Starbucks Project Report

Project Definition

1. **Project Overview**

   To apply my AWS MLE NanoDegree learnings, I chose the Starbucks Capstone Challenge. The goal is to improve marketing ROI by sending targeted offers to the right audience. The program provides the data, including simulated customer behavior on the Starbucks rewards mobile app. Starbucks sends an offer to mobile app users every few days, which can be an advertisement or a discount offer. Some users may not receive offers during certain weeks.

2. **Problem Statement**

   The main target is to make promotion offer events more effective, and there are multiple approaches for that, among which I chose to build a model that predicts whether or not someone will respond to an offer. In addition, as this is a MLE program, I productionized the classifier on AWS to be able to predict with new data.

   It provides 3 files:

   - portfolio.json - containing offer ids and meta data about each offer (duration, type, channel, etc.)
   - profile.json - demographic data for each customer (age, gender, income, etc.)
   - transcript.json - records for transactions, offers received, offers viewed, and offers completed, timestamp of event happen and the amount of money spent on a purchase

   The data is really messy and needs a lot of processing, but finally I converted them to one structural dataframe. Given the data structure, I decided to use traditional machine learning models, among which XGBoost has good performance and fast speed.

   Although there are 3 types of offer, I add the offer type as feature, so that I can predict all offer types with one model, in which case, I'll have more training data. Therefore, I'll build a XGBoost binary classifier. To benchmark the performance I'll train a logistic regression model first. Modeling part is done on AWS for a production environment. I also get report to analyze model performance, as well as feature importance using sagemaker.

3. **Metrics**

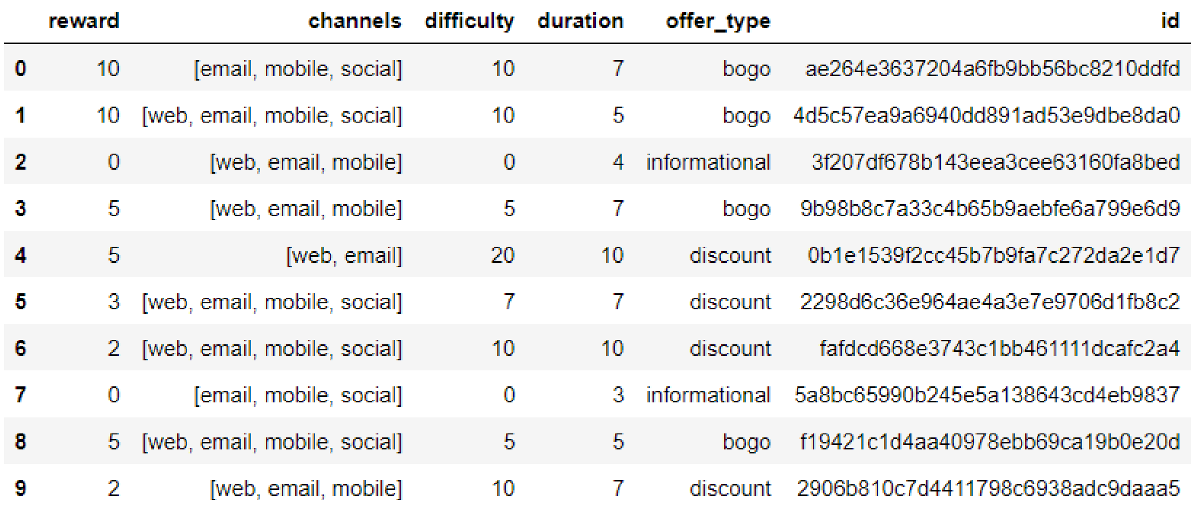
   There are multiple metrics to assess binary classification model, such as precision, recall, f1-score, and so on. I use the AUC score (also known as the area under the ROC curve) for model selection/comparison. This metric takes into account the trade-off between the true positive rate and the false positive rate for different classification thresholds. The more it’s close to 1, the better performance the model has. In training report created by sagemaker, there are more evaluation such as overall accuracy, and confusion matrix.

Analysis (Data Exploration)

1. **offer portfolio**

   Here is the overview of the features

   - id (string) - offer id
   - offer_type (string) - type of offer ie BOGO, discount, informational
   - difficulty (int) - minimum required spend to complete an offer
   - reward (int) - reward given for completing an offer
   - duration (int) - time for offer to be open, in days
   - channels (list of strings).



	reward	channels	difficulty	duration	offer_type	id
0	10	[email, mobile, social]	10	7	bogo	ae264e3637204a6fb9bb56bc8210ddfd
1	10	[web, email, mobile, social]	10	5	bogo	4d5c57ea9a6940dd891ad53e9dbe8da0
2	0	[web, email, mobile]	0	4	informational	3f207df678b143eea3cee63160fa8bed
3	5	[web, email, mobile]	5	7	bogo	9b98b8c7a33c4b65b9aebfe6a799e6d9
4	5	[web, email]	20	10	discount	0b1e1539f2cc45b7b9fa7c272da2e1d7
5	3	[web, email, mobile, social]	7	7	discount	2298d6c36e964ae4a3e7e9706d1fb8c2
6	2	[web, email, mobile, social]	10	10	discount	fafdcd668e3743c1bb461111dcafc2a4
7	0	[email, mobile, social]	0	3	informational	5a8bc65990b245e5a138643cd4eb9837
8	5	[web, email, mobile, social]	5	5	bogo	f19421c1d4aa40978ebb69ca19b0e20d
9	2	[web, email, mobile]	10	7	discount	2906b810c7d4411798c6938adc9daaa5

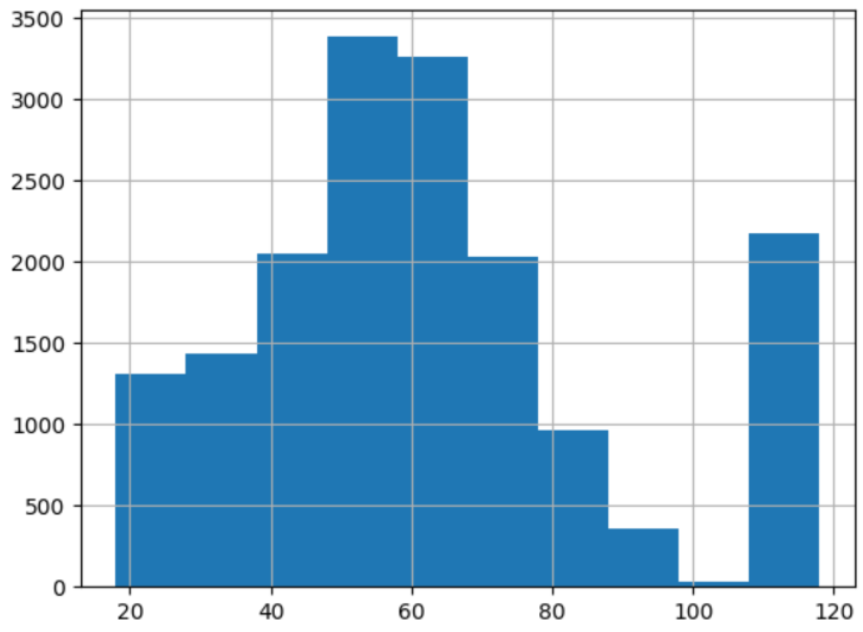
There are 10 offers in total. I converted duration from days to hours to match with the transaction data.

2. **Customer Profile**

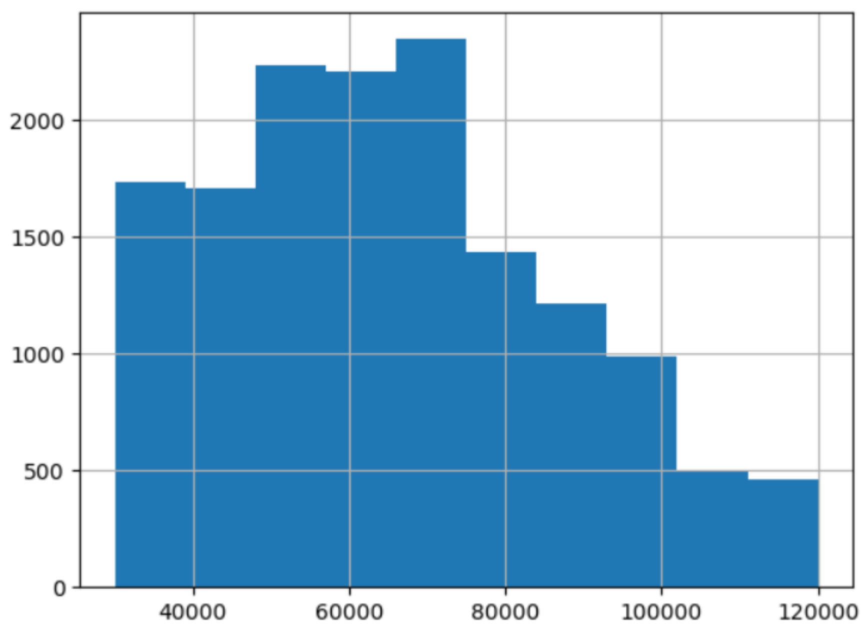
   It’s demographic data for each customers with following features

   - age (int) - age of the customer
   - became_member_on (int) - date when customer created an app account
   - gender (str) - gender of the customer (note some entries contain 'O' for other rather than M or F)
   - id (str) - customer id
   - income (float) - customer's income

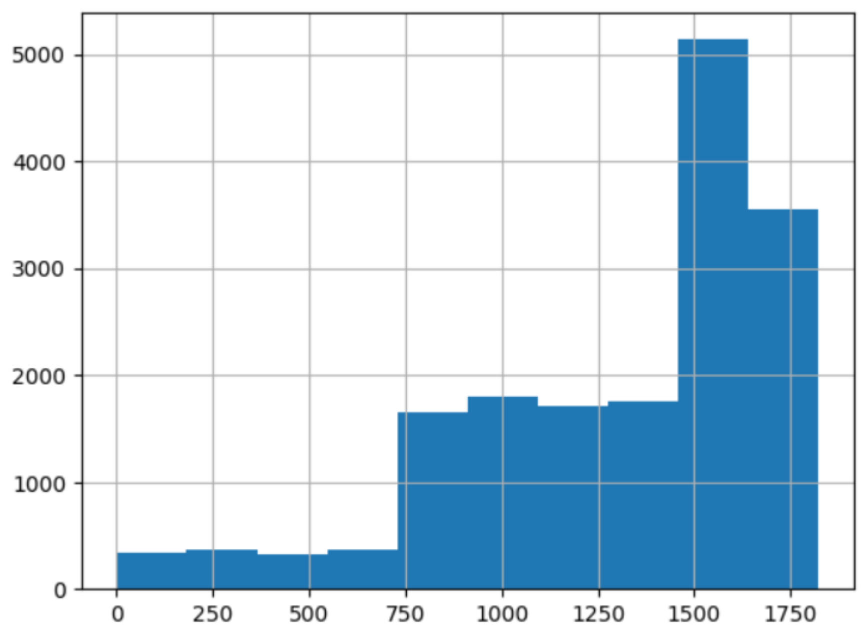
   There are 17000 customers in the dataset. `gender` and `income` contains missing values, and below in the hist plot for `age`, there are anomalies of 118. I’ll replace them with NA later.



`Income` distribution looks normal below. Most of them are below 80k, and the distribution is a bit right skewed.



I transformed feature `became_member_on` to date format and create a new feature `member_duration` which represents number of days being member since the earliest date of `became_member_on`. And below is the distribution of `member_duration` . It implies that most customers have been members for quite long.



I also renamed `id` to `person` to match with transaction data

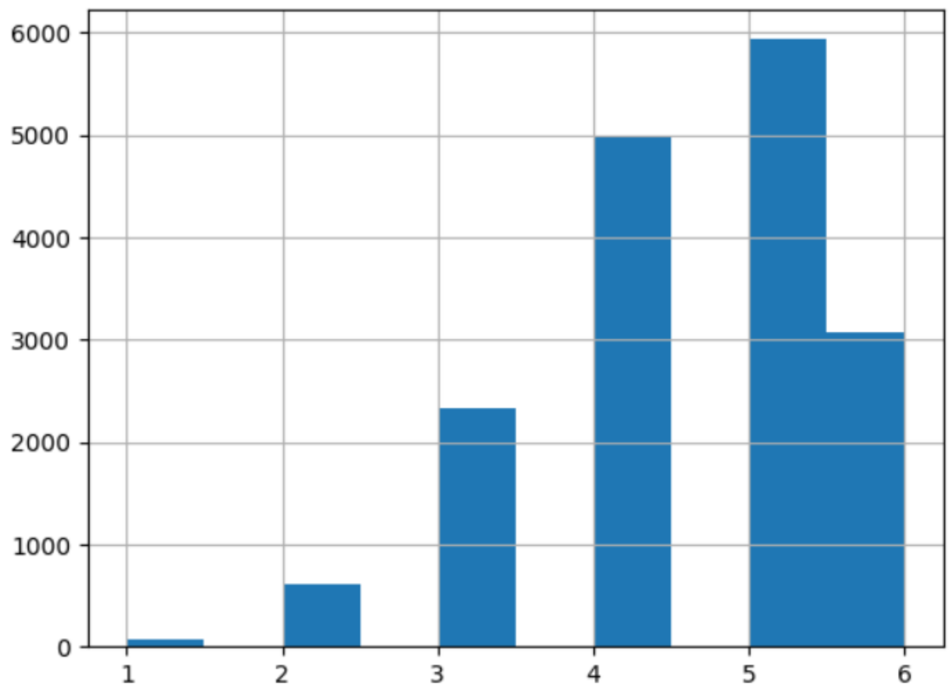
3. **Transactional records (transcript)**

There are 4 columns and 306534 records.

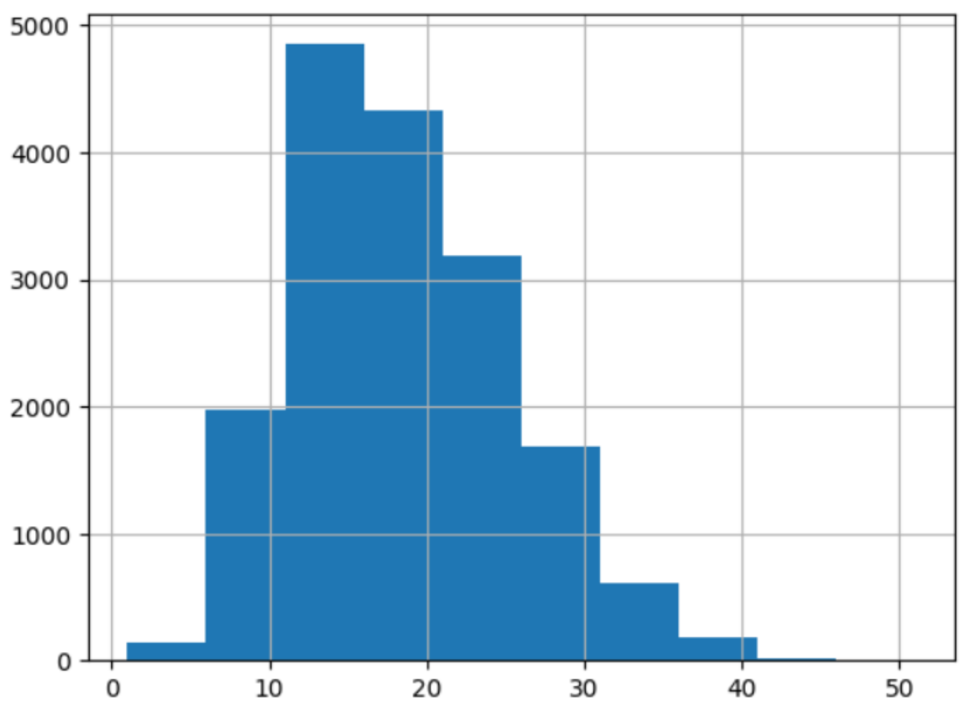
- event (str) - record description, including offer received, offer viewed, transaction, offer completed.
- person (str) - customer id
- time (int) - time in hours since start of test. The data begins at time t=0
- value - (dict of strings) - either an offer id or transaction amount depending on the record

I cleaned `value` column, breaking down it into three new columns - `amount`, `offer_id`, `reward`, which are from dictionary keys under `value` .

Firstly I checked distribution of offers per customer received, so mostly of them received between more than 3 offers, with a maximum of 6 during the test period.



In hist plot for events per customer, most people has around 20 records.



Around 30k BOGO and Discount offers are sent, and only 15k informational offer are distributed.

Then I created a new feature: `total_amount` the total amount of transactions that customer consumed by the timestamp.

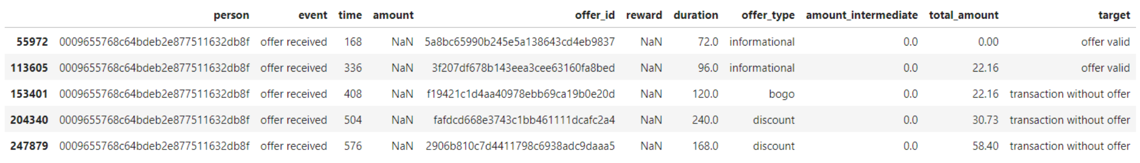
Methodology

1. **Data Preprocessing**
   a. Convert transcript to dataset of offer result

      For each customer, I filtered out the records of offer received, as a starting point to build the final dataset. For each record, I filtered out all activities in offer duration for this offer (since there is no offer_id for transaction event, I'll include it for each valid offer duration). Then I reversed the events order for the labeling function `check_offer_result()` . The logic in the function is that: For informational offer, if a transaction happened after offer viewed, it's a successful offer, if customer purchased without viewing offer, it's noise data should be removed later in the dataset. If customer didn't view or didn't purchase, the offer failed.

For BOGO and discount offer, it’s similar logic just replace transaction with offer completed. I created `target` column in this way.

Looping for all customers and all offers they received, I got the final dataset (`offer_df` in my notebook).



	person	event	time	amount	offer_id	reward	duration	offer_type	amount_intermediate	total_amount	target
55972	0009655768c64bdeb2e877511632db8f	offer received	168	NaN	5a8bc65990b245e5a138643cd4eb9837	NaN	72.0	informational	0.0	0.00	offer valid
113605	0009655768c64bdeb2e877511632db8f	offer received	336	NaN	3f207df678b143eea3cee63160fa8bed	NaN	96.0	informational	0.0	22.16	offer valid
153401	0009655768c64bdeb2e877511632db8f	offer received	408	NaN	f19421c1d4aa40978ebb69ca19b0e20d	NaN	120.0	bogo	0.0	22.16	transaction without offer
204340	0009655768c64bdeb2e877511632db8f	offer received	504	NaN	fafdcd668e3743c1bb461111dcafc2a4	NaN	240.0	discount	0.0	30.73	transaction without offer
247879	0009655768c64bdeb2e877511632db8f	offer received	576	NaN	2906b810c7d4411798c6938adc9daaa5	NaN	168.0	discount	0.0	58.40	transaction without offer

b. other feature engineering

Then I dropped noise records, converted `target` to dummy (only `offer_valid` is marked as 1). After merging the rest columns in portfolio and profile with this `offer_id`, I also processed missing value for `gender` (for `age` and `income` I left it NA because XGBoost can deal with that), and processed categorical columns to dummies. I created three more features, `recieved_total` - total offer received before the timestamp , `valid_total` - total offer succeeded before the timestamp, and `success_rate` which is just `valid_total/recieved_total` .

Here is the full columns I have in the final dataset.

```
['person', 'time', 'offer_id', 'duration', 'total_amount', 'target',
       'age', 'income', 'member_duration', 'reward', 'difficulty', 'web',
       'email', 'mobile', 'social', 'bogo', 'discount', 'informational', 'F',
       'M', 'O', 'Unknown', 'recieved_total', 'valid_total', 'success_rate']
```

Finally create train and test set by 0.8:0.2 and save the data. I didn’t shuffle the dataset, because it’s actually time series data, and the test data can only happen after train data.

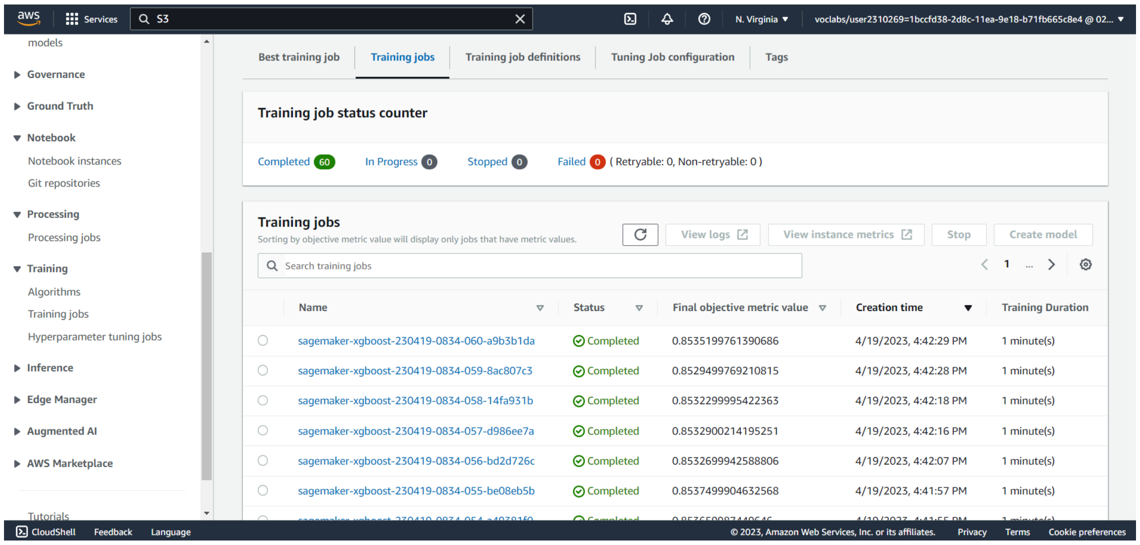
2. **Implementation**

I use the prepared train and test dataset mentioned above for model training and evaluation. I mainly use AUC to select and assess the model, but also checked other criteria e.g. accuracy score.

a. Firstly I created a logistic classifier as baseline. I filled missing values as it doesn’t accept NA. I used `LogisticRegression` from `sklearn` .

b. I developed an XGBoost classifier on AWS Sagemaker. This part is done on AWS. It's convenient with AWS image container and other sagemaker functions.

   First I processed the data to be the format XGBoost can accept, then uploaded to S3. I started with hyperparameter tuning job with `sagemaker.tuner`, and selected the best ones as the hyperparameters for my final model. Below is the hyperparameter tuning jobs.



Training job status counter

Completed 60 In Progress 0 Stopped 0 Failed 0 (Retryable: 0, Non-retryable: 0)

Training jobs

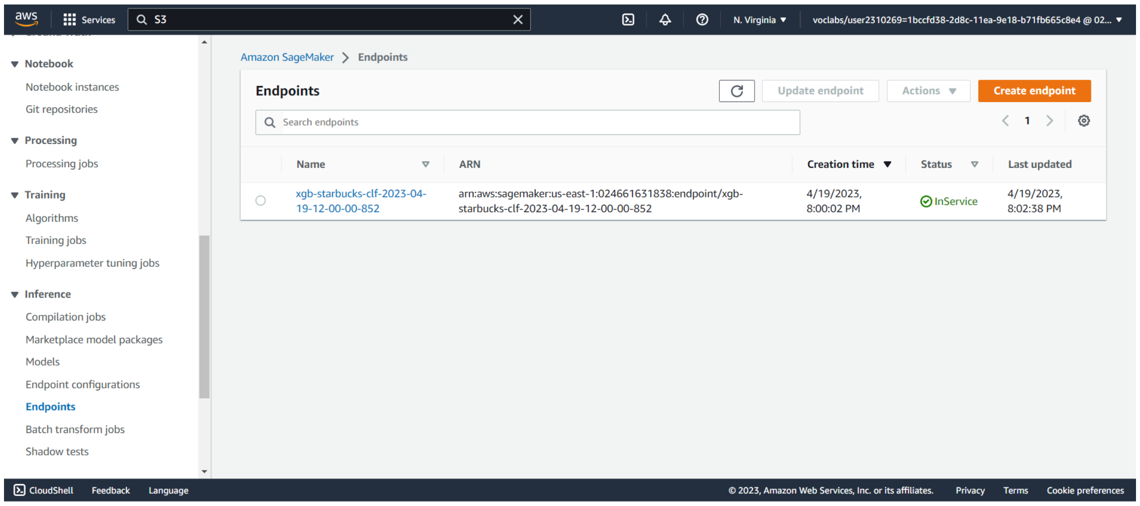
Sorting by objective metric value will display only jobs that have metric values.

Name	Status	Final objective metric value	Creation time	Training Duration
sagemaker-xgboost-230419-0834-060-a9b3b1da	Completed	0.8535199761390686	4/19/2023, 4:42:29 PM	1 minute(s)
sagemaker-xgboost-230419-0834-059-8ac807c3	Completed	0.8529499769210815	4/19/2023, 4:42:28 PM	1 minute(s)
sagemaker-xgboost-230419-0834-058-14fa931b	Completed	0.8532299995422363	4/19/2023, 4:42:18 PM	1 minute(s)
sagemaker-xgboost-230419-0834-057-d986ee7a	Completed	0.8532900214195251	4/19/2023, 4:42:16 PM	1 minute(s)
sagemaker-xgboost-230419-0834-056-bd2d726c	Completed	0.8532699942588806	4/19/2023, 4:42:07 PM	1 minute(s)
sagemaker-xgboost-230419-0834-055-be08eb5b	Completed	0.8537499904632568	4/19/2023, 4:41:57 PM	1 minute(s)

   When fit the final model, I added a rule to generate the XGBoost training report for model evaluation using `sagemaker.debugger` . Below is the core modeling part.

```
xgboost_container = image_uris.retrieve("xgboost", region, "1.2-1")
estimator=Estimator(
    role=role,
    image_uri=xgboost_container,
    base_job_name="xgb-starbucks-clf",
    instance_count=1,
    instance_type="ml.m5.xlarge",
    output_path=f's3://{bucket}/{prefix}/output/',
    hyperparameters=hyperparameters,
    rules=rules,
)
estimator.fit({'train': train_input, 'validation': validation_input},
              wait=True)
```

After model trained, I deployed it with Sagemaker endpoint, and successfully made predictions by invoking it. Below is the screenshot of endpoint.



Name	ARN	Creation time	Status	Last updated
xgb-starbucks-clf-2023-04-19-12-00-00-852	arn:aws:sagemaker:us-east-1:024661631838:endpoint/xgb-starbucks-clf-2023-04-19-12-00-00-852	4/19/2023, 8:00:02 PM	InService	4/19/2023, 8:02:38 PM

3. **Refinement**

I started with Logistic Regression, which gives me an AUC score `0.7085` . This is already a good starting point.

For the XGBoost classifier, I created a hyperparameter job to improve my model. The tuning result is displayed in the notebook `train_and deploy.ipynb` . I tuned 60 jobs for three hyperparameters. `eta` is similar to learning rate, `max_depth` is to avoid overfitting, `num_round` is number of total iterations.

```
hyperparameter_ranges = {'eta': ContinuousParameter(0.01,1.0),
                         'max_depth': IntegerParameter(5, 10),
                        'num_round': IntegerParameter(50, 200)
                        }
```

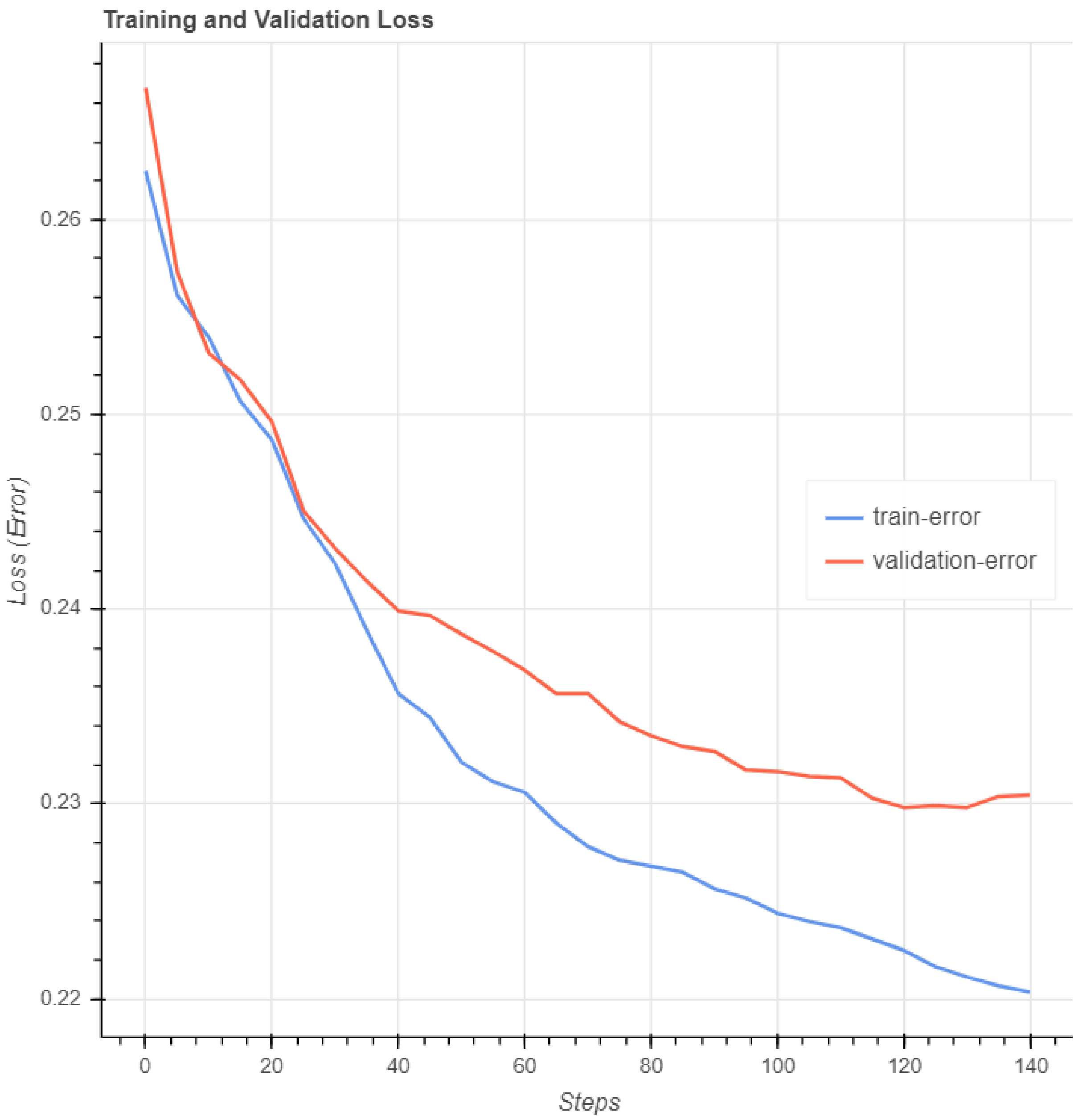
The lowest AUC score is `0.82584` , while the best AUC is `0.85442` with hyperparameters below

```
{'eta': '0.07903618528752897',
                 'max_depth': '5',
                 'num_round': '142'}
```

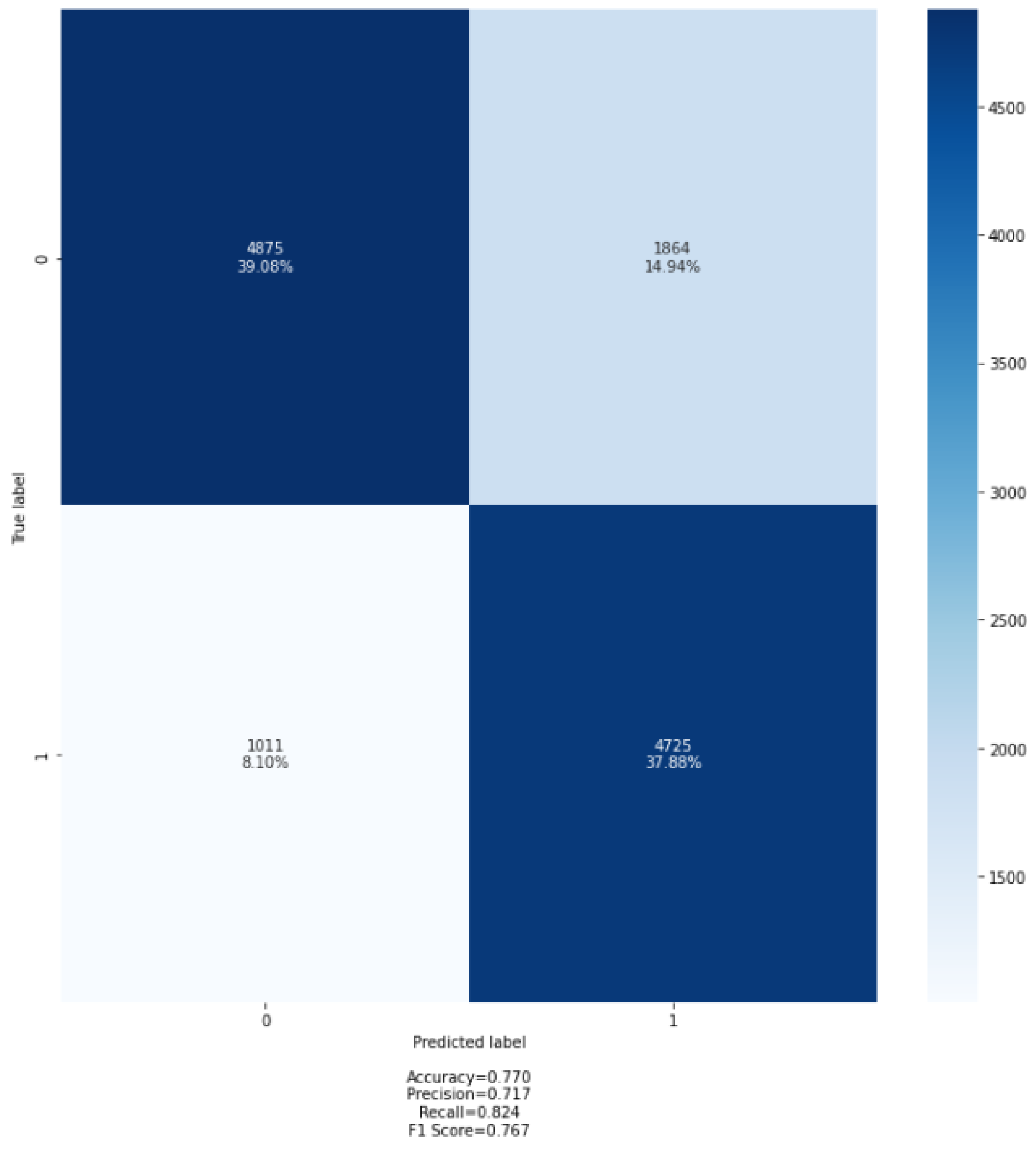
Results

1. **Model Evaluation**

   Checking the loss plot in training report, both training loss and validation loss goes down until end of the iterations, however, the validation loss is higher than training loss, showing that my model could be overfitting a bit.

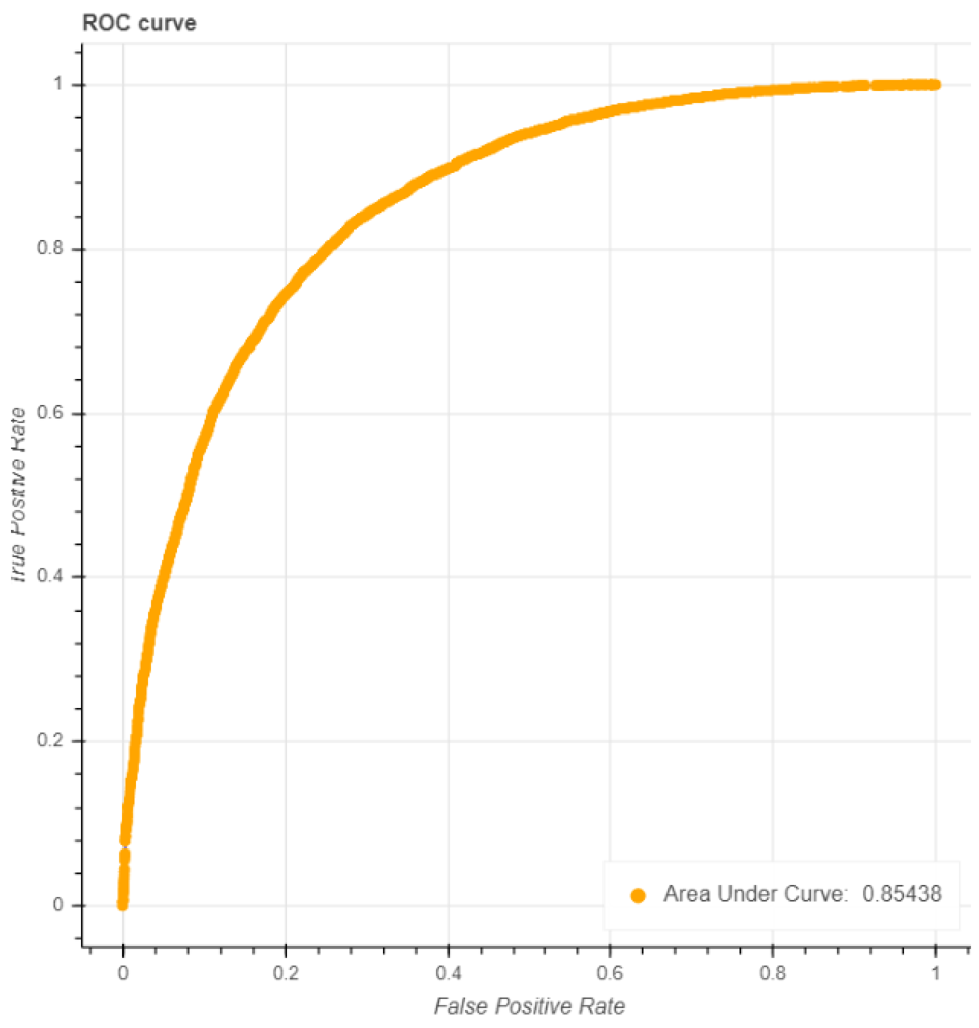


Here is the confusion matrix. The cells on the principal diagonal shows the True Positive counts,   and the off-diagonal cells count the number of misclassified predictions. The percentage of False Positive is almost twice as False Negative.



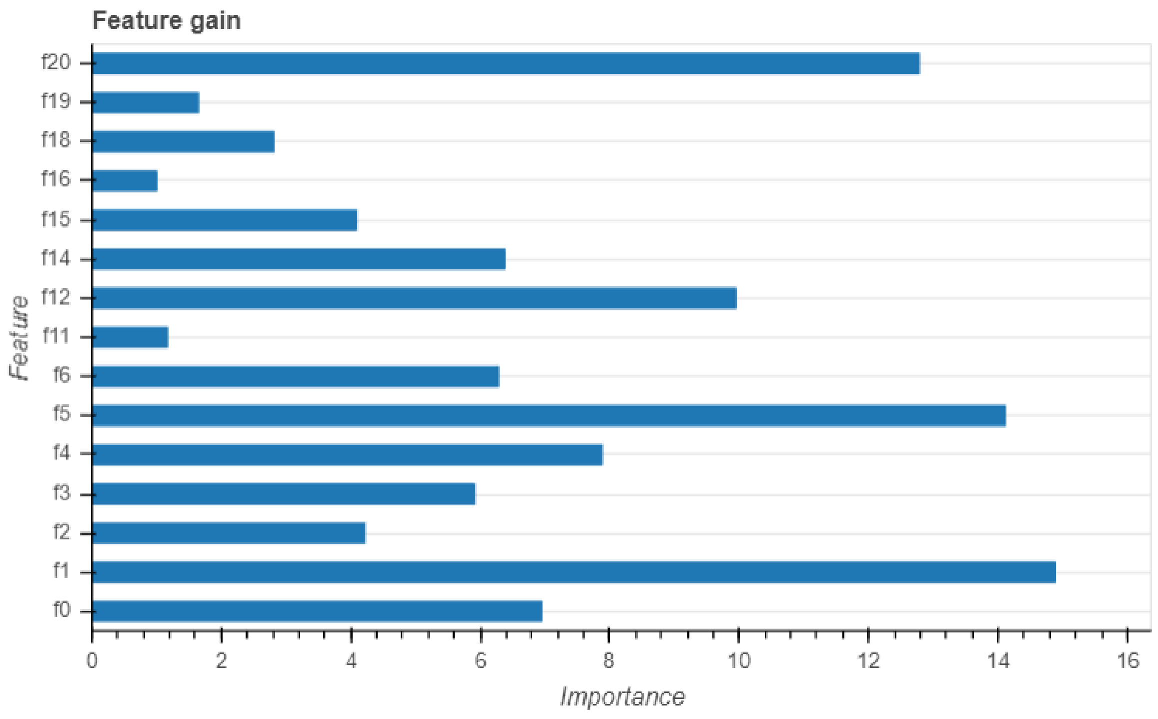
Accuracy=0.770
Precision=0.717
Recall=0.824
F1 Score=0.767

ROC curve looks satisfying and close to 1.



2. **Feature Importance**

   The most 3 important features f1 - `total_amount`, f5 - `reward`, f20 - `success_rate`. `total_amount` and `success_rate` are new features I added, indicating that customer has more purchase and more successful offers is more likely to affect by the offer. `reward` also contributes a lot to offer success.
   `discount` (f12) and `member_duration` (f4) are also important features, maybe Starbucks can focus on discount offer and target audience by `member_duration` for higher offer success rate.



Conclusion

1. **Reflection**

   From data analysis, data processing, to model tuning / training and deployment, I managed to predict customer reaction to an offer with an XGBoost classifier with AUC score of more than 0.85, which can help Starbucks to distribute the offer more precisely. The deployed model is on AWS, and it can be invoked easily to make new predictions.

   As a side product I also found some important features to take into consideration when sending offers, including customer purchase and offer reaction in the past, offer type and reward, etc.

   It's quite challenging to clean the data and convert it to appropriate format. I chose to create tabular dataset, but I also want to explore if it's possible to build it in time series. Apart from binary classification, it's also interesting to build model predicting conversion rate offer received → offer viewed → offer success, and model forecasting amount spent given an offer.

2. Since now I have the feature importance, I can do a feature selection, and fit it to XGBoost model again to see if it mitigate overfitting.

I can try neural network as well, which is more complicated, but may have better performance than XGBoost.