by each person.
18. Create succ\_tried\_offers\_counts df and succ\_tried\_offers\_summary df.
19. Get summary of all offers: offers\_summary
20. Merge offers\_summary and succ\_tried\_offers\_summary df
21. Check if this merging makes any sense by checking different conditions.
22. Now find below stats:
    1. Amount of non-offers spending
    2. No of non-offers transactions
    3. Avg spending per transaction for no offers.
23. Take a subset df from succ\_tried\_offers as offers\_transactions
24. Merge this offers\_transactions with transcript\_transaction to create transaction\_labeled. From this df, select only those rows where spend\_during\_offer == 0. This will be transactions made for non-offer: transactions\_non\_offer
25. Find summary of transaction info for non-promotional situations. This will be non\_offer\_summary
26. Now create below ratios: for each type of offer
    1. No. of this type of offers received
    2. No. of successful offers
    3. No. of tried offers
    4. % of successful offers
    5. % of tried offers
    6. Total amount spent on this offer
    7. No. of transactions on this offer
    8. Avg spending per transaction for this offer
27. To calculate all above rations: we need to create a separate df for each offer. e.g offer\_0\_summary to offer\_9\_summary
28. Merge all the above 10 offer df with profile and create a new df: all\_data. Check for duplicates
29. Now calculate:
    1. Total spending
    2. Total received
    3. Total successful
    4. Total tried
    5. Total no. of transactions
    6. Total avg. spending per transaction
30. Now find:
    1. Total spending for each family BOGO/Discount/Informational
    2. Total received for each family BOGO/Discount/Informational
    3. Total successful for each family BOGO/Discount/Informational
    4. Total tried for each family BOGO/Discount/Informational
    5. Total no. of transactions for each family BOGO/Discount/Informational
    6. Total avg. spending per trans. for each family BOGO/Discount/Informational
31. All the above data will be in all\_data df. Fill all null values with 0
32. Now our model should be able to distinguish between
    1. Customer received the offer but did not act on them
    2. Customer did not received the offer at all
33. For this, we will compute below stats:
    1. Proportion of total spending that was spent on offer-0/…/offer-9 and no\_offer
    2. Proportion of total spending that was spent on each offer family type
    3. Proportion of total no. of transactions that was made on offer-0/…/offer-9 and no\_offer
    4. Proportion of total no. of transactions that was made on each offer family type
    5. Proportion of total no. of offers received that belongs to offer-0/…/offer-9
    6. Proportion of total no. of offers received that belongs to each offer family type
34. Apply one hot encoding for the Gender column. (M, F, None, Other)
35. Also apply label encoding for the gender column: Create new column gender\_enc with numeric values.
    1. No. 0, 1 & 3 are for M, F and O and No. 2 will be for Gender = None
36. Store all\_data into a csv file
37. Separate data with missing values from the data without missing values
    1. data\_model: data without missing values
    2. data\_predict: data with missing values
38. The gender\_enc of data\_model will contain only 0, 1, 3 values. All the records with value = 2 in gender\_enc will be in data\_predict
39. Split data\_model into train and test
40. Create X\_train and X\_test by dropping a few unnecessary columns.
41. Apply feature scaling. Use StandardScalar
42. Apply pca
43. Use stratified K fold and XGB regressor for predicting age and income. Use XGB classifier for predicting gender
44. Train the model, fit the model and predict for data\_model dataset. Save the best model and use it to predict data\_predict. Replace the null values with predicted values in data\_predict
45. After replacing all the null values with predicting values for age, income and gender, save the dataset as new\_profile.csv

**generate\_monthly\_data.ipynb:**

Here we will create a time-series monthly summary that tracks the transactional behavior of customers during promotional and non-promotional situations.

1. Step no. 1 to 16 from input\_missing\_values.ipynb are the same in this file.
2. Once we get succ\_tried\_offers, we need to find all the failed offers that means customers did not spend any money on these offers.
3. We have all offers sent by the company in transcript\_offer\_received, we have a list of offers on which customers did spend money in succ\_tried\_offers. Now the difference between these 2 df will give us the failed offers.
4. Get the succ\_tried\_offers\_summary from succ\_tried\_offers by using agg method.
5. Merge all\_offers with succ\_tried\_offers\_summary. After merging, rows with NaN in successful\_offer, tried\_offer, failed\_offer are failed offers.
6. Fill the Nan from successful\_offer and tried\_offer with 0 and failed\_offers with 1
7. Separate the rows where failed\_offer has 1 value and get the df: failed\_offers
8. Use function assign\_month\_num to convert the days into appropriate months assuming a month contains 30 days
9. After getting month numbers, get the dataset: monthly\_failed\_offers We now have a monthly summary of offers that failed. Customers did not spend anything here
10. Now produce monthly summary of customers transactional behavior
11. Create offer\_transactions df from succ\_tried\_offers. Then merge transcript\_trans and offer\_transaction df to create transaction\_labeled df
12. This transaction\_labeled df will have NaN values in the spend\_during\_offers column which are nothing but records for non-offer transactions.
13. Fill Nan with 0 and respective offer\_id column as 10.
14. The shape of transaction\_labeled and transcript\_trans should be same
15. Apply assign\_month\_num function on transaction\_labeled and transcript\_offer df
16. Get a monthly\_transactions df from transaction\_labeled by using group\_by on month\_num, per\_id, offer\_id
17. Now find which customer did not spend anything during non-promotional days and month during which this event happened
18. To find this, we need to track the instances of offer\_id\_10 which is no-offer situation
19. Apply assign\_month\_num on original transcript df and new df will be named as transcript\_month
20. We will add only those individuals whom we have seen so far in the transcript. This ensures that we are tracking individuals only after they become a member of starbucks.
21. After applying the above condition, we will get a df called non\_offer\_trans.
22. Merge non\_offer\_trans with monthly\_transactions df to find the months when a person did not perform any trans. associated with no offers.
23. Fill null values in monthly\_amt\_spent and num\_trans column with 0 after merging
24. Create a df for records where during these months, individuals receive no offer and make no transaction. For this select rows from non\_offer\_trans where value in the num\_trans is 0. This df is named as no\_offer\_no\_trans
25. Now combine below 3 summaries:
    1. Monthly summary of transactions (Step no. 16)
    2. Monthly summary of failed offers (Step no. 9)
    3. Monthly summary of no non-promotional spending occurred (Step no. 24)
26. Resulting data frame will give us:
    1. Which type of offers a customer received, how much he spent on them and how many transactions he carried out for them
    2. Did customers make any non promotional transactions? If so, how much he spent and how many transactions he performed.
27. Resulting data frame after concating the 3 df mentioned in step 25 will be named as monthly\_data.
28. Create a new column called amt\_spent\_per\_transaction and calculate the values by dividing monthly\_amt\_spent by num\_trans. Fill null with 0
29. Next, find if an individual can receive the same offer twice a month?
30. This will ensure that: Any customer will not receive the same offer twice a month. If a customer receives an offer which is ending in the next month, then he will not receive the same offer in that ending month as well. Resulting df: all\_offers\_summary
31. This will give the df: all\_offer\_summary.
32. Create a df (diff\_month\_df) of such offers which end in a different month than started.
33. Merge all\_offers\_summary with diff\_month\_df to create all\_offers\_summary\_check
34. Now calculate the profit generated by each instance in the monthly data
35. To calculate the profit, use 3 rules:
    1. If offer is successful then profit = monthly revenue - cost of offer
    2. If offer is not successful then profit = monthly revenue
    3. If transactions were not made as a part of the offer then profit = monthly revenue since no cost involved.
36. Get offer\_cost df by selecting offer\_id, difficulty and reward columns from portfolio
37. Get monthly\_data records where num\_offers = 1
38. Merge monthly\_data with offer\_cost. Fill null values in the reward and difficulty column with 0.
39. Informational offers do not have any cost. Compute the cost for BOGO and Discount offers and create a column called cost in monthly\_data
40. Calculate the profit. Create column: has\_profit and add value 1 where we have profit > 0. Drop reward and difficulty columns
41. As a part of feature engineering, we will calculate the cumulative values and moving averages of total spending, no. of transactions and profit for all 10 offers.
42. Max value in the month\_num col. is 23. Create month 24 as placeholder month to calculate cumulative sums for rows with NaN to be removed later.
43. Create df placeholder\_month
44. Concat monthly\_data and placeholder\_month df.
45. Create offer\_type\_df from portfolio. Create no\_offer\_df for offer\_id\_10 i.e. no offer instances. Add no\_offer\_df in offer\_type\_df. Merge monthly\_data and offer\_type\_df
46. Apply generate\_cumulative\_stats function for monthly\_amt\_spend, num\_trans and profit columns in the monthly\_data df
47. Add new columns for these cumulative sums
48. Remove the placeholder\_month records.
49. Calculate the following:
    1. cumulative spend per transaction per offer id
    2. cumulative profit per transaction per offer id
    3. Cumulative total spend
    4. Cumulative num. of transactions
    5. Cumulative profit
50. Similarly calculate moving average for all the above factors mentioned in the step 49
51. Apply create\_lag\_feature function and apply it on monthly\_data
52. Remove the first 2 months as there will not be any lags for the first 2 months.
53. Apply one hot encoding for offertype column in the portfolio df and get portfolio\_enc df
54. Add a new record in the no\_offer\_df
55. Create a column for the no\_offer (offer\_id\_10) indicator variable.
56. Append no\_offer\_df to portfolio\_enc df
57. Apply one hot encoding for gender in the profile df and get profile\_enc df
58. Merge monthly\_data and profile\_enc df
59. Print the distribution of has\_profit label
60. Save monthly\_data as monthly\_data\_rolling.csv
61. Check the monthly\_data

**Model\_offer\_0.ipynb:**

1. Import monthly\_data\_rolling.csv
2. Add seasonality features.
3. Apply generate\_offer\_monthly\_data function to get the subset of monthly\_data relevant to the current offer (offer\_0)
4. Plot the graphs
5. Create train, valid and test datasets for baseline model.
6. Then create train valid and test datasets for uplift models.
7. Apply standardization