# Entrega de ofertas Starbucks

Definir o problema

A Starbucks é uma grande rede de cafés, presentes em todo o mundo. Suas vendas chegam a mais de 4 bilhões de dólares no ano de 2021.

A rede possui clientes fiéis, que consomem com regularidade e garante parte do sucesso da empresa.

Porém, a rede de cafeterias é reconhecida pelo seu bom marketing, atraindo e mantendo novos clientes a cada dia.

Uma das principais estratégias para isso são as ofertas promocionais. Gerando um sentimento de compensação para o cliente final, as ofertas costumam chamar atenção e aumentar a receita das lojas.

Os dados analisados aqui veem de uma simulação feita pela própria Starbucks do padrão de consumo dos seus clientes, a partir do recebimento de ofertas via web, e-mail e rede social.

As ofertas são de três tipos: compre um, leve outro; informacional e desconto. As ofertas BOGO e desconto tem uma recompensa diferente, assim como uma quantia mínima para ser recompensado.

O objetivo de uma boa campanha de envio de ofertas é aumentar a quantidade total transacionada pelos consumidores.

Com auxílio dos dados e de técnicas de machine learning é possível otimizar uma campanha. A partir das respostas dos usuários a ofertas já enviadas, é possível entender qual perfil de usuário responde melhor a qual tipo de oferta e, dessa forma, direcionar as ofertas da forma mais otimizada.

Com os dados disponíveis, esse artigo mostra como os dados foram tratados, visualizados e modelados para fazer recomendações de quais ofertas são mais melhor para cada perfil de usuário.

# Tratar os dados

## Os dados

Foi fornecido três bases de dados:

Profile – dataset com dados dos usuários presentes na base transcript

* gender – sexo do usuário
* Age – idade do usuário
* Income – renda do usuário
* Id – hash
* Duration – a duração da oferta, no qual se considera o tempo de influencia dela sobre o usuário.

Portifolio – base com informações sobre as 10 ofertas diferentes disponíveis para serem ofertadas.

Transcript – base com os eventos que aconteceram no experimento.

Time – tempo que foi registrado o evento

Value – dicionário com os valores dos diferentes eventos.

Offer id - para ofertas enviadas, visualizadas e completas

Reward – caso uma oferta seja completa, a quantidade ganha pelo usuário ao completar aquela oferta.

Amount – quantidade transacionada, caso seja um evento de transação de valores.

Person – usuário referente ao evento

Event – classificação do evento que aconteceu

FOTO DAS TABELAS AQUI

A base transcript é a principal para a análise.

Os eventos registrados no log são de 4 tipos:

Offer Received – um usuário recebeu a oferta

Offer Viewed – um usuário visualizou a oferta

Offer Completation – um usuário completou a oferta, ganhando uma recompensa

Transaction – quando um usário faz uma transação.

## **Criação de ids numéricos em profile e portifólio**

Os ids de usuário e de ofertas vieram na base no formato de hash. Para transformar em um formato mais fácio de visualizar, foi criado o script mapper\_id.py, com a função mapper\_ids(), responsável por pegar os ids em hash e criar um dicionário de conversão para id inteiros.

O mapper\_ids() é utilizado para os dois casos de ids hash, coluna person em profile e id no portifólio.

Dentro do script os dicionários criados são exportados como jsons, de forma que possam ser utilizados em qualquer outro script e também mantém a repordudilidade desse tratamento de informações. Os arquivos ficam alocados dentro da pasta mapper\_id.

## Data Preparation

Todo o tratamento dos dados está no script ‘Data Prepartion.py’.

O script inicia fazendo a carga das bases fornecidas e os dicionários criados pelo script mapper\_id.py.



## Tratamento da transcript

A dificuldade no tratamento de dados está na disposição deles.

A base transcript é uma time line dos acontecimentos. Para extrair o que se deseja, o efeito de uma oferta sobre o comportamento de compra de um usário, é necessário extrair a informação do tempo e congelar ela em uma tabela resumo.

O tipo de tabela imaginada foi um cruzamento de usuários e ofertas. Nessa tabela, se dispõe quais ofertas cada usuário recebeu e como ele respondeu (se visuailizou, se completou e quanto transacionou influenciada por ela). A partir da timeline é que se pode avaliar se o usuário fez a transação dentro do tempo de duração limte de cada oferta. Algumas trasanções são soltas, sem necessariamente terem sido feitas por conta de uma oferta.

Alguns defasios e seus contornos no tratamento de dados:

1. Algumas ofertas podem ser completadas sem mesmo terem sido vistas. Nesse caso, não se considera a transação realizada para completar a oferta. Se irá considerar somente transações influenciadas por ofertas após a oferta ter sido visualizada. Além disso, considera-se que aquela completação não é válido, pois o usuário não tomou ideia da existência da oferta, o que supõe-se que ele não foi influenciado para completar ela.
2. Algumas ofertas são recebidas e visualizadas pelo o usuário, porém, antes de uma transação, outra oferta é recebida e visualizada. Nesse caso, qualquer transação que venha depois da segunda oferta, será contabilizada para esta segunda, anulando o efeito da primeira.
3. A mesma oferta pode ser entregue para um usuário mais de uma vez. É possível que o usuário complete essa oferta repetida mais uma vez na base transcript. Para esse caso, os dados serão agrupados para usuário e oferta, considerando a média de vezes que o usuário visualizou a oferta, média de quantas vezes ele completou e soma de quanto ele transacionou influenciado por aquela oferta.
4. Algumas ofertas são visualizadas após o prazo de validade delas. Nesse caso, a visualização é considerada não válida e qualquer transação realizada após essa visualização não é contabilizada.

O gatilho para salvar tratar as bases parte da função generate\_user\_offer():



A função itera sobre todos os usuários presentes na base transcript. A função create\_user\_offer\_df() retorna um dataframe. Os dataframes são unidos via função do pandas concat. A base é salva em um arquivo csv para ser trabalhada nos próximos scripts.

A função create\_user\_offer\_df() recebe um id (hash) e cria um dataframe com a função get\_offer\_table\_user(), principal função para tratar a base transcript. Depois, aplica a função group\_offer\_df(), responsável por agrupar as informações no formato de usuário versus ofertas. No final, a função cria uma coluna user\_id para identificar o usuário no qual as informações são a respeito.



A função get\_offer\_table\_user() tem quatro partes. Ela recebe como argumento o código do usuário (em hash) e faz uso das outras variáveis definidas no começo do script. Aqui, vou explicar como funciona cada parte.

1. Separar as bases

São 4 eventos diferentes dentro de transcript: offer received, offer viewed, offer completed and transaction.

Nessa primeira parte do código, esses quatro eventos são separados em dataframes exclusivos: received\_df, viewed\_df, complete\_df and transaction\_df.

Cada um deles vai possuir as informações relativas as suas fases.



Para criação dos dataframes, usa-se a função get\_subset()



Uma das facilidades dessa função get\_subset() é extrair os valores dos dicionários, a partir do argumento de entrada dict\_keys.

A segunda parte da função é analisar a timeline, utilizando as 4 bases extraídas na primeira parte.

O fluxo segue o seguinte raciocínio:

1. Merge entre a base de recebidas e visualizadas. Se mais de uma oferta for entregue para os clientes, há um filtro que garante que a visualização foi realizada depois que aquela oferta foi recebida e antes da próxima oferta. Cria-se a coluna que informa o período máximo no qual aquela oferta tem influencia. Usa-se a função validate\_view() para validar visualizações feitas dentro do período válido. Ao final, o tempo no qual a próxima oferta é visualizada é disposto em uma coluna. Essa informação funcionará para filtrar as transações das próximas ofertas.
2. Merge da base anteiror com a base de completadas. O objetivo é saber quais das ofertas foi completada pelo usuário. Alguns problemas nessa mesclagem é que a mesma oferta pode ter sido completada mais de uma vez no tempo. Por isso, usa-se a estratégia de drop\_duplicates. Para contabilizar as completações válidas, as quais aconteceram após o usuário ver a oferta, a coluna completed\_after\_view é criada. Ao final, a função guarantee\_viewed() é utilizada para garantir que todas as ofertas recebidas pelo usuários originalmente estejam na saída dessa etapa. Isso é necessário porque o filtro da função merge\_and\_filter pode retirar ofertas repetidas que foram “completas” do tempo de recebimento.
3. Para as transações, o merge não fucionaria. Ao invés disso, a tabela de trasanções é iterada, linha a linha, para contabilizar o total por cada oferta enviada para o usuário. A regra para a transação é que a transação tenha sido feita antes da visualização da próxima oferta; antes do período máximo de validade da oferta; e antes da oferta tenha sido completada.



1. No final, duas colunas são criadas, viewed and completed, contabilizando como 0 ou 1 se a oferta foi visualizada e se ela foi completada.

A terceira parte da função get\_offer\_table\_user() é cria duas colunas são criadas, viewed and completed, contabilizando como 0 ou 1 se a oferta foi visualizada e se ela foi completada.



A quarta e a última parte da função é seleciona as colunas que importam para a aplicação da próxima função.



A função group\_offer\_df() recebe a saída da função get\_offer\_table\_user() e agrupada nas métricas que serão utilizadas no restante da análise de dados.



Para cada métrica, foi-se utilizada a agregação conveniente. Por exemplo, para a taxa de visualização, usou-se a média, enquanto para as transações foi utilizada a soma. Sobre as métricas criadas:

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

No final da função create\_user\_offer\_df(), uma coluna com a identificação do usuário é criada e a saída é um dataframe. Esse dataframe possui todas as ofertas recebidas por cada usuários e como eles responderam a cada oferta.

## Criando coluna com id integer



A função generate\_datasets() trata as bases de profile e portifólio para criar uma coluna extra, com o id no tipo int, não hash.

Salva esses arquivos como csv para serem utilizados posteriormente.

## Tratamento da Profile

A base de usuários é uma grande fonte de informações sobre o perfil de cada um.

As colunas da base profile são tanto numéricas (age, income), categóricas (gender) e de tempo (became\_member\_on).

Alguns usuários têm informações em branco, o que se torna necessário tratar antes de analisar os dados.

### Preenchendo os vazios

As colunas gender, age e income possuem vazios. Age é codificado como 118 para vazios.

Na base, são 12% de linhas vazias, representando 2040 usuários.

A estratégia para preenchimento de vazios, nesse caso, foi utilizar a mediana para as colunas de age e income. Assim, a distribuição numérica dos valores não se altera significativamente.

Já para o gender, criou-se uma nova classa, NI (not indentify). Dessa forma, se tem uma forma de rastreio dessas linhas vazias posteriormente.

Para a coluna de data, became\_member\_on, derivou-se uma nova feature para a tabela. Com a data de início como membro, pode-se calcular o tempo em anos que aquele usuário é membro. Como não se sabe o dia exato da extração dos dados, usou-se a data do usuário mais novo dentro do dataset como padrão. Assim, todos os tempos calculados são relativos a essa data mais recente. A coluna rela\_member\_years guarda esses valores.



### Clusterização dos usuários

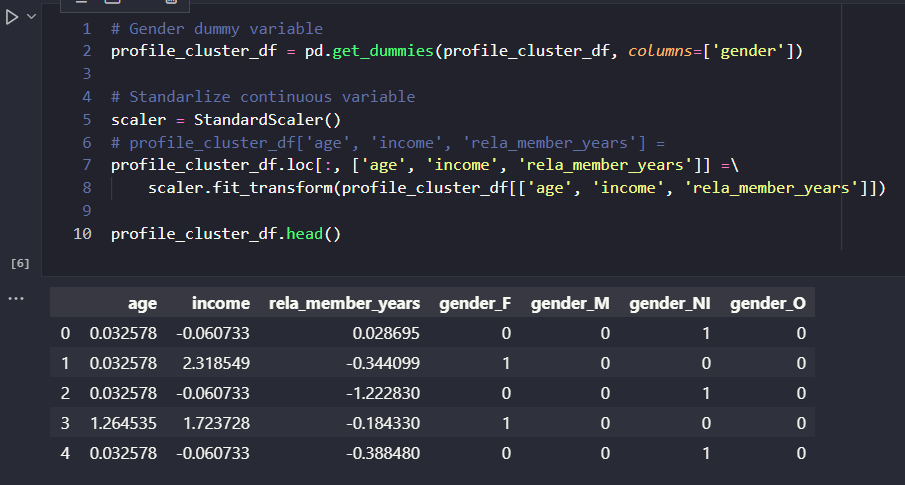
Profile tem 4 colunas que caracterizam o usuário: age, income, gender and time as member.

Há diversas combinações possíveis entre os valores dessas colunas.

Um método não supervisionado pode revelar classes dos usuários que podem ser importantes no momento da modelagem.

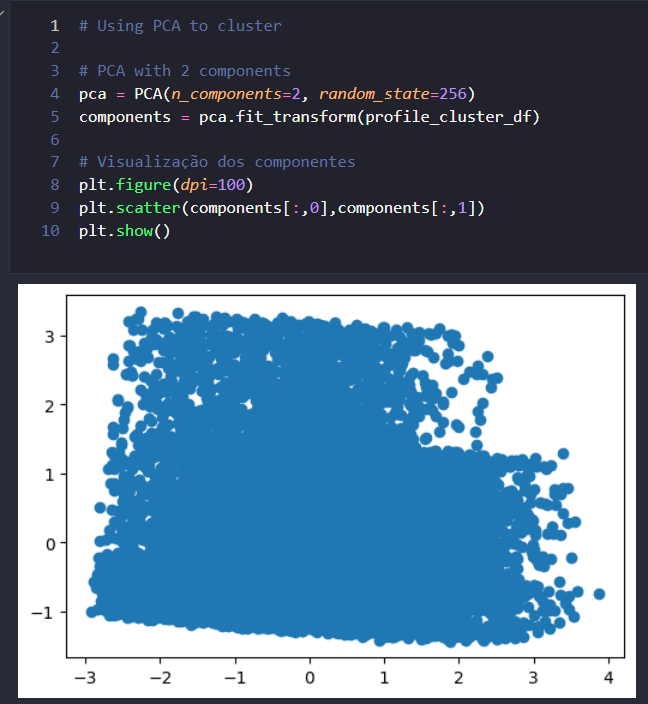
Uma clusterização é realizada para revelar esses grupos.

Primeiro, a coluna gender, uma variável categórica é transformada em variável dummy.

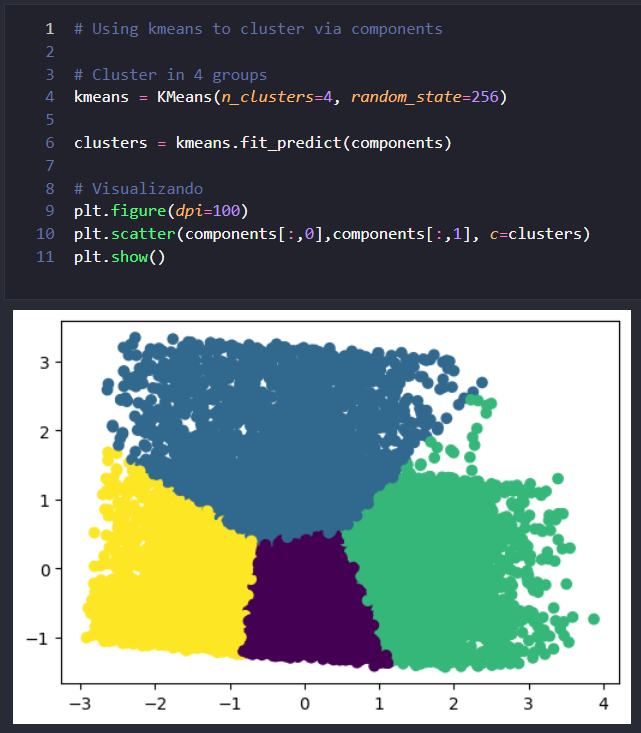


Depois, as variáveis contínuas (age, income, rela\_member\_years) são standardize para retirar a influência de grandes números.

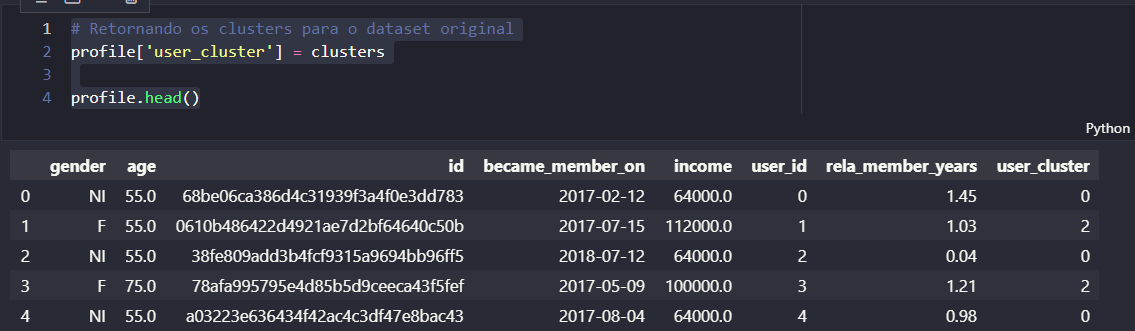
Após esses tratamentos, uma redução de dimensionalidade é executada para redução das 7 variáveis para 2 componentes principais.



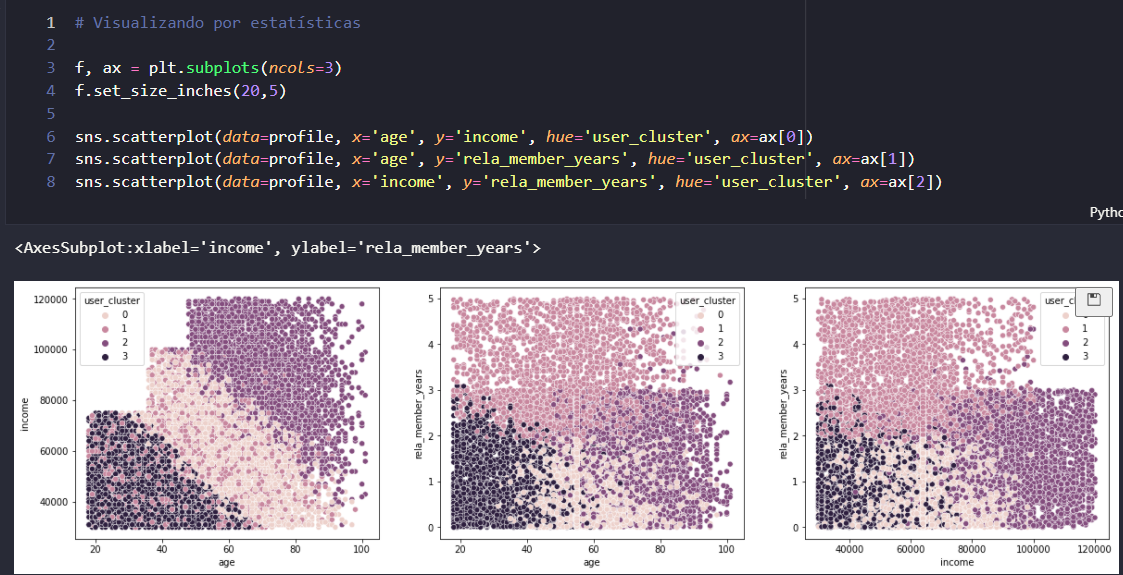
Utilizando o método de k-means, separa-se 4 grupos dentro das duas variáveis.

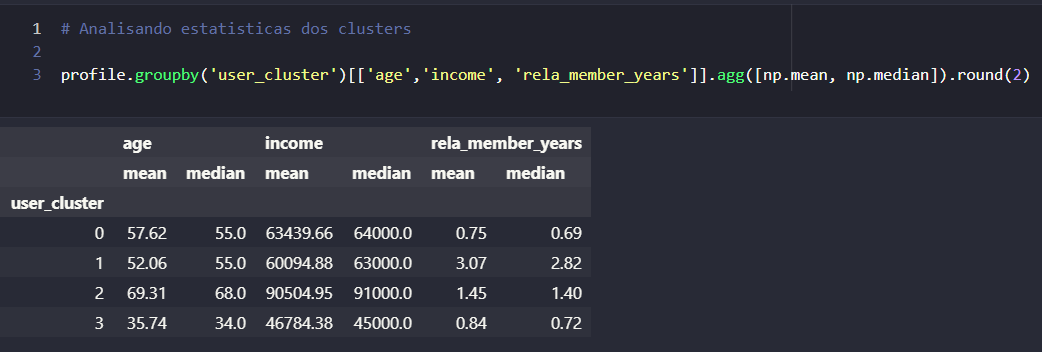


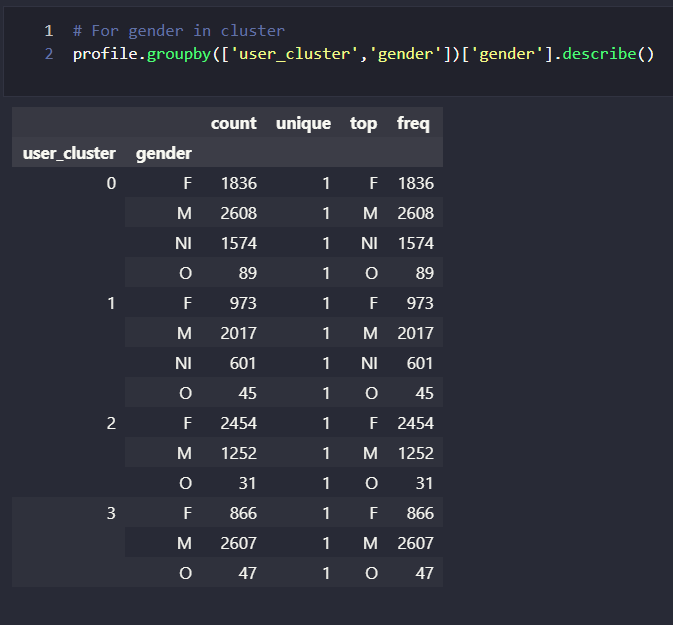
Com os clusters criados, volta essa informação para a base original de profile.



Com essa nova classe, podemos identificar grupos de usuários.







Observando as médias e as disposições dos grupos nas variáveis, pode-se definir os clusters encontrados como:

* Cluster 0: Meia idade, renda média e membro recente
* Cluster 1: Meia idade, renda média e membros antigos
* Cluster 2: Alta idade, renda alta e membro recente
* Cluster 3: Baixa idade, renda baixa e membro recente

Quanto ao gender, não se destaca nenhum grupo com classe dominante. Todos têm uma maioria de male.

A base é extraída como csv para ser utilizada pelo demais processos.

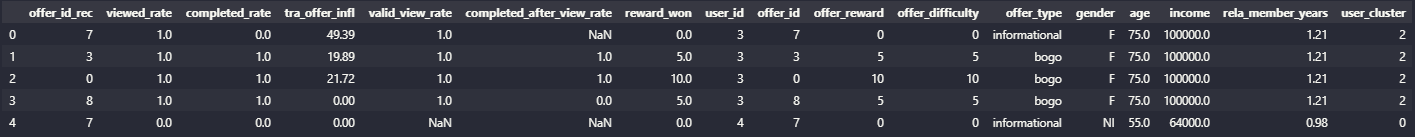
# Data Exploration

A exploração dos dados permite enxergar distribuições e tendências das informações tratadas.

A entrada principal para a análise dos dados é a tabela de usuário-oferta, criada a partir do script Data Preparation.

A partir dela, faz-se um merge com as bases de profile e portifólio.





Nessa fase de exploração, as variáveis contínuas são agrupadas em seus quartis. Esse agrupamento permite enxergar comportamentos gerais dos dados.



Último tratamento necessário é definir as métricas que serão visualizadas.

Do tratamento realizados, as três métricas que serão utilizadas serão:

- valid\_view\_rate: indica se a oferta foi visualizada dentro do período de influencia (até o período máximo). Valores de 0 ou vazios indicam que a oferta ou foi visualizada após o período válido ou não chegou a ser visualizada.

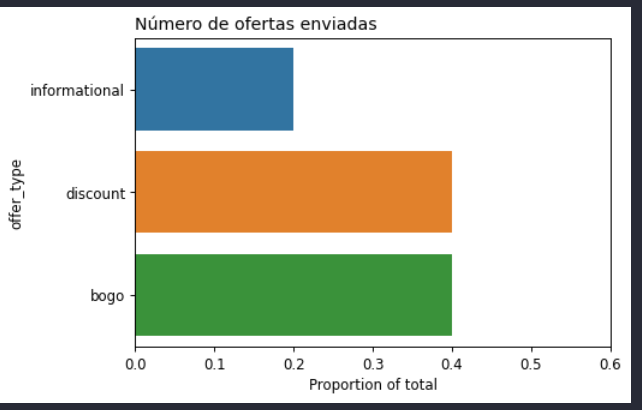
- completed\_after\_view\_rate: indica se a oferta foi completada após ser visualizada. Valores de 0 indicam que o usuário não foi influenciado a completar a oferta, já que não chegou a visualizar. Valores vazios indicam que a oferta não foi completa.

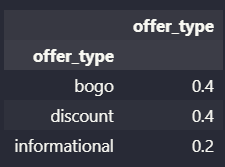
Para utilizar essas duas colunas, substitui-se os valores NaN por 0.



Alguns pontos sobre os dados.

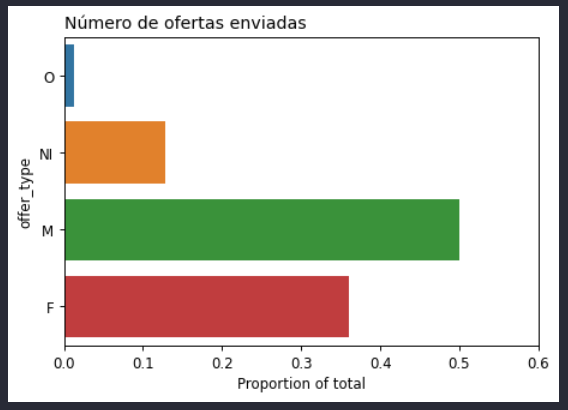
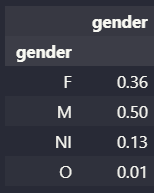
Aqui mostrarei alguns pontos interessantes sobre os dados. Uma visualização mais completa pode ser vista no notebook Data Exploration.ipynb.





A quantidade de ofertas enviadas dentro da base se distribui igual a quantidade de ofertas disponíveis dentro de portifólio.

O mesmo comportamento é visto na distribuição de usuários, no qual os percentuais se distribuem igual a base profile.



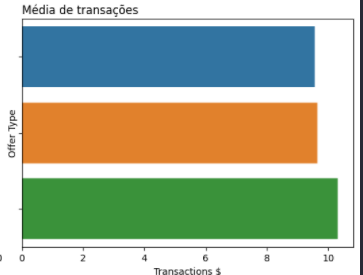
Isso indica que a simulação garantiu equidade nas distribuições de ofertas e usuários disponíveis, não priorizando nenhum grupo em específico. Quanto a quantidade de usuários, isso acontece porque a simulação usou praticamente todos os usuários. De fato, somente 6 usuários não receberam oferta alguma.

Em relação ao tipo de oferta, nota-se que as taxas de visualização se distribuem próximo a distribuição da quantidade de canais que as ofertas são distribuídas.

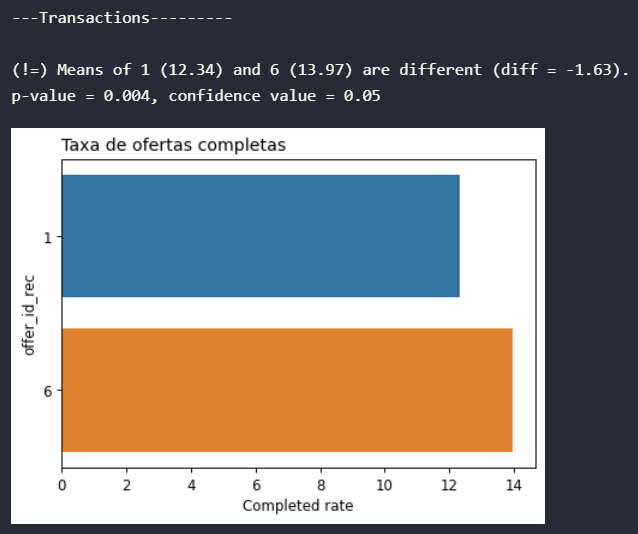


Para a taxa de completos, nota-se que as ofertas de desconto têm taxa de completos maior que as ofertas do tipo BOGO. Informational não tem taxa de completo, já são ofertas somente de propaganda, que não oferecem recompensas.

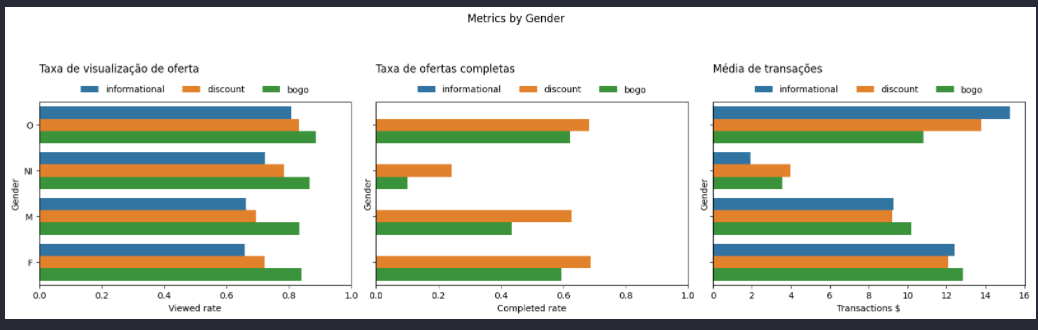
Para transações, a média de transações para usuários que visualizaram ofertas do tipo BOGO são maiores, no geral, porém as ofertas são maior que para os outros tipos. Porém, nota-se que a ofera 6, do tipo desconto se destaca das demais no quesito transações.



Comparando essa oferta com o segundo lugar, a oferta BOGO com id 1, a diferença de $ 1.63 em média é estatisticamente significativa (utilizando teste t para comparação entre médias)



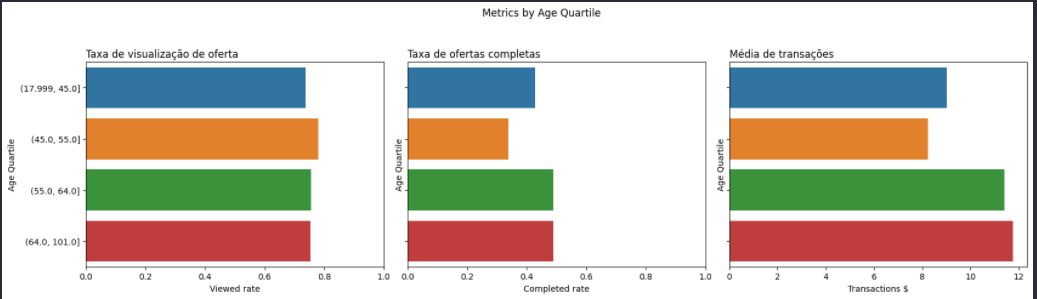
Por gender.



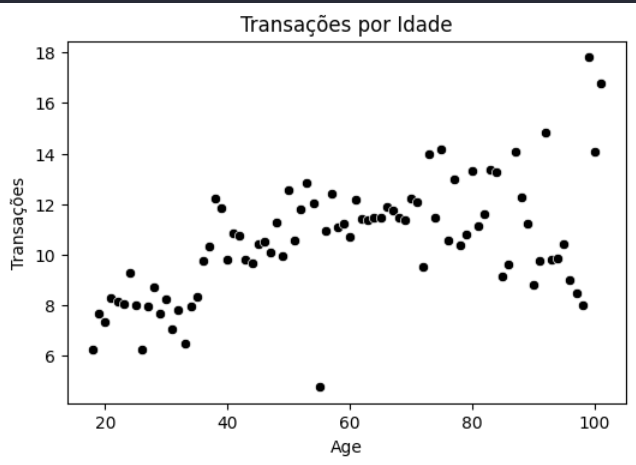
O grupo others têm maiores transações média pela quantidade da amostra, que chega a ser menor. O que pode ser destacado também é que mulheres tem médias de transações maiores que homens, com maiores transações para ofertas do tipo BOGO e segundo para as informationals.

Entre os grupos, mantém-se a tendência de completar mais ofertas de desconto que de BOGO.

Por idade, os números apresentam uma correlação positiva. Quanto maior a idade, mais ofertas são completas e maior a média de transação.

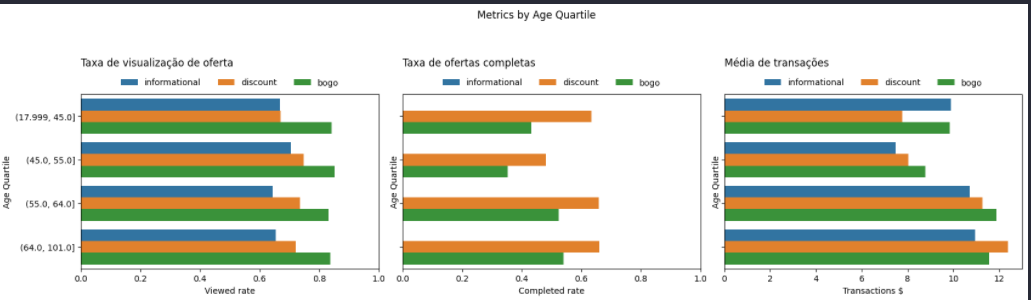


Olhando os pontos sem ser por grupos

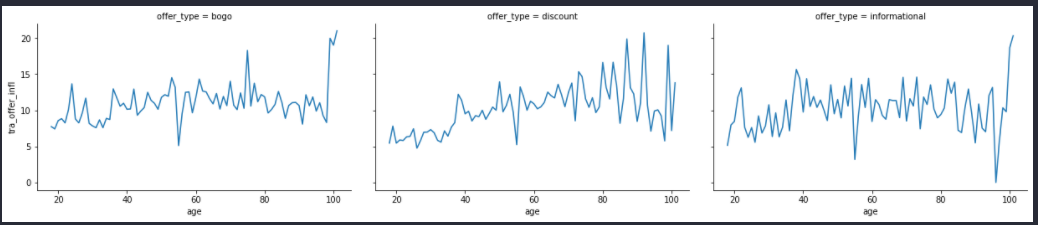


Nota-se que idades intermediárias e avançadas tem maiores transações.

Abrindo por tipo de oferta, para o grupo de idade de 55 a 64 anos, há uma média de transação maior para BOGO, enquanto para grupo de idade mais avançada, 64 a 101, a média é maior para ofertas de desconto.

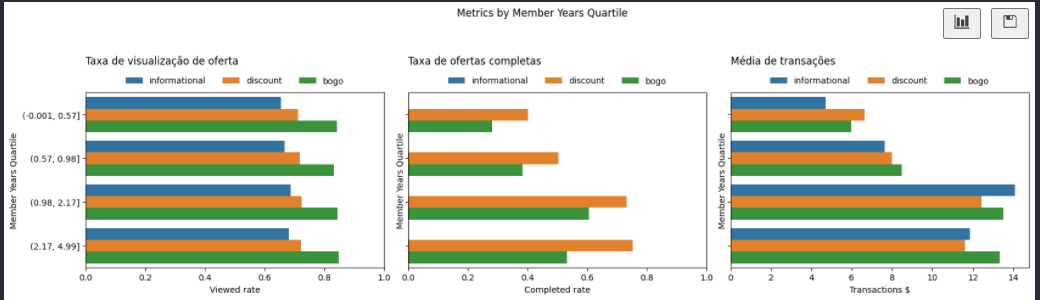


O gráfico abaixo mostra que as transações para ofertas de desconto (gráfico do centro) aumentam na região avaçanda de idade.



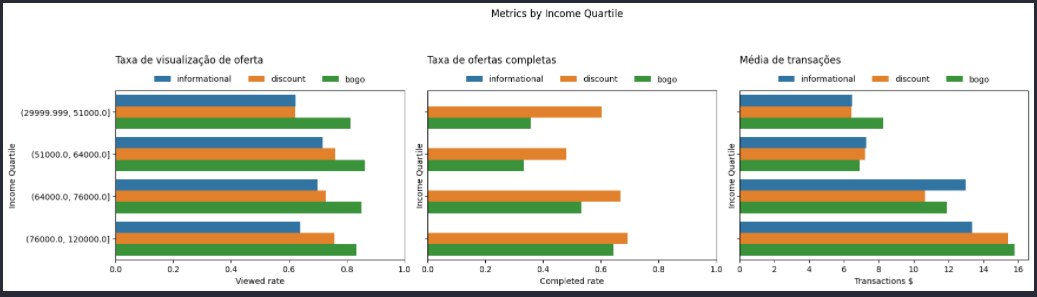
Quanto a taxa de completas, ela se equivalem para os dois grupos de idades e se destacam para as ofertas do tipo desconto.

Olhando para a coluna de tempo de cadastro, nota-se que membros mais antigos têm maiores médias de transações.



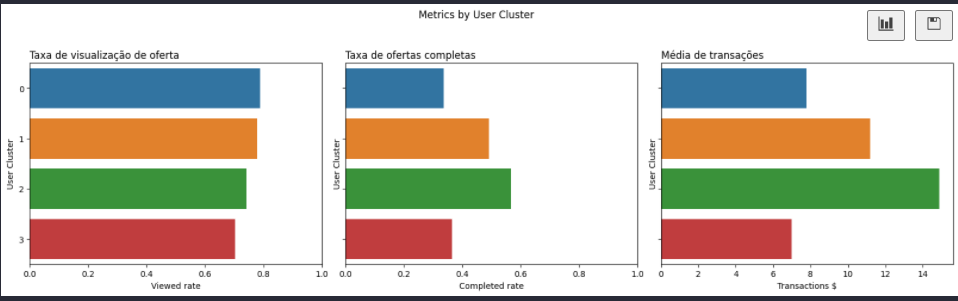
Nota-se, ainda que membros com tempo entre 1 e 2 anos, fazem transações maior para informational, quando membro de mais de 2 anos transacionam mais com ofertas do tipo BOGO. A taxa de completos segue o mesmo padrão, com desconto com maior taxa.

Olhando renda, a obviedade acontece. Quando maior a renda, mais o usuários transaciona em média. A diferença entre as médias entre o último e penúltimo grupo chega a ser $4.



Outro ponto a comentar é que grupo com maior renda tem maior média para ofertas do tipo BOGO, enquanto para o penúltimo grupo a oferta de preferência é as informational. Para a taxa de completados, nota-se que usuários com renda menor (primeiro grupo), tem maiores taxas para ofertas do tipo desconto comparada as ofertas do tipo BOGO.

Por ultimo, observando por cluster de usuários criados anteriormente, nota-se que divisão dos grupos se expressão nas métricas.



O cluster 2 tem destaque positivo na taxa de completos e transações, enquanto os números são menores para o cluster 1 e 3. Olhando novamente as descrições identificadas para cada cluster:

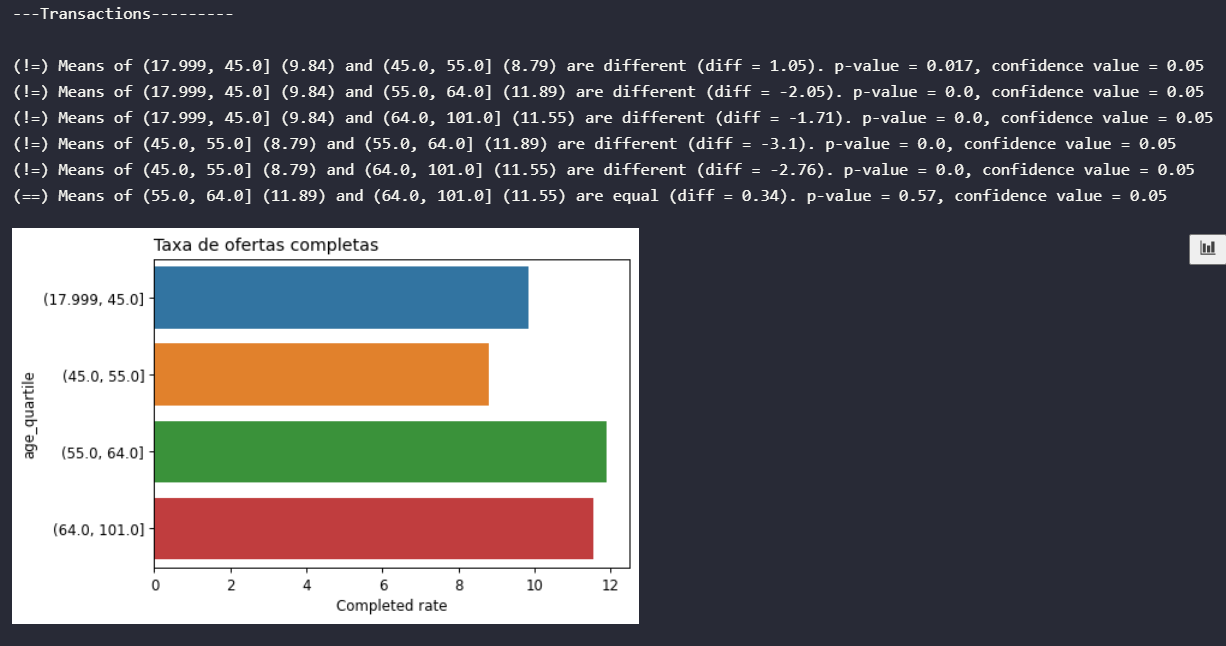
* Cluster 0: Meia idade, renda média e membro recente
* Cluster 1: Meia idade, renda média e membros antigos
* Cluster 2: Alta idade, renda alta e membro recente
* Cluster 3: Baixa idade, renda baixa e membro recente

O cluster 2 se identifica como de alta idade, renda alta e membro recente. O destaque nos números veem do fato de terem rendas maiores e serem membros novos. Os cluster 0 e 3 tem renda média oe baixa e são membros recentes. O cluster 1 é o segundo nas métricas, muito por conta de se identificar que são membros mais antigos (talvez mais fiéis).

Os gráficos mostram bastante para nós do comportamento dos números.

Para tirar uma conclusão estatísticas dos valores, usa-se teste t para comparação entre médias para comprovar que realmente as diferenças observadas em alguns grupos são significativas, ou seja, podemos afirmar que são diferentes.

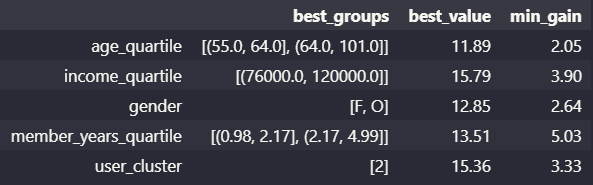
Abaixo mostro o teste para as diferenças obervadas nos quartis de idades.



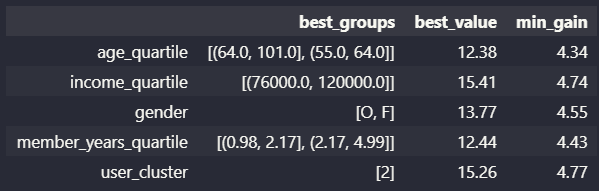
O teste estatístico mostrou que as diferenças entre os grupos de idade avançada e idade mais novos é significativa.

Usando essas estatísticas, pode-se determinar, por tipo de oferta, quais os melhores grupos em termos das métricas. Se um grupo tem métrica estatisticamente igual a outro, eles são posicionados juntos. O melhor valor é maior média entre os grupos. O mínimo ganho é a diferença para o segundo grupo significamente diferente.

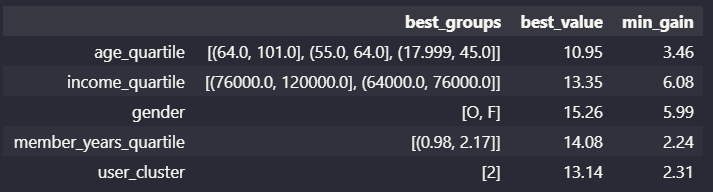
Para média de transações em ofertas do tipo BOGO:



Para média de transações em ofertas do tipo Discount:



Para média de transações em ofertas do tipo Informational:



Nota-se que para as ofertas do tipo informationa, não há grande diferenças nos diferentes grupos. Para idade e renda, somente um grupo ficou de fora.

Entre BOGO e Discount, nota-se que os ganhos mínimos para o discount são maiores, ou seja, os grupos com métricas de menor valor tem maiores diferenças.

O mesmo resumo pode ser feito por grupo de usuário. Baseado no maior valor da métrica, qual o melhor tipo de oferta a ser entregue para cada conjunto de usuários.

Por exemplo, para os grupos de idade, com base na média de transações.

Os grupos de idade avançadas não tem grande diferenças entre as ofertas, enquanto grupo de idade menores (18 a 45) tem melhores números com informational e bogo e não tem bons números com ofertas do tipo discount.

Por gender, nota-se que mulheres não têm diferenças entres os tipos de ofertas, já homens tem melhores números com BOGO comparado aos outros dois tipos.



# Data Modeling

A validação estatística realizada na exploração dos dados já pode ser utilizada para orientar a distribuição das ofertas.

Nessa seção essa orientação de ofertas tenta ser resolvida via modelo de machine learning. A vantagem em comparação com as validações estatísticas anteriores é que modelo consegue lidar com mais variáveis de perfilamento ao mesmo tempo, o que pode otimizar os resultados.

Aqui duas métricas serão os alvos nos modelos. A primeira será a taxa de ofertas completas e a outra a quantidade de transação.

Quanto a taxa de completos, a ideia é classificar, com base nas informações demográficas, se um usuário vai ou não completar certa oferta.

Quanto a transação, a ideia é analisar duas vertentes. Uma é prever quanto um usuário vai transacionar a partir da visualização de uma oferta. A outra é tentar apenas classificar se aquele usuário vai transacionar algum valor quando visualizar aquela oferta.

## Classificação para taxa de completos

Antes de iniciar a modelagem, é necessário tratar a disposição das informações.

Alguns usuários receberam a mesma oferta mais de uma vez, o que gera nos dados fracionários taxa de completos. De primeira, considera-se qualquer taxa de completos acima de 0 como 1, indicando que o usuário completou a oferta após visualizá-la.



Após isso, separa-se as colunas que importam para a modelagem dos dados.



Com a base definida, parte para o passo-a-passo de modelagem, explicitado no código abaixo.

As colunas no dataset são:

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

Dois tranfromers foram utilizados com os dados. Para as colunas do tipo categóricas, usou-se um one hot enconder para transformar as informações categóricas em variáveis dummies.

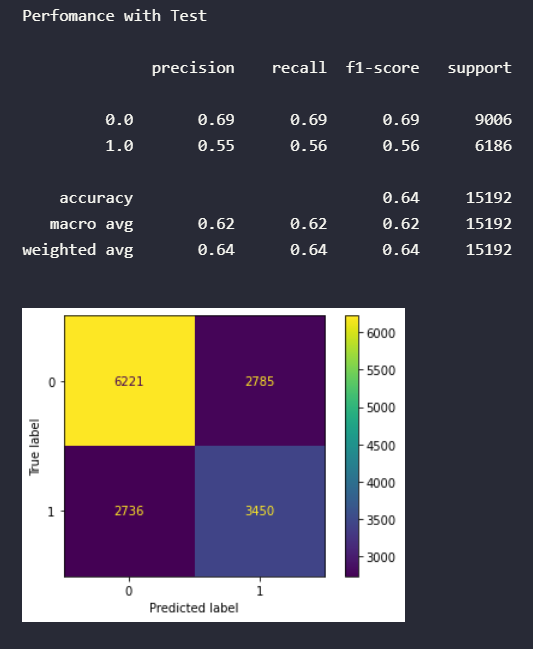
Para as variáveis numéricas, usou-se um StandartScaler, para escolonar os números, evitando que a ordem de grandeza dos números possam atrapalhar na modelagem.

O classificador escolhido foi a árvore de decisão. Foi setado o valor ‘balanced’ para o parêmtro de class\_weight, a fim de permitr que o estimator faça oversample da amostra. Isso evita que classe predominante atrapalhe na modelagem.

Transformers and classicators são colocados em um pipeline. A vantagem de um pipeline é que os daos de treinamento não contaminar os dados de teste, por exemplo, no escolonamento das variáveis numéricas.

É executado o fit no pipeline com os dados de trainemtno



A função evaluate\_model recebe o modelo criado e as bases de treino e teste e plota os resultados. Observando o desempenho desse p rimeiro modelo.

A métrica de avaliação usada aqui será o recall. Como ele representa o quanto das classes reais o modelo conseguiu acertar, ela se encaixa no propósito da modelagem.

Os reusltados mostram claramente um overfitting para os dados de treino, no qual o recall para a classe 1 ficou em 99%, enquanto no teste o valor chegou a 56%.

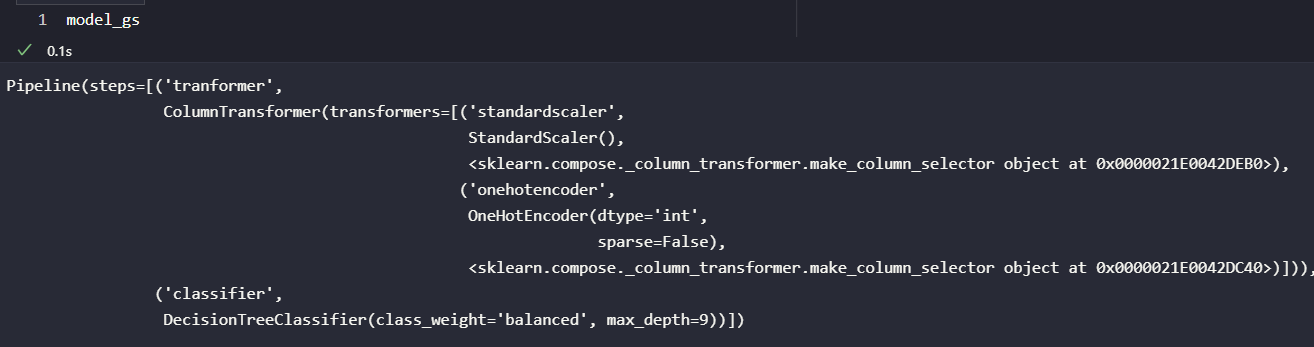
Um dos fatores que pode causar isso em uma árvore de decisão é a profundidade da árvore. Quando maior a profundidade alcançada para os dados de treino, mas a árvore se encaixa aos dados de treino.

Isso pode ser avaliado em um GridSearch, pocurando o melhor resultado para o recall em um range relativamente pequeno. A validação cruzada realizada no GridSearch garantirá que o overffiting seja evitado.

Na classe DecisionTree, esse parâmetro é o max\_depth. Além desse parâmetro, também testa-se o parêmetro de critério entre ‘gini’ e ‘entropy’.

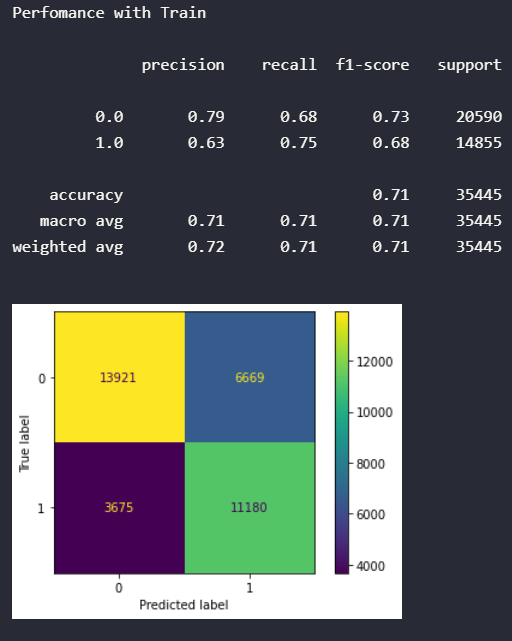


A saída desse grid Search é o modelo modelo em relação a métrica definida (recall).



Nota-se o valor pequeno de profundidade (9).

Avaliando os resultados



Após o gridsearch, os resultados parecem melhores. Com recall para os dados de teste de 73%, o modelo pode ser utilizado.

Não se busca modelos com recall muito altos, sem receio do underfitting. Para o proposito da modelagem, um modelo mais generalista talvez traduza melhor o comportamento de vários usuários e novos usuários que possam surgir.

## Implementação do modelo

O modelo criado pode ser usado para determinar quais classificar se cada usuário vai completar uma oferta.

A implementação do modelo é feita na função best\_user\_offer\_to\_send(). A função tem como entrada o id de um usuário.

Primeiro, a função filtra o dataset profile, contendo as informações do usuário.

Depois, pega-se a base de portifólio e expande ela dentro da base profile do usuário.

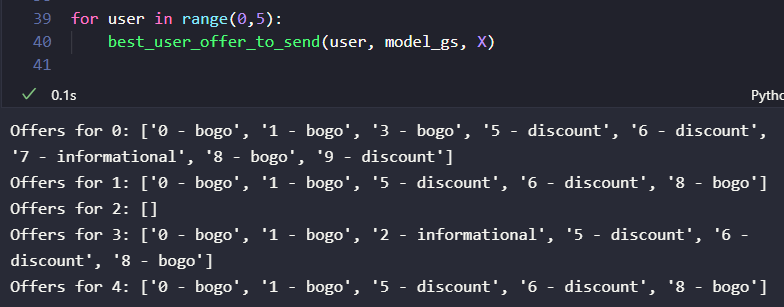
Com isso, cria-se uma tabela simulando que o usuário recebeu todas as ofertas.

Antes de entrar no modelo, é necessário garantir que a base está com as colunas na mesma ordem na qual o modelo foi treinado.

Com isso, uma coluna de previsão de completar pode ser adicionado usando o modelo treinado. Com a previsão feita, as melhores ofertas são aquelas que o modelo previu que seriam completadas.



Aplicando a função para os cinco primeiros usuários, essas são as recomendações:



Para o usuário com id = 0, todas as ofertas têm previsão de serem completas. Já para o usuário id = 1, apenas 0, 1, 5, 6 e 8 tem previsão de serem completadas.

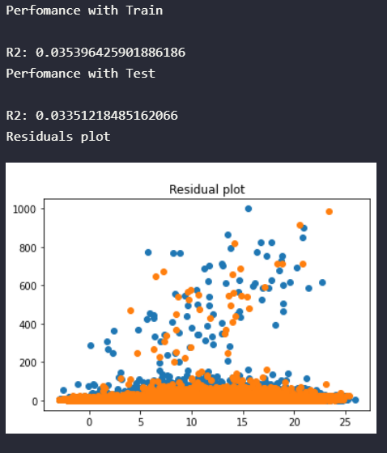
Para o usuário com id = 2, não há ofertas que o modelo previsão que seriam completadas. Esse resultado pode se alterar com o tempo. Em um experimento com mais dados, talvez o perfil do usuário id = 2 tenha preferência para alguma oferta e o modelo entenda isso.

## Transações.

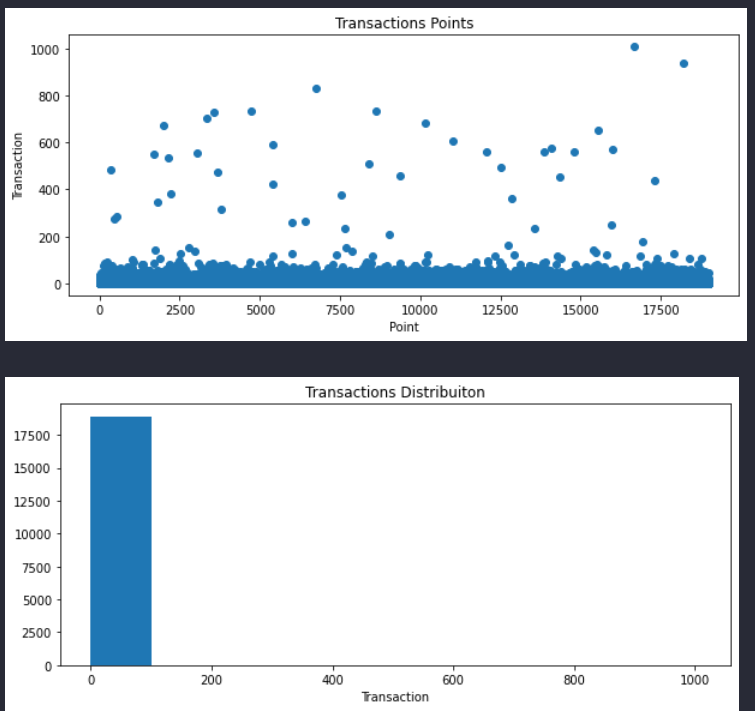
Pela natureza do tipo de dados, as transações são uma candidata para regressão linear. A tentanitva com esse modelo é realizado no código abaixo.



Porém, os resultados encontrados não são viáveis para utilização. O r² mostra valores bem baixos, próximos a 0, mostrando que o modelo não consegue se ajustar bem aos dados. Observando o gráfico de resíduos, os erros são significativos em praticamente qualquer faixa de valores previstos.



O que acontece é que a maior parte dos valores de transação são próximos de 0, com cauda a esquerda bem acentuada.



Assim, os dados apresentam baixa variabilidade para o modelo de regressão entender e poder prever os valores de teste.

Uma estratégia para poder viabilizar a utilização da variávei de transação é adptar o modelo para o modelo de classificação.

A mudança consiste em considerar que qualquer transação, ou seja, valores de transação acima de zero, são indícios que o usuário fez compras influenciado por aquela oferta. Assim, divide-se duas classes:

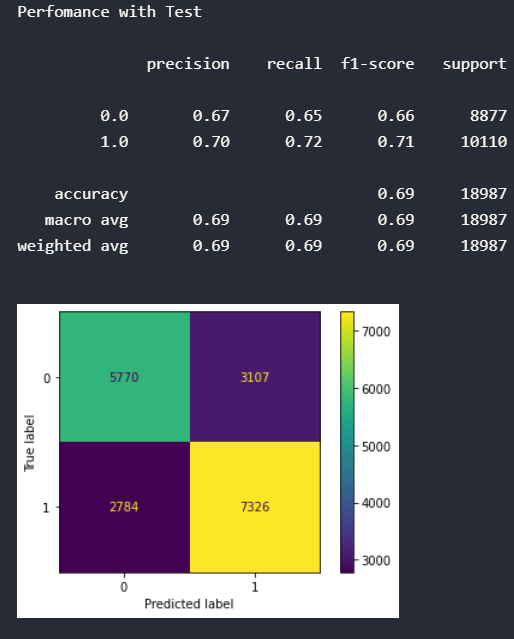
* 0: não houve transações. Tra\_offer\_infl == 0.
* 1: houve alguma transação. Tra\_offer\_infl > 0.

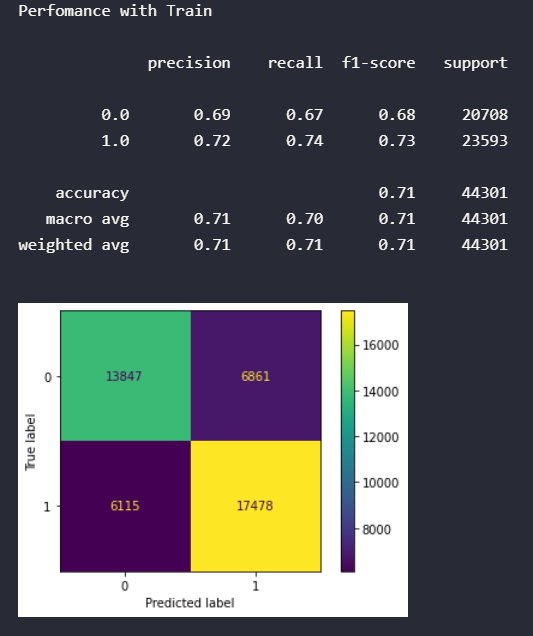
O código abaixo coloca essa regra no dataset.



Com isso, usa-se as mesmas etapas anteriores são seguidas para criar o modelo de classificação. Depois de criado, usa-se GridSearch para procurar o melhor parâmetro para a profundidade.



O resultado desse modelo pode ser visto abaixo



Um resultado parecido com o modelo para classificação da taxa de completadas. Novamente, o ponto aqui é talvez criar um modelo mais generalista, que conseguirá se encaixar em mais de uma situação, aquém do que foi treinado.

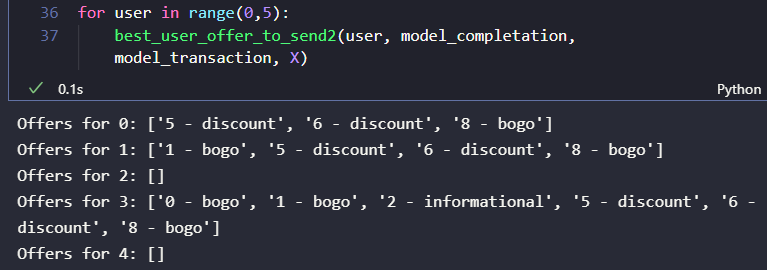
## Combinando os dois modelos

Com o segundo modelo criado, pode-se combinar ambos para gerar melhores orientações para envio de ofertas. A ideia é enviar ofertas que são previstas de serem completas pelo usuário e também possuirão alguma transação feita por ele. A função best\_user\_offer\_to\_send2() implementa isso.



A novidade nessa função é a criação da coluna transaction\_pred. Com ela, o modelo de transação é utilizado, prevendo se aquele usuário com o seu perfil irá transacionar cada oferta. Além disso, a coluna score multiplica as colunas de previsão. Com isso, valores de 1 indicam que tanto o usuário vai completar quanto vai transacionar. O 0 pode indicar que o usuário é previsto completar, mas não transacionar ou transacionar sem completar a oferta.

Aplicando essa função para os 5 primeiros usuários novamente



Nota-se que para o usuário id==0, usando o primeiro modelo, todas as ofertas seriam cabíveis de serem enviadas. Com a combinação dos dois modelos, reduziu-se para as ofertas 5, 6 e 8. Isso significa que, pelo modelo, somente essas três ofertas tem previsão de também transacionar, além de serem completas.