# Machine Learning Engineer Nanodegree

## Capstone Project Report

**Artem Shuvalov**

**31 October, 2021**

## Overview

# I. Definition

**Project Overview**
In 2021, the need for the data analysis among the modern corporations is more important than ever before. During the COVID-year, the majority of the companies lost the opportunity to serve the customers in offline-stores, making online services the only connection to their clients. Therefore, the importance of the websites and applications has risen to the levels never seen before.

Starbucks was not an exception, but it was prepared with its state-of-art mobile phones application, which allows users to make orders online, predict the queues and send special offers to the clients.

The reason why I took this project is that I am a CEO of TravelTech startup in Russia, which has an embedded recommendation system. So, the problem stated in this Project is relevant to the realms of Machine Learning I work with.

**Problem Statement**
The Starbucks would like to analyze how different demographic groups respond best to different types of offers in their application. The task is to combine historical transaction, demographic and offer data to determine the algorithm for mapping of the customer and the most relevant for him\her\them offer type. The evaluation of the performance can be measured using the historical data as well.

The problem to be solved is to find the optimal model, which will help to get the highest conversion of the clients, i.e. maximize the number of people making the orders among those who saw the each type of offer in the application.

It is worth to compare the model's results with the current conversion values for each type of offer. There is no reason to include reward-free offers, as the can increase the client's retention, but would not affect the conversion metrics, we chose the most relevant one.

**Metrics**
The conversion is going to be analyzed using the standard Machine Learning metrics:

1) Precision and Recall
The formulas are the following:

$$Precision = \frac{True\ Positives}{True\ Positives + False\ Positives}$$

It means the number of correct results divided by the number of all results.
In terms of the Project, it relates to the number of correctly predicted conversions over the number of actions assessed as conversions.

$$Recall = \frac{True\ Positives}{True\ Positives + False\ Negatives}$$

It means the number of correct results divided by the number of results that have to be retrieved.
In terms of the Project, it relates to the number of correctly predicted conversions over the number of all conversions.

2) F-1 score - combines Precision and Recall.
The formula is the following:

$$Recall = \frac{2 \times Precision \times Recall}{Precision + Recall}$$

It measures of test's accuracy on a dataset, what is basically the harmonic mean of the Precision and Recall.

1) AUC-ROC curve - area under the Receiving Operating Characteristic curve.
   This measure has quite convolved analytical formula to put it into proposal. Basically, it is the integral which measures the relation of the True Positive Rate, or Recall, to the False Positive Rate:

$$False\ Positive\ Rate = \frac{False\ Positive}{False\ Positive + True\ Negative}$$

   It means the probability that the model ranks a random positive example than a random negative example.

## II. Analysis

**Data Exploration**
There were 3 datasets in this project:

1) Portfolio - consists of the information about the offer provided to the client, including it's type (BOGO/Buy one get one, discount or informational/reward-free), difficulty to achieve for the client, reward, duration and channels.

There were 10 offers with the following features:

- id – id of the offer
- offer_type – type of the offer
- Channels – channels of offer mareting
- Difficulty – the amount customer had to spend to start the offer
- Reward – reward of the offer
- Duration – duration of the offer in days

2) Profile - includes the demographical information about the customers.

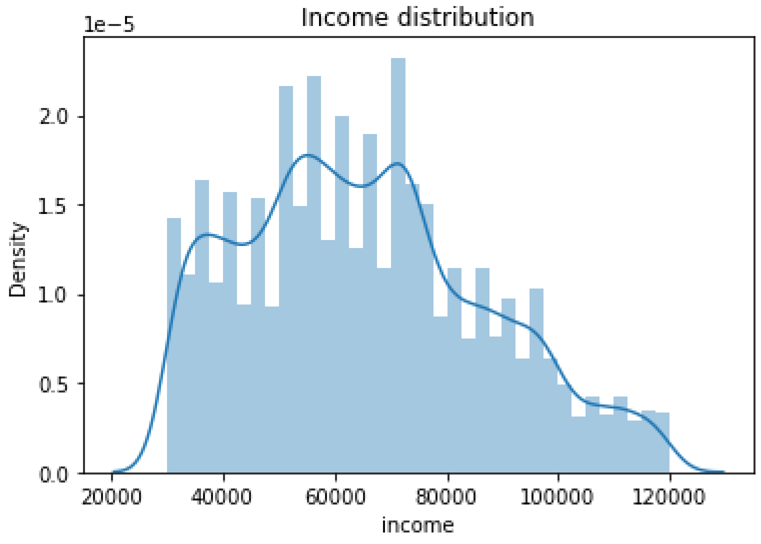
It contained the information about 17,000 people with the following features:

- Id – id of the customer
- Gender – Male, Female or Other
- Age – age of the customer
- Income – income of the customer
- Became_member_on – the date when the customer downloaded the application

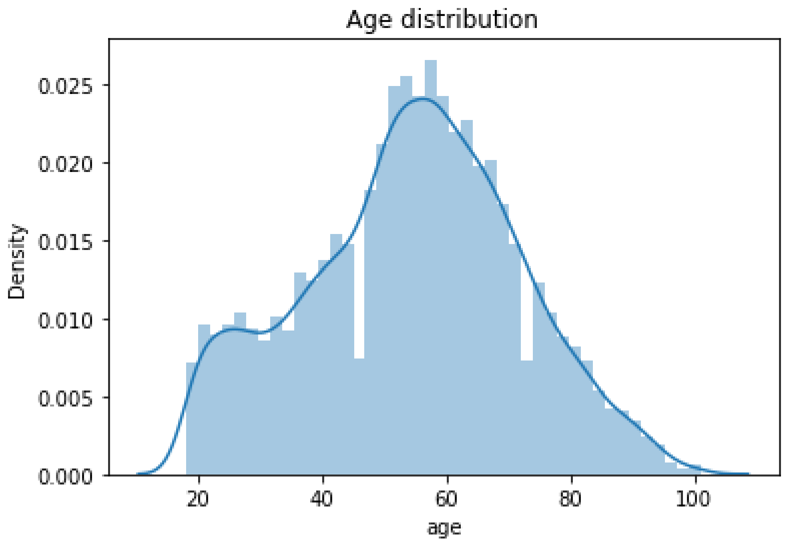
The value with missing income, gender or age were dropped, what resulted in 14,825 records after.

Here is the distribution of the demographic variables:

- We see that the income distribution is not normal and skewed with the average of 70,000-80,000 USD

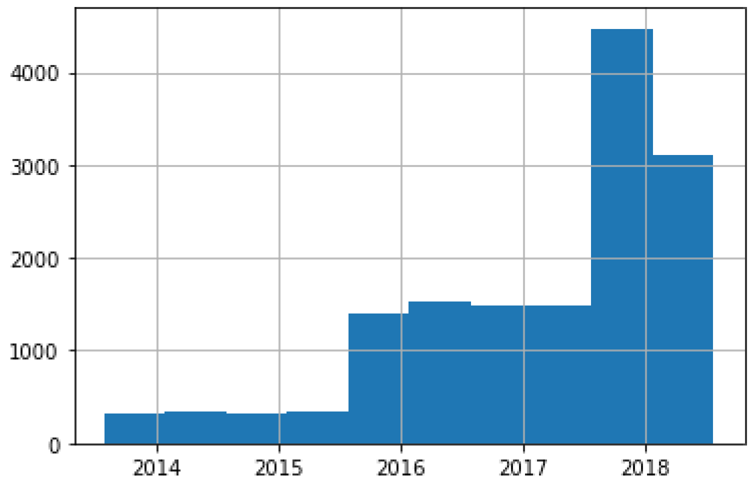


- While we see that the age is distributed normally with the average of 60 years old



- The most part of the users arrived after 2017

```
# The major share of the customers arrived after the 2017
profile_df['became_member_on'].hist()
plt.show()
```



3) Transcript - contains the information about the actions with the offer in the customer's application as well as transactions.

The dataset contained 306,534 records and the following features:

- person – id of the customer
- event – the type of event happened
- time – when the event happened, measured in hours
- value – additional data about the record, including offer_id, reward and amount

The value feature was split in 3 parts for further analysis.
It was seen that there were:

- For each completed offer, there was transaction with the same time
- For each viewed offer there was received offer
- Not all viewed offer were completed

In order to analyze the behavior of the clients, the new dataset with Customer Journey was created.

4) Customer Journey

The baseline for the dataset was Transcript dataset, so:

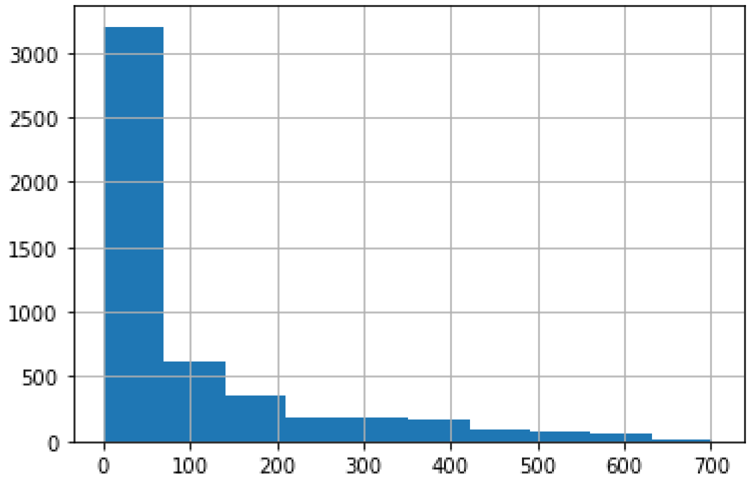
- For each offer viewed, the completed offer was concatenated
- For each offer received, the viewed offer was joined
- For each completed offer, the relative transaction was added

The final data had 143,843 records. After that we had to join Portfolio and Profile dataset, as well as create new features.

## New Features

As the beginning, new features for all 3 targeted offers were created. For Bogo and Discount offers, the value of 1 was set if the viewed offer was completed. For the informational offer, 1 was set if the transaction happened after some time after the view. The difference between median and 25% quartile was around 24 hours, so it was taken as the threshold:

```
count    4948.000000
mean       97.612773
std       136.946582
min         0.000000
25%        12.000000
50%        36.000000
75%       126.000000
max       702.000000
```



For each offer types, the following conversions were received:

| OfferType | Conversion |
|---|---|
| Bogo | 52.51% |
| Discount | 67.50% |
| Informational | 41.87% |

Furthermore, this dataset was augmented with the following features:

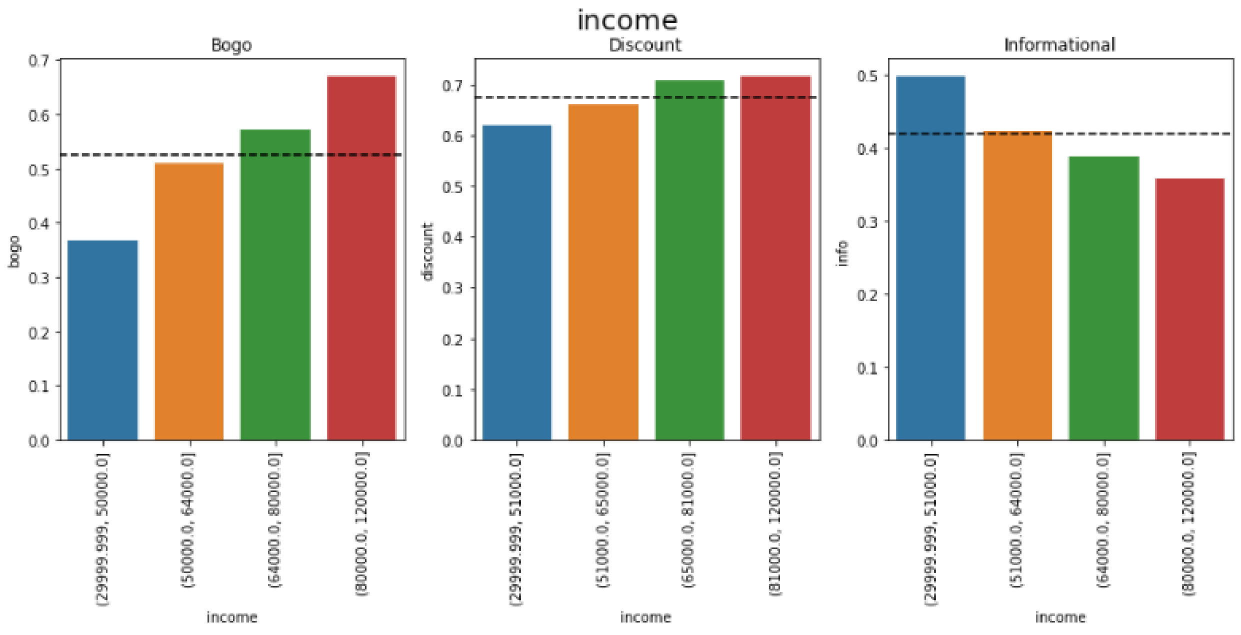
- Became_member_from – the number of months since the start of the membership
- Time
- Number of viewed offers
- Number of completed offers
- Average reward from offers
- Average time from receival to offer view
- Average time from receival to offer completion

- Number of transactions
- Mean amount of transactions

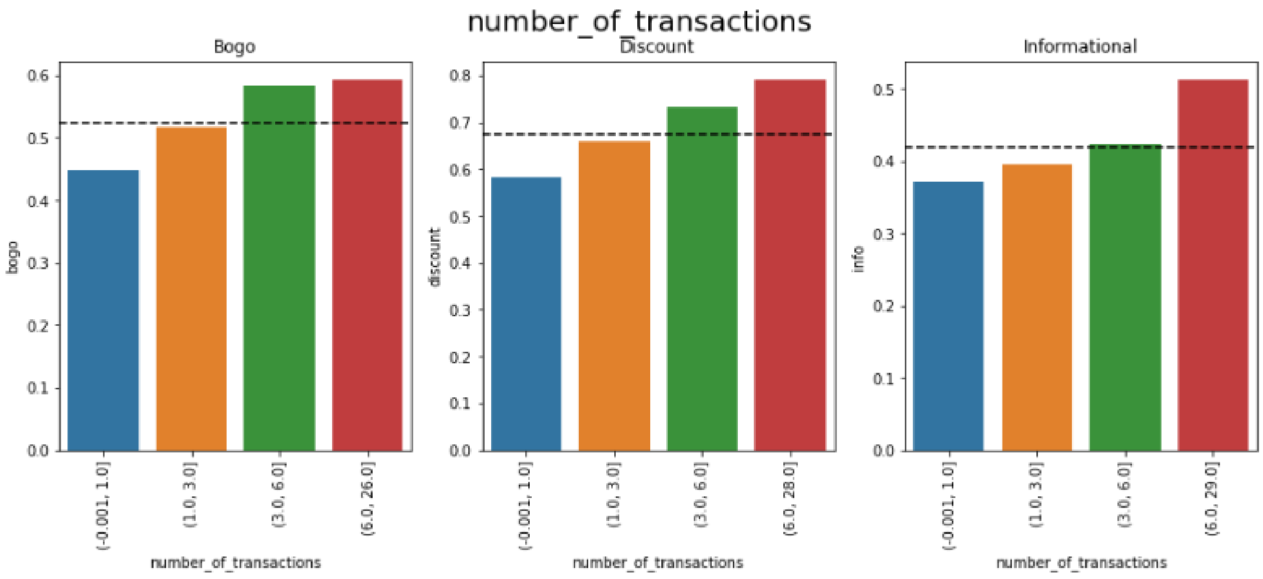
**Exploratory Visualization**

As we collected some information about the clients, we can tell something about their behavior:

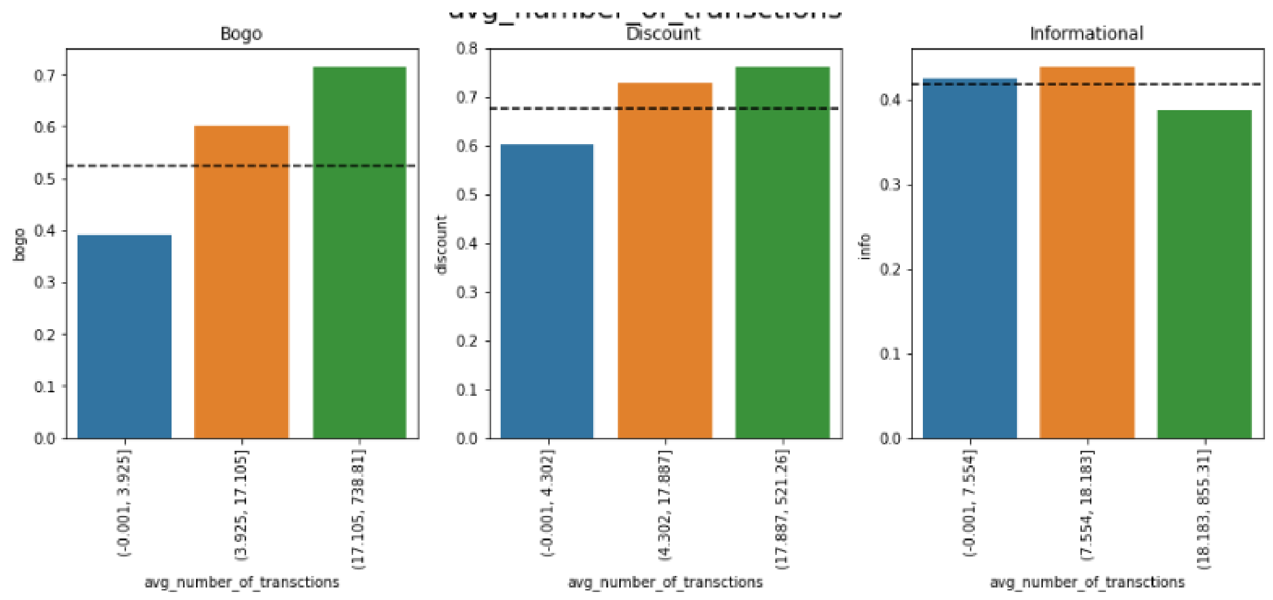
- Higher-income clients prefer Discount and Bogo offers, while the lower-income prefer Informational ones



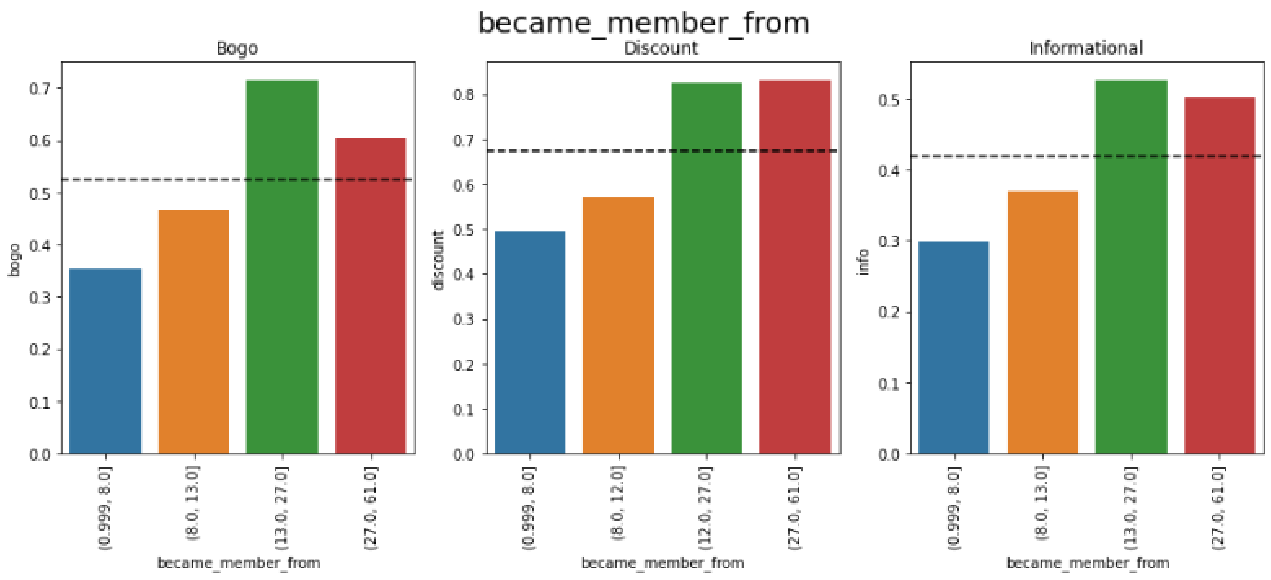
- The clients who are more frequent customers are more interested in Informational offers



- Customers who spend more money, prefer Bogo offers



- The longer the customers are members of the application, the more they tend to track the offers



**Algorithms and Techniques**

In order achieve the aim of finding the best offer type for each customer, the following Machine Learning algorithms were used:

- Linear model – the standard AWS SageMaker's LinearLearner was used. The model predicts an event if the output is higher than threshold, based on the input features
- XGBoost – this is the version of gradient boosting model, which models decision trees in special manner, trying to adjust the error of the previous step

**Benchmark**

As the benchmark for these models, the historical conversion rates for each offer types were taken. The models' performances were measured using the precision scores, mentioned in the Metrics section.

## III. Methodology

**Data Preprocessing**

For each offer type the specific datasets were created, which contained only information related to the viewed offer with the following customer's steps. Furthermore, the data was processed in the following way:

- The missing numerical values were replaced with median, as the most robust statistics. The categorical value, gender, was replaced with "Other" value

```
# Preprocessing for gender. Replace with "O", then One-Hot Encoding
gender_pipe = Pipeline(
    [
        ('imputer', SimpleImputer(strategy='constant', fill_value='O')),
        ('ohe', OneHotEncoder())
    ])

# Preprocessing for numerical features. Replace with median and standardize
num_pipe = Pipeline(
    [
        ('imputer', SimpleImputer(strategy='median')),
        ('scaler', StandardScaler())
    ])
```

- As ML models can work only with numerical values, the Age was transformed using One-Hot encoding method, which creates binary feature for each category

```
# Join via ColumnTransformer
preprocessing = ColumnTransformer(transformers=[
    ('gender_pipeline', gender_pipe, ['gender']),
    ('numeric_pipeline', num_pipe, variables[1:])
])
```

- Data standardization, which balanced the distribution, subtracting mean and dividing by the standard deviation of the numerical features

- Data was sampled, making balanced ration between the offer types

```
rus = RandomUnderSampler(random_state=seed)
X_tot, y_tot = rus.fit_resample(df.drop(target_name, 1), df[target_name])
```

- Data was divided into Train-Validation-Test sets, divided by the offer types

```
# Split the data
X, X_val, y, y_val = train_test_split(
    X_tot, y_tot, test_size=args.val_split_ratio,
    stratify=y_tot, random_state=seed)

X, X_test, y, y_test = train_test_split(
    X, y, test_size=args.test_split_ratio / (1 - args.val_split_ratio),
    stratify=y, random_state=seed)
```

The following steps were done using the SKLearnProcessor with the code available in ./model_utils/preprocess.py.

**Implementation**

The Hyperparameter tuning has to start from some baseline, which was set.
Here is the example of such default parameters and further tuning, based on the set metrics, F1-score for our case:

```
# Set the model image
container = get_image_uri(session.boto_region_name, 'xgboost', '0.90-1')

for prefix in ['bogo', 'discount', 'info']:
    # Initialize XGBoost
    xgb = sagemaker.estimator.Estimator(container,
        role,
        train_instance_count=1,
        train_instance_type='ml.c4.xlarge',
        output_path=f's3://{bucket}/CapstoneProjectStarbucks/{prefix}/model',
        sagemaker_session=session,
        base_job_name=prefix + '-')

    xgb.set_hyperparameters(max_depth=4,
                            eta=0.1,
                            gamma=4,
                            min_child_weight=6,
                            colsample_bytree=0.5,
                            subsample=0.6,
                            early_stopping_rounds=10,
                            num_round=200,
                            seed=1)

    # Initialize Tuner
    xgb_hyperparameter_tuner = HyperparameterTuner(estimator=xgb,
                    objective_metric_name='validation:f1',
                    objective_type='Maximize',
                    max_jobs=20,
                    max_parallel_jobs=4,
                    hyperparameter_ranges = {
                         'max_depth': IntegerParameter(2, 6),
                         'eta'      : ContinuousParameter(0.01, 0.5),
                         'gamma': ContinuousParameter(0, 10),
                         'min_child_weight': IntegerParameter(2, 8),
                         'colsample_bytree': ContinuousParameter(0.2, 1.0),
                         'subsample': ContinuousParameter(0.3, 1.0),
                    },
                    base_tuning_job_name=prefix + '-xgb-tuning')
```

For Linear model the system works in the similar way, while the number of parameters to fit is lower:

```python
# Train the Linear Model
for prefix in ['bogo', 'discount', 'info']:
    # Create instance of LinearLearner
    ll = LinearLearner(role,
           train_instance_count=1,
           train_instance_type='ml.c4.xlarge',
           predictor_type='binary_classifier',
           output_path='s3://{}/CapstoneProjectStarbucks/{}/model'.format(\
                                                     bucket, prefix),
           sagemaker_session=session,
           binary_classifier_model_selection_criteria='f1',
           epochs=100,
           use_bias=True,
           optimizer='adam',
           loss='auto',
           wd=0,
           normalize_data=True,
           unbias_data=True,
           early_stopping_patience=5,
           learning_rate=0.01,
           balance_multiclass_weights=True)

    # Create Sets from the Local data as the inputs to the Linear model
    train = pd.read_csv(f'./data/{prefix}/{prefix}_train.csv', header=None)
    train_data = ll.record_set(train.drop(0, 1).values.astype('float32'),
                   labels=train[0].values.astype('float32'), channel='train')

    valid = pd.read_csv(f'./data/{prefix}/{prefix}_train.csv', header=None)
    validation_data = ll.record_set(valid.drop(0, 1).values.astype('float32'),
                labels=valid[0].values.astype('float32'), channel='validation')

    # Fit the model
    ll.fit([train_data, validation_data], logs=False)
```

**Refinement**

In order to assess the results, the Transformer method of SageMaker is used, in order to compute queries:

```
# Create Transformers
for prefix in ['bogo','discount', 'info']:
    print(prefix)

    # Create XGB Transformer
    transformer = best_model[prefix]['ll']['model'].transformer(\
                        instance_count = 1, instance_type = 'ml.m4.xlarge')

    # Batch transform
    transformer.transform(f's3://{bucket}/CapstoneProjectStarbucks/{prefix}/{prefix}_test.csv',
                    content_type='text/csv', split_type='Line', logs=False)

    transformer.wait(logs=False)

    # Download the data
    session.download_data(f'./data/{prefix}/{prefix}_ll_preds', bucket,
                key_prefix='/'.join(transformer.output_path.split('/')[3:]))

    # Create Linear Transformer
    transformer = best_model[prefix]['xgb']['model'].transformer(\
                     instance_count = 1, instance_type = 'ml.m4.xlarge')

    # Batch transform
    transformer.transform(f's3://{bucket}/CapstoneProjectStarbucks/{prefix}/{prefix}_test.csv',
       content_type='text/csv', split_type='Line', wait=True, logs=False)

    # Download the data
    session.download_data(f'./data/{prefix}/{prefix}_xgb_preds', bucket,
                key_prefix='/'.join(transformer.output_path.split('/')[3:]))
```

## IV. Results

**Model Evaluation and Validation**

Here are the results of the algorithms for each offer type:

**Bogo**

| Model | F1-Score |
|---|---|
| XGBoost | 74.48% |
| LinearLearner | 71.83% |

With the following metrics and confusion matrix:

XGBoost model

| | value |
|---|---|
| **accuracy** | 71.45% |
| **balanced_accuracy** | 71.45% |
| **precision** | 70.29% |
| **recall** | 74.32% |
| **f1** | 72.25% |
| **average_precision** | 75.14% |

|  |  |
|---|---|
| **AUC** | 77.02% |

|  | **Predicted 0** | **Predicted 1** |
|---|---|---|
| **True 0** | 68.58% | 31.42% |
| **True 1** | 25.68% | 74.32% |

So we see 70.29% conversion versus 52.51% historic one.

**Discount**

| **Model** | **F1-Score** |
|---|---|
| XGBoost | 73.58% |
| LinearLearner | 71.24% |

With the following metrics and confusion matrix:

XGBoost model

|  | **value** |
|---|---|
| **accuracy** | 65.76% |
| **balanced_accuracy** | 65.77% |
| **precision** | 73.55% |
| **recall** | 49.32% |
| **f1** | 59.04% |
| **average_precision** | 65.33% |
| **AUC** | 69.49% |

|  | **Predicted 0** | **Predicted 1** |
|---|---|---|
| **True 0** | 82.22% | 17.77% |
| **True 1** | 50.68% | 49.31% |

So we see 73.55% conversion versus 67.50% historic one.

**Informational**

| **Model** | **F1-Score** |
|---|---|
| XGBoost | 65.50% |
| LinearLearner | 69.48% |

With the following metrics and confusion matrix:

Linear model

| value | |
|---|---|
| accuracy | 59.94% |
| balanced_accuracy | 59.94% |
| precision | 56.29% |
| recall | 89.02% |
| f1 | 68.97% |
| average_precision | 65.46% |
| AUC | 67.73% |

| | Predicted 0 | Predicted 1 |
|---|---|---|
| True 0 | 30.86% | 69.13% |
| True 1 | 10.97% | 89.02% |

So we see 56.29% conversion versus 41.87% historic one.

**Justification**
After application of XGBoost and Linear models, the positive results were received. We see that for all offer types the conversion rate increased with the new model.

## V. Conclusion

**Reflection**
During the task, the specific datasets were analyzed in order to increase the conversion rates for each offer type for Starbucks application clients. The customer journey was reconstructed in order to analyze the customer's behavior from offer view to the transaction. Different new features were introduced and the data was processed. After application of XGBoost and Linear models, the positive results were received. We see that for all offer types the conversion rate increased with the new model.

**Improvement**
In order to improve the model, more detailed information about clients' purchases could be useful. Moreover, new models such as Deep Neural Networks, Random Forest could be applied in order to check the performance.