# Delivery of Starbucks offers

Starbucks is a large network of cafes, present all over the world. Its sales reach more than 4 billion dollars in 2021.

The network has loyal customers, who consume regularly and guarantee part of the company's success.

However, the coffee shop network is recognized for its good marketing, attracting and maintaining new customers every day.

One of the main strategies for this are promotional offers. Generating a feeling of compensation for the end customer, the offerors usually draw attention and increase the revenue of the stores.

The data analyzed here come from a simulation made pela own Starbucks of the consumption pattern of its customers, from the receipt of offers via web, email and social network.

Offers are of three types: buy one, take another; informational and discount. Bogo and discount offers have a different reward as well as a minimum amount to be rewarded.

The goal of a good bid submission campaign is to increase the total amount transed by consumers.

With the help of data and machine learning techniques it is possible to optimize a campaign. From users' responses to offers already submitted, you can understand which user profile responds best to which type of offer and thus target the offers in the most optimized way.

With the data available, this article shows how data has been handled, visualized, and modeled to make recommendations of which offers are best for each user profile.

# Handle the data

## The data

Three databases were provided:

Profile - dataset with user data present in the transcript database

* gender - user sex
* Age - user age
* Income - user income
* Id - hash
* Duration - the duration of the offer, in which the duration of influence of the user is considered.

Portifolio - based on information about the 10 different offers available to be offered.

Transcript - based on the events that happened in the experiment.

Time - time that was logged the event

Value - dictionary with the values of the different events.

Offer id - for offers submitted, viewed and complete

Reward – if an offer is complete, the amount earned by the user when completing that offer.

Amount - traded quantity, if it is a value transaction event.

Person - user referring to the event

Event - classification of the event that happened

PHOTO OF THE TABLES HERE

The transcript base is the main one for the analysis.

The events logged in the log are of 4 types:

Offer Received - a user received the offer

Offer Viewed - a user viewed the offer

Offer Completation - a user completed the offer, earning a reward

Transaction - when a user makes a transaction.

## **Creation of numeric ids in profile and portfolio**

User ids and offers came on the basis of hash format. To transform into a more cute format to visualize, the mapper\_id.py script was created, with the function mapper\_ids(), responsible for taking the ids in hash and creating a conversion dictionary for integer ids.

The mapper\_ids() is used for both cases of hash ids, person column in profile, and id in portfolio.

Within the script the created dictionaries are exported as jsons, so that they can be used in any other script and also maintains the repordudility of this processing of information. The files are allocated within the mapper\_id.

## Data Preparation

All data processing is in the 'Data Prepartion.py' script.

The script starts by loading the supplied bases and the dictionaries created by the script mapper\_id.py.



## Transcript treatment

The difficulty in data processing is at their disposal.

The base transcript is a time line of events. To extract what you want, the effect of an offer on the buying behavior of a user, it is necessary to extract the time information and freeze it in a summary table.

The type of table imagined was a cross of users and offers. In this table, it is available which offers each user received and how he responded (if he saw, completed and how much he traded influenced by it). From the timeline is that it is possible to evaluate whether the user made the transaction within the limte duration of each offer. Some sanctions are released without necessarily being made on account of an offer.

Some lags and their contours in the processing of data:

1. Some offers can be completed without even being seen. In this case, the transaction carried out to complete the offer is not considered. You will only consider transactions influenced by offers after the offer has been viewed. Moreover, it is considered that this completion is not valid, because the user had no idea of the existence of the offer, which is assumed that he was not influenced to complete it.
2. Some offers are received and viewed by the user, however, before a transaction, another offer is received and viewed. In this case, any transaction that comes after the second offer, will be accounted for this second, canceling the effect of the first.
3. The same offer can be delivered to a user more than once. It is possible for the user to complete this offer repeated once again in the base transcript. For this case, the data will be grouped for user and offer, considering the average of times the user viewed the offer, average of how many times he completed and sum of how much he traded influenced by that offer.
4. Some offers are displayed after their expiration date. In this case, the preview is considered not valid and any transaction performed after that view is not counted.

The trigger to save handle the bases part of the generate\_user\_offer():



The function ites over the users present in the base transcript. The function create\_user\_offer\_df() returns a dataframe. The dataframes are joined via the pandas concat function. The base is saved to a csv file to be worked on in the next scripts.

The function create\_user\_offer\_df () receives an id (hash) and creates a dataframe with the get\_offer\_table\_user(), the main function for handling the base transcript. It then applies the group\_offer\_df(), responsible for grouping the information in user versus offers format. In the end, the function creates a user\_id to identify the user in whom the information is about it.



The function get\_offer\_table\_user() has four parts. It receives user code (in hash) as an argument and makes use of the other variables defined at the beginning of the script. Here, I'll explain how each part works.

1. Separate the bases

There are 4 different events within transcript: offer received, offer viewed, offer completed and transaction.

In this first part of the code, these four events are separated into unique dataframes: received\_df, viewed\_df, complete\_df, and transaction\_df.

Each of them will have the information relating to their phases.



For the creation of the dataframes, the function of the get\_subset()



One of the facilities of this get\_subset() function is to extract the values from the dictionaries from the input argument dict\_keys.

The second part of the function is to analyze the timeline, using the 4 bases extracted in the first part.

The flow follows the following reasoning:

1. Merge between the received and viewed base. If more than one offer is delivered to customers, there is a filter that ensures that the view was performed after that offer was received and before the next offer. You create the column that tells you the maximum period in which that offer influences. The function uses the validate\_view() to validate visualizations made within the valid period. At the end, the time at which the next offer is viewed is arranged in a column. This information will work to filter transactions for upcoming offers.
2. Merge from the anteiror base with the base of completed. The goal is to know which of the offers was completed by the user. Some problems with this merge is that the same offer may have been completed more than once in time. Therefore, the strategy of drop\_duplicates. To account for valid completes, which happened after you see the offer, the completed\_after\_view is created. At final, the guarantee\_viewed() is used to ensure that all offers received by users are originally in the output of this step. This is necessary because the filter of the merge\_and\_filter can withdraw repeated offers that have been "complete" from the receive time.
3. For transactions, merge would not fucionara. Instead, the back-up table is iterate, line by line, to account for the total for each offer sent to the user. The rule for the transaction is that the transaction was made before the next offer was seen; before the maximum validity period of the offer; and before the offer has been completed.



1. In the end , two columns are created, viewed and completed, counting as 0 or 1 if the offer was viewed and whether it was completed.

The third part of the get\_offer\_table\_user() function is created two columns that are created, viewed and completed, counting as 0 or 1 if the offer was viewed and whether it was completed.



The fourth and last part of the function is selects the columns that matter to the application of the next function.



The function group\_offer\_df() receives the output of the get\_offer\_table\_user() function and grouped into the metrics that will be used in the rest of the data analysis.



For each metric, convenient aggregation was used. For example, for the display rate, the average was used, while for the transactions the sum was used. About the metrics you create:

About metrics

* viewed\_rate - the rate of views for an offer
* completed\_rate - the rate of completes for an offer
* tra\_offer\_infl - the total of transaction because of an offer
* valid\_view - for viewed offers, the rate of vizualizations in
* validy period of an offer
* completed\_after\_view\_rate - for complete offer, the rate of that
* was complete after was visualize
* reward - for complete offer, the total of reward won by user

At the end of create\_user\_offer\_df(), a column with the user ID is created and the output is a dataframe. This dataframe has all the offers received by each user and how they responded to each offer.

## Creating column with id integer



The gen erate\_datasets() function handles the profile and portfolio bases to create an extra column, with the id in type int, not hash.

Saves these files as csv for later use.

## Profile Treatment

The user base is a great source of information about each other's profile.

The base profile columns are both numeric (age, income), categorical (gender) and time (became\_member\_on).

Some users have blank information, which makes it necessary to handle before analyzing the data.

### Filling the voids

The gender, age and income columns have voids. Age is encoded as 118 for voids.

On the basis, there are 12% of empty rows, representing 2040 users.

The strategy for filling voids, in this case, was to use the median for the age and income columns. Thus, the numerical distribution of values does not change significantly.

For gender, a new class, NI (not indentify) was created. In this way, you have a trace form of these empty lines later.

For the date column, became\_member\_on, a new feature for the table was derived. With the start date as a member, you can calculate the time in years that user is a member. As the exact day of data extraction is not known, the date of the younger user within the dataset was used as the default. Thus, all calculated times are relative to that most recent date. The column rela\_member\_years saves these values.



### User clustering

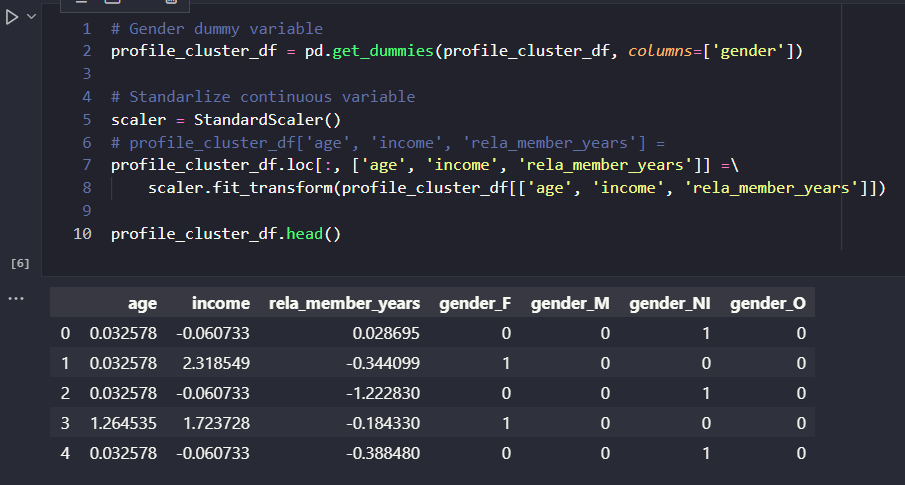
Profile has 4 columns that characterize the user: age, income, gender and time as member.

There are several possible combinations between the values of these columns.

An unsupervised method can reveal user classes that may be important at the time of modeling.

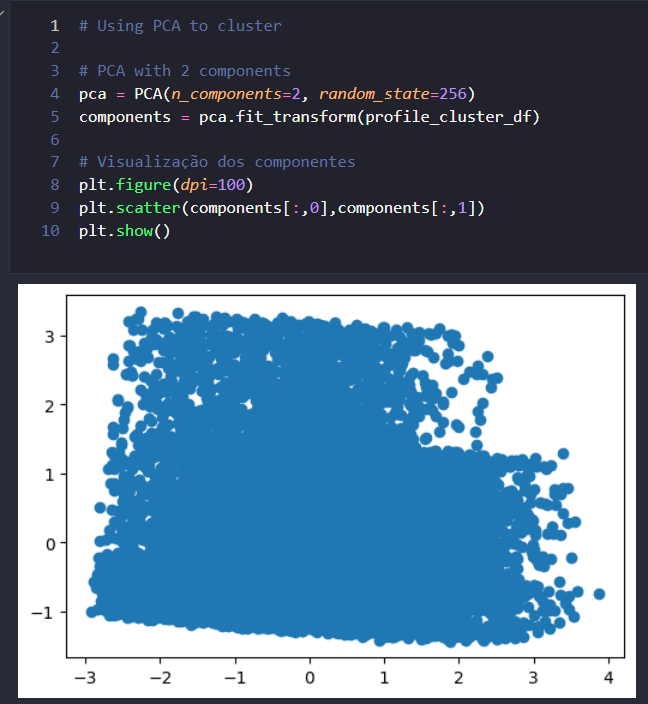
A clustering is performed to reveal these groups.

First, the gender column, a categorical variable is transformed into a dummy variable.

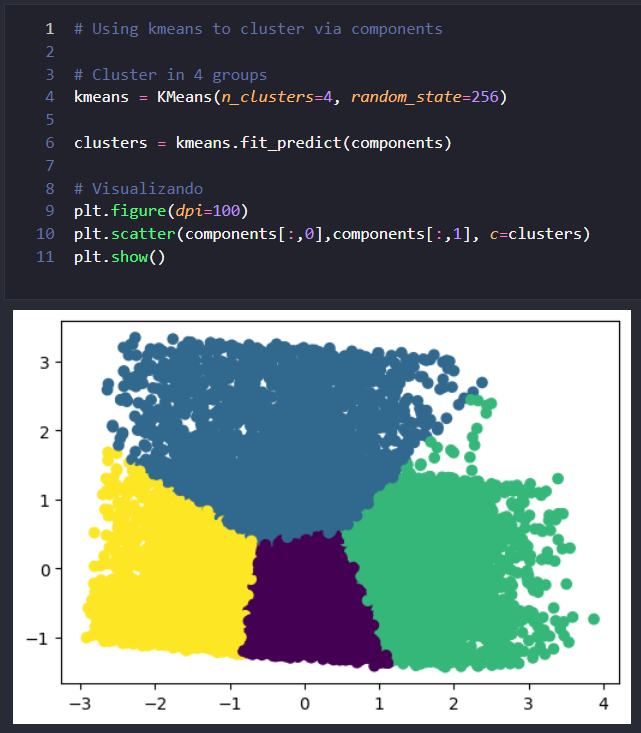


Then, continuous variables (age, income, rela\_member\_years) are standardized to remove the influence of large numbers.

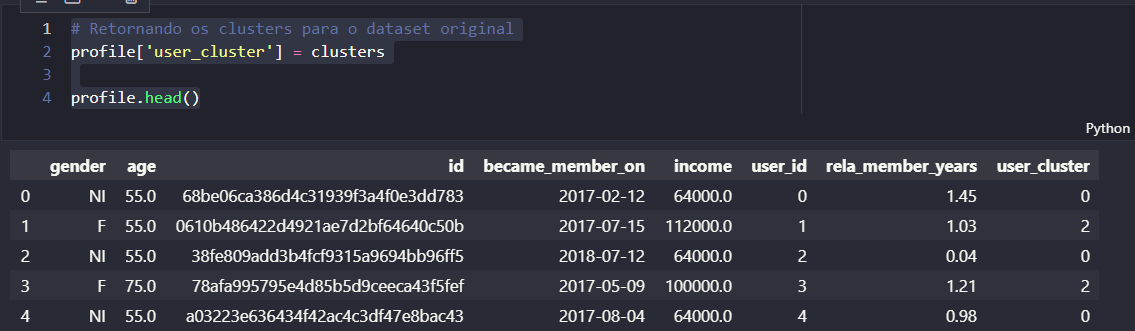
After these treatments, a dimensionality reduction is performed to reduce the 7 variables to 2 main components.



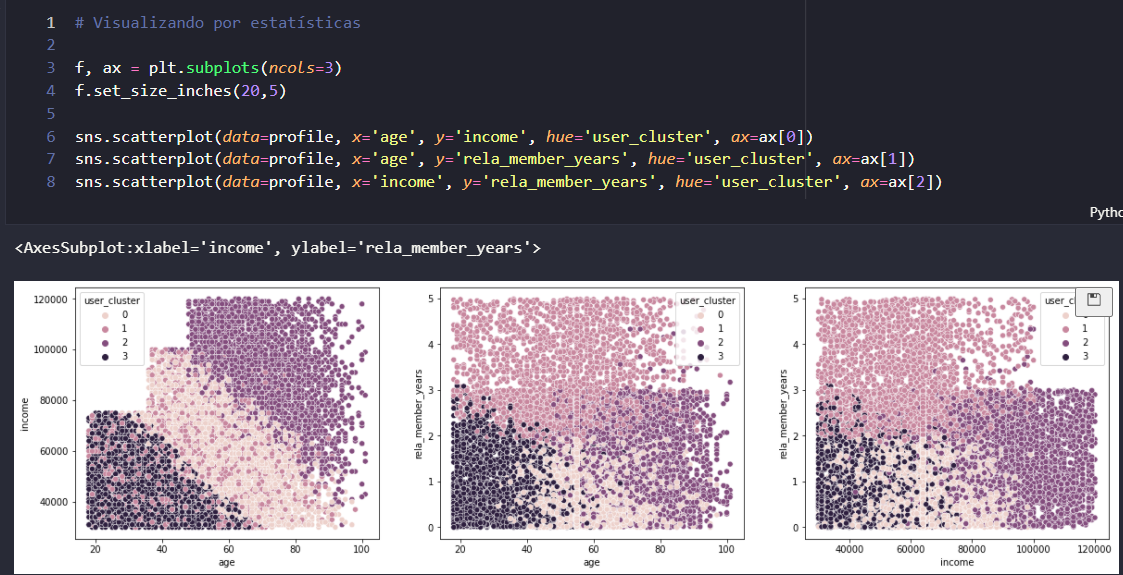
Using the k-means method, 4 groups are different within the two variables.

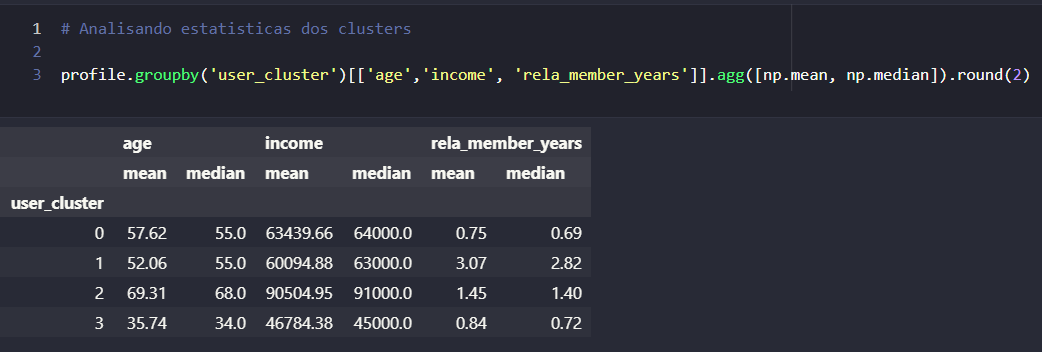


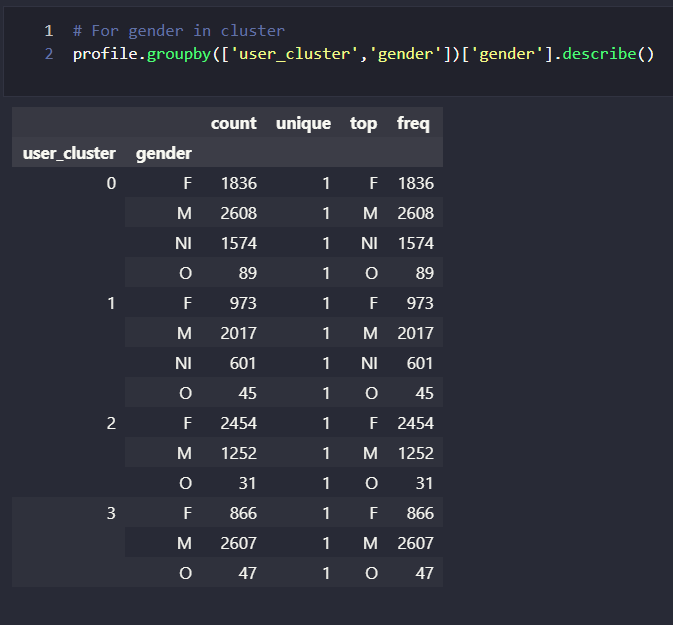
With clusters created, you return this information to the original profile base.



With this new class, we can identify user groups.







Observing the means and the arrangements of the groups in the variables, one can define the clusters found as:

* Cluster 0: Middle age, middle income and recent member
* Cluster 1: Middle age, middle income, and old limbs
* Cluster 2: High age, high income and recent member
* Cluster 3: Low age, low income, and recent membership

As for gender, no group with a dominant class stands out. Everyone has a male majority.

The base is extracted as csv to be used by other processes.

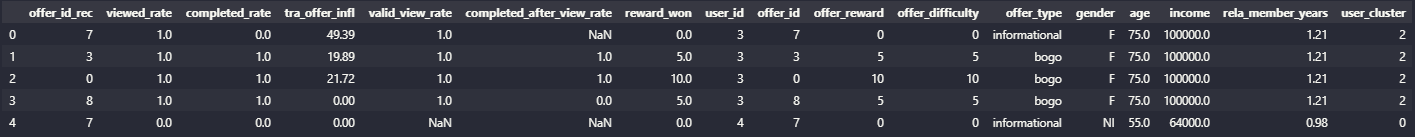
# Data Exploration

The exploration of the data allows to see distributions and trends of the information treated.

The main entry for data analysis is the user-offer table, created from the Data Preparation script.

From it, a merge is made with the profile and portfolio bases.





In this exploration phase, continuous variables are grouped into their quartiles. This grouping allows you to see general behaviors of the data.



Last treatment required is to define the metrics that will be viewed.

From the treatment performed, the three metrics that will be used will be:

- valid\_view\_rate: indicates whether the offer was displayed within the influence period (up to the maximum period). Values of 0 or empty indicate that the offer was either viewed after the valid period or was not seen.

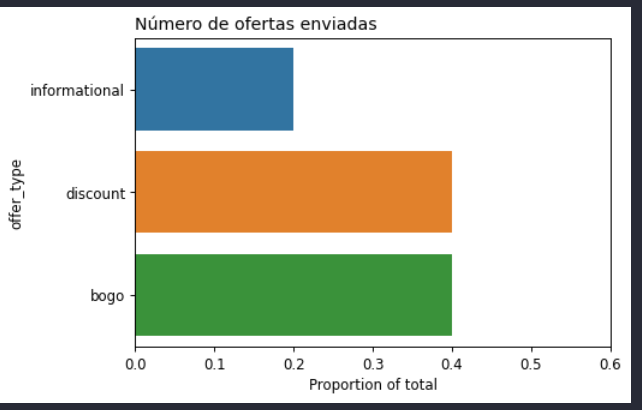
- completed\_after\_view\_rate: indicates whether the offer has been completed after viewing. Values of 0 indicate that the user was not influenced to complete the offer, since it did not come to view. Empty values indicate that the offer was not complete.

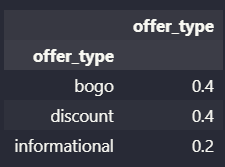
To use these two columns, you replace the NaN values with 0.



Some points about the data.

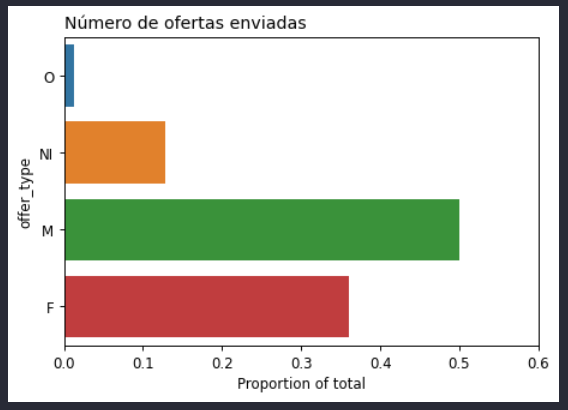
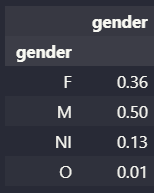
Here I will show some interesting points about the data. A more complete view can be seen on the Data Exploration.ipynb notebook.





The number of offers submitted within the base is distributed equal to the amount of offers available within the portfolio.

The same behavior is seen in the distribution of users, in which the percentages are distributed equal to the base profile.



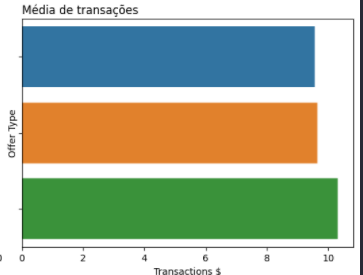
This indicates that the simulation has shown equity in the distributions of offers and available users, not prioritizing any particular group. As for the number of users, this happens because the simulation used virtually all users. In fact, only 6 users received no offer.

Regarding the type of offer, it is noted that the viewing rates are distributed close to the distribution of the number of channels that the offers are distributed.

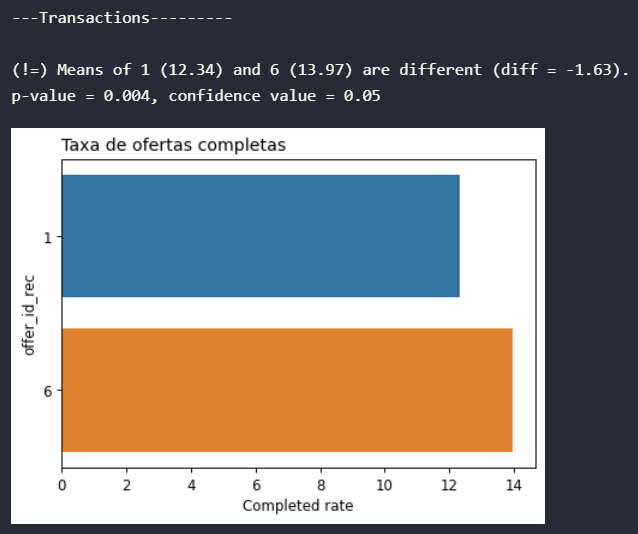


For the full rate, it should be noted that discount offers have a higher full rate than BOGO-type offers. Informational has no full fee, they are already advertising-only offers, which do not offer rewards.

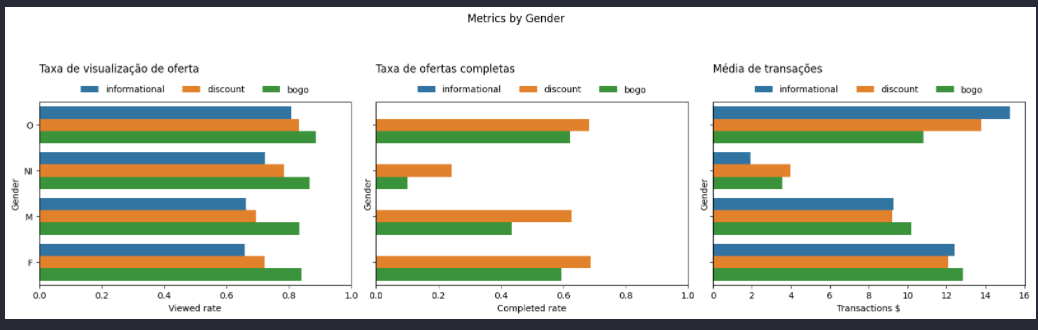
For transactions, the average transaction scan for users who have viewed BOGO-type offers is higher overall, but the offers are higher than for other companies. However, it is noted that the discount type 6 of the discount type stands out from the other in the iterate transactions.



Comparing this offer with the second place, the BOGO offer with id 1, the difference of $ 1.63 on average is statistically significant (using t-test for comparison between means)



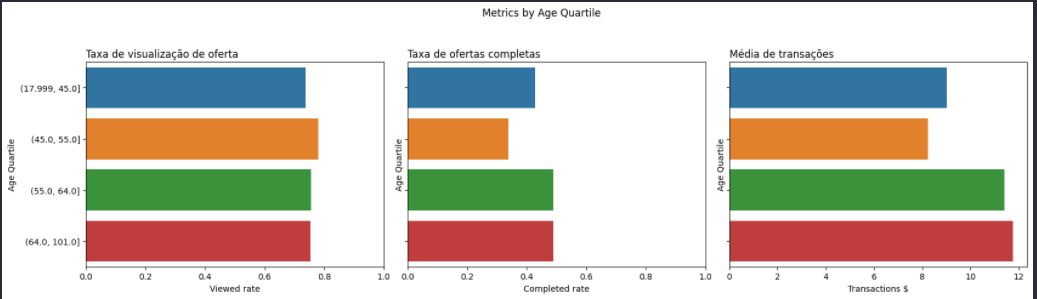
By gender.



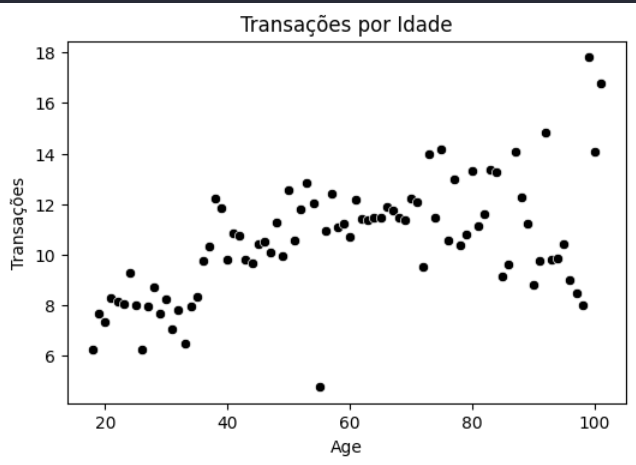
The others group have higher average transactions by the sample quantity, which is smaller. What can also be highlighted is that women have higher transaction averages than men, with higher transactions for BOGO-type offers and second to informationals.

Among the groups, there is a tendency to complete more discount offers than bogo.

By age, the numbers present a positive correlation. The older the age, the more offers are complete and the higher the average transaction.

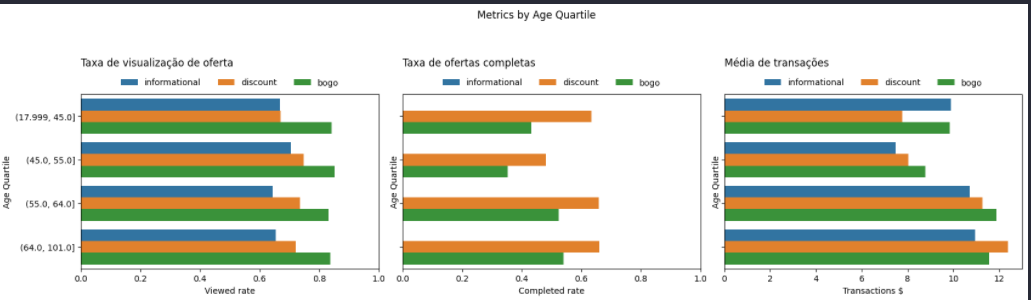


Looking at the dots other than by groups

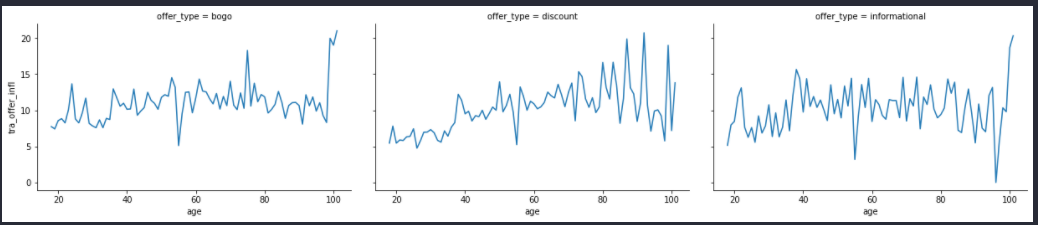


It is noted that intermediate and advanced ages have higher transactions.

Opening by offer type, for the age group from 55 to 64 years, there is a higher transaction average for BOGO, while for older age group, 64 to 101, the average is higher for discount offers.

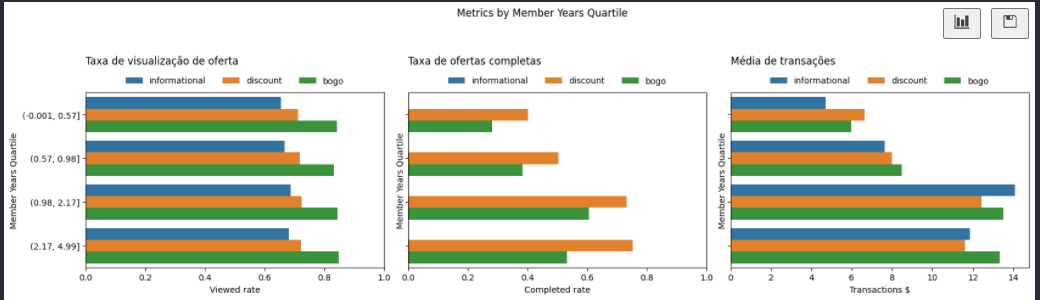


The chart below shows that transactions for discount offers (center chart) increase in the age-old avacity region.



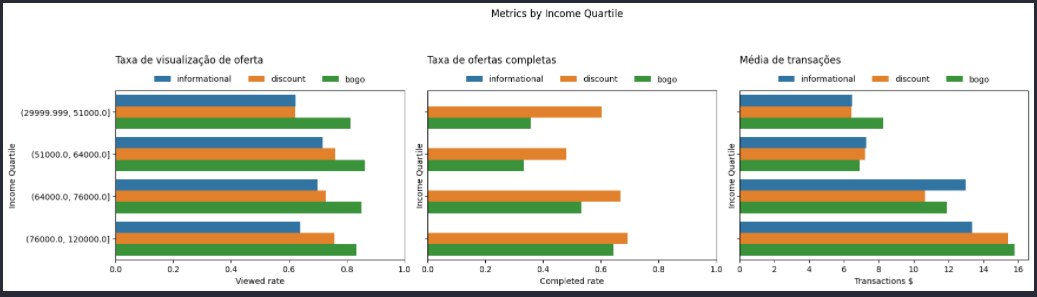
As for the full rate, it is equivalent to the two age groups and stand out for discount type offers.

Looking at the registration time column, it is noted that older members have higher averages of transactions.



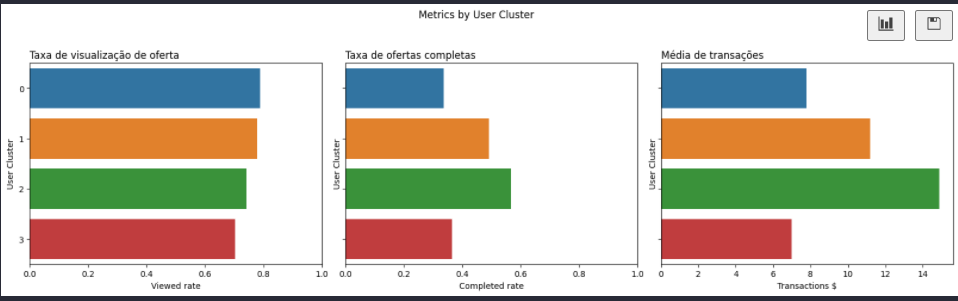
It is noted that members with a time between 1 and 2 years, make larger transactions for informational, when members of more than 2 years trade more with BOGO-type offers. The full rate follows the same standard, discounted with higher rate.

Looking at income, the obviedehappens. The higher the income, the more the average users transact. The difference between the averages between the last and penultimate group is up to $4.



Another point to comment is that group with higher income has higher average for BOGO type offers, while for the penultimate group the preference offer is informational. For the completed rate, it is noted that users with lower income (first group), have higher rates for discount offers compared to bogo type offers.

Finally, looking at clusterof previously created users, it is noted that dividing groups is expression in metrics.



Cluster 2 has positive prominence in the rate of complete and transactions, while the numbers are smaller for cluster 1 and 3. Looking again at the descriptions identified for each cluster:

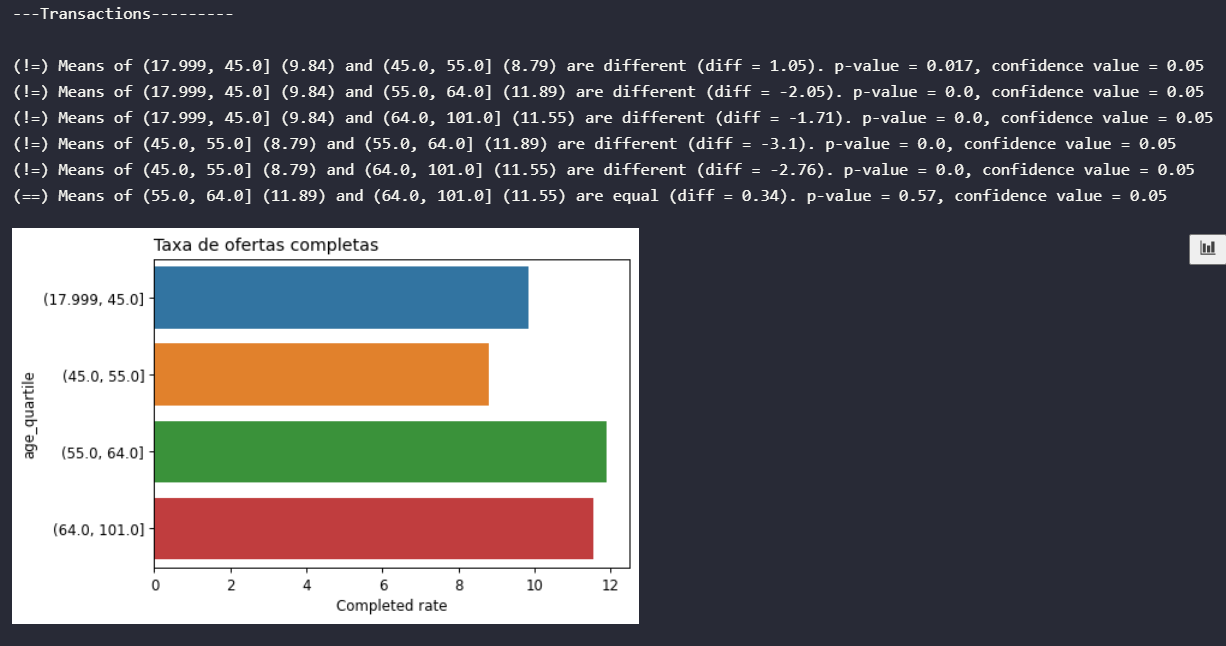
* Cluster 0: Middle age, middle income and recent member
* Cluster 1: Middle age, middle income, and old limbs
* Cluster 2: High age, high income and recent member
* Cluster 3: Low age, low income, and recent membership

Cluster 2 identifies itself as high-age, high-income, and recent member. The highlight in the numbers are the fact that they have higher incomes and are new members. Cluster0 and 3 have low average income and are recent members. Cluster 1 is second in metrics, much to identify that they are older (perhaps more faithful) members.

The graphs show a lot for us of the behavior of the numbers.

To draw a statistical conclusion of the values, we use the t-test to compare means to prove that the differences observed in some groups are significant, that is, we can state that they are different.

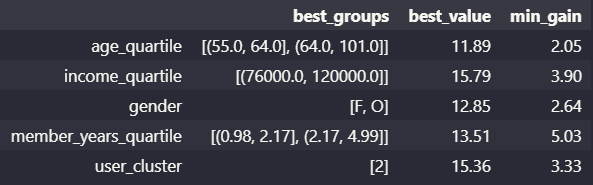
Below I show the test for the differences observed in the quartiles of ages.



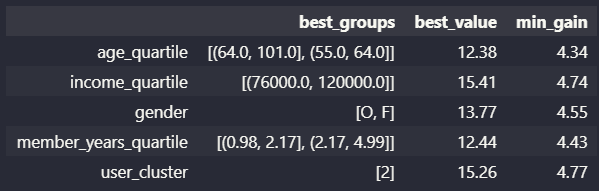
The statistical test showed that the differences between the older age and younger age groups is significant.

Using these statistics, you can determine, by offer type, which groups are best in terms of metrics. If one group has a metric statistically equal to another, they are positioned together. The best value is higher average among groups. The minimum gain is the difference for the second group significantly different.

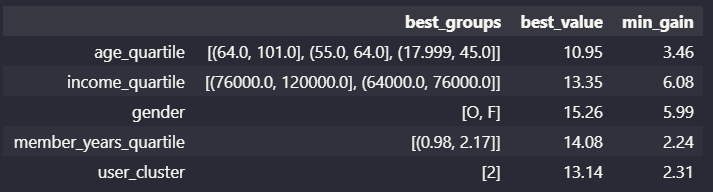
For average transactions in BOGO-type offers:



For average transactions in Discount offers:



For average transactions in Informational offers:



Note that for informationa-type offerings, there are no major differences in different groups. For age and income, only one group was left out.

Between BOGO and Discount, it is noted that the minimum gains for the discount are higher, that is, the groups with lower value metrics have the largest differences.

The same summary can be done by user group. Based on the highest metric value, what is the best type of offer to deliver to each set of users.

For example, for age groups, based on the average of transactions.

The older groups do not have great differences between the offers, while age group minors (18 to 45) have better numbers with informational and bogo and do not have good numbers with discount type offers.

By gender, it is noted that women do not have differences between the types of offers, already men have better numbers with BOGO compared to the other two types.



# Data Modeling

The statistical validation performed in the exploration of the data can already be used to guide the distribution of the offers.

In this section this offer guidance tries to be resolved via machine learning model. The advantage compared to previous statistical validations is that model can handle more profiling variables at the same time, which can optimize results.

Here two metrics will be the targets in the models. The first will be the full offer fee and the other the transaction amount.

As for the full rate, the idea is to rate, based on demographic information, whether or not a user will complete a certain offer.

As for the transaction, the idea is to analyze two strands. One is to predict how much a user will trade from the view of an offer. The other is to try to sort whether that user will trade any value when viewing that offer.

## Classification for full rate

Before you start modeling, you must handle the arrangement of the information.

Some users have received the same offer more than once, which generates in the fractional data full rate. First, any complete rate above 0 is considered as 1, indicating that the user completed the offer after viewing it.



After that, you can seed the columns that matter for data modeling.



With the base defined, part for the step-by-step modeling, explained in the code below.

The columns in the dataset are:

* Categoriacal target: completed\_after\_view\_rate
* Categorical: gender, user\_cluste r and offer\_type
* Continuos: age, income, rela\_member\_years

Two tranfromers were used with the data. For categorical type columns, a one hot enconder was used to transform categorical information into dummies variables.

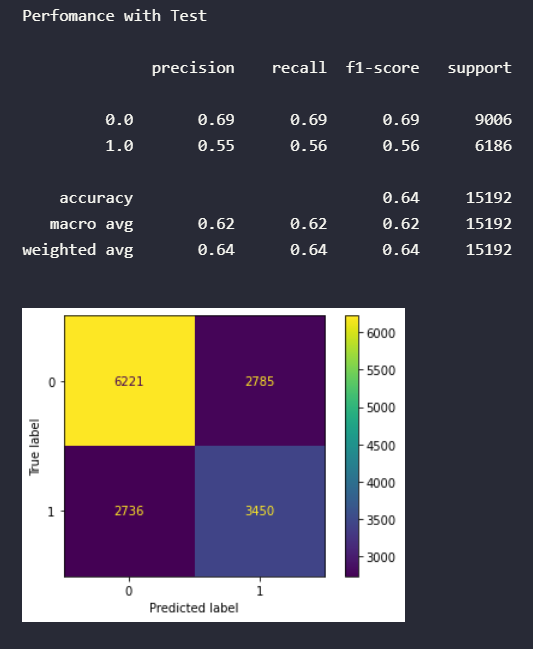
For the numerical variables, a StandartScaler was used to colonize the numbers, preventing the order of magnitude of the numbers from hindering the modeling.

The classifier chosen was the decision tree. The 'balanced' value was set to the class\_weight to allow the estimator to oversample. This prevents the predominant class from hindering na modeling.

Transformers and classicators are placed in a pipeline. The advantage of a pipeline is that training data does not contaminate test data, for example, in the scolonation of numeric variables.

Fit is performed in the pipeline with trainemtno data



The evaluate\_model receives the model created and the training and test bases and plots the results. Observing the performance of this model p rimeiro.

The evaluation metric used here will be the recall. Because it represents how much of the actual classes the model has managed to hit, it fits the purpose of modeling.

The reusltados clearly show an overfitting for the training data, in which the recall for class 1 was at 99%, while in the test the value reached 56%.

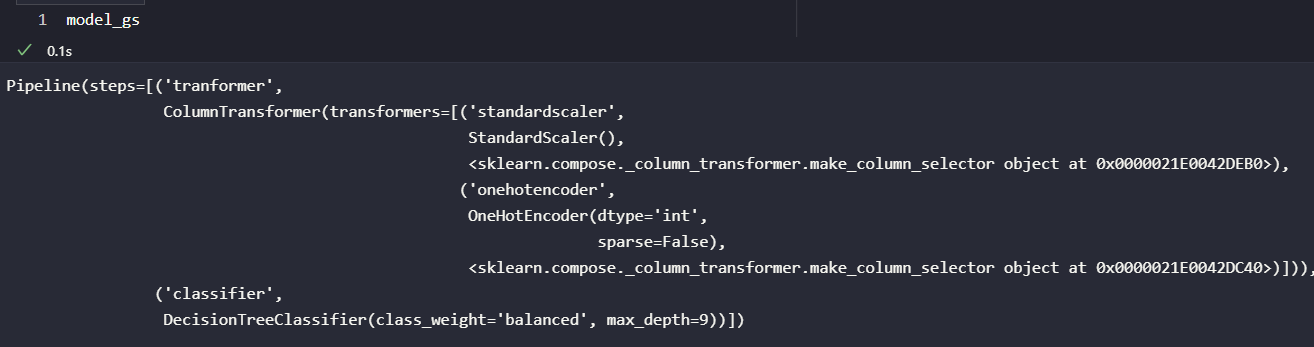
One of the factors that can cause this in a decision tree is the depth of the tree. When the depth reached for the training data, the tree fits the training data, the greater.

This can be evaluated in a GridSearch, pusing the best result for the recall in a relatively small range. Cross-validation performed in GridSearch will ensure that overffiting is avoided.

In the DecisionTree class, this parameter is the max\_depth. In addition to this parameter, the criterion parameter between 'gini' and 'entropy' is also tested.

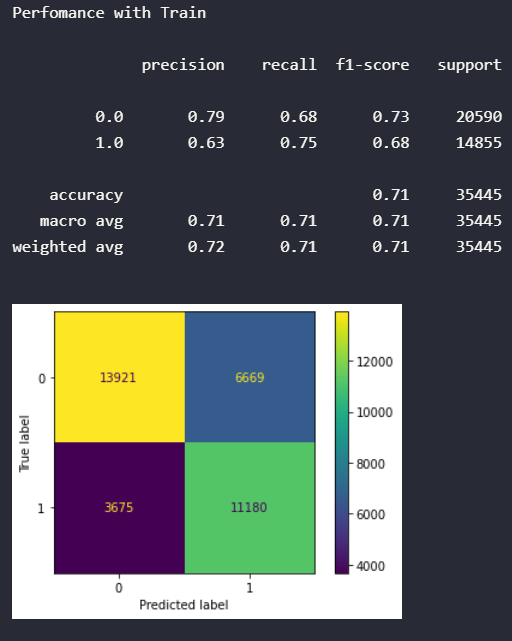


The output of this grid Search is the model model against the defined metric (recall).



Note the small depth value (9).

Evaluating results



After gridsearch, the results look better. With recall for 73% test data, the model can be used.

You don't search for models with very high recalls, without fear of underfitting. For the purpose of modeling, a more generalist model may better translate the behavior of several users and new users that may arise.

## Implementation of the model

The created template can be used to determine which ones to rate whether each user will complete an offer.

The implementation of the model is done in the best\_user\_offer\_to\_send(). The role is entered as a user's id.

First, the function filters the dataset profile, containing user information.

Then you take the portfolio base and expand it within the user's profile base.

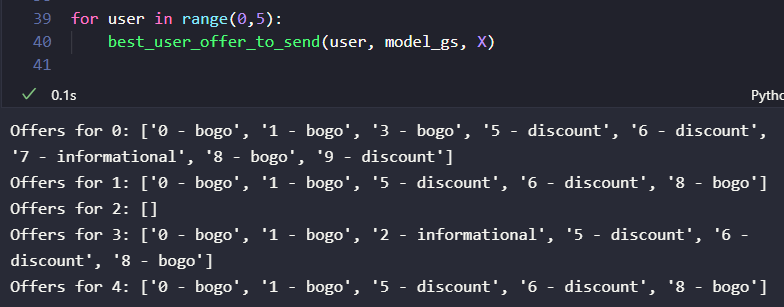
With this, a table is created simulating that the user has received all offers.

Before you enter the model, you must ensure that the base has the columns in the same order in which the model was trained.

With this, a complete prediction column can be added using the trained model. With the prediction made, the best offers are those that the model predicted would be completed.



Applying the role to the first five users, these are the recommendations:



For the user with id = 0, all offers are expected to be complete. For user id = 1, only 0, 1, 5, 6 and 8 is expected to be completed.

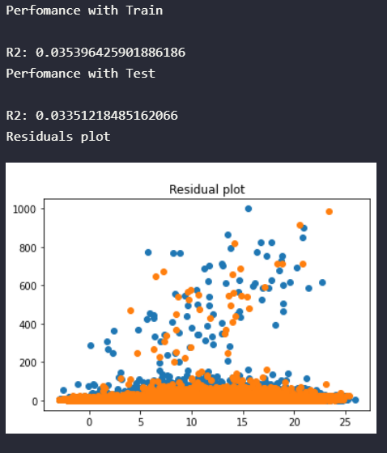
For the user with id = 2, there are no offers that the prediction model would be completed. This result may change over time. In an experiment with more data, the user profile id = 2 might have a preference for some offer, and the model understands that.

## Transactions.

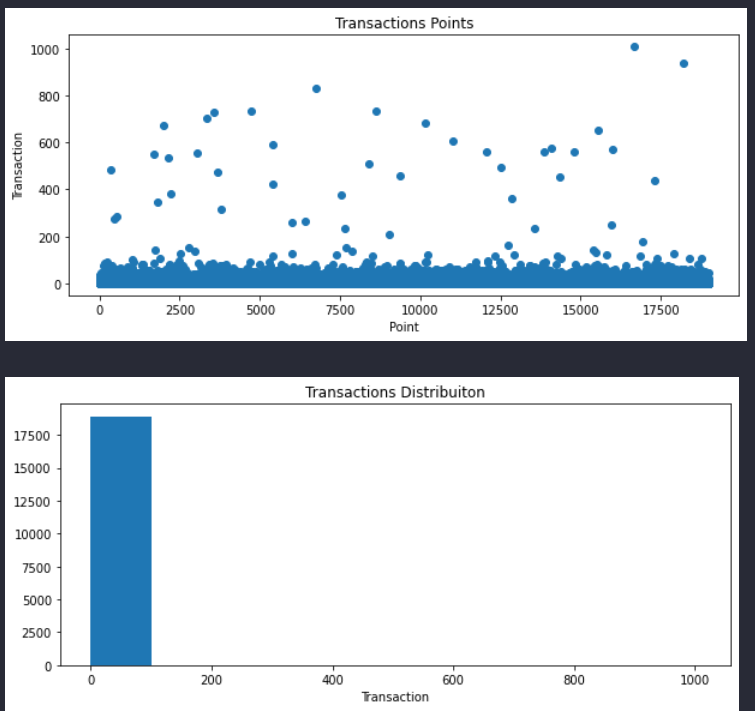
By the nature of the data type, transactions are a candidate for linear regression. The tentanitvaus with this model is performed in the code below.



However, the results found are not feasible for use. R² shows very low values, close to 0, showing that the model cannot fit the data well. Looking at the residual sgraph, errors are significant in virtually any predicted range of values.



What happens is that most transaction values are close to 0, with left tail sharp.



Thus, the data present low variability for the regression model to understand and be able to predict the test values.

One strategy to be able to make it possible to use the transaction variable is to adptar the model for the classification model.

The change consists in considering that any transaction, that is, transaction values above zero, are indications that the user made purchases influenced by that offer. Thus, two classes are divided:

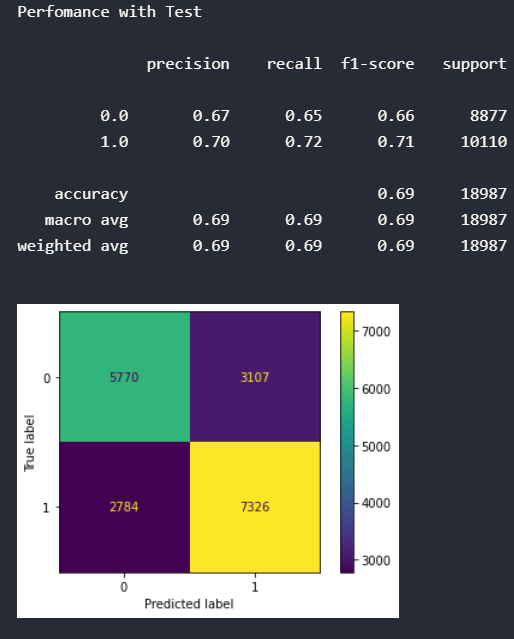
* 0: There were no transactions. Tra\_offer\_infl == 0.
* 1: There was some transaction. Tra\_offer\_infl > 0.

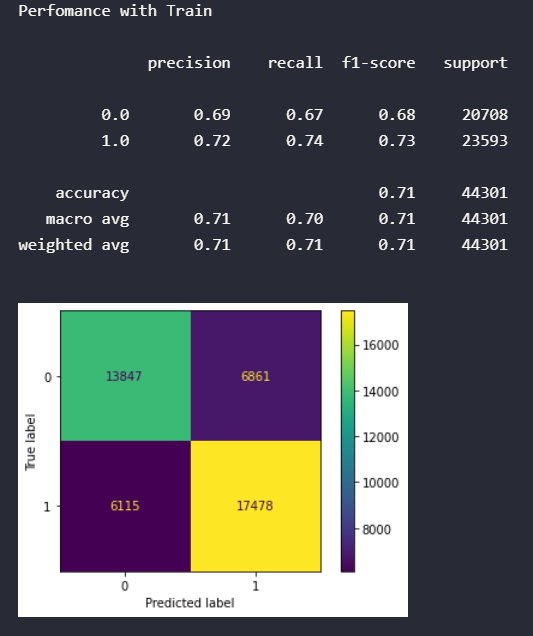
The code below puts this rule in the dataset.



With this, you use the same previous steps that are followed to create the classification model. Once created, you use GridSearch to look for the best parameter for depth.



The result of this model can be seen below



A result similar to the model for classification of the completed rate. Again, the point here is perhaps to create a more generalist model, which will be able to fit into more than one situation, below what has been trained.

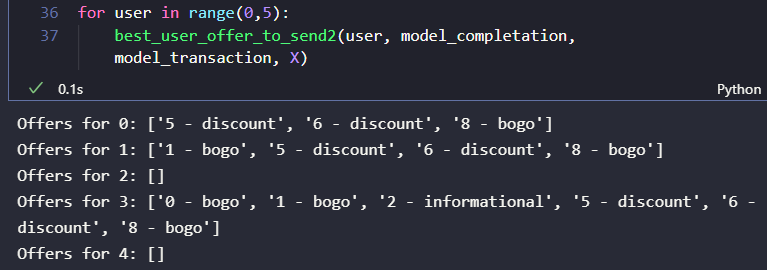
## Combining the two models

With the second model created, you can combine both to generate better guidelines for sending offers. The idea is to send offers that are expected to be completed by the user and will also have some transaction made by the user. The best\_user\_offer\_to\_send2() implements this.



The novelty in this function is the creation of the transaction\_pred. With it, the transaction model is used, predicting whether that user with your profile will trade each offer. In addition, the score column multiplies the prediction columns. With this, values of 1 indicate that both the user will complete and will trade. 0 can indicate that the user is expected to complete, but not trade or trade without completing the offer.

Applying this function to the first 5 users again



Note that for user id==0, using the first template, all offers would be appropriate to be sent. With the combination of the two models, it was reduced to offers 5, 6 and 8. This means that, by model, only these three offers are expected to also trade, in addition to being complete.