STARBUCKS OFFER PERSONALIZATION

COSC2789 -Practical Data Science

Group 12

INTRODUCTION

Starbucks is being well known as the largest coffeehouse company in the world at the moment. However, it is also being famous as a world-leading data-driven company that utilizes the use of data to elevate their business. One of the most famous business data-driven business approaches of Starbucks is personalization promotion. Starbuck uses the data gathered from the Starbucks reward mobile app and performs analysis to send the most suitable offer to a customer. In the final assignment of the course Practical Data Science, a group of four students will try to simulate the analysis process to build a model that can predict which type of offer is effective for the customer.

TABLE OF CONTENTS

01
BUSINESS OBJECTIVE AND HYPOTHESIS

02
DATA SCIENCE PROCESS

03
RESULT AND DISCUSSION

BUSINESS OBJECTIVE AND HYPOTHESIS

01

PROBLEM STATEMENTS

What is the most effective offer that customer is most likely to use?

BUSINESS UNDERSTANDING

The process of an offer are:

- For BOGO and discount:

offer_recieved ---> offer_viewed ---> offer_completed ---> transaction

- For informational offer:

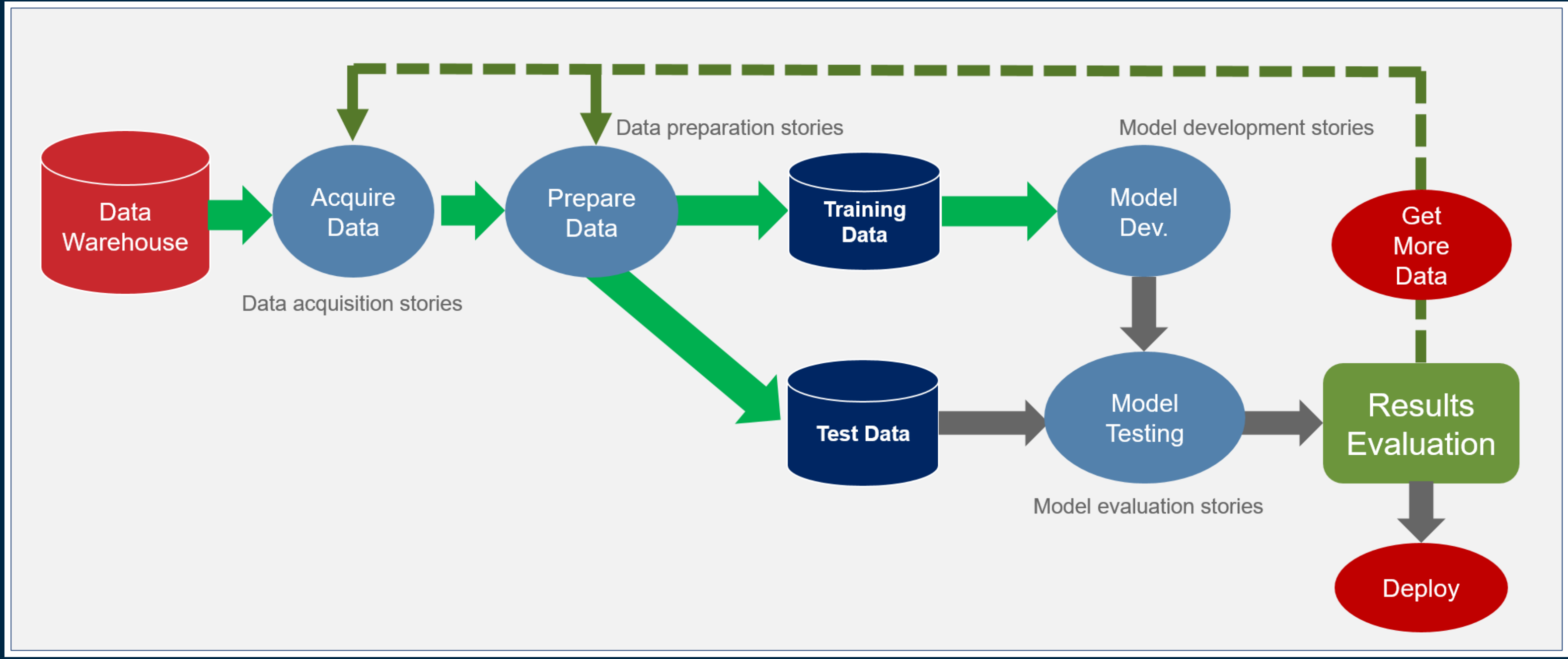
offer_recieved ---> offer_viewed ---> transaction

=> An offer is considered success if the offer status is viewed

02

DATA SCIENCE PROCESS

ROADMAP



Source: Agile Data Science – Addendum – The Burndown

DATA SET

- portfolio.json: offer id and its relevant data
- profile.json: customer demographic data
- transcript.json: record for transactions, offers received, offers viewed, and offers completed

DATA PREPARATION

Portfolio.json cleaning steps:

- Change the offer_id to integer value
- One hot encoded the channel and offer type column
- Drop unnecessary columns

```
portfolio_new, offer_id_encoded = cleanPortfolio(portfolio)
portfolio_new
```

[20]:

	reward	difficulty	duration	offer id	email	mobile	social	web	offer_type_bogo	offer_type_discount	offer_type_informational
0	10	10	7	1	1	1	1	0	1	0	0
1	10	10	5	2	1	1	1	1	1	0	0
2	0	0	4	3	1	1	0	1	0	0	1
3	5	5	7	4	1	1	0	1	1	0	0
4	5	20	10	5	1	0	0	1	0	1	0
5	3	7	7	6	1	1	1	1	0	1	0
6	2	10	10	7	1	1	1	1	0	1	0
7	0	0	3	8	1	1	1	0	0	0	1
8	5	5	5	9	1	1	1	1	1	0	0
9	2	10	7	10	1	1	0	1	0	1	0

DATA PREPARATION

Profile.json cleaning steps:

- Normalize the customer id
- Drop columns with missing gender and age
- Format the date in became_member_on column
- Create the member duration columns
- One hot encoded gender
- Change the age over 100 to NaN

```
profile_new, cust_id_encoded = cleanProfile(profile)
profile_new
```

[22]:

	age	became_member_on	income	customer_id	membership_duration	gender_F	gender_M	gender_O
1	55.0	2017-07-15	112000.0	1	376	1	0	0
3	75.0	2017-05-09	100000.0	2	443	1	0	0
5	68.0	2018-04-26	70000.0	3	91	0	1	0
8	65.0	2018-02-09	53000.0	4	167	0	1	0
12	58.0	2017-11-11	51000.0	5	257	0	1	0
...	...	...	...	...	...	...	...	...
16995	45.0	2018-06-04	54000.0	14821	52	1	0	0
16996	61.0	2018-07-13	72000.0	14822	13	0	1	0
16997	49.0	2017-01-26	73000.0	14823	546	0	1	0
16998	83.0	2016-03-07	50000.0	14824	871	1	0	0
16999	62.0	2017-07-22	82000.0	14825	369	1	0	0

14825 rows × 8 columns

DATA PREPARATION

Transcript.json cleaning step:

- Map the customer and offer id
- Sort the data by customer and time
- Split the “value” collumn
- Fill the N/A value in amount and reward column
- Split the event column
- Change the time to hours to days

```
transcript_new = cleanTranscript(transcript, offer_id_encoded, cust_id_encoded)
transcript_new
```

[25]:

	customer_id	time	amount	offer id	reward	event_offer completed	event_offer received	event_offer viewed	event_transaction
0	1.0	0.75	21.51	NaN	0.0	0	0	0	1
1	1.0	6.00	32.28	NaN	0.0	0	0	0	1
2	1.0	17.00	0.00	4.0	0.0	0	1	0	0
3	1.0	21.00	0.00	3.0	0.0	0	1	0	0
4	1.0	22.00	23.22	NaN	0.0	0	0	0	1
...	...	...	...	...	...	...	...	...	...
306529	NaN	29.75	0.00	6.0	3.0	1	0	0	0
306530	NaN	29.75	4.48	NaN	0.0	0	0	0	1
306531	NaN	29.75	0.00	7.0	2.0	1	0	0	0
306532	NaN	29.75	2.20	NaN	0.0	0	0	0	1
306533	NaN	29.75	4.05	NaN	0.0	0	0	0	1

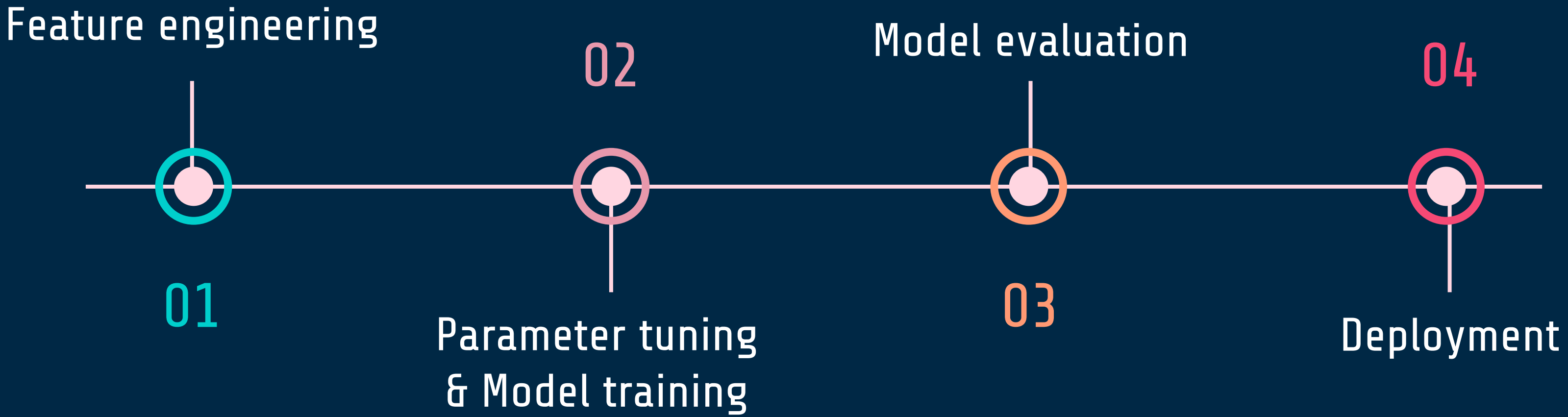
306534 rows × 9 columns

BUSINESS UNDERSTANDING

Merge the portfolio and profile data to transcript

```
[31]: transcript_new.info()

<class 'pandas.core.frame.DataFrame'>
Int64Index: 306534 entries, 0 to 306533
Data columns (total 26 columns):
 #   Column                    Non-Null Count   Dtype
---  ------                    --------------   -----
 0   customer_id               272762 non-null  float64
 1   time                      306534 non-null  float64
 2   amount                    306534 non-null  float64
 3   offer id                  167581 non-null  float64
 4   amount_rewarded           306534 non-null  float64
 5   event_offer completed     306534 non-null  uint8
 6   event_offer received      306534 non-null  uint8
 7   event_offer viewed        306534 non-null  uint8
 8   event_transaction         306534 non-null  uint8
 9   offer_reward              167581 non-null  float64
 10  difficulty                167581 non-null  float64
 11  duration                  167581 non-null  float64
 12  channel_email             167581 non-null  float64
 13  channel_mobile            167581 non-null  float64
 14  channel_social            167581 non-null  float64
 15  channel_web               167581 non-null  float64
 16  offer_type_bogo           167581 non-null  float64
 17  offer_type_discount       167581 non-null  float64
 18  offer_type_informational  167581 non-null  float64
 19  age                       272664 non-null  float64
 20  became_member_on          272762 non-null  datetime64[ns]
 21  income                    272762 non-null  float64
 22  membership_duration       272762 non-null  float64
 23  gender_F                  272762 non-null  float64
 24  gender_M                  272762 non-null  float64
 25  gender_O                  272762 non-null  float64
dtypes: datetime64[ns](1), float64(21), uint8(4)
memory usage: 55.0 MB
```

DATA MODELLING



FEATURES ENGINEERING

- ✓ Drop all the column with nulls value
- ✓ Drop duplicate
- ✓ Make target column “offer_succeed”
- ✓ Select the feature

```
[74]: Y

[74]: 2         0.0
      3         0.0
      5         1.0
      6         0.0
      7         1.0
               ...
      272754    0.0
      272755    0.0
      272757    1.0
      272759    1.0
      272760    0.0
      Name: offer_succeed, Length: 148754, dtype: float64
```

```
[71]: X.info()

<class 'pandas.core.frame.DataFrame'>
Int64Index: 148754 entries, 2 to 272760
Data columns (total 19 columns):
 #   Column                    Non-Null Count   Dtype
---  ------                    --------------   -----
 0   time                      148754 non-null  float64
 1   offer id                  148754 non-null  float64
 2   amount_rewarded           148754 non-null  float64
 3   offer_reward              148754 non-null  float64
 4   difficulty                148754 non-null  float64
 5   duration                  148754 non-null  float64
 6   channel_email             148754 non-null  float64
 7   channel_mobile            148754 non-null  float64
 8   channel_social            148754 non-null  float64
 9   channel_web               148754 non-null  float64
 10  offer_type_bogo           148754 non-null  float64
 11  offer_type_discount       148754 non-null  float64
 12  offer_type_informational  148754 non-null  float64
 13  age                       148754 non-null  float64
 14  income                    148754 non-null  float64
 15  membership_duration       148754 non-null  float64
 16  gender_F                  148754 non-null  float64
 17  gender_M                  148754 non-null  float64
 18  gender_O                  148754 non-null  float64
dtypes: float64(19)
memory usage: 27.7 MB
```

Model training

- Logistic Regression
- Random Forest Classifier
- AdaBoost Classifier
- LightBGM Classigier

Parameter Tunning

- Research about each model to chose the most important parameters
- Create a list of possible value for each parameter
- Using GridSearchCV to train model and select the best params

```
[52]: %%time

model =  LogisticRegression()

parameters = {
    'C': [ 1, 10, 20, 30],
    'max_iter': [1000, 4000, 10000]
}

log_reg = GridSearchCV(model, parameters, refit=True)
log_reg.fit(X_train, y_train)

print('Best Score: ', log_reg.best_score_*100, '\nBest Parameters: ', log_reg.best_params_)

Best Score:  63.888261668265464
Best Parameters:  {'C': 10, 'max_iter': 1000}
CPU times: user 11min 32s, sys: 6min 23s, total: 17min 56s
Wall time: 2min 32s
```

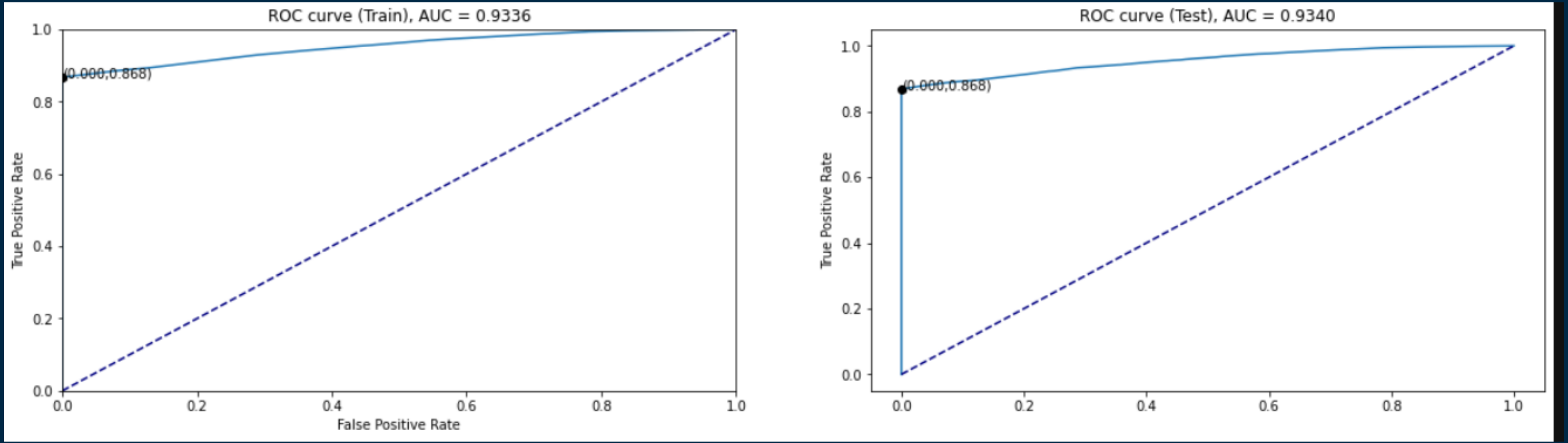
```
[61]: %%time

model =  LGBMClassifier()

parameters = {
    'num_leaves': [6,18,36,52],
    'boosting_type' : ['gbdt', 'dart'],
    'max_depth ': [5,10,15, None],
    'min_data_in_leaf': [20, 30, 50, 100]
}

lgbm_clf= GridSearchCV(model, parameters, verbose=2, cv=5, n_jobs=-1)
lgbm_clf.fit(X_train, y_train)

print('Best Score: ', lgbm_clf.best_score_*100, '\nBest Parameters: ', lgbm_clf.best_params_)

Fitting 5 folds for each of 128 candidates, totalling 640 fits
[Parallel(n_jobs=-1)]: Using backend LokyBackend with 8 concurrent workers.
[Parallel(n_jobs=-1)]: Done  25 tasks      | elapsed:   13.8s
[Parallel(n_jobs=-1)]: Done 146 tasks      | elapsed:   47.8s
[Parallel(n_jobs=-1)]: Done 349 tasks      | elapsed:  2.0min
[Parallel(n_jobs=-1)]: Done 640 out of 640 | elapsed:  5.6min finished
[LightGBM] [Warning] max_depth is set=-1, max_depth= will be ignored. Current value: max_depth=-1
[LightGBM] [Warning] Unknown parameter: 5
[LightGBM] [Warning] min_data_in_leaf is set=20, min_child_samples=20 will be ignored. Current value: min_data_in_leaf=20
Best Score:  92.68253872533427
Best Parameters:  {'boosting_type': 'gbdt', 'max_depth ': 5, 'min_data_in_leaf': 20, 'num_leaves': 6}
CPU times: user 7.88 s, sys: 6.86 s, total: 14.7 s
Wall time: 5min 35s
```

Model Evaluation – models' score

	Model	Accuracy Score	F1 Score
0	LogisticRegression	0.662230	0.643438
1	RandomForestClassifier	0.772109	0.788178
2	AdaBoostClassifier	0.927364	0.929598
3	LGBMClassifier	0.927364	0.929598

Figure Training model report

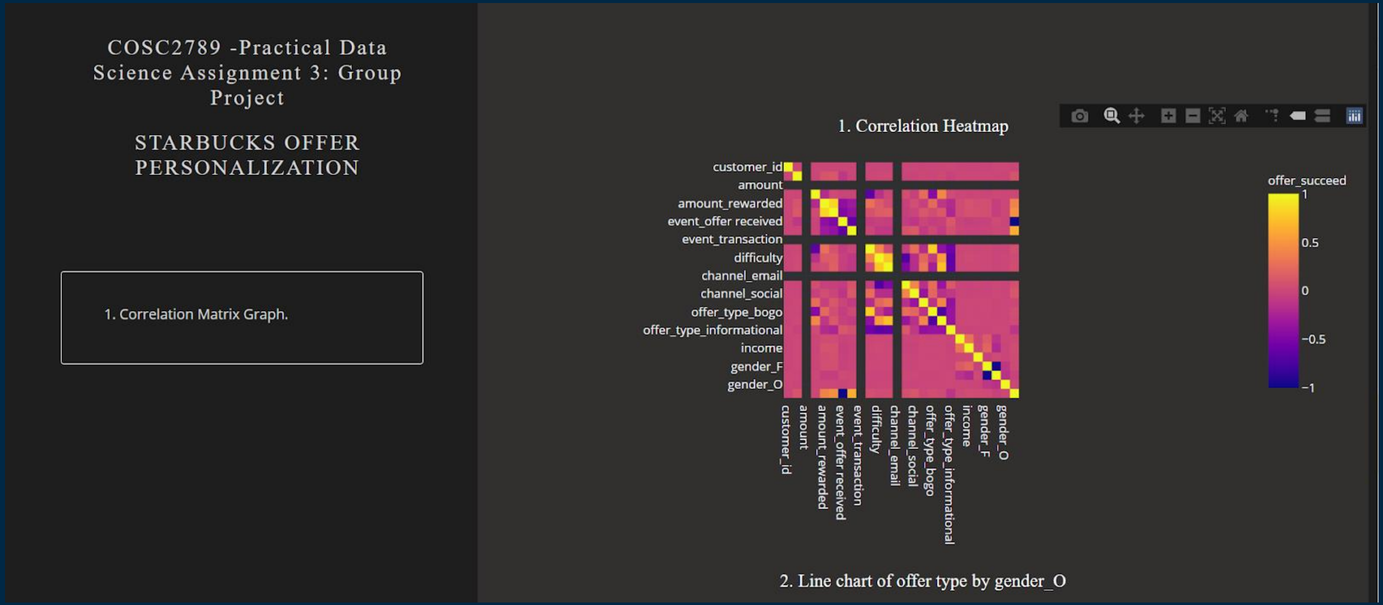
Model Evaluation – ROC graph



Deployment – API

HTTP methods	Route	Description
GET	/api/evaluate/<model_name>	Return a model score
GET	/api/predict/<model_name>	Return a list of result predicted by model
POST	/api/predict_offer_effective	Return the result predict whether offer effective or not against specific customer

Deployment – Dashboard



03

RESULT & DISCUSSION

Conclusion

In conclusion, from the data of customer behavior of Starbucks application, we have applied the process of data science to draw some insights and build a predictive model to evaluate the effectiveness of an offer based on the customer profile. The first part of this process is data cleaning which is one of the most challenging parts in this project, because the raw data is in JSON type with dictionary structure. Thus a tremendous amount of work has to be done when dealing with data such as one hot encoding categorical features, drop null value, merge the data sets. After finished cleaning, in the EDA part we have analyzed the data to acquire some insights about the customer spending trend with different offer types and customer membership duration over time. Finally, we have performed parameter tuning with 4 different binary classification models and found out that the LightGBMClassifier is the most effective model with the accuracy and f1 score over 0.9. In the future, some improvement can be done with this project. From the information, we can make more features to increase the accuracy of the model, other high performance models such as XGBoost or CatBoost could be taken into account and the project can be taken to a further step which is to predict the best offer type for a specific customer.

Q&A