Starbucks Offer Recommendation

Table of Contents

1. Overview

We want to identify which demographic group is more likely to respond to a particular offer based on their demographics characteristics.

However, what if we want to do this for new customers? We would obtain a powerful tool to evaluate all new customers and provide them with the offer that best suits their needs right from the start.

3 models are used: best channel to contact, spend model and time model to complete the offer.
The application is displayed in Gradio where the interface is quite practical: you can choose the client's gender, as well as age and income.
By sending the data, the application provides the most suitable offer from the portfolio according to your profile.

2. Input Data

We have information about the customer profile, offer portfolio, and transactions, all of them in json format:

- 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

3. Strategy for solving the problem

The project aims to estimate the best recommendation for a new customer, using only gender, age, and annual income as input data. Based on these three variables, three different models are generated to help us choose the best offer:

- Best channel to contact model; multi-output classifier model
- Spending model; regression model
- Time to complete the offer model; regression model

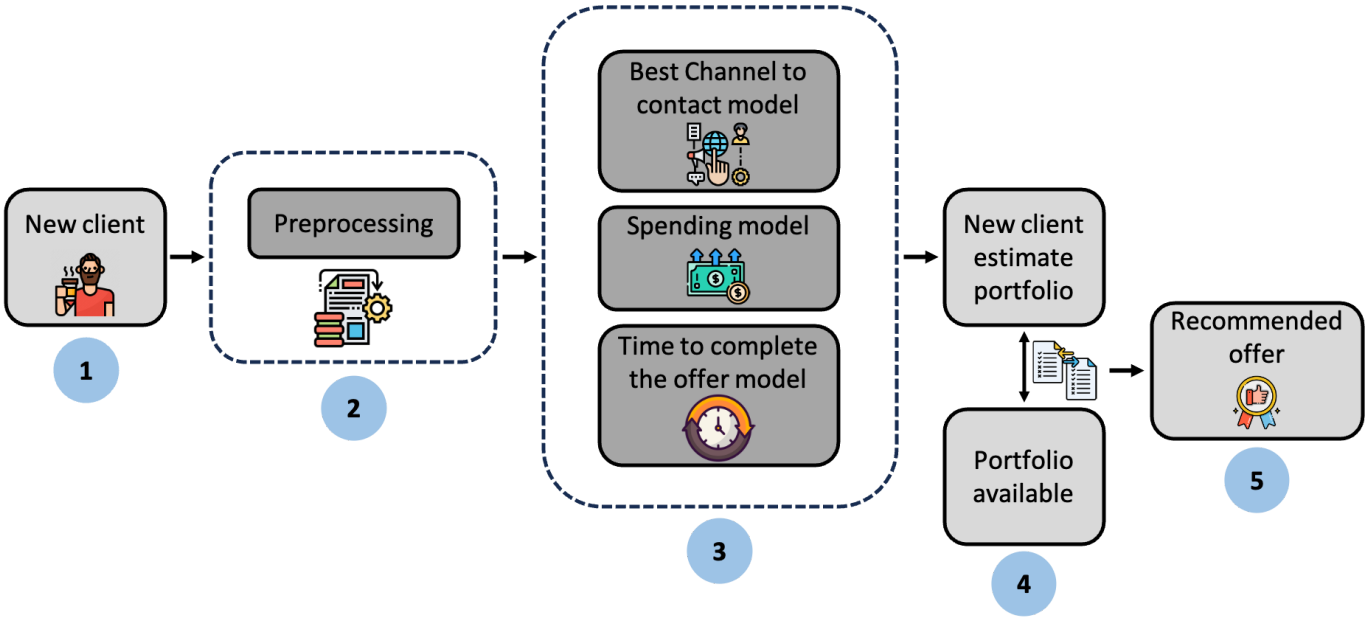
By obtaining these three results for a new customer, we can approximate the closest offer that aligns with their needs.

4. Discussion of the expected solution

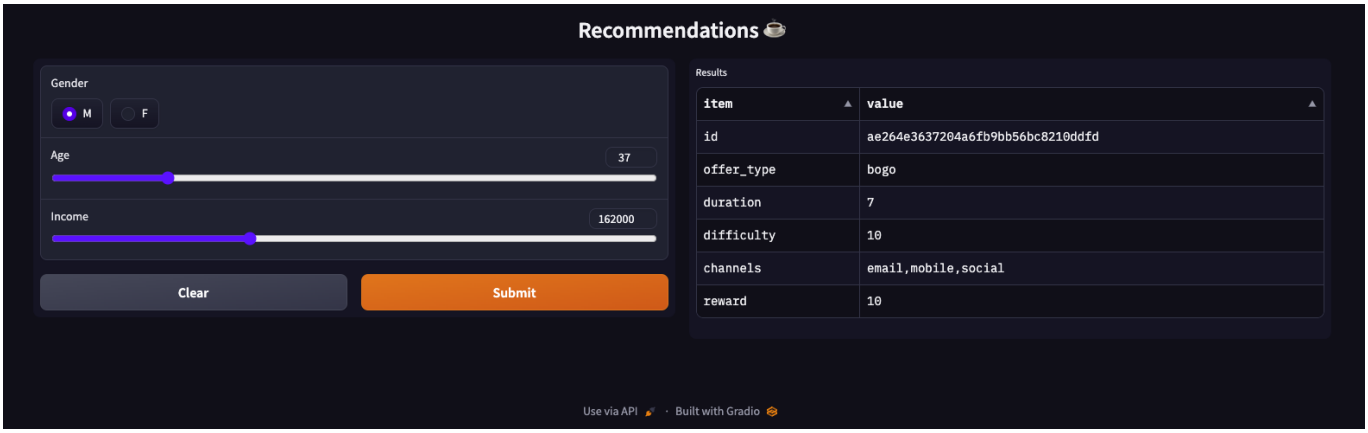
Below is the diagram of the solution that we will detail:

1. New customer information enters the flow
2. Customer information is pre-processed
3. The preprocessed information is entered into each of the models and the result is obtained for each one.

4. The estimated channel, expense and time are compared with the entire available portfolio based on Euclidean distance to identify which offer is closest to the estimate
5. The recommended offer is returned



General view of project:



5. Metrics

Since these are three different models, the metrics we will use will also be different:

- Best channel to contact model: as a multi-output classifier model, for each target (in this case, contact channels), we will obtain the confusion matrix, as well as precision, recall, and F1 score for an overall assessment of the model's quality.
- Spending model: as a regression model, we will use mean squared error and square root as goodness-of-fit measures for the model.
- Time to complete the offer model: similarly, as a regression model, we will use mean squared error and square root as goodness-of-fit measures for the model.

6. EDA

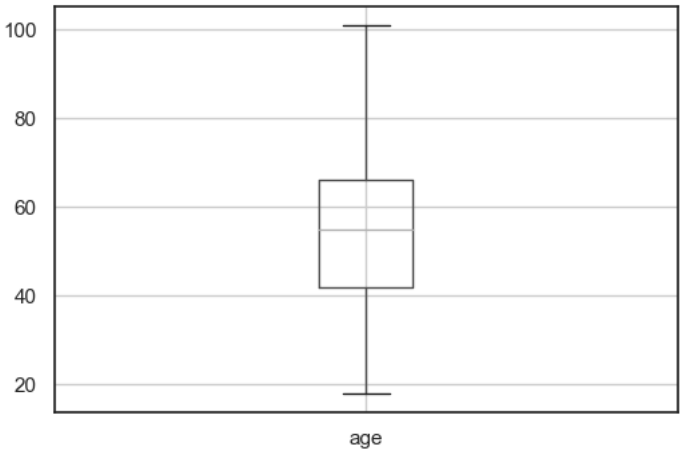
6.1. Profile

- We create a new variable called 'seniority' to see the distribution of customer seniority, however since we will focus in new customers, we can ignore that variable. Income variable has values within what is

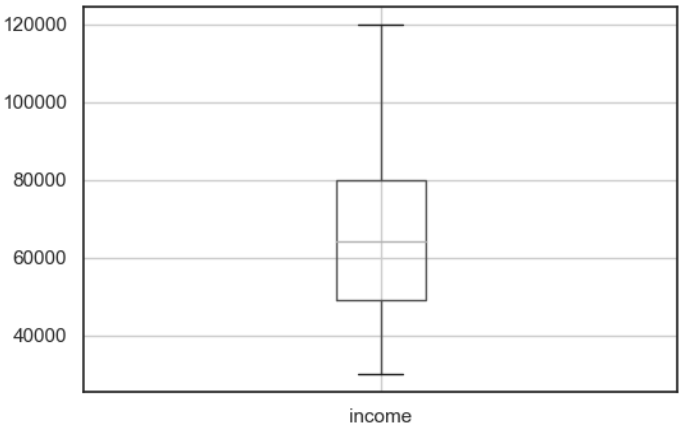
allowed. Male gender is the most popular in the population.

	age	income	seniority
count	14,825.00	14,825.00	17,000.00
mean	54.39	65,404.99	517.45
std	17.38	21,598.30	411.22
min	18.00	30,000.00	0.00
25%	42.00	49,000.00	208.00
50%	55.00	64,000.00	358.00
75%	66.00	80,000.00	791.00
max	101.00	120,000.00	1,823.00

- For the age variable we see the distribution is almost normal.



- For the income variable we see the 50% of clients have an income of 64,000 or less.



6.2. Portfolio

- The maxim reward for completing and offer is 10 and the minimun is 0. The average spend to complete and offer is 7.7, is higher than the average reward (4.2). The average duration to complete an offer is 1 week.

	reward	difficulty	duration
count	10.000000	10.000000	10.000000
mean	4.200000	7.700000	6.500000
std	3.583915	5.831905	2.321398
min	0.000000	0.000000	3.000000
25%	2.000000	5.000000	5.000000
50%	4.000000	8.500000	7.000000
75%	5.000000	10.000000	7.000000
max	10.000000	20.000000	10.000000

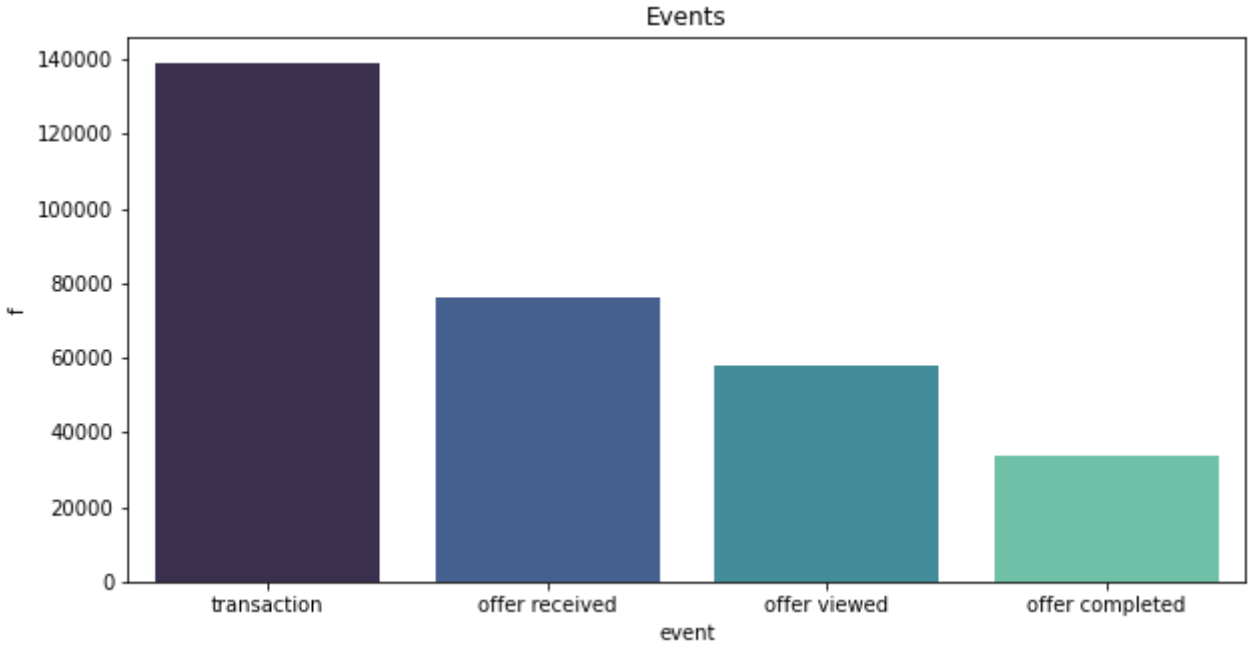
- There is four types of bogo offers, where the average reward is 7.5, with the same difficult and average duration of six days. All of this have the email and mobile channel, and 75% for web and social media.
  There is four types of discount offers, where the average reward is 3, with the difficult of 11.75 and average duration of nine days. All of this have the email and web channel, 75% for mobile and 50% for social.
  There is two types of informational offers, difficulty and reward are zero, and average duration of 4 days. All of this have the email and mobile channel, and 50% for web and social media.

offer_type	id	reward	difficulty	duration	flag_email	flag_mobile	flag_web	flag_social
bogo	4	7.5	7.50	6.0	1	1.00	0.75	0.75
discount	4	3.0	11.75	8.5	1	0.75	1.00	0.50
informational	2	0.0	0.00	3.5	1	1.00	0.50	0.50

6.3. Transcript

- 45% of events are transactions, and only 11% of events complete the offer.
  If we see unique person in each event, more of the 95% of persons have completed transactions, offer received of viewed, but only 75% of 17,000 persons have completed the offer.

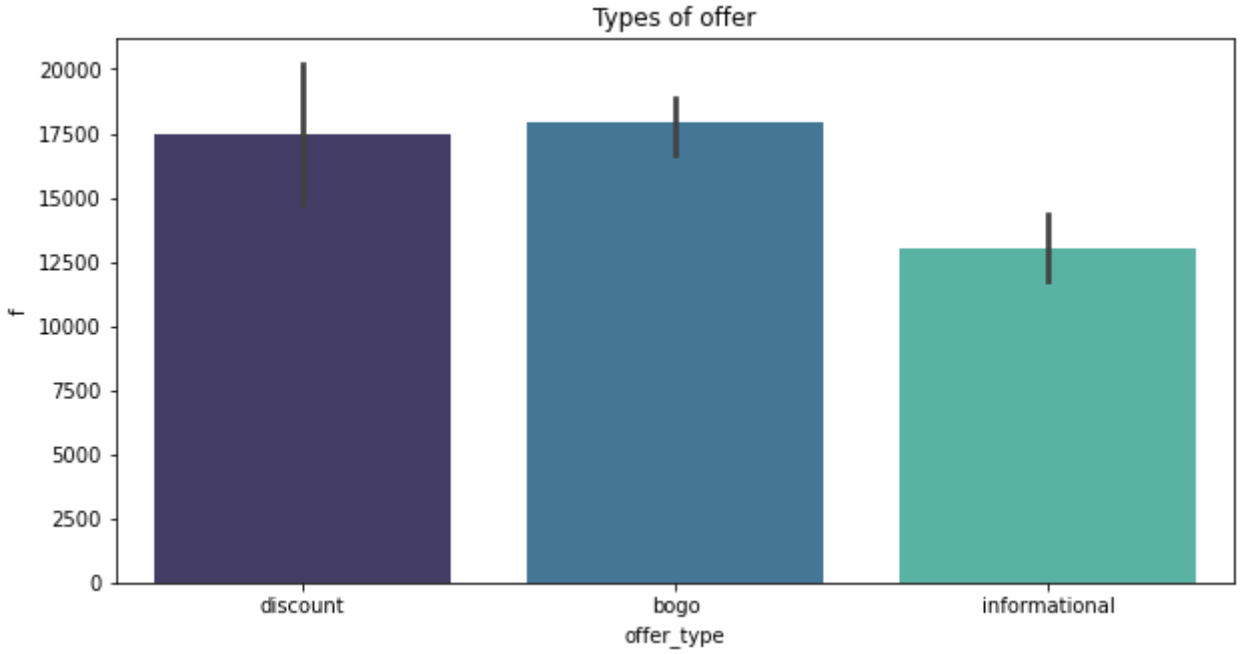
	person			
	count	<lambda_0>	nunique	<lambda_1>
event				
transaction	138953	45.33	16578	97.52
offer received	76277	24.88	16994	99.96
offer viewed	57725	18.83	16834	99.02
offer completed	33579	10.95	12774	75.14



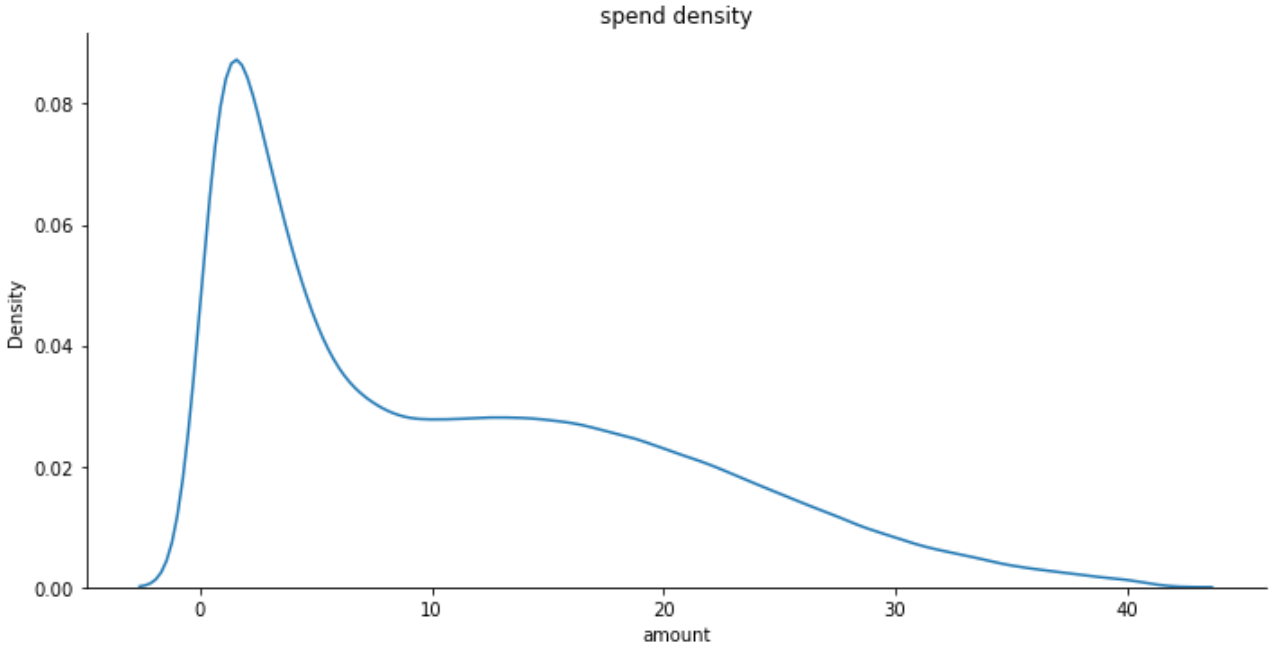
- The average amount for transaction is 13, and the average reward of all the transactions is 5. The time average in days since start of test to complete the offer is 17 days.

	amount		reward		time	
	mean	median	mean	median	<lambda_0>	<lambda_1>
event						
offer completed	NaN	NaN	4.904137	5.0	16.71	18.00
offer received	NaN	NaN	NaN	NaN	13.86	17.00
offer viewed	NaN	NaN	NaN	NaN	14.76	17.00
transaction	12.777356	8.89	NaN	NaN	15.90	16.75

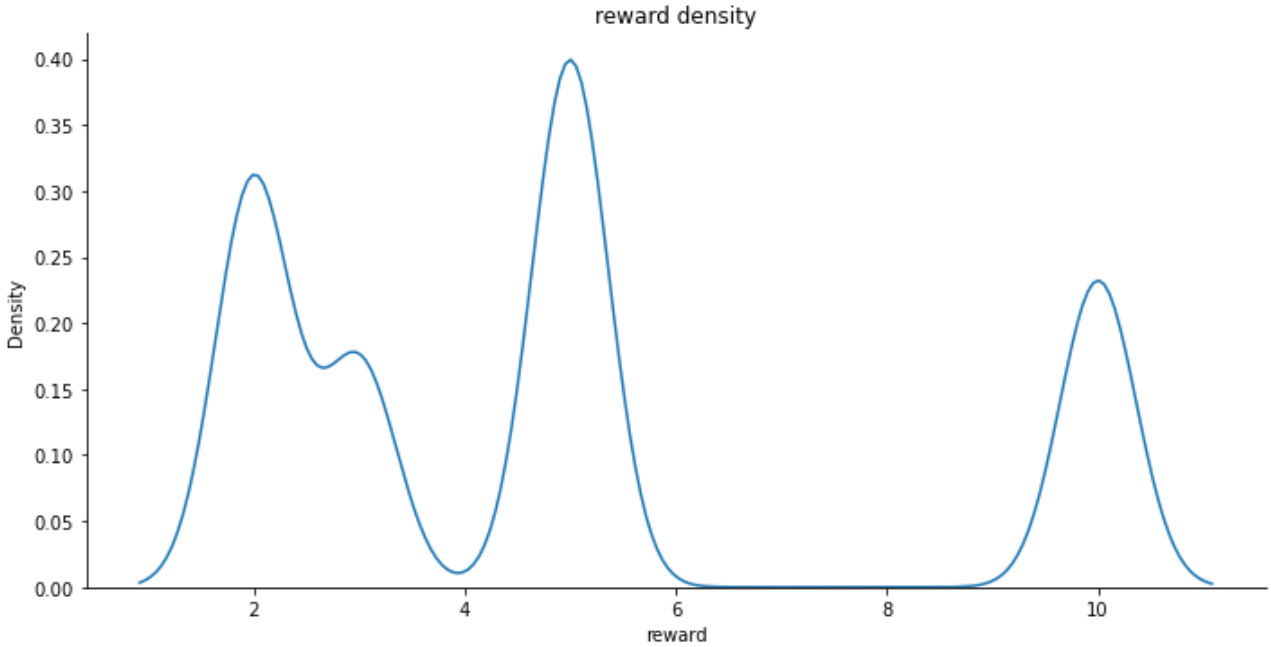
- The most popular offers are discount and bogo.



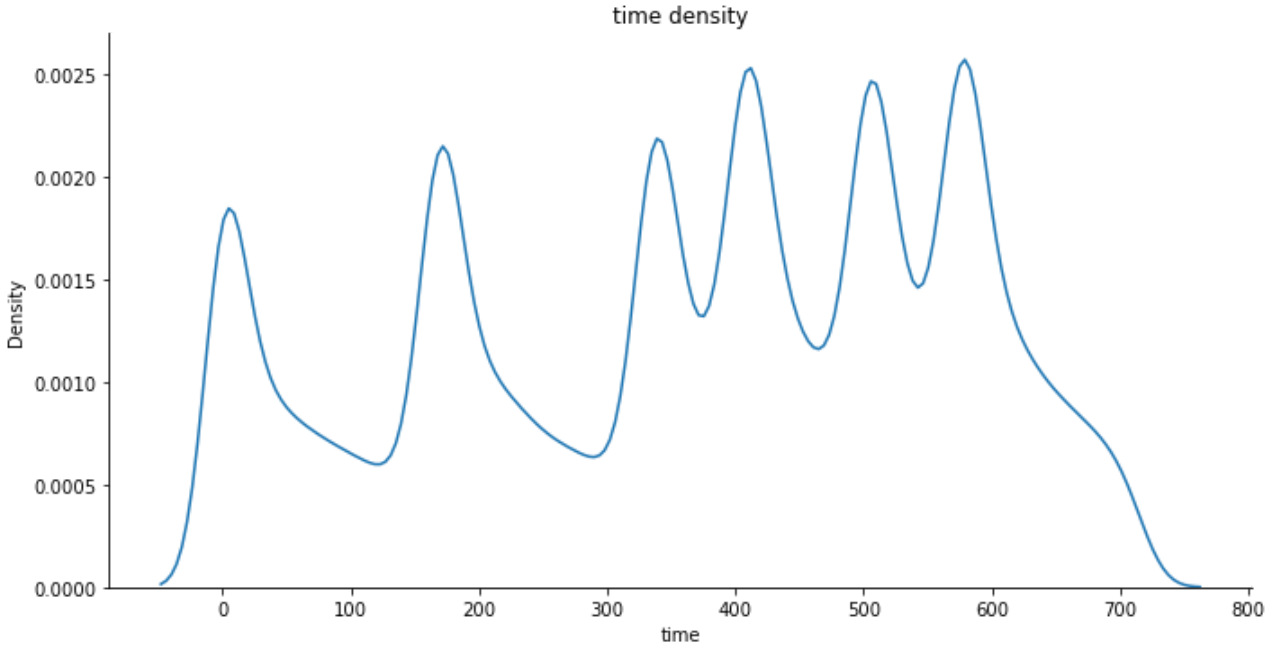
- The average spend is between 0 and 10.



- The rewards more populars are around 2, 5 and 10.



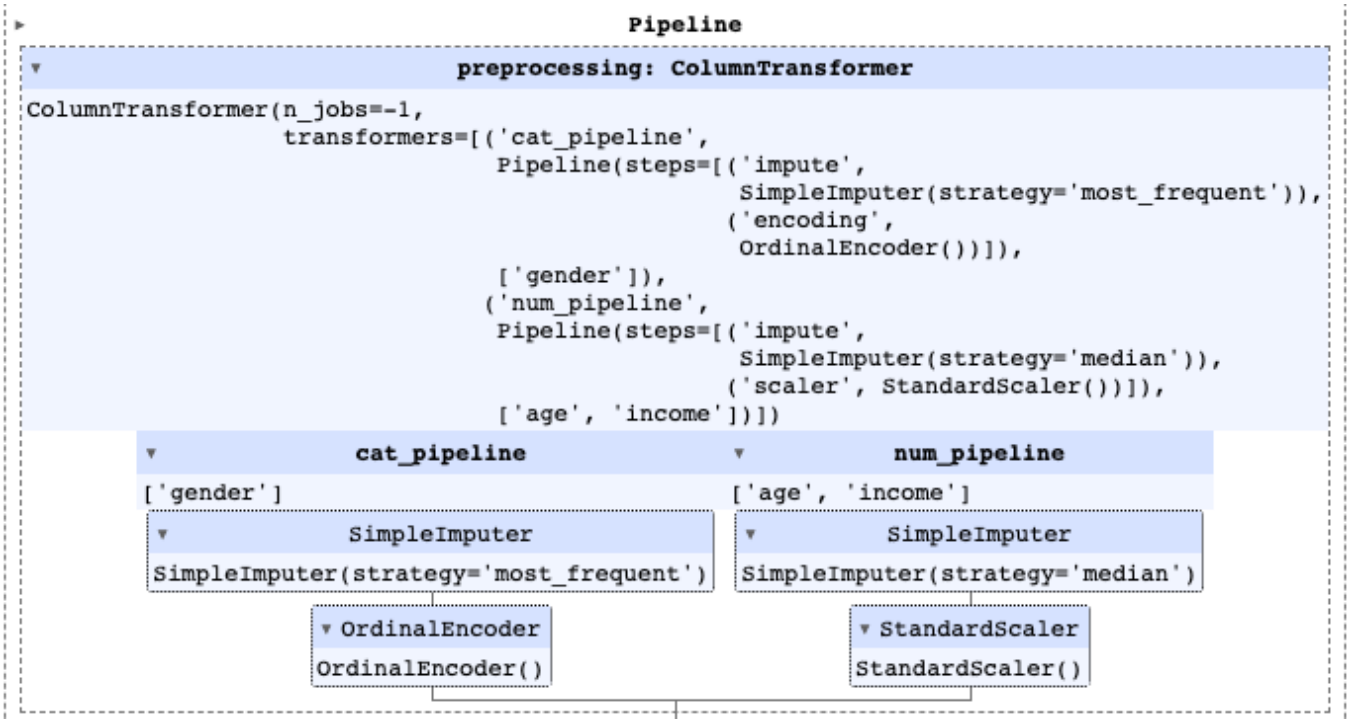
- The most popular times are between 300 and 600 hours.



7. Data Preprocessing

We have 3 input variables: age, gender and income. We will classify them into numerical and categorical, for each of them we will opt for a different strategy:

- numeric features; we will use the median to impute null values, avoiding being affected by extreme values. In addition, we will do a standard normal transformation since the prior distribution of the data suggests us to have this distribution.
- categorical features; we'll use the mode to impute null values and encode it so it can be fed into the model.



8. Modeling

8.1. Best Channel to contact model

We have created a target for each channel for model training. We do not consider the email channel since all offers are sent through this.
We will use a random forest classifier as multioutput classifier since we have 3 targets.

```
X = df1[['gender', 'age', 'income']]
Y = df1[['flag_mobile','flag_web','flag_social']]
targets = Y.columns
X_train, X_test, Y_train, Y_test = train_test_split(X, Y, test_size = 0.2,
random_state = 42)

model = Pipeline(steps = [
    ('preprocessing',column_transformer),
    ('model', MultiOutputClassifier(RandomForestClassifier(n_jobs = -1)))
])
```

8.2. Spending model

We focus only on transactions to estimate spending for each profile

```
df1 = df[df['event']=='transaction']
X = df1[['gender', 'age', 'income']]
y = df1['amount']

X_train, X_test, y_train, y_test = train_test_split(X, y, test_size = 0.2,
random_state = 42)

model = Pipeline(steps = [
    ('preprocessing',column_transformer),
    ('model', RandomForestRegressor(random_state = 0))
])
```

8.3. Time to complete the offer model

We need to define the universe of customers who saw a certain offer, and after a certain time completed it. This is to find out which customer profile takes more or less time to complete an offer. Let's take this client as an example.

```
pr = df[(df['id_customer']=='ffff82501cea40309d5fdd7edcca4a07')&
        (df['id_offer']=='2906b810c7d4411798c6938adc9daaa5')]
[['id_offer','id_customer','event','time']]

pr = pr[pr['event'].isin(['offer viewed','offer completed'])]
```

	id_offer	id_customer	event	time
133074	2906b810c7d4411798c6938adc9daaa5	ffff82501cea40309d5fdd7edcca4a07	offer viewed	354
143788	2906b810c7d4411798c6938adc9daaa5	ffff82501cea40309d5fdd7edcca4a07	offer completed	384
168022	2906b810c7d4411798c6938adc9daaa5	ffff82501cea40309d5fdd7edcca4a07	offer viewed	414
168024	2906b810c7d4411798c6938adc9daaa5	ffff82501cea40309d5fdd7edcca4a07	offer completed	414
258362	2906b810c7d4411798c6938adc9daaa5	ffff82501cea40309d5fdd7edcca4a07	offer completed	576
262475	2906b810c7d4411798c6938adc9daaa5	ffff82501cea40309d5fdd7edcca4a07	offer viewed	582

We noticed that the client took 30 hours to complete the offer the first time (time diff), then 0 hours, and finally completed the offer but then saw it, the latter should not count.

```
pr['time_diff'] = pr.groupby(['id_customer','id_offer'])['time'].diff()
pr['flag'] = pr['event'].map({'offer viewed':1, 'offer completed':-1})
pr['flag_diff'] = pr['flag'].cumsum()
```

	id_offer	id_customer	event	time	time_diff	flag	flag_diff
133074	2906b810c7d4411798c6938adc9daaa5	ffff82501cea40309d5fdd7edcca4a07	offer viewed	354	NaN	1	1
143788	2906b810c7d4411798c6938adc9daaa5	ffff82501cea40309d5fdd7edcca4a07	offer completed	384	30.0	-1	0
168022	2906b810c7d4411798c6938adc9daaa5	ffff82501cea40309d5fdd7edcca4a07	offer viewed	414	30.0	1	1
168024	2906b810c7d4411798c6938adc9daaa5	ffff82501cea40309d5fdd7edcca4a07	offer completed	414	0.0	-1	0
258362	2906b810c7d4411798c6938adc9daaa5	ffff82501cea40309d5fdd7edcca4a07	offer completed	576	162.0	-1	-1
262475	2906b810c7d4411798c6938adc9daaa5	ffff82501cea40309d5fdd7edcca4a07	offer viewed	582	6.0	1	0

We create the model:

```
X = df1[['gender', 'age', 'income']]
y = df1['time_diff'] / 24 # days

X_train, X_test, y_train, y_test = train_test_split(X, y, test_size = 0.2,
random_state = 42)
pipe = Pipeline(steps = [
    ('preprocessing',column_transformer),
    ('model', RandomForestRegressor(random_state = 0))
])
```

9. Hyperparameter Tuning

9.1. Best Channel to contact model

In the case of this model, we will not apply hyperparameter tuning because it is a multioutput model and requires many resources. However, this model, as we will see in the results section, does not require it since it has very good results.

9.2. Spending model

We have chosen the 3 main hyperparameters that control overfitting and model performance:

- n_estimators: number of trees to create. The higher the number of trees, the more robust and accurate the model will be.
- max_depth: depth of each tree. Control model complexity and avoid overfitting.
- min_samples_split: minimum sample to split an internal node in a tree. Helps prevent splits that can be too specific and lead to overfitting.

In this model we have 2 main approaches because the target has outliers:

- Estimate the cost of the new client

```
parameters = {
    'model__n_estimators': [100, 200, 300],
    'model__max_depth': [3, 5],
    'model__min_samples_split': [50, 100],
}

pipe = Pipeline(steps = [
    ('preprocessing',column_transformer),
    ('model', RandomForestRegressor(random_state=0))
])

model = GridSearchCV(pipe, parameters, cv = 5, n_jobs = -1)
model.fit(X_train, y_train)
best_model = model.best_estimator_
```

- Estimate the interval where the spending of the new customer can be found

```
cuts = np.percentile(df1['amount'].values, q = np.arange(5, 100, 5))
df1['amount'] = np.digitize(df1['amount'], bins = cuts, ) + 1

X = df1[['gender', 'age', 'income']]
y = df1['amount']

X_train, X_test, y_train, y_test = train_test_split(X, y, test_size = 0.2,
random_state = 42)

parameters = {
    'model__n_estimators': [100, 200, 300],
    'model__max_depth': [3, 5],
    'model__min_samples_split': [50, 100],
}

pipe = Pipeline(steps = [
    ('preprocessing',column_transformer),
    ('model', RandomForestRegressor(random_state=0))
])

model = GridSearchCV(pipe, parameters, cv = 5, n_jobs = -1)
model.fit(X_train, y_train)
best_model = model.best_estimator_
```

It will depend on the performance of each model according to the target mentioned above.

9.3. Time to complete the offer model

We tuned the model for estimating the time to complete the offer.

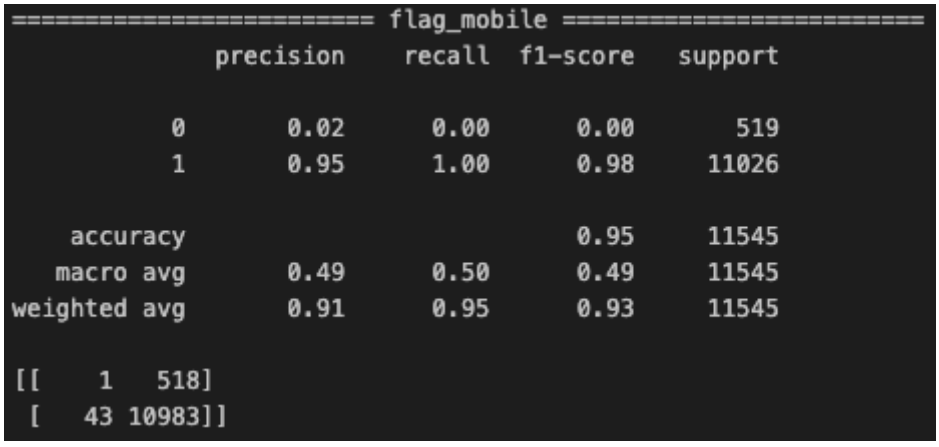
```
parameters = {
    'model__n_estimators': [100, 200, 300],
    'model__max_depth': [3, 5],
    'model__min_samples_split': [50, 100],
}

pipe = Pipeline(steps = [
    ('preprocessing',column_transformer),
    ('model', RandomForestRegressor(random_state=0))
])

model = GridSearchCV(pipe, parameters, cv = 5, n_jobs = -1)
model.fit(X_train, y_train)
best_model = model.best_estimator_
```

10. Results

10.1. Best Channel to contact model

Since we have 3 targets, we will get a report for each one:

- Mobile target, the accuracy of the model is 0.95 and the f1 score for the true class is 0.98 (out of 100 customers who used this channel, 98 were recognized by the model)



```
========================= flag_mobile =========================
              precision    recall  f1-score   support

           0       0.02      0.00      0.00       519
           1       0.95      1.00      0.98     11026

    accuracy                           0.95     11545
   macro avg       0.49      0.50      0.49     11545
weighted avg       0.91      0.95      0.93     11545

[[    1   518]
 [   43 10983]]
```

- Web target, the accuracy of the model is 0.73 and the f1 score for the true class is 0.84 (out of 100 customers who used this channel, 84 were recognized by the model)

```
========================= flag_web =========================
              precision    recall  f1-score   support

           0       0.23      0.08      0.12      2616
           1       0.77      0.92      0.84      8929

    accuracy                           0.73     11545
   macro avg       0.50      0.50      0.48     11545
weighted avg       0.65      0.73      0.68     11545

[[ 208 2408]
 [ 711 8218]]
```

- Social target, the accuracy of the model is 0.71 and the f1 score for the true class is 0.81 (out of 100 customers who used this channel, 81 were recognized by the model)

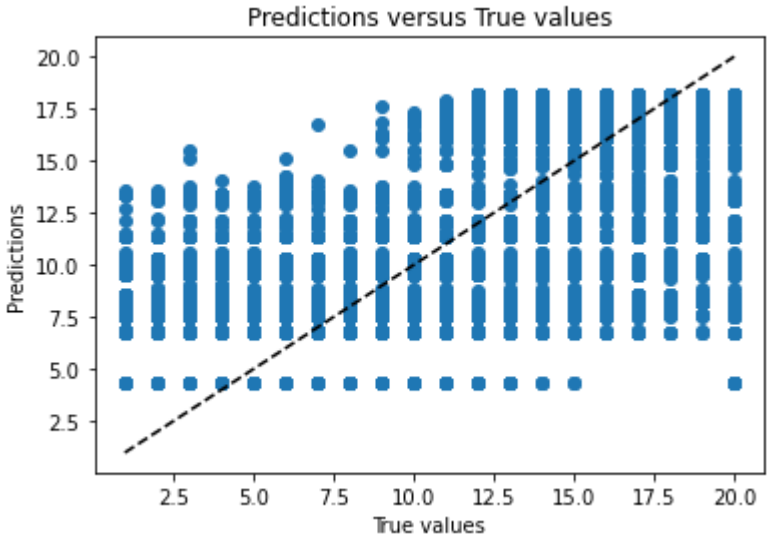
```
======================== flag_social ========================
              precision    recall  f1-score   support

           0       0.32      0.12      0.17      3088
           1       0.74      0.91      0.81      8457

    accuracy                           0.70     11545
   macro avg       0.53      0.51      0.49     11545
weighted avg       0.63      0.70      0.64     11545

[[ 365 2723]
 [ 786 7671]]
```

10.2. Spending model

We report 2 regression indicators:

- Mean Squared Error (MSE): 15.70
- Root Mean Squared Error (RMSE): 3.96

This means that we can estimate with precision of +/- 3.96 intervals of the true value.

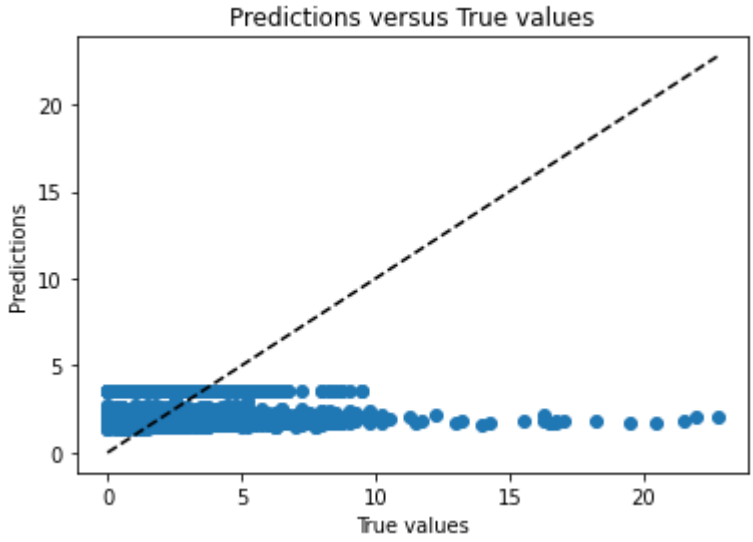


10.3. Time to complete the offer model

We report 2 regression indicators:

- Mean Squared Error (MSE): 4.42
- Root Mean Squared Error (RMSE): 2.10

This means that we can estimate with precision of +/- 2 days of the true value.

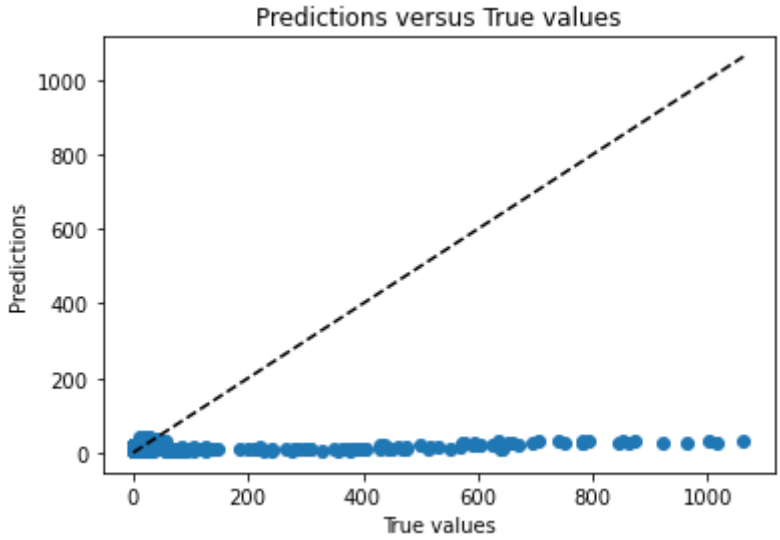


11. Comparision table

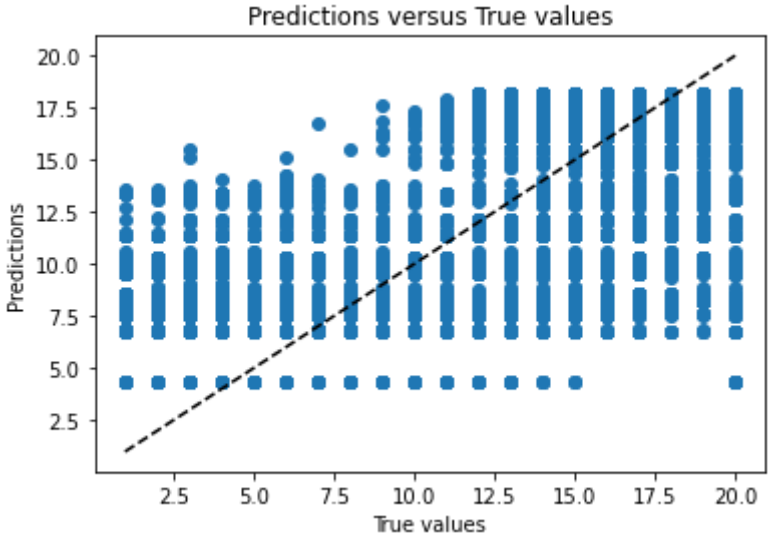
11.1. Spending model

As we mentioned in the previous sections, we have 2 approaches to this model:

- Estimate real spending; the error is very large
  Mean Squared Error (MSE): 879.85
  Root Mean Squared Error (RMSE): 29.66



- Estimate the range of actual spending; the error was reduced
  Mean Squared Error (MSE): 15.70
  Root Mean Squared Error (RMSE): 3.96



12. Conclusion

The problem began by knowing how the demographic group can influence when offering an offer.
On my side I raised the bet and asked myself what would happen if he is a new client? I found this question a bit more interesting.
Being a new client, it is reasonable that the estimates are not the best, but they give us a north of what to offer to these clients.

13. Improvements

It is possible to improve this project in 2 ways:

- We have only used one type of algorithm for each model. Perhaps carrying out a benchmark with more models will give us a greater number of alternatives
- Try more hyperparameters. For hardware reasons, I only tried with a small set of hyperparameters.

14. Acknowledgment

I want to recognize and recommend this great post from Freecodecamp that helped me a lot to improve and clean my code.