Machine Learning Engineer Nanodegree
Capstone Project

Hao Yun

December 7, 2022

I Definition

Project Overview

A successful reward program can benefit a company in attracting and retaining customers. A variety of rewards are provided in the market. People may respond differently to certain type of rewards. Comparing with giving the whole population the same reward, providing variant rewards at an individual personalized level not only helps companies improve the profitability, but also improves customer experience.

Starbucks has provided a dataset containing simulated data that mimics customer behavior on the Starbucks rewards mobile app [1]. Analyzing demographic data and transactional data helps Starbucks understand cutsomers better. And contributing a statistical model or a machine learning model based on these data can help marketers know customer preferences, evaluate products and services, predict trends in the future and make new marketing strategies.

Problem Statement

The Starbucks dataset exists as three separate files including offer information, demographic data and records of transactions during the test month. In this capstone project, two main problems can be solved by analyzing this dataset. One objective is to predict how much a customer will spend at Starbucks during the experiment period. Another is to predict the probability that a customer will complete a reward, so that Starbucks can come up with new strategies to determine what kind of reward will be more appropriate for a certain customer.

The problem requires the predictions of continuous output variables. This is obviously a supervised learning problem, more precisely, a regression problem. The target variables are the total amount that each customer spends at Starbucks and the probability that each customer completes offers. The main task is to learn a mapping from multiple input variables to numerical variables. Various kinds of algorithms can be used to solve this problem. In this project, three algorithms will be implemented: linear regression, xgboost and neural network.

After modeling, output values of an amount and a probability will be predicted from the input data. Then the result will be compared with the true values.

Metrics

Mean Absolute Error (MAE) is a measure of errors between paired observations expressing the same phenomenon and is calculated as the sum of absolute errors divided by the sample size [2]:

$$MAE = \frac{\Sigma_{i=1}^{n}|y_i - \hat{y}_i|}{n} \tag{1}$$

In this project, the main task is to solve a regression problem. MAE is an appropriate metric that can be used to quantify the performance of a regression model. A low MAE means that predicted values are close to true values.

Root Mean Square Error (RMSE) is another popular metric for continuous variables [3]:

$$RMSE = \sqrt{\frac{\Sigma_{i=1}^{n}(y_i - \hat{y}_i)^2}{n}} \tag{2}$$

RMSE penalizes the higher difference more than MAE. And MAE is robust to outliers whereas RMSE is not. Both input data and output data will be normalized in a certain range because it's necessary for neural network. There won't be great difference between predicted values and true values. In this case, MAE is a better metric to evaluate models.

II Analysis

Data Exploration and Visualization

Starbucks dataset contains three json files:

- portfolio.json - containing offer ids and meta data about each offer (duration, type, etc.)
- profile.json - demographic data for each customer (customer id, age, income, etc.)
- transcript.json - records for transactions, offers received, offers viewed, and offers completed

17000 customers personal data is provided in profile. There are 10 different offers in this test. And 17000 customers generated 306534 transaction records during 30 days.

The first step is to combine these files, then select variables that will be used in the model.

i) Univariate Analysis

According to the problems to be solved, the target variables are the total amount that each customer spends at Starbucks (*total_amount*) and the probability that each customer completes offers (*p_completed*). Both target variables are numeric and continuous.

The experiment lasted about a month. The plot below shows that the majority of the customers spent less than $400 during this period. Evidently it exists some outliers among the variable *total_amount*. And *total_amount* has a wide range, while *p_completed* has a small range. So the target variables still need some processing.

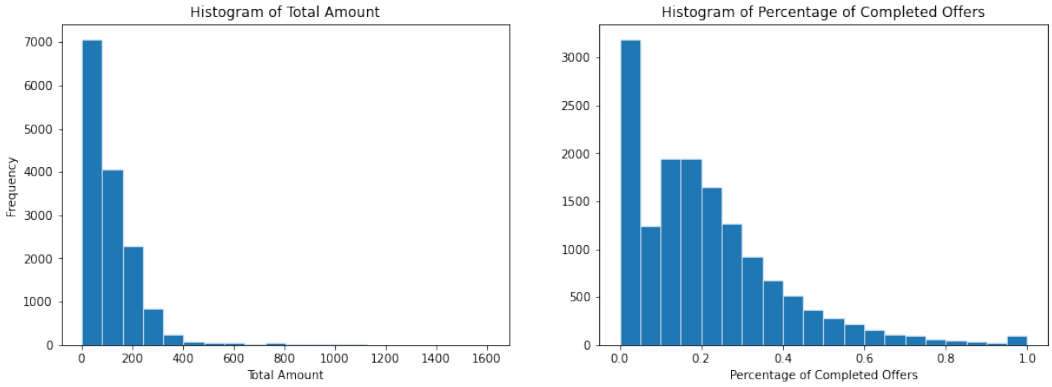

Figure 1: Distribution of Target Variable

And 5 variables are selected as input:

- age (int) - age of the customer
- gender (str) - gender of the customer (note some entries contain 'O' for other rather than M or F)
- income (float) - customer's income
- n_received (int) - the number of times the customer received offers
- p_viewed (float) - the proportion of offers the customer have viewed

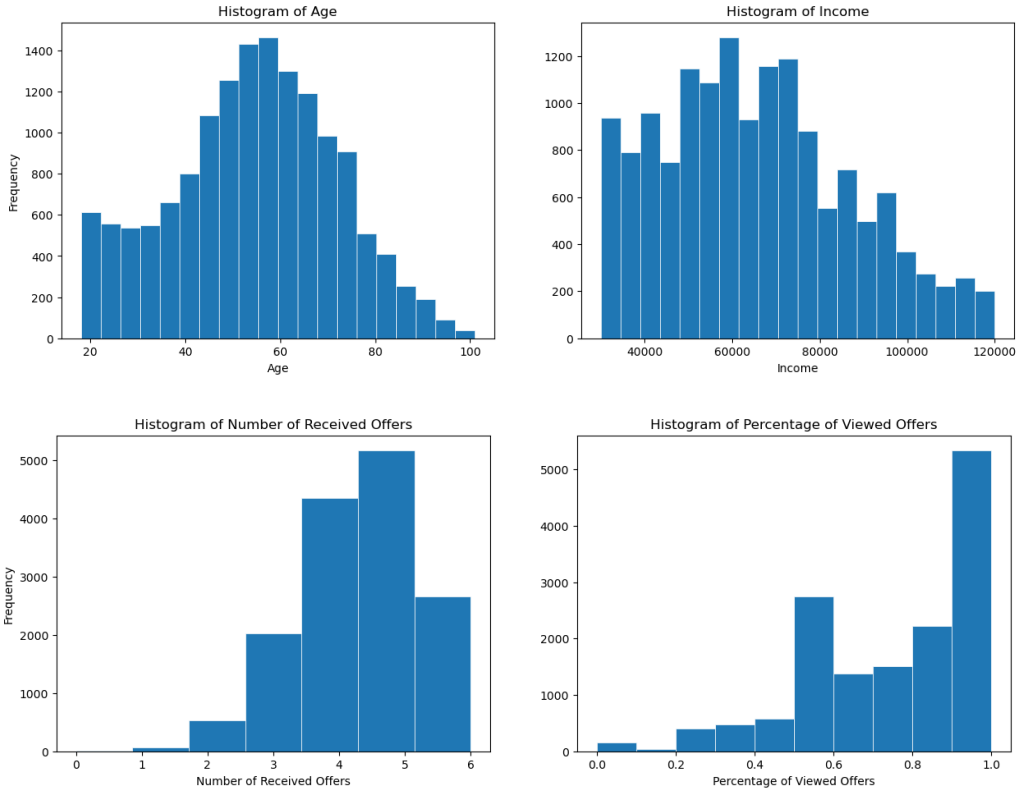

Figure 2: Distribution of Input Variable

gender is a categorical variable, the others are numerical variables. *age*, *gender* and *income* are from the file profile.json. The other variables may need more explanation.

1. total_amount and p_completed

In this simulated data, there are three different kinds of offers: an advertisement for a drink and two actual rewards including discount and BOGO (buy one get one free). Each offer has a validity period. If a customer opens the offer and has records of transactions during this validity period, we can assume that the customer is influenced by this offer. Unfortunately, by analyzing the data, I found that the real scenarios are complicated. The following situations make the problems more difficult:

1) When customers receive a bogo/discount reward, offer status (event) can be: offer received, offer viewed and offer completed. But there are only two possible status for an advertisement: offer received and offer viewed. There isn't a condition to determine whether this offer is completed.

2) It's possible for a customer to receive more than one offer at the same time. One transaction can be used for multiple offers as long as the transaction occurs during the validity period. Therefore we can't recognise which transaction is affected by which offer.

3) Some transactions happens during a validity period of an offer, but the offer isn't completed.

4) An offer has been completed but the customer never opens the offer.

As a simplification, if a bogo/discount offer isn't completed, the transactions happens during the validity period might be considered as transactions affected by an advertisement. And when an offer is completed, the customer has spent at least an amount of money which is equal to the difficulty of this offer. So the amount of completed offers is calculated from the difficulty of these offers. Additionally, total amount and completed amount must have a high correlation, because customers who spend more money may have higher completed amount values. Therefore, percentage of completed offers is selected as a target feature.

2. n_received and p_viewed

The dataset description doesn't mention the rules for sending offers to customers. However, the histogram above shows that the frequency customers receive the offers is quite different. This may influence the amount customers spend at Starbucks. A customer who receives more offers may buy more products. So I count the number of times each customer received offers. For the same reason, to avoid high correlation between the number of times that customer received offers and viewed offers, the proportion of offers the customer have viewed is selected as input.

3. other variables

In fact, the dataset contains more than these variables, like $difficulty$ (minimum required spend to complete a reward), *duration* (time for offer to be open, in days), *time* (time in hours that describe when a transaction happens since start of test), etc. These variables are useful, but it isn't meaningful to use them directly to predict the amount. Instead, they are used to generate new variables mentioned above.

There is another variable *became_member_on* that describes date when customer created an account. In general, the length of time a person becomes a member will influence customer behavior. But in this dataset, the time when this test started is not provided. This variable doesn't seem useful and is not selected.

Below, an example of input variables and target variables:

gender	age	income	n_received	p_viewed	total_amount	p_completed
F	55	112000	2	0	77.01	0.0649
F	75	100000	4	1	159.27	0.1256
M	68	70000	4	0.75	57.73	0.2598
M	65	53000	6	1	36.43	0.9607

Table 1: Profile

ii) Bi-variate Analysis

Pearson correlation coefficient can show correlations between two numeric variables. The table below shows that *income* and *total_amount* have the largest correlation coefficient. This means that high-income customers spend more at Starbucks.

	age	income	total_amount	p_completed	n_received	p_viewed
age	1.000000	0.306703	0.105787	-0.001228	-0.005827	0.049157
income	0.306703	1.000000	0.315033	-0.052571	-0.006450	0.069018
total_amount	0.105787	0.315033	1.000000	-0.168666	0.090215	0.175591
p_completed	-0.001228	-0.052571	-0.168666	1.000000	0.187172	-0.000305
n_received	-0.005827	-0.006450	0.090215	0.187172	1.000000	-0.061819
p_viewed	0.049157	0.069018	0.175591	-0.000305	-0.061819	1.000000

Table 2: Pearson Correlation Coefficient

Making scatter plot or box plot is another powerful method to find out the relationship between two variables. *age* and *income* also have a high correlation coefficient. So I create a scatter plot.

Figure 3: Relationship between age and income

Figure 3 shows that younger people have lower income. It's reasonable since young people always have less work experience. Few of them can get high income.

Intuitively, there may be a high correlation between some variables, for example *n_received* and *total_amount*, because customers will spend more when they receive more offers. But in the table of correlation coefficient, the actual coefficient is 0.090215 which is close to 0. So I make a box plot.

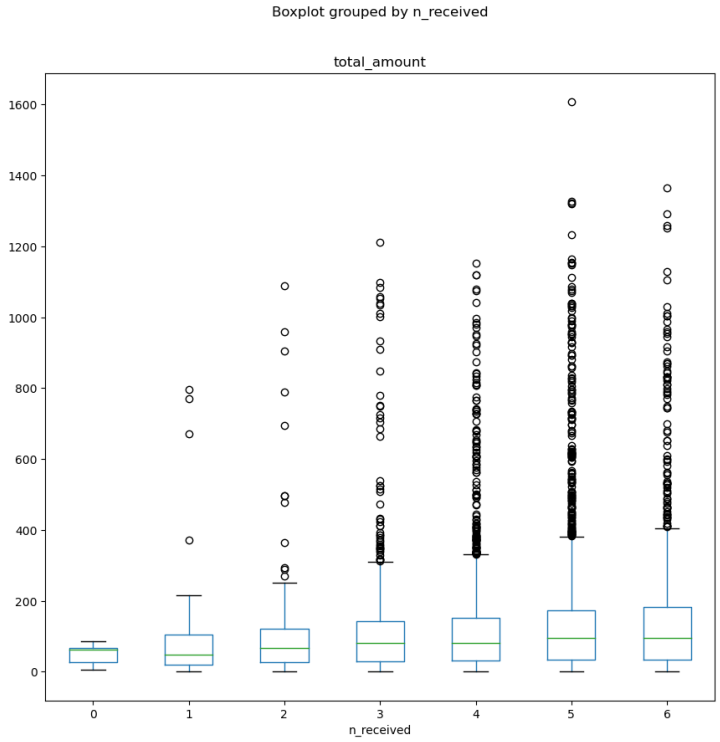

Figure 4: Relationship between n_received and total_amount

The result is as expected. There is a trend between *n_received* and *total_amount*. Another important point to note is that it exists outliers. This may be the reason that the correlation coefficient of the two variables is close to 0.

After analyzing the correlations between numeric variables, find the relationship between categorical variable and numeric variables. Figure 5 shows that *gender* may have an effect on *total_amount*. But the difference between *gender* and *p_completed* is not significant.

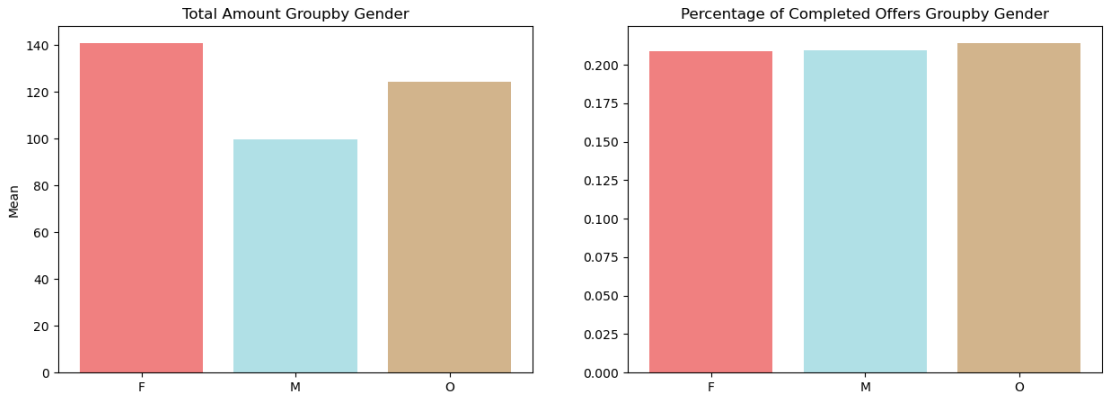

Figure 5: Relationship between gender and total_amount/p_completed

iii) Missing Value Treatment

After combining the files, profile contains 17000 observations and 7 variables. There are missing values in the dataframe. *gender* and *income* have 2175 missing values for the same samples which record 118 for *age*. 118 is obviously a default value for the variable *age*. In this case, if I replace all missing values with the same value of mean, median or mode, these samples will be considered to be similar. This might affect the accuracy of the model. So to delete these samples is a better way. After deleting miss values, profile has 14825 samples left.

iv) Outlier Treatment

The box plot and the distribution show that there is outliers. Z-score is a statistical measure that can be used to detect outliers:

$$z = \frac{x - \mu}{\sigma} \tag{3}$$

where μ is mean of the variable, and σ is standard deviation.

After the preprocessing, all variables will follow standard normal distribution. Z-score greater than +3 or less than -3 is considered as outliers. Then I remove the observations that has an absolute value of the z-score greater than 3. Finally, profile has 14110 observations.

Algorithms and Techniques

The problem to be solved is a regression problem. The target variables are the total amount that each customer spends at Starbucks (*total_amount*) and the probability that each customer completes offers (*p_completed*). In this project, three algorithms are implemented:

i) Linear Regression

Linear Regression is always used for finding out a linear relationship between variables and forecasting. In this problem, the task is to learn a mapping from 5-dimention input to 2-dimention output, then to predict the target variables. So it's a multivariate regression. And it's an appropriate and simple method to solve this problem. A linear regressor can be directly implemented by using the package `sklearn`.

ii) XGBoost

Considering the relationship between input and output may be non-linear, XGBoost and neural network which are non-linear algorithms are implemented to solve the problem. XGBoost is an effective algorithms and can get a high prediction accuracy. But xgboost doesn't support multiple output. `sklearn` provided a module `sklearn.multioutput.MultiOutputRegressor` which enables model to fit one regressor per target.

iii) Neunal Network

There are different types of neural networks, such as Artificial Neural Network (ANN), Recurrent Neural Network (RNN) and Convolutional Neural Network (CNN) [4]. RNN is generally used to solve time series problems, or to deal with language problems. CNN is designed specifically for computer vision. It's a powerful algorithm to handle the pixel data.

ANN consists of neurons, synapses, weights, biases, and functions, and is used to model complex non-linear relationships. An ANN is created by using `TensorFlow` and `keras`.

Benchmark

A multivariate linear regression model will be used as a benchmark model, because it's easy to implement and efficient to train. MAE measures, on average, the absolute values of the differences between predicted values and true values. It is used to determine whether other models are better compared with the linear regression model.

Since the data is scaled, after fitting the linear regressor, MAE is 0.1401. The scaled predictions can't show whether the model is good or not. So I unscale the predictions to origin form. And MAE for the two target is 30.4583. MAE of *total_amount* is 60.7934. MAE of *p_completed* is 0.1231. This means that the prediction of *total_amount* has an error of $60.7934 on average, and that the error of the probability that a customer complete offers is 12.31% on average.

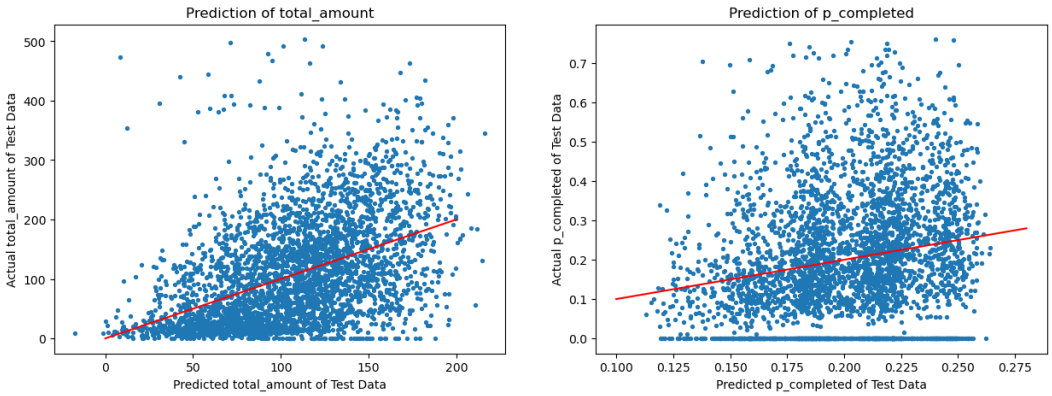

Figure 6: Results of Predictions Compared with Actual Data

The figure above shows that the majority of the points are distributed around the line $x = y$. But many of them are not close to the line. For the points whose actual value is 0, the predictions of linear regression are not accurate.

III Methodology

Data Preprocessing

Data processing contains the following steps:

- Data Conversion

- Feature Extraction
- Missing Values and Outliers Treatment
- Spliting Training Data and Test Data
- Feature scaling

i) Data Conversion

Converting data into a concise format can simplify analyzing and present information more clearly. In Starbucks dataset, some variables need to be converted. Before this, the first step is to create three dataframe for each json file.

id in dataframe profile & person in dataframe transcript

The two variables in different files represent the same information of customer id. They are in format long strings. It's not easy to identify each customer. So I replace customer id with integers.

value in dataframe transcript

In dataframe transcript, the variable *value* is in format dictionary which contains only one element and can be considered as additional information for the column *event*. If a customer behavior is related to an offer, *value* will be an id corresponding to this offer. And if a customer has purchased a product, *value* will be the amount this customer spent.

I separate dataframe transcript into trans and event. Dataframe trans contains all transactional events. Dataframe event contains all information about offer including offer id and status of offer i.e. offer received, offer viewed and offer completed.

Then I transform variable *value* in both dataframe, convert the dictionary format into its value. After transformations, *value* in dataframe trans is float representing amount of each transaction. And *value* in dataframe event is string representing offer id.

value in dataframe event and id in datafram portfolio

The two variables in different files represent the same information of offer id. Treatment for these two variables are same as customer id.

ii) Feature Extraction

This dataset is provided without a specific purpose on kaggle. Users can come up with some marketing strategies by analyzing this dataset. I want to predict the amount a customer will spend at Starbucks during the experiment period and to predict the probability that a customer will complete a reward. So I need to get the two target features. Additionally, not all variables can be used directly in models. I need to extract more variable from existing data.

total_amount

A function `calculate_amount` is created to calculate the amount have been spent of each customer. Parameters are an integer as customer id and a datafram containing transactional records which is the dataframe trans here. It returns a float which is total amount the customer has spent. Then loop through each customer

id using this function to get the *total_amount* list, and insert this list in dataframe profile.

```
def calculate_amount(id, trans_df):
    """ Given an id and a dataframe, calculate the total amount
                                        of money a person spent
    :param id: int, person id
    :param trans_df: dataframe, transcript with information of
                                        amount
    :return: float, the total amount of money the person (id)
                                        spent
    """
    return trans_df.loc[trans_df['person'] == id]['amount'].sum()
```

p_completed

In order to get more information in detail, a function `user_transcript` is created. It takes an customer id and a dataframe transcript as input, and returns a sub-dataframe with all actions of this customer.

```
# get one user's transcript
def user_transcript(user_id, transcript):
    """ Given a user id, create a dataframe containing all
                                        actions of this user
    :param user: int
    :param transcript: a dataframe (transcript)
    :return: a subset of the dataframe corresponding to the given
                                         id
    """
    user_actions = transcript.loc[transcript['person'] == user_id
                                        ]
    return user_actions
```

More information can be found by seeing one customer's transcript.

	person	event	time	amount	offer_id	offer_type
53174	0	offer received	168	NaN	discount4	discount
85290	0	offer viewed	216	NaN	discount4	discount
110828	0	offer received	336	NaN	discount1	discount
130147	0	offer viewed	348	NaN	discount1	discount
135224	0	transaction	360	0.35	NaN	NaN
150596	0	offer received	408	NaN	discount3	discount
163374	0	offer viewed	408	NaN	discount3	discount
167626	0	transaction	414	0.74	NaN	NaN
182544	0	transaction	444	1.89	NaN	NaN
201570	0	offer received	504	NaN	discount2	discount
214274	0	offer viewed	504	NaN	discount2	discount
218392	0	transaction	510	5.04	NaN	NaN
230411	0	transaction	534	0.38	NaN	NaN
237364	0	transaction	552	2.12	NaN	NaN
237365	0	offer completed	552	NaN	discount3	discount
237366	0	offer completed	552	NaN	discount2	discount
245122	0	offer received	576	NaN	discount3	discount
262137	0	offer viewed	582	NaN	discount3	discount
274519	0	transaction	606	4.61	NaN	NaN
283959	0	transaction	630	0.06	NaN	NaN
301915	0	transaction	696	5.21	NaN	NaN

Figure 7: Transcript of Customer 0

To check the frequency of each offer in dataframe transcript, I found that *event* which describe offer status has difference between reward offers and advertisement. If an offer type is bogo or discount, when offers are sent to customers,

it can be three status: offer received, offer viewed and offer completed. But for advertisement, there are only two possible status: offer received and offer viewed. Because advertisement doesn't have an difficulty or reward, there is no condition to determine whether it is completed.

From customer 0's transcript, we can see that this customer received respectively discount3 and discount2 at time 408 and 504. The three transactions which happened from 510 to 552 helped customer 0 completed the two offers at the same time.

This also happens for advertisement. It's possible for a customer to receive more than one offer at the same time. Therefore I can't recognise a transaction is affected by an advertisement or a reward.

More difficulties have been discussed in the previous section. So I calculate the amount of completed offers for each customer ($offer\_amount$) with function `calculate_amount`. To avoid high correlation between $total\_amount$ and $offer\_amount$, I calculate the percentage of completed offers ($p\_completed$). Then insert $p\_complete$ into dataframe profile.

n_received

Considering the frequency customers receive the offers is quite different. This may influence the amount customerss spend at Starbucks. A customer who receives more offers may buy more products. So I count the number of times customer have received offers with function `count_offers`.

```
def count_offers(user_transcript, offer_status):
    """ Given a user's transcript, count how many times the
                                        status appears
    :param user_transcript: dataframe, a user transcript
    :param offer_status: str, 'offer received' or 'offer viewed'
    :return: int, times the str appears
    """
    status_df = user_transcript.loc[user_transcript['event'] ==
                                        offer_status]
    return status_df.shape[0]
```

This function takes in a datafram and an offer status, and counts the times the status appears. After loop through each customers, insert $n\_received$ into dataframe profile.

p_viewed

Using the function `count_offers` again, I get a new variables n_viewed. Then I calculate p_viewed. For the same reason as $p\_completed$, I keep the variable p_viewed as input.

iii) Missing Values and Outliers Treatment

Missing values and outliers are deleted from dataframe profile. The reasons and rules are explained in the previous section. The figure below shows the datafram of missing values.

Outliers are detected by z-score. Value which has an absolute value of the z-score greater than 3 is regarded as outlier.

After this step, profile has 14110 observations.

	gender	age	became_member_on	income	total_amount	p_completed	n_received	p_viewed
0	None	118	20170212	NaN	20.40	0.833333	5	1.00
2	None	118	20180712	NaN	14.30	0.000000	2	1.00
4	None	118	20170804	NaN	4.65	0.000000	5	0.60
6	None	118	20170925	NaN	0.00	0.000000	5	1.00
7	None	118	20171002	NaN	0.24	0.000000	5	0.80
...	...	...	...	...	...	...	...	...
16980	None	118	20160901	NaN	25.25	0.198020	5	0.60
16982	None	118	20160415	NaN	31.02	0.548034	4	0.75
16989	None	118	20180305	NaN	0.76	0.000000	5	1.00
16991	None	118	20160116	NaN	21.88	0.000000	4	0.25
16994	None	118	20151211	NaN	12.78	0.000000	4	1.00

2175 rows × 8 columns

Figure 8: Transcript of Customer 0

iv) Spliting Training Data and Test Data

I use `sklearn.model_selection.train_test_split` to split the dataframe into training data and test data. 25% samples are allocated to the test data.

v) Feature scaling

I use `sklearn.preprocessing.MinMaxScaler` to normalize training data. And it should be noted that the test data should be scaled with the same scaler as training data. After fitting estimators and prediction, the predicted values should be unscaled using the same scaler.

For neural network, data normalisation or standardization is a required step and very important. However in a linear regression and xgboost, this step is not necessary. Considering different variables have quite different range in this dataset, data will be scaled for all models in this project.

Implementation

In this project, the task is to learn a mapping from 5-dimention input to 2-dimention output. Three algorithms are implemented to solve the problem. All algorithms are built-in machine learning algorithms provided by python libraries.

The models can be implemented by the following steps:

1) Creating an estimator

2) Fitting the estimator with the scaled training data

3) Predicting on test data

4) Calculating MAE for predictions

5) Unscaling predicted data

6) Comparing the results with actual data

However, in the process of implementation, there are still differences among the models.

i) Linear Regression

Linear regression is selected as benchmark model. It supports multiple outputs directly. A linear regression estimator can be created by using `sklearn.linear_model.LinearRegression`. Then the steps are the same as the above process.

ii) XGBoost

XGBoost is different from linear regression, because this algorithm is designed for predicting a single numeric value. Fortunately, `sklearn` provided a module `sklearn.multioutput.MultiOutputRegressor` . This strategy enables model to fit one regressor per target.

I use library `xgboost` to create an estimator. Many parameters in xgboost can be trained. I select eight of them to not use the default values:

- n_estimators = 500
- max_depth = 5
- min_child_weight = 3
- learning_rate = 0.025
- gamma = 0.2
- subsample = 0.9
- colsample_bytree = 0.7
- reg_lambda = 0.5

Then `sklearn.multioutput.MultiOutputRegressor` helps me to fit the estimator for both target features using training data. The last steps are the same as before.

iii) Neunal Network

This algorithm is implemented with `TensorFlow` and `keras`. A multilayer perceptron (MLP) is a fully connected class of artificial neural network (ANN).

Different number of layers and nodes of each layer has been tried for the network. Finally I create a MLP with 3 non-linear hidden layers. The first two layers have 10 nodes each. The last layer has 2 nodes since the output is 2-dimention. For each layer I use `TensorFlow` built-in activation function `tensorflow.nn.sigmoid`.

There is one more step for this algorithm. The step is to compile model after creating the MLP model. `tensorflow.keras.Model.compile` configures the model for training. It defines the optimizer, the loss function and the metrics:

- optimizer = tf.keras.optimizers.Adam(learning_rate=0.001)
- loss = tf.keras.losses.mean_squared_error
- metrics = [tf.keras.metrics.MeanAbsoluteError()]

After compilation, fit model with training data, predict target variables using trained model, calculate MAE and compare results with actual data.

Refinement

This part is mainly for XGBoost regressor, since the result of multioutput-xgboost is just a little bit better than linear regression which is benchmark model in this project.

	linear regressor	multioutput-xgboost regressor
total_amount MAE	60.7934	60.0927
p_completed MAE	0.1231	0.1224

Table 3: MAE of Benchmark Model and XGBoost Model

In fact, there is a disadvantage when using `MultiOutputRegressor`. Although `MultiOutputRegressor` can fit one regressor per target, the parameters of the regressors are same for different target features. So I split two target features and train the models separately for the two target features.

This time, I want to search specified parameter values for the two models. I use `sklearn.model_selection.GridSearchCV` for parameter tuning. It's a powerful function provided in `sklearn`. It's efficient and allows more than one processes running in parallel for training jobs. But this can take long time if there are many parameters. For example, in this problem of predicting one feature, I chose 9 parameters in xgboost estimator. Each parameter has 3 values to try. There will be 3^9 models to train if I don't divide the work into two parts. Thus for each `XGBRegressor`, I use `GridSearchCV` two times to get the parameters that make each model perform the best.

Parameters for total_amount:

- n_estimators: 700
- max_depth: 6
- min_child_weight: 5
- learning_rate: 0.02
- gamma: 0.1
- subsample: 0.8
- colsample_bytree: 0.6
- reg_lambda: 0.8
- reg_alpha: 0.6

Parameters for p_completed:

- n_estimators: 800
- max_depth: 5
- min_child_weight: 3
- learning_rate: 0.05
- gamma: 0.5
- subsample: 0.9
- colsample_bytree: 0.6
- reg_lambda: 0.4
- reg_alpha: 0.5

After getting the best estimators, make predictions separately using the two estimators for different features. Then I combine the results, since if I don't combine them, the predictions can't be unscaled. Finally I can compare the result with benchmark model and multioutput-xgboost model.

	linear regressor	multioutput-xgboost	2-xgboost
total_amount MAE	60.7934	60.0927	59.9109
p_completed MAE	0.1231	0.1224	0.1225

Table 4: MAE of Benchmark Model and XGBoost Models

Comparing with the results of multioutput-xgboost, by training estimator separately, MAE of *total_amount* is reduced a bit but not too much. Predictions of *p_completed* are not as good as multioutput-xgboost estimator.

IV Results

Model Evaluation and Validation

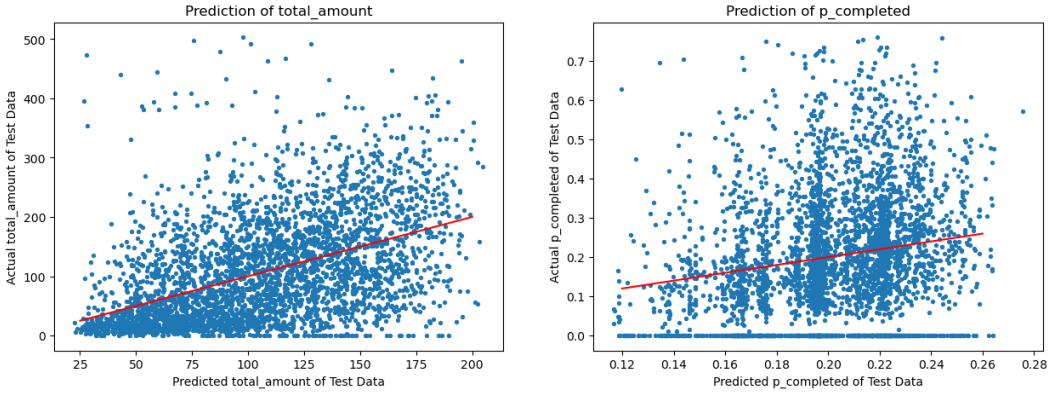

Figure 9: Results of Predictions Compared with Actual Data - XGBoost

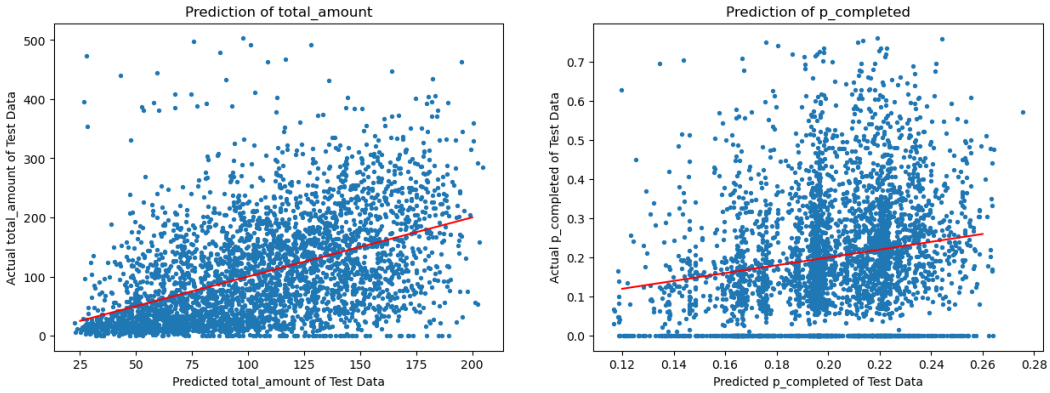

Figure 10: Results of Predictions Compared with Actual Data - MLP

Figure 9 and Figure 10 show that predictions of XGBoost model and MLP model are similar with benchmark model. Most of the points located around the line $x = y$. But the variance of errors is not small. The two algorithms still don't solve the problem of inaccuracy when predicting samples with actual value 0.

Comparing these algorithms, XGBoost which was trained separately for each target feature performs the best among the three models. It's better to predict target variables using this model.

	total_amount MAE	p_completed MAE
linear regressor	60.7934	0.1231
multioutput-xgboost	60.0927	0.1224
2-xgboost	59.9109	0.1225
MLP (ANN)	60.2638	0.1232

Table 5: MAE of All Models

To guarantee that the model will work for similar unknown data. I repartitioned the dataset into training data and test data to evaluate the model performance. Then create xgboost estimators using the same parameters and fit estimators with new training data. Parameters of the best estimators are mentioned in previous section.

MAEs of predictions on new data are 59.4792 for *total_amount* and 0.1264 for *p_completed*. Errors of *total_amount* are even lower than origin data. But predictions of *p_completed* are not as good as before. The prediction of *total_amount* has an error of $59.4792 on average, and that the error of the probability that a customer complete offers is 12.64% on average.

Justification

The performances of the three algorithms are quite similar. MAEs of each model are listed in Table 5. XGBoost gets the best result with MAE values equal to 59.9109 and 0.1225 for *total_amount* and *p_completed* respectively.

Linear regression was selected as benchmark model in this project. Comparing with the benchmark model, XGBoost estimator performs better than it. This may because XGBoost is a non-linear algorithm. And I trained two models for different target features, and tuned specific parameters for each model. This step may improve the model performance.

However the accuracy isn't improved too much. I think that there may be some problems with the data. Figure 9 and Figure 10 show that there still some outliers in variable *total_amount*, although some of them are already deleted. And there are many customers who never bought products or completed an offer. This may make models inaccurate.

Considering Starbucks aims to find an appropriate strategy to send offers to customers. The two target variables are selected to solve this problem. This result with such error can still provide some ideas. In general, there is no harm in sending offers for company.

V Conclusion

Reflection

I did this project following a general machine learning workflow:

- Data Retrieving
- Data Processing
- Training Model
- Parameter Tuning
- Model Evaluation

The first two steps can be found in `Data.ipynb` and the others are in `model.ipynb`.

This project is extremely open. Only a dataset is provided. I am free to analyze in a way that I'm interested in. I decide to find the relationships between customers and offers so that I can come up with a strategy of sending offers.

In fact, data processing isn't easy for this dataset, since it contains three file with different structures. And I have to extract features on my own. However this is also the aspect that I think is the most interesting. The situations in this experiment are complicated. Like I mentioned in section Data Exploration, it's impossible to identify a transaction was affected by which offer. I found these interesting situations by checking many customers transcripts. I have done my best to extract information that I think may be useful.

Then I used processed data to develop models. I thought there will be obvious difference between each model. But the results of each model are close. This may demonstrate that all models work well. It may be the data lead to this result.

Furthermore, it's just a simplified experiment which simulated data with only one product. Starbucks has actually dozens of products. When these data are mixed together, data processing will be more difficult.

Fortunately, I think there is a high tolerance for this problem. Because sending rewards to customers is inherently random. In real-world scenario, Starbucks can provide a standard. For example, if the probability that a customer completes a reward is greater than 50%, this customer can receive more rewards. This don't means do not sent rewards to the population with a probability lower than 50%. But Starbucks can send them a small amount of rewards or send them advertisement. Because they will spend at Starbucks whether they receive the reward or not.

Improvement

I discussed the potential problem in Reflection. Data can significantly affect model performance. In this project, many features are extracted from Starbucks dataset. There may be some information that I missed.

The feature *p_completed* is obtained by calculating difficulty of completed offers only. In fact, many transactions happened during an offer's validity period without completing this offer. So true value of *p_completed* would be a little bit higher than the data I used in the model. I set this rule because some of these transactions happened in a validity period of another offer.

There is a variable *become_member_on* describes date when customer created an account. In general, the length of time a person becomes a member will influence customer behavior. Starbucks dataset doesn't provide when the experiment started. But the date data is useless for predictions.

If more information can be saved in Starbucks application, the problems caused by data can be solved easily. And more useful features can be extracted. I think this will improve model performance.

References

[1] Starbucks app customer rewards program data

[2] Mean absolute error - Wikipedia

[3] MAE and RMSE — Which Metric is Better?

[4] What's the principal difference between ANN,RNN,DNN and CNN?