2024

StarBucks Capstone Project

BSH DATA SCIENTIST NANODEGREE - UDACITY

GOLDARAZ MARTIN, IÑAKI

Problem Statement:

This project explores simulated data from the Starbucks Rewards app, mimicking user interactions with periodic offers, from ads to discounts. The challenge lies in understanding varied offer distributions among users and identifying demographic segments with optimal responses.

Project goal:

Using transaction, demographic, and offer data, our goal is to create a model which can predict offer completion based on user demographics and offer attributes.
The results on said model should give us an answer on whether a client will use and buy an offer or not based exclusively on this.

Metrics:

After an initial phase of cleaning and understanding the data we are working with, the idea is to develop a model and based on its performance results, evaluate it. Said metrics include:

- Accuracy: It measures overall correctness of the model.

$$\text{Accuracy} = \frac{\text{Correct predictions}}{\text{All predictions}}$$

- Precision: Measures the correctness of positive predictions.
- Recall: Measures the ability to identify positive instances
.
- F1 score: Represents the balance between precision and recall.
- Support: Provides information about the distribution of classes.

Content Index:

1. Introduction 3
1.1 Data Dictionary: 3
2. Data Cleaning and Preparation. 5
2.1 Portfolio Dataset: 5
2.2 Profile Dataset: 5
2.3 Transcript Dataset: 6
3. Exploratory Analysis and Visualization. 6
3.1 Demographics. 6
3.2 Offer Related Variables. 8
4 Modelling and Predictions. 10
4.1 Results summary: 13
5 Conclusions. 14
5.1 Reflections and Improvements: 14

1. Introduction

In this project, we will be exploring a dataset that simulates customer behavior on the Starbucks Rewards mobile app. Starbucks regularly sends out offers, which can be advertisements or actual deals like discounts or BOGO (buy one get one free), creating a varied experience for users. The challenge at hand is to decipher patterns in user responses to different offers, considering not all users receive the same offer.

The task is to merge transaction, demographic, and offer data to identify which demographic groups respond most favorably to specific offer types. It is worth noting that this dataset is a simplified version of the actual Starbucks app, featuring a single product in its simulator, while Starbucks offers a diverse product range.

Each offer in the dataset has a validity period, even informational ones, influencing customer behavior for a specified time after receiving the ad. For instance, a BOGO offer may be valid for only 5 days. The dataset includes transactional data showing user purchases, offer timestamps, views, and completions. Importantly, users may make purchases on the app without having received or seen an offer. As we navigate this data, our goal is to build insights and develop a model that can predict user responses to different offers based on demographic and transactional information.

1.1 Data Dictionary:

The data is contained in three files:

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

Here is the schema and explanation of each variable in the files:

portfolio.json

- 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)

profile.json

- 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

transcript.json

- event (str) - record description (ie transaction, offer received, offer viewed, etc.)
- 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

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

(10, 6)

	age	became_member_on	gender	id	income
0	118	20170212	None	68be06ca386d4c31939f3a4f0e3dd783	NaN
1	55	20170715	F	0610b486422d4921ae7d2bf64640c50b	112000.0
2	118	20180712	None	38fe809add3b4fcf9315a9694bb96ff5	NaN
3	75	20170509	F	78afa995795e4d85b5d9ceeca43f5fef	100000.0
4	118	20170804	None	a03223e636434f42ac4c3df47e8bac43	NaN

(17000, 5)

	event	person	time	value
0	offer received	78afa995795e4d85b5d9ceeca43f5fef	0	{'offer id': '9b98b8c7a33c4b65b9aebfe6a799e6d9'}
1	offer received	a03223e636434f42ac4c3df47e8bac43	0	{'offer id': '0b1e1539f2cc45b7b9fa7c272da2e1d7'}
2	offer received	e2127556f4f64592b11af22de27a7932	0	{'offer id': '2906b810c7d4411798c6938adc9daaa5'}
3	offer received	8ec6ce2a7e7949b1bf142def7d0e0586	0	{'offer id': 'fafdcd668e3743c1bb461111dcafc2a4'}
4	offer received	68617ca6246f4fbc85e91a2a49552598	0	{'offer id': '4d5c57ea9a6940dd891ad53e9dbe8da0'}

(306534, 4)

As we can see in the last DataFrame pictured, column event registers the offer status:

- Offer received
- Offer viewed
- Transaction
- Offer completed

In the same fashion, we can observe the offer_type column:

- Bogo: Buy one get one free. Clients spend X amount to get back that same value.
- Discount: Clients need to spend X amount to get back a fraction of said amount.
- Informational: There is not a direct reward for the client, but neither any amount of money needed to be spent for it.

2. Data Cleaning and Preparation.

Each one of the Datasets is treated and analyzed separately depending on its information and how it is presented.

2.1 Portfolio Dataset:

The main idea is to create additional columns based on the source of the offer, instead of only one column with the offer type.
Additionally some renaming of the columns for easier work later on:

	difficulty	duration	offer_id	offer_type	reward	email	mobile	social	web
0	10	7	ae264e3637204a6fb9bb56bc8210ddfd	bogo	10	1	1	1	0
1	10	5	4d5c57ea9a6940dd891ad53e9dbe8da0	bogo	10	1	1	1	1
2	0	4	3f207df678b143eea3cee63160fa8bed	informational	0	1	1	0	1
3	5	7	9b98b8c7a33c4b65b9aebfe6a799e6d9	bogo	5	1	1	0	1
4	20	10	0b1e1539f2cc45b7b9fa7c272da2e1d7	discount	5	1	0	0	1

2.2 Profile Dataset:

For this dataset we want mainly want two things, one is to clarify the id in here from the order id in the Portfolio dataset, and second to obtain a more clear date on when the client became a member from its original format.

	age	became_member_on	gender	customer_id	income
0	118	2017-02-12	None	68be06ca386d4c31939f3a4f0e3dd783	NaN
1	55	2017-07-15	F	0610b486422d4921ae7d2bf64640c50b	112000.0
2	118	2018-07-12	None	38fe809add3b4fcf9315a9694bb96ff5	NaN
3	75	2017-05-09	F	78afa995795e4d85b5d9ceeca43f5fef	100000.0
4	118	2017-08-04	None	a03223e636434f42ac4c3df47e8bac43	NaN

2.3 Transcript Dataset:

Similarly to the portfolio dataset, we need to create new columns for the event column. The column 'value' needs some editing as well, dividing it into two subcolumns (amount and offer_id).

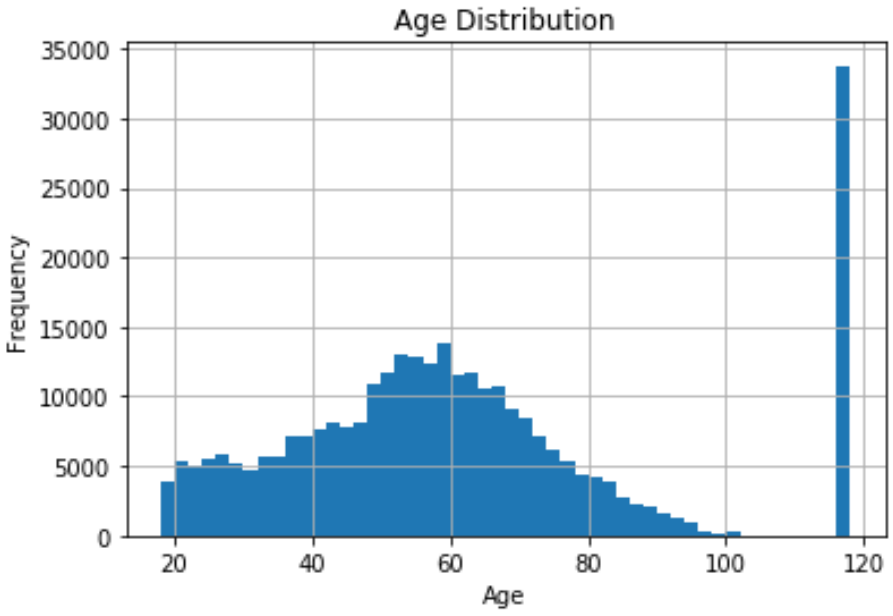
	customer_id	time	offer completed	offer received	offer viewed	transaction	offer_id	amount
0	78afa995795e4d85b5d9ceeca43f5fef	0	0	1	0	0	9b98b8c7a33c4b65b9aebfe6a799e6d9	NaN
1	a03223e636434f42ac4c3df47e8bac43	0	0	1	0	0	0b1e1539f2cc45b7b9fa7c272da2e1d7	NaN
2	e2127556f4f64592b11af22de27a7932	0	0	1	0	0	2906b810c7d4411798c6938adc9daaa5	NaN
3	8ec6ce2a7e7949b1bf142def7d0e0586	0	0	1	0	0	fafdcd668e3743c1bb461111dcafc2a4	NaN
4	68617ca6246f4fbc85e91a2a49552598	0	0	1	0	0	4d5c57ea9a6940dd891ad53e9dbe8da0	NaN

3. Exploratory Analysis and Visualization.

Once we have the datasets properly organized, it is time to make some preliminary investigations, just to understand better the data we are working with. Then, based on this information it is time to make some decisions on how we want to continue further studies, whether it is using predictive models, statistical regressions or even Neural Networks or other techniques.

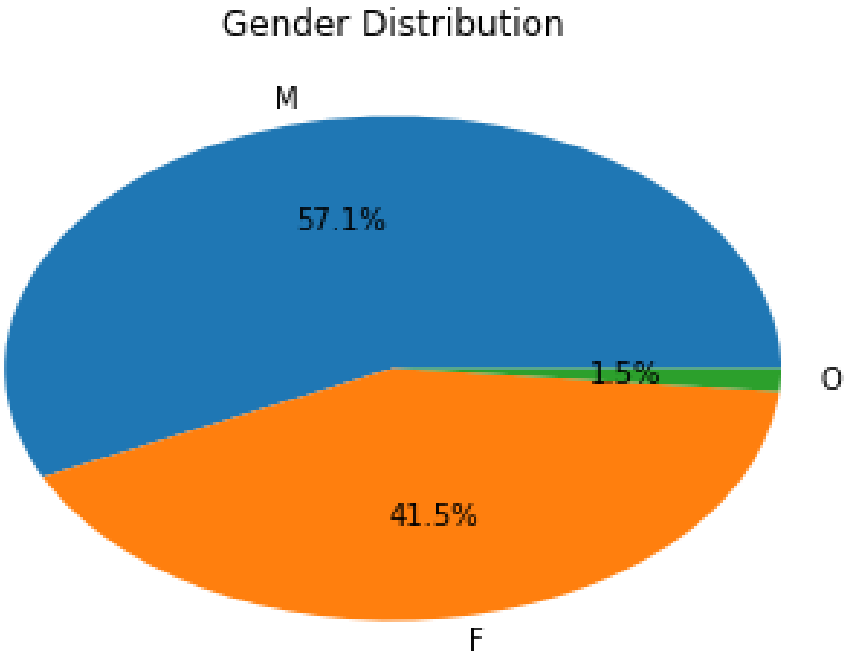
3.1 Demographics.

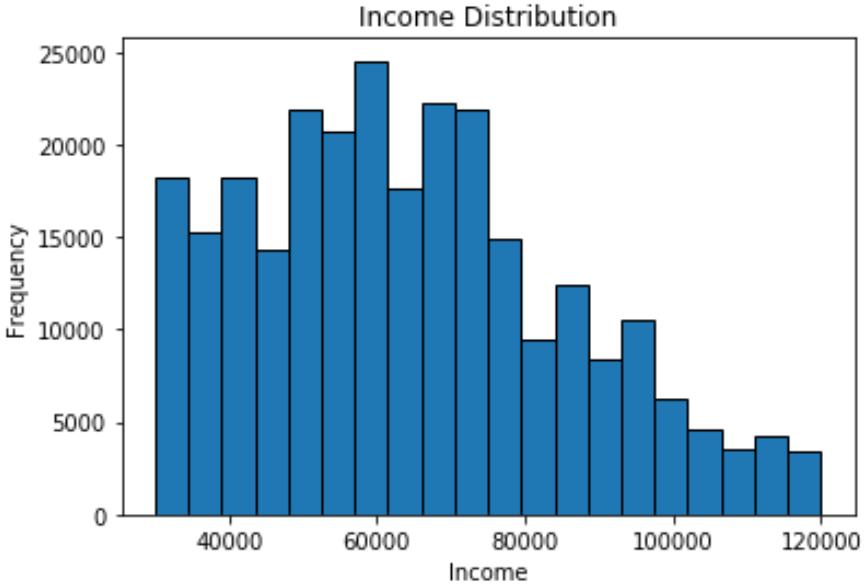
The age of the clients is represented in the following plot, although I personally expected it to be more skewed towards young people since Starbucks is a chain usually more popular in younger generations.



There are 33772 inputs of customers being 118 years old, which is probably a wrong input/missing data.

We can also take a deeper look into the demographics by looking at the income and the gender distribution on the data:



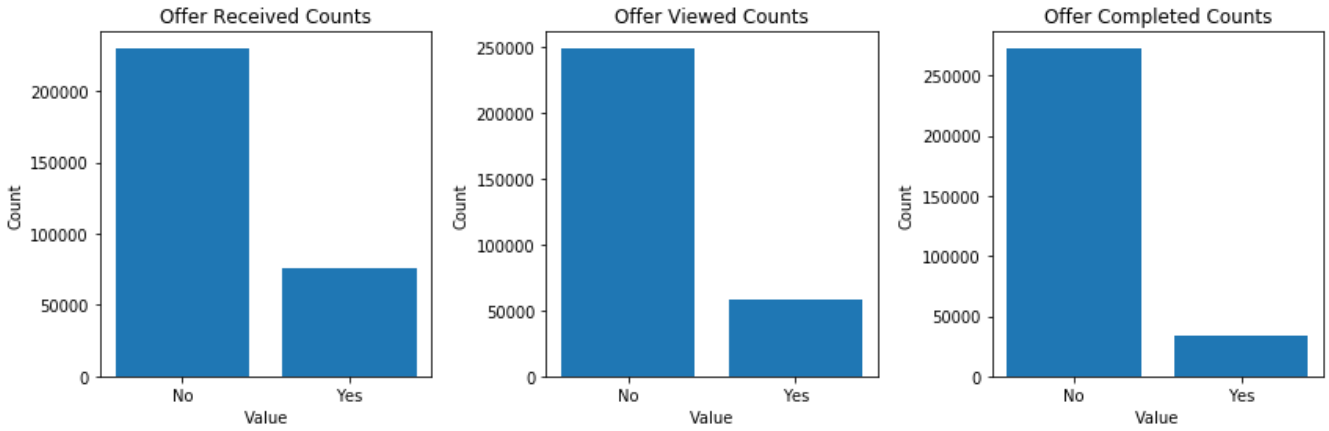


count	272762.000000
mean	64337.000755
std	21243.762941
min	30000.000000
25%	48000.000000
50%	62000.000000
75%	78000.000000
max	120000.000000

There are quite a few NaN values in the Income column, however after a little bit of cleaning we obtain the average income, which is in the 60-65k range, with a standard deviation of around 21k.

3.2 Offer Related Variables.

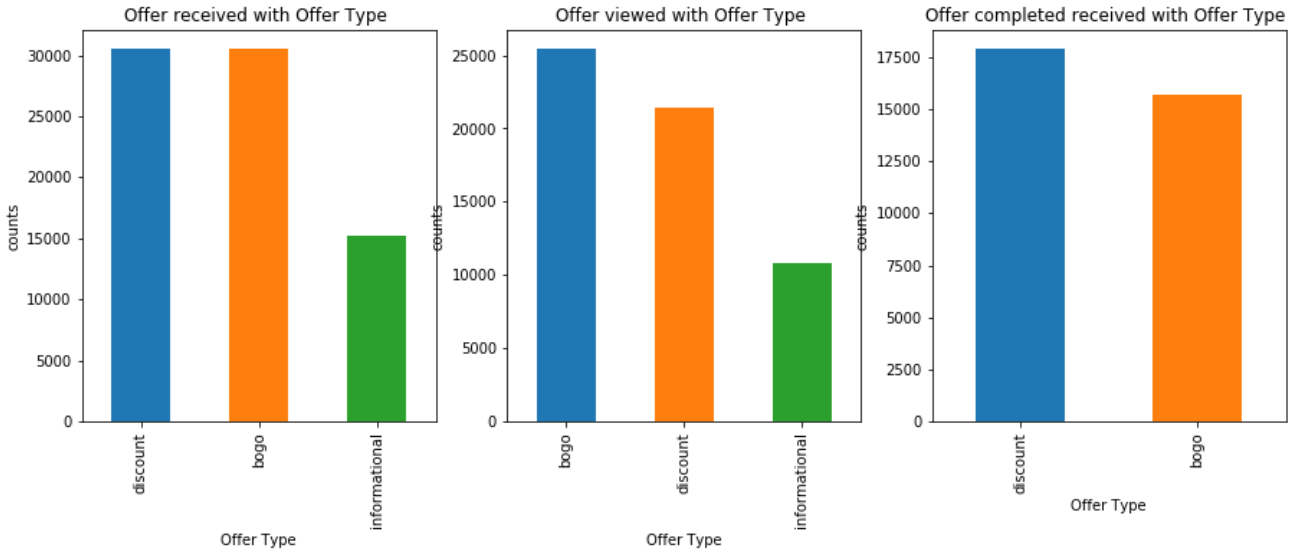
The idea in this part is to dive deeper into the amount of offers every customer receives, how they are received and such insights.
On a first approach, we can look at the distribution on the offer category:




```
Proportion of 'Yes' values in Offer Received: 24.88%
Proportion of 'Yes' values in Offer Viewed: 18.83%
Proportion of 'Yes' values in Offer Completed: 10.95%
```

In general, the offer completion are distributed in a fairly standard way, It is also interesting to look at it from the point of view where once the offer is viewed by the client, there are relatively high chances of it being completed.

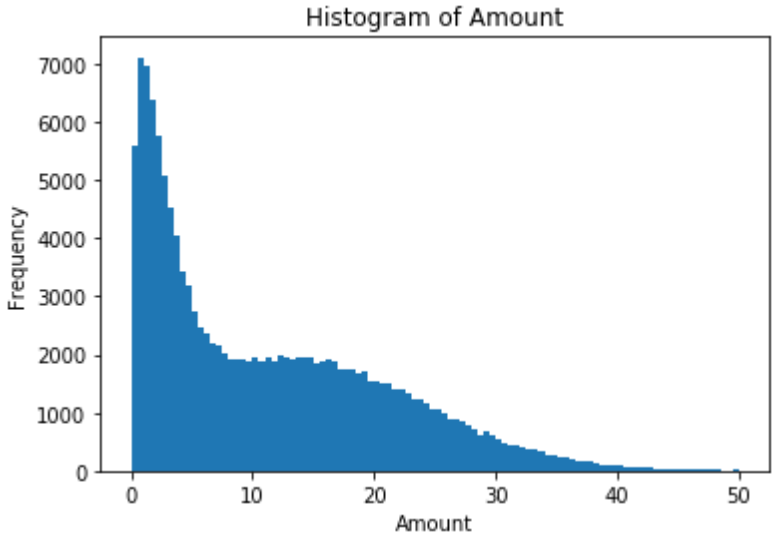
Now, to see how everything is shaped up based also in the offer category we can work something:




```
Proportion of 'bogo' offers completed over received: 51.38%
Proportion of 'bogo' offers completed over viewed: 61.57%
Proportion of 'discount' offers completed over received: 58.64%
Proportion of 'discount' offers completed over viewed: 83.52%
```

There are also quite a few other indicator we can look at to have a better understanding of the offers and the customer behavior:

A good metric is the amount of money saved by the customer depending on the offer completion. Once we finish cleaning the information from the column, we observe the following distribution:



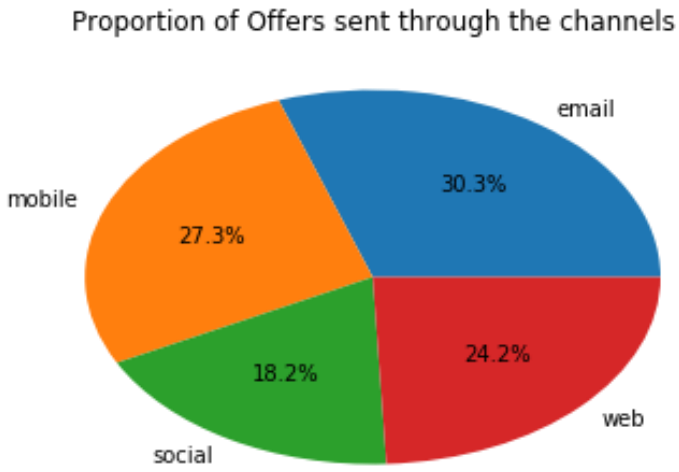
count	138953.000000
mean	12.777356
std	30.250529
min	0.050000
25%	2.780000
50%	8.890000
75%	18.070000
max	1062.280000

It is a skewed distribution on the left side with 50% of the total money savings from the customers being under 10$.

On a last note, it is interesting to look at the channel from where the offers are received.

The circle graph represents how frequent each channel is used to send the offers, with mail being the predominant one.

On the other hand social media is the least used.



4 Modelling and Predictions.

So, after the initial understanding of the datasets we are working with we arrived at seeing a good overview on what info we have, the customers attributes and the posible insights we are looking for. Despite the complexity of the information we have available i want to dive further into the relationship between demographics and the kind of offers that are more suitable to the customers based on it.

For the modelling we will start with a model as simple as posible and evaluate the results obtained with it, before diving into more complex challenges or models where we can find ourselves with problems such as overfitting or the need for more computational resources. We will use a regression model.

The code used is as it follows, which will be explained more in detail subsequently:

```
## 3.1 Data split and Model Training:
from sklearn.preprocessing import StandardScaler, OneHotEncoder
from sklearn.ensemble import RandomForestClassifier
from sklearn.model_selection import train_test_split
from sklearn.metrics import classification_report, accuracy_score


scaler = StandardScaler()

# Identify categorical columns for one-hot encoding
categorical_cols = ['gender', 'offer_type']

# Identify numerical columns for scaling
numerical_cols = ['age', 'income', 'reward', 'difficulty']

# Replace zero values in 'income' with the mean value
all_data['income'] = all_data['income'].replace(0, all_data['income'][all_data['income'] > 0].mean())

# Handle missing values
all_data.fillna(0, inplace=True)

# Handle infinite values
all_data.replace([np.inf, -np.inf], 0, inplace=True)

# Preprocess categorical columns with one-hot encoding
X_categorical = pd.get_dummies(all_data[categorical_cols], drop_first=True)

# Concatenate one-hot encoded categorical columns with numerical columns
X = pd.concat([all_data[numerical_cols], X_categorical], axis=1)
y = all_data['offer completed']

# Split the data
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)

# Scale numerical columns
X_train[numerical_cols] = scaler.fit_transform(X_train[numerical_cols])
X_test[numerical_cols] = scaler.transform(X_test[numerical_cols])

# Train the model
model = RandomForestClassifier()
model.fit(X_train, y_train)

# Make predictions
y_pred = model.predict(X_test)

# Evaluate the model
print("Accuracy:", accuracy_score(y_test, y_pred))
print("Classification Report:\n", classification_report(y_test, y_pred))
```

1. **Data Preprocessing:**

    - **Standardization (StandardScaler):** This technique was applied to numerical features, ensuring that they were on a consistent scale. The StandardScaler scales features to have a mean of 0 and a standard deviation of 1. This is particularly crucial for algorithms sensitive to the magnitude of features, such as the Random Forest classifier.

    - **One-Hot Encoding (OneHotEncoder):** Categorical columns ('gender', 'offer_type') were encoded into binary vectors, creating a numerical representation of categorical data. The drop-first parameter was set to mitigate multicollinearity by removing one category from each encoded feature.

    - **Handling Zero Income Values:** Zero values in the 'income' column were replaced with the mean of non-zero entries. This preprocessing step aimed to address potential bias introduced by zero incomes, ensuring a more representative feature.

    - **Handling Missing and Infinite Values:** Missing values were imputed with zeros, and infinite values were replaced with zeros. This strategy aimed to maintain data integrity and facilitate model training.

2. **Feature Concatenation:**

    - **Combining Categorical and Numerical Features:** The one-hot encoded categorical columns were concatenated with numerical features ('age', 'income', 'reward', 'difficulty') to form the feature matrix (X). This step ensured that the model could effectively utilize both categorical and numerical information for predictions.

3. **Data Splitting:**

    - **Train-Test Split (train_test_split):** The dataset was split into training and testing sets (80% training, 20% testing) to assess the model's generalization performance on unseen data.

4. **Numerical Feature Scaling:**

    - **Scaling Numerical Features:** Standardization was applied to numerical columns in both the training and testing sets using the previously fit scaler. This step is crucial for algorithms like Random Forest that are sensitive to the scale of features, ensuring fair treatment of each feature during model training.

5. **Model Selection and Training (RandomForestClassifier):**

    - **Random Forest Classifier:** Chosen for its robustness and ability to handle complex relationships within the data, the Random Forest algorithm constructs multiple decision trees and combines their outputs for enhanced accuracy and generalization. The algorithm operates on the principles of ensemble learning, making it resilient to overfitting.

- **Parameter Settings:** Default parameters were used for simplicity in this example, but in a real-world scenario, parameter tuning (e.g., using grid search) could be employed to optimize the model's performance. The RandomForestClassifier offers various hyperparameters, such as the number of trees and tree depth, allowing customization based on the dataset characteristics.

6. **Prediction and Evaluation:**

- **Model Prediction (model.predict):** The trained Random Forest model was used to generate predictions (y_pred) on the test set.
- **Model Evaluation (accuracy_score, classification_report):** Standard classification metrics, including accuracy, precision, recall, and F1 score, were utilized to comprehensively evaluate the model's performance. The classification_report provides a detailed breakdown of these metrics for each class.
-

The results from this model were as it follows:

```
Accuracy: 0.726198645463
Classification Report:
             precision    recall  f1-score   support

          0       0.79      0.90      0.84     26796
          1       0.03      0.01      0.02      6721

avg / total       0.63      0.73      0.68     33517
```

This underperformance, especially in the positive cases leaded me to play around a little and try to get a better prediction model by adding some new parameters.

4.1 Results summary:

First Iteration:
Accuracy: 0.726

Observations:
The model showcased effectiveness in predicting instances where the offer was not completed (class 0).
However, its performance in predicting completed offers (class 1) was suboptimal, reflected by low precision, recall, and F1-score.

Second Iteration:
Accuracy: 0.706

Changes Made:
Two extra parameters were introduced (duration and offer_received).

Observations:
The incorporation of additional parameters contributed to a slight enhancement in the model's ability to predict completed offers.
Despite this improvement, challenges persisted in providing satisfactory results for completed offers.

Third Iteration:
Accuracy: 0.664

Changes Made:
The parameter 'offer received' was replaced with 'offer viewed.'

Observations:
The modification negatively affected the model's predictive performance, particularly for completed offers.

The best overall result yielded from the second iteration:

```
Accuracy: 0.705612077453
Classification Report:
              precision    recall  f1-score   support

          0       0.81      0.83      0.82     26796
          1       0.25      0.23      0.24      6721

avg / total       0.70      0.71      0.70     33517
```

However, it still is not quite the expected accuracy, especially in the positive cases.

5 Conclusions.

In the pursuit of developing a machine learning model to predict whether an offer is completed, we undertook three iterations, focusing on refining the predictive capabilities, particularly for the positive class.

The model consistently excelled in identifying instances where the offer was not completed (class 0) but encountered difficulties in accurately predicting completed offers (class 1). In summary, while our model demonstrated satisfactory accuracy, improvements are essential, especially in predicting completed offers, maybe a broader and more balanced dataset we could achieve a better result prognosis.

5.1 Reflections and Improvements:

The project personally was a great challenge to put to test all of the skills and knowledge acquired through the course.
The biggest challenge was probably deciding what route to take or what results we are looking forward to obtain, mostly because there are chances where due to the nature of the dataset we may not obtain the expected results despite a good work of analysis/modelling.
Still, this kind of problems are the real challenges one faces in the daily work, so it felt like a great practice for “real world” tasks.

Further iterations involving careful feature engineering, hyperparameter tuning, and potentially exploring alternative algorithms could be also a deciding factor for better results.
Using Datasets with broader information and maybe for different product categories could also aid into obtaining results more stable in the predictions.

Still, as I previously mentioned there are chances that predicting accurately whether a customer may use an offer or not based only on its demographics could be out of scope as a reasonable result.

Thanks for reading this far!