Poli Júnior & Total Pass

Construção de um modelo de Machine Learning para a previsão de Churn Rate

Bernardo Asztalos Teixeira
bernardo.teixeira@polijunior.com.br

Pedro Lucas Nakandakari Mascarenhas
pedro.nakandakari@polijunior.com.br

Tomaz Pineda do Amaral Gurgel
tomaz.gurgel@polijunior.com.br

André Caserta de Castro Helena
andre.helena@polijunior.com.br

Gerente Comercial: Henrique Gouveia Aguiar
Gerente de Projetos: Mateus Matzkin Gurza
Período do Projeto: 18/12/2023 a 03/05/2024

Sumário

Resumo

O projeto Poli Júnior & Total Pass visa à criação de um modelo preditivo de churn, a fim de entender melhor os motivos que levam a uma rotatividade dos clientes. A metodologia utilizada no processo pode ser resumida em:

1. Definição do problema
2. Coleta de dados
3. Limpeza e preparação dos dados
4. Engenharia de Features
5. Segmentação dos clientes
6. Otimização do regressor
7. Análise dos resultados

Ao fim do projeto, obteve-se um regressor que utiliza um método segmentado e outro não segmentado de previsão de churn. A equipe da Poli Júnior buscou fazer um código computacionalmente otimizado, de modo a permitir que a Total Pass conseguisse executar frequentemente a análise realizada. Com isso, a equipe de marketing e de vendas possuirá um relatório automatizado para sugerir ações data driven.

1. Introdução 3

1.1 Entregas do Projeto

As entregas do projeto serão realizadas em dois tipos de formato: Jupyter Notebooks e PDF. Estas podem ser encontradas em https://drive.google.com/drive/folders/1UYzDJZRai-5fr6G8USkP4EDk0nGGjafn?usp=sharing.

- **'RelatorioPoliJunior.pdf'**: este relatório, que detalha as metodologias utilizadas e as conclusões obtidas no projeto.
- **'Clusterizacao.ipynb**: relata o passo a passo realizado durante a segmentação dos clientes.
- **'AnaliseClusterizacao.ipynb'**: explica a segmentação realizada e realiza análises pertinentes para uso durante a otimização do regressor.
- **'Tabelas Poli Júnior'**: Tabelas feitas pela Poli Júnior, responsáveis pelo treinamento do clusterizador.
- **'Regressor.ipynb'**: este é o notebook central do projeto, responsável por armazenar todo o pipeline do projeto, estando inclusa a clusterização. Pode ser executado periodicamente para a obtenção de relatórios atualizados.
- **'AnaliseRegressor.ipynb'**: possui análises relevantes sobre o desempenho do regressor.

1.2 Bases Utilizadas

Além das bases fornecidas pela Total Pass, alguns arquivos foram criados pela equipe da Poli Júnior para o funcionamento adequado do clusterizador e do regressor. Assim, o projeto utitiza dois tipos de base:

- Bases Total Pass: fornecidas no início do projeto, devem ser atualizadas pela equipe Total Pass a cada execução do regressor ou clusterizador.
- Bases Poli Júnior: criadas pela Poli Júnior, são estáticas e não devem ser atualizadas com o tempo.

Para o regressor, são usadas as seguintes bases:

- Bases Total Pass: daily_token_validation, employees, gyms, freezes, subscriptions, subscriptions_feature_dataset.
- Bases Poli Júnior: features, tendencia_de_utilizacao, df_cluster.

Para o clusterizador, são usadas as seguintes bases:

- Bases Total Pass: subscriptions_feature_dataset.
- Bases Poli Júnior: features, tendencia_de_utilizacao.

2. Engenharia de Features

Neste tópico, serão apresentadas as funções utilizadas para o preparo de cada uma das features empregadas no clusterizador e no regressor. Algumas dessas features são originais da Total Pass, enquanto outras foram criadas pela Poli Junior com os dados fornecidos.

2.1 Temporais

2.1.1 Age

Função: calcular_age
Inputs: CUT_OFF_DATE
Outputs: Dataframe com idade de cada usuário para o dia atual
Como foi obtida?
Conversão da coluna que contém a data de nascimento do usuário para o formato date time e após isso calculo simples da idade dele tomando em consideração o dia em que o código está sendo rodado.

2.1.2 Tendência de Utilização

Função: calcular_tendencia_utilizaçao
Bibliotecas necessárias: futures, tqdm, pymannkendall
Inputs: CUT_OFF_DATE
Outputs: Dataframe com as métricas de tendência de utilização
Como foi obtida?

- Importação das bibliotecas: Inicialmente, foram importadas das bibliotecas, ThreadPoolExecutor, tqdm e original_test.
- Pré-processamento: Foram realizadas algumas alterações na base fornecida daily_token_validation, alterando os tipos de dado, de float para datetime nas colunas de data. Além disso, foi removida uma coluna sem dados e duas colunas novas foram adicionadas: mês e ano. Como queremos ver todos os dados anteriores à data limite, um novo dataframe "df1"foi criado apenas com esse período de tempo.
- Agrupamento dos Dados: Primeiramente, os dados são agrupados por employee_id, year e month usando a função groupby do Pandas. O método size é utilizado para contar o número de registros para cada combinação dessas variáveis, e esses contagens são armazenadas na coluna num_utilizacoes.
- Criação de Tabela Pivô: Após o agrupamento, os dados são reorganizados em uma tabela pivô utilizando o método pivot, com employee_id

como índice e combinações de year e month como colunas. Os valores são preenchidos com num_utilizacoes e qualquer valor ausente é substituído por zero (fillna(0)).

- Cálculo da Variação: Uma nova coluna chamada variacao é adicionada à tabela pivô. Ela é calculada utilizando o método diff ao longo das colunas (axis=1), que computa a diferença entre cada coluna e sua precedente para mostrar a variação mensal no número de utilizações. Qualquer valor ausente após a diferenciação é substituído por zero, e as diferenças de todas as colunas são somadas para cada linha.

- Definição e Processamento Concorrente: Uma função process_row é definida para processar cada linha da tabela pivô usando uma função hipotética original_test, que retorna várias estatísticas e tendências baseadas nos dados. O processamento é feito de forma concorrente utilizando ThreadPoolExecutor, que permite executar múltiplas instâncias da função process_row em paralelo. Cada resultado é coletado e monitorado com a ajuda do tqdm, que fornece uma barra de progresso.

- Compilação dos Resultados: Os resultados de cada execução paralela são coletados em uma lista e convertidos em um DataFrame resultados_df. Uma coluna (valor_p) é removida e as colunas são renomeadas para refletir seus conteúdos mais claramente.

- Retorno do DataFrame Final: O DataFrame final é retornado, contendo employee_id, informações sobre tendência, valores p e estatísticas normalizadas para cada funcionário.

2.1.3 Status da Academia

Função: calcular_gym_status

Inputs: CUT_OFF _DATE

Outputs: Dataframe com o status da academia mais utilizada pelo cliente.

Como foi obtida?

Código agrupa pelo employee_ID e gym_ID, contando o número de vezes do uso de cada academia. Ao selecionar a mais utilizada, verifica seu status.

2.1.4 Modalidade Preferida

Função: calcular_pref_modality_utilizaçoes

Inputs: CUT_OFF _DATE

Outputs: Dataframe com a modalidade mais utilizada de cada usuário.

Como foi obtida? Inicialmente, as tabelas gyms e daily token validation foram agrupadas. A partir disso, foi feito um agrupamento por employee ID e

main modality, com o intuito de identificar, para cada usuário, a modalidade preferida na academia em que frequenta.

2.1.5 Número de Academias Distintas

Função: calcular_num_distinct_gyms

Inputs: CUT_OFF _DATE

Outputs: Dataframe com o número de academias distintas utilizadas por um mesmo usuário.

Como foi obtida? Da tabela daily token validation, os employee id e gym id foram agrupados, para contar o número de academias distintas utilizadas por um mesmo usuário.

2.1.6 Usos por Semana

Função: calcular_uses_per_week

Inputs: CUT_OFF _DATE

Outputs: Dataframe com a média de utilizações semanal dos usuários.

Como foi obtida? Da tabela daily token validation, usou-se a coluna de validated at minute para criar duas novas colunas, a de ano e semana. Assim, agruparam-se os employee ID, ano e semana, e foi calculada uma média semanal de utilizações para cada usuário.

2.1.7 Horário Mais Frequente

Função: calcular_most_frequent_hour

Inputs: CUT_OFF _DATE

Outputs: Dataframe com o horário mais frequente de utilização de cada usuário.

Como foi obtida? Da tabela daily token validation, usou-se a coluna de validated at minute para verificar o horário em que o cliente mais utilizou a academia.

2.2 Atemporais

2.2.1 Payment Source

Foi obtida a partir da coluna payment _source, da tabela subscriptions fornecida.

2.2.2 Type Mapped

Função: type_mapped

Outputs: Variável binária do tipo do usuário, sendo 0 caso Holder e 1 caso Relative.

Como foi obtida?

A variável foi obtida a partir da leitura e transformação dos dados da coluna type da base employees_id

2.2.3 Gender Mapped

Função: gender_mapped

Outputs: Variável binária com o gênero do usuário, sendo 0 caso masculino e 1 caso feminino

Como foi obtida? Foi obtida a partir da leitura da coluna gender da base employees e pela substituição dos valores para variáveis binárias

2.2.4 Binário Fee

Função: Binario_fee

Outputs: Variável binaria sendo 1 caso o plano que o usuário esteja contratando exija o pagamento de uma taxa e 0 caso contrário.

Como foi obtida? Feature foi obtida a partir da substituição dos valores da coluna fixed_fee_value pelos valores binários

2.2.5 Fee

Função: Fee

Outputs: Valor da taxa paga pelo usuário

Como foi obtida? Renomeação da coluna $\text{fixed}_{f}ee_{v}alue$

2.2.6 Num Gyms Within Radius

Função: NumGymsWithinRadius

Outputs: Número de academias que estão num raio de 5 kilometros do endereço do usuário

Como foi obtida? Foi obtida a partir da verificação de se os valores de latitude e longitude das academias estavam dentro da região considerada, isto é, se a latitude estava dentro do interevalo da latitude do endereço do cliente somados 5 kilometros assim como decrescidos 5 kilometros. O mesmo foi realizado para a longitude. Após determinar caso de facto a latitude e longitude estavam nesse intervalo, foi então efetivamente verificada se a distância das duas tuplas era menor ou igual a 5 kilometros. A feature foi executada de tal maneira para requerir menos processamento computacional.

2.2.7 Num Gyms Near Company

Função: NumGymsNearCompany

Outputs: Número de academias que estão num raio de 5 kilometros do endereço da empresa

Como foi obtida? Foi obtida a partir da verificação de se os valores de latitude e longitude das academias estavam dentro da região considerada, isto é, se a latitude estava dentro do interevalo da latitude do endereço da empresa somados 5 kilometros assim como decrescidos 5 kilometros. O mesmo foi realizado para a longitude. Após determinar caso de facto a latitude e longitude estavam nesse intervalo, foi então efetivamente verificada se a distância das duas tuplas era menor ou igual a 5 kilometros. A feature foi executada de tal maneira para requerir menos processamento computacional.

2.2.8 Distância Cliente Empresa

Função: distancia_empresa

Outputs: DIstancia entre o endereço da empresa e o endereço do cliente

Como foi obtida? A feature foi obtida a partir do processamento de dados das coluna de latitude e longitude tanto da empresa quanto do cliente utilizando a biblioteca geopy.

3. Clusterização

Uma hipótese da equipe da Poli Júnior presume que a segmentação de clientes permite uma aderência maior do modelo preditivo aos dados, por meio da criação de um algoritmo para cada cluster. Para tal, foram utilizadas as bases 'subscriptions_features_dataset', 'tendencias_de_utilizacao' e 'features'.

3.1 Processamento dos Dados

Primeiramente, um pré-processamento é realizado. Esse processo consiste essencialmente em transformar variáveis categóricas em numéricas e tratar valores nulos. Depois disso, realiza-se uma amostragem de 16.000 clientes, a fim de reduzir o tempo computacional do algoritmo K-Means, sem afetar significativamente o resultado.

Além disso, a normalização dos dados é uma etapa necessária para a clusterização, uma vez que as diferentes escalas das features poderiam gerar um falso viés de distância entre os dados, isto é, variáveis de escalas maiores seriam consideradas mais distantes do que de escalas menores. Para tal, o algoritmo MinMaxScaler da biblioteca Scikit-Learn foi utilizado. Essa função realiza a seguinte transformação nos dados:

$$X_{scaled} = \frac{X - X_{min}}{X_{max} - X_{min}} \tag{3.1}$$

Para a análise de quais features são mais importantes, realizou-se uma Análise das Componentes Principais (PCA), um algoritmo responsável por realizar uma redução de dimensionalidade nos dados. Isso ocorre por meio da criação de features resultantes de uma combinação linear das features inciais, de modo a preservar a maior quantidade de informação dos dados originais com uma quantidade menor de variáveis. Com isso, é possível verificar as categorias mais e menos influentes nessa criação. A Figura 3.1 revela que aproximadamente 93% das variância do dataset original pode ser explicado por 9 combinações lineares das features originais.

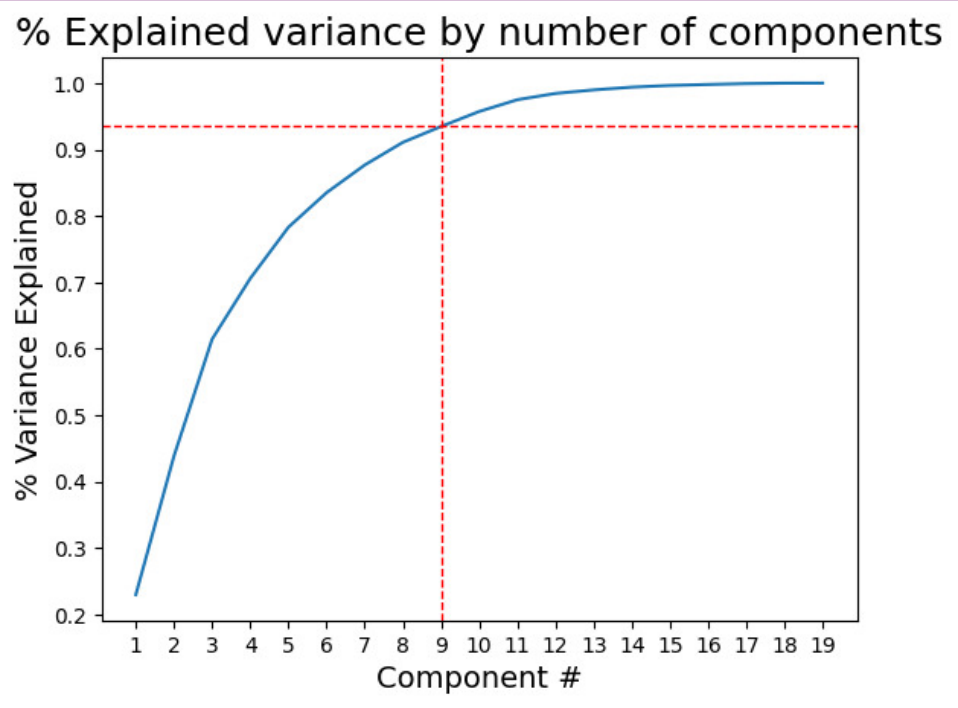

Figura 3.1: Variância Explicada em função das componentes principais.

A Tabela 3.1 representa a combinação linear das nove componentes principais. A primeira, na linha 0, é formada por 0.513 * primeira feature - 0.005 * segunda feature + 0785 * terceira feature, etc. Apenas as primeiras 6 features foram representadas..

	0	1	2	3	4	5
0	0.513	-0.005	0.785	-0.018	-0.009	-0.002
1	-0.300	-0.004	-0.201	-0.021	-0.008	-0.003
2	0.774	-0.011	-0.578	-0.010	-0.012	-0.005
3	-0.195	0.048	0.058	0.042	0.007	0.008
4	0.015	0.005	0.006	0.994	0.003	-0.002
5	0.050	0.022	-0.032	0.004	0.007	-0.009
6	0.017	0.062	-0.015	-0.021	-0.004	-0.001
7	0.076	0.010	-0.032	0.060	0.033	0.024
8	0.002	0.044	0.003	-0.047	0.012	-0.003
9	-0.007	-0.037	0.053	-0.041	0.035	0.005

Tabela 3.1: Combinação linear das 9 componentes principais dos dados iniciais (apenas 6 colunas de features iniciais foram colocadas para melhor apresentação visual).

Essa análise permite concluir que a feature 0 e 3 são mais relevantes do

que as outras, uma vez que estas aparecem significativamente em muitas das 9 componentes principais. Para a seleção de features realizadas, considerou-se que variáveis que possuírem uma influência maior que 0.05 em alguma das nove componentes principais serão consideradas. Com isso, obteve-se o seguinte resultado:

- Variáveis consideradas: Gender Mapped, Type Mapped, Pref Modality, Num Gyms Near Company, Payment Source, Uses per Week, Upgrade, Num Gyms Within Radius, Most Frequent Hour, Age, Distancia Cliente Empresa, Normalized Stats (tendências de utilização), Utilizações.
- Variáveis desconsideradas: Count Subscriptions, Numero de IDs Distintos, Binario Fee, Fee, Gym Status, Num Distinct Gyms.

Com isso, 6 das 19 variáveis foram descartadas.

3.2 K-Means

Finalmente, é possível aplicar um algoritmo de clusterização. O escolhido pela equipe da Poli Júnior foi o K-Means, enquanto a principal métrica analisada foi o **Silhouette Score**, que mede a distância intra e inter-cluster a partir do SWC (silhueta média sobre todos os objetos):

$$SWC = \frac{1}{N}\sum_{i=1}^{N} s(i) \tag{3.2}$$

onde

$$s(i) = \frac{b(i) - a(i)}{max(a(i), b(i))} \tag{3.3}$$

Nessa equação, a(i) equivale a distância média do objeto i para as outras instâncias do mesmo cluster (distância média intra-cluster) e b é a distância média do objeto i para as instâncias do cluster mais próximo, excetuando o cluster da própria instância (distância inter-cluster).

O coeficiente de Silhouette pode variar entre -1 e +1. Um valor próximo a +1 significa que a instância está dentro de seu próprio cluster e distante de outros clusters, indicando uma boa clusterização. Já coeficientes próximos de 0 indicam que o objeto está na fronteira entre clusters, revelando um agrupamento duvidoso. Por fim, valores próximos de -1 indicam que a instância pode ter sido atribuída ao cluster errado.

A Figura 3.10 exibe a avaliação do Silhouette Score para a clusterização realizada. Com isso, concluí-se que o número ideal de clusters para a segmentação é 8, por possuir maior coeficiente.

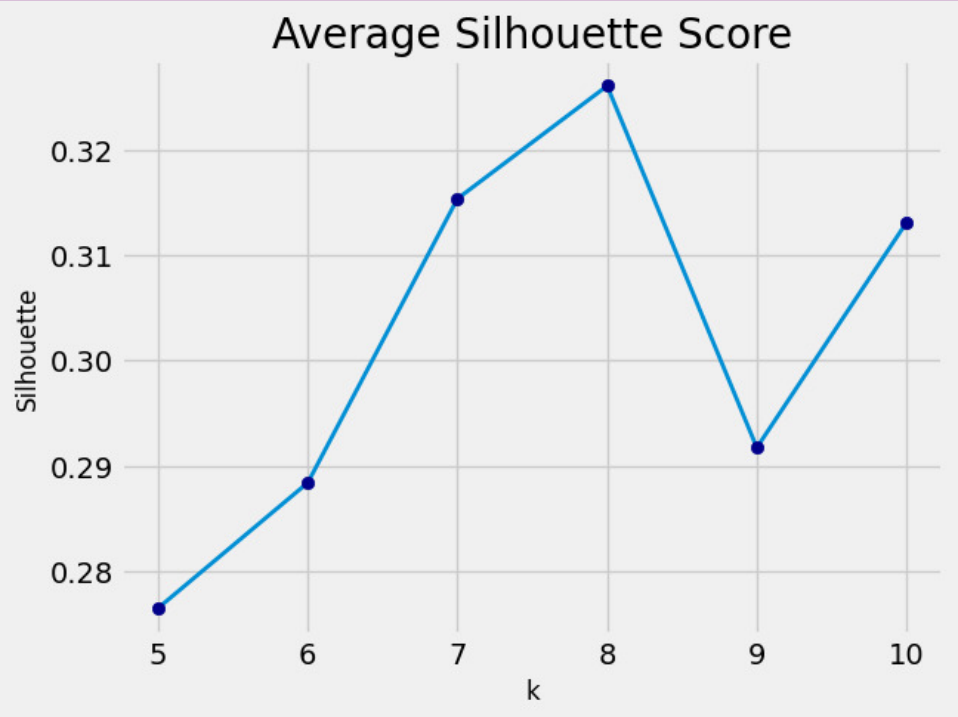

Figura 3.2: Coeficiente de Silhouette por número de clusters (k).

Uma maneira de observar se a clusterização foi bem sucedida é observar o gráfico TSNE, também da biblioteca Scikit-Learn. Este algoritmo busca a melhor projeção bidimensional de um espaço hiperdimensional. Nesse caso, a Figura 3.3 exibe a melhor projeção encontrada para o espaço de 13 dimensões criado pelas 13 features.

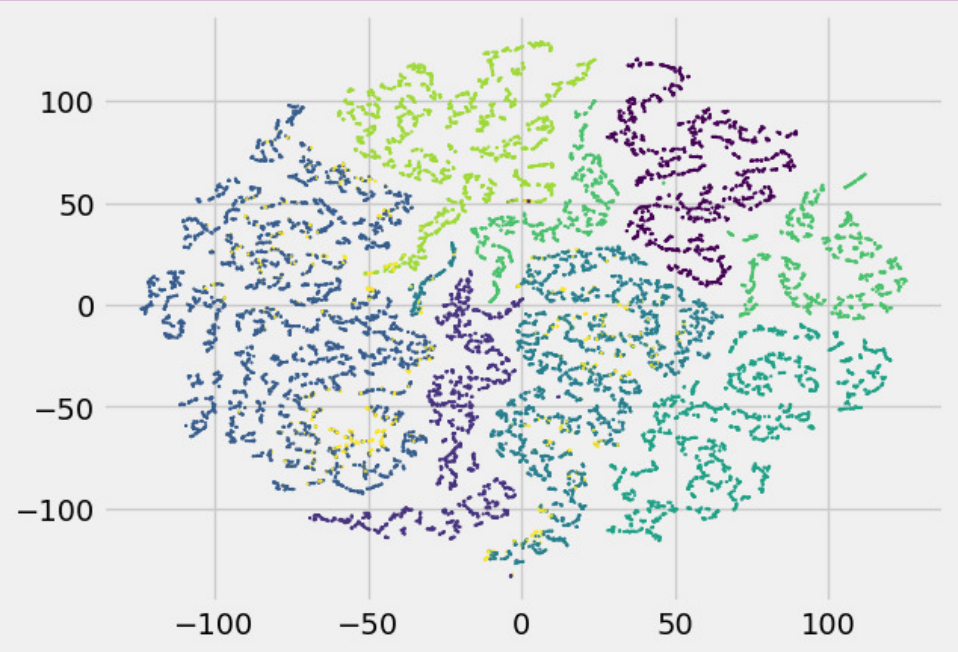

Figura 3.3: Melhor projeção bidimensional encontrada pelo TSNE. Escalas arbitrárias.

3.3 Análise dos Clusters

Inicialmente, realizou-se um cálculo das estatísticas descritivas para as variáveis numéricas de cada cluster, para possibilitar uma visão ampla de como estas variáveis estão sendo distribuídas. Dessa forma, foram calculadas a média, o desvio padrão, o mínimo, o máximo, a mediana, o primeiro e o terceiro quartis. Para méritos de uma melhor visualização, também foram plotados os gráficos mais adequados para a visualização de cada uma das features numéricas.

- Idade

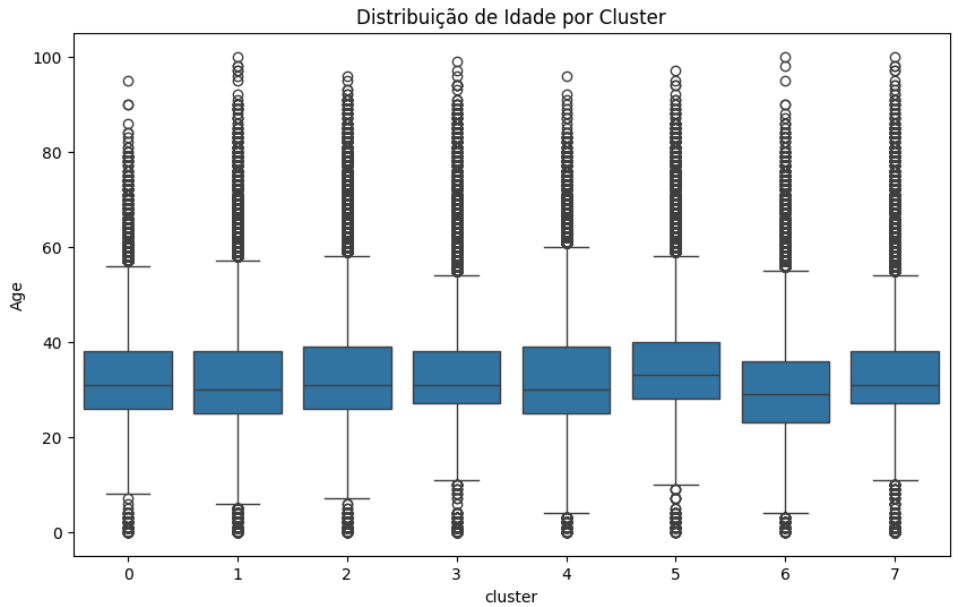

Figura 3.4: Boxplot das idades por cluster

Para as idades, têm se que a variação é muito discreta. Dessa forma, conclui-se que não foi um fator tão decisivo para a separação dos clientes em clusters.

- Método de pagamento

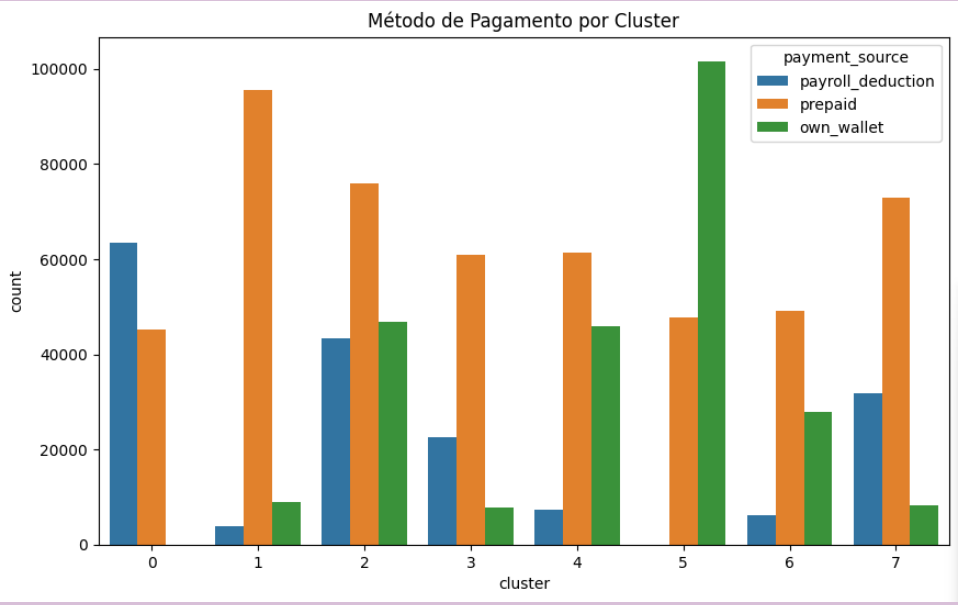

Figura 3.5: Número de clientes por método de pagamento por cluster

Em contrapartida, para os métodos de pagamento nota-se uma diferença muito clara ao realizarmos comparações entre os clusters. Pode-se notar, por exemplo, que houve uma concentração de clientes que realizam o pagamento com o próprio dinheiro no cluster 5, e quase nenhum cliente com o status de payment_deduction.

- Gênero

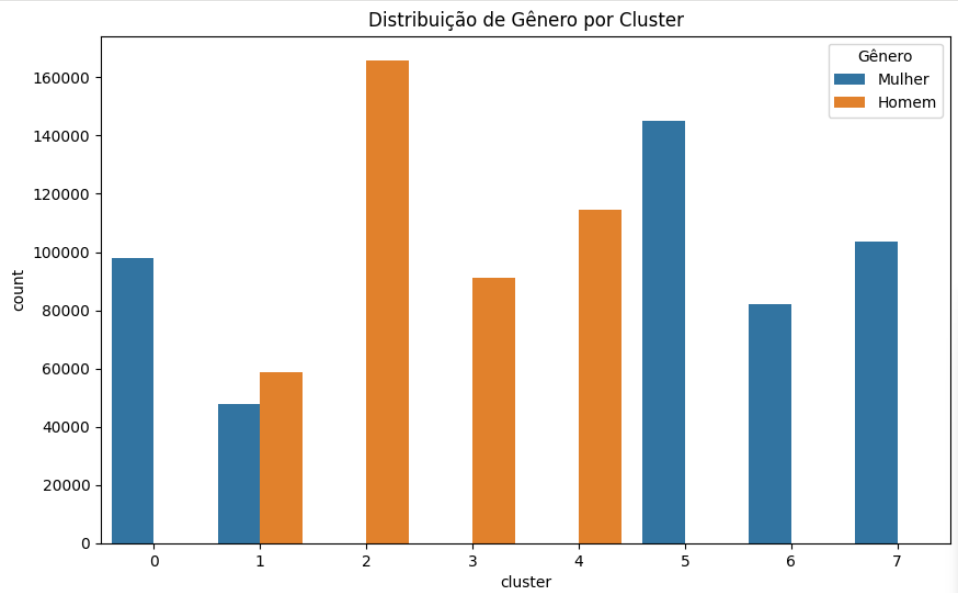

Figura 3.6: Distribuição de gênero por cluster

Assim como na seção anterior, o gênero parece ser um critério decisivo importante, tendo em vista a presença de apenas um cluster com uma distribuição evidentemente balanceada de gêneros, quandp em todos os outros 7 há uma concentração extrema de apenas um gênero.

- Número de academias em um raio de 5km

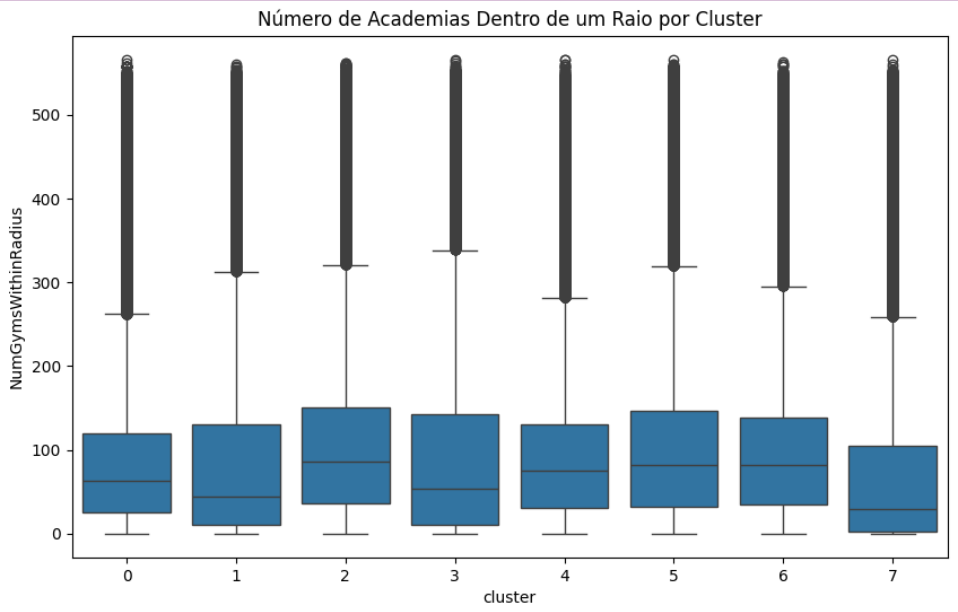

Figura 3.7: Número de academias dentro de um raio de 5 quilômetros do cliente por cluster

Para esta feature, há também uma distribuição discreta, sem uma diferença gritante entre os clusters.

- Número de academias próximas à empresa

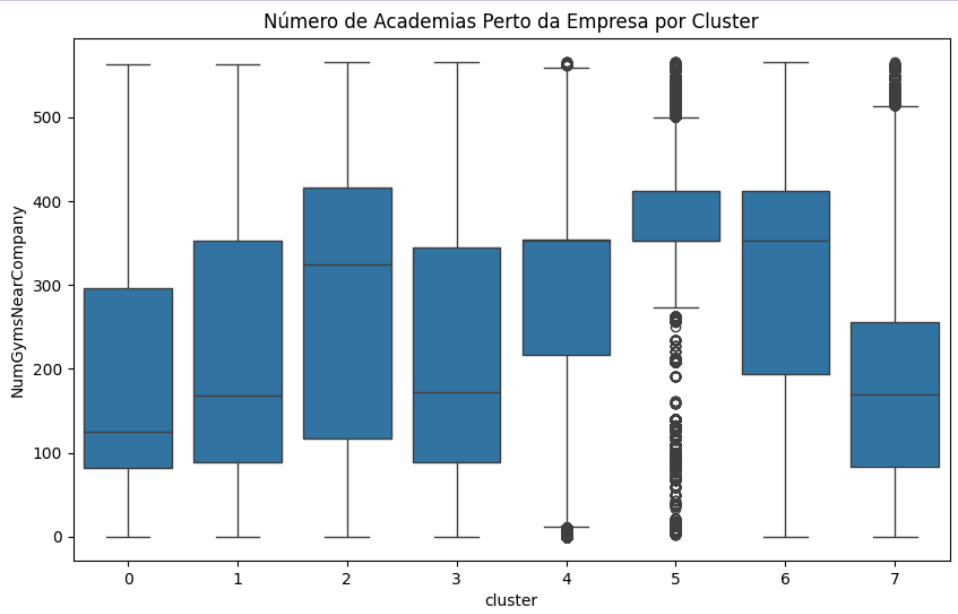

Figura 3.8: Número de academias próximas à empresa em que o cliente trabalha

Aqui, vemos mais uma vez uma métrica em que, embora seja menos balanceada que a anterior, não parece afetar tanto na distribuição.

- Upgrade

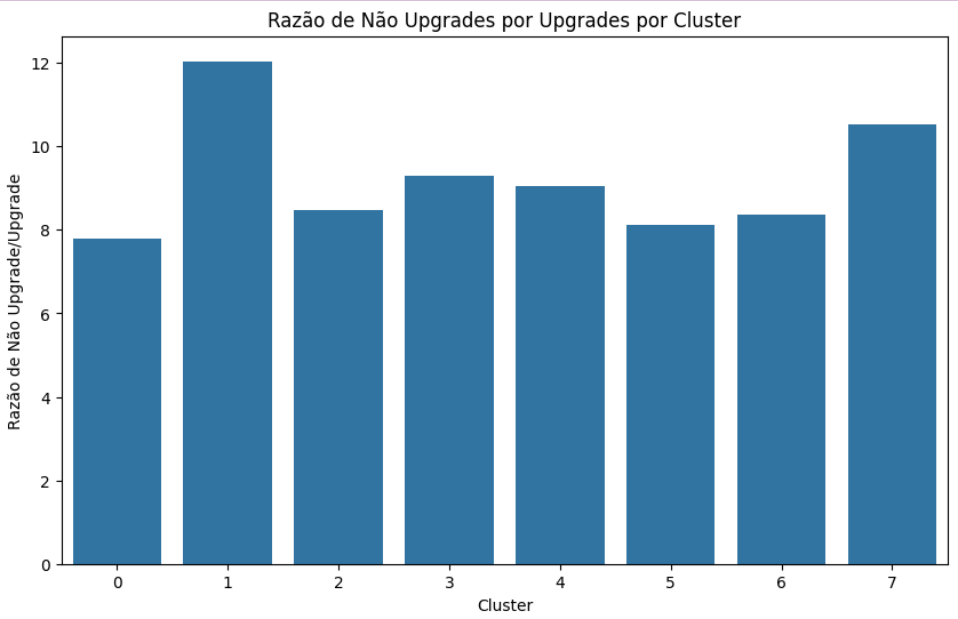

Figura 3.9: Razão entre número de clientes que realizaram e não realizaram upgrade

Novamente, com exceção dos clusters 1 e 7 que apresentam uma razão maior em comparação com as demais, o resultado parece ser balanceado.

- Status da assinatura

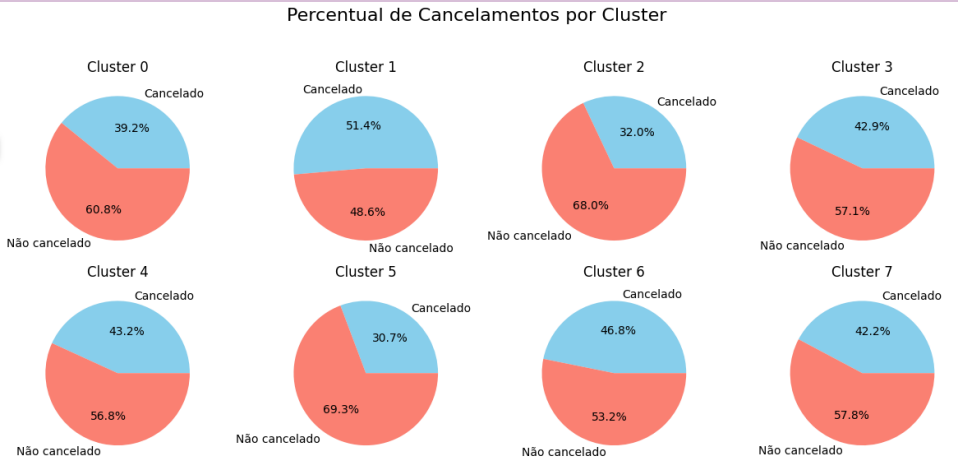

Figura 3.10: Gráficos de pizza para o status da assinatura em cada cluster

Por último, vizualizamos as distribuições percentuais entre clientes com planos cancelados e ativos, uma das métricas mais relevantes. Embora

não haja diferenças gritantes, podemos notar um maior número de cancelados nos clusters 1 e 6, assim como um menor número nos clusters 2 e 5. Como há distinções visíveis entre os clusters, consideramos que este foi um fator que influenciou a separação dos clientes.

3.3.1 Conclusão

Com base nas análises, ficou claro que o algoritmo tomou as métricas de método de pagamento, gênero e meio de pagamento como as mais decisivas ao realizar a distribuição dos clientes em clusters. Embora não tão evidentes, todas as mínimas diferenças entre cada um dos clusters, quando levamos em conta as outras features, não devem ser desprezadas. O intuito desta análise é apenas ter uma visão um pouco mais ampla de como essa clusterização foi feita. Finalmente, pode-se concluir que a clusterização foi balanceada, tendo em vista os tamanhos de cada cluster, e a uniformidade na maioria das métricas.

4. Regressor de Churn

O principal objetivo do projeto é a modelagem de um algoritmo para previsão da porcentagem de churn para clientes ativos do Total Pass. Pensando nas ações estratégicas que podem ser tomadas, privilegiou-se aumentar a precisão para clientes mais propensos a cancelar, isto é, obter a maior porcentagem de acertos em clientes caracterizados por alta porcentagem da chance de churn.

A próxima Seção introduz o conceito teórico que guia o funcionamento do regressor.

4.1 Introdução Teórica

Primeiramente, vale ressaltar o problema enfrentado. O Total Pass deseja, a partir de dados anteriores ao presente, prever o comportamento de seus clientes ativos em um período de tempo, definido como dt. Ilustrando melhor, espera-se predizer como a linha representada pela Figura 4.1 se comporta no período entre hoje e hoje + 30 dias, no caso em que $dt = 30$.

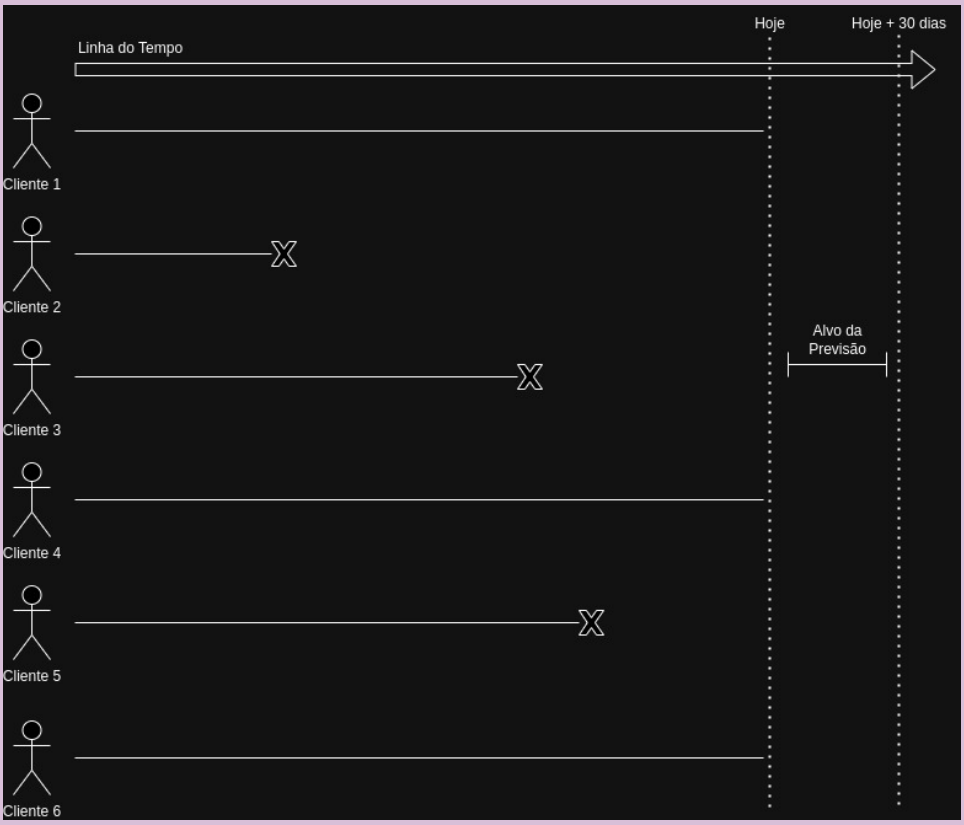

Figura 4.1: Alvo da previsão do regressor: período entre hoje e hoje + dt dias.

No entanto, o Aprendizado de Máquina é impossível de ser aplicado nessa situação, haja vista a necessidade de dados de treino e de teste para a criação do modelo. Portanto, a estratégia que foi implementada foi treinar o algoritmo de Machine Learning em uma data anterior ao presente, definida por *cut off date*, que dividirá a base de dados em treino e teste.

A Figura 4.2 apresenta esse cenário. Observa-se que todo período antes de 'Data' será considerado dado de treino, enquanto o período entre 'Data' e 'Data + 30 dias' comporá os dados de teste. Para que isso seja adequadamente realizado, é necessário recalcular todas as features em relação ao Cut Off Date, momento para o qual a inferência será realizada. Além disso, clientes como o cliente 2, que cancelou antes do Cut Off Date, ou clientes que criaram o plano depois do Cut Off Date serão desconsiderados, isto é, apenas os clientes ativos nesta data serão utilizados para o treinamento do modelo.

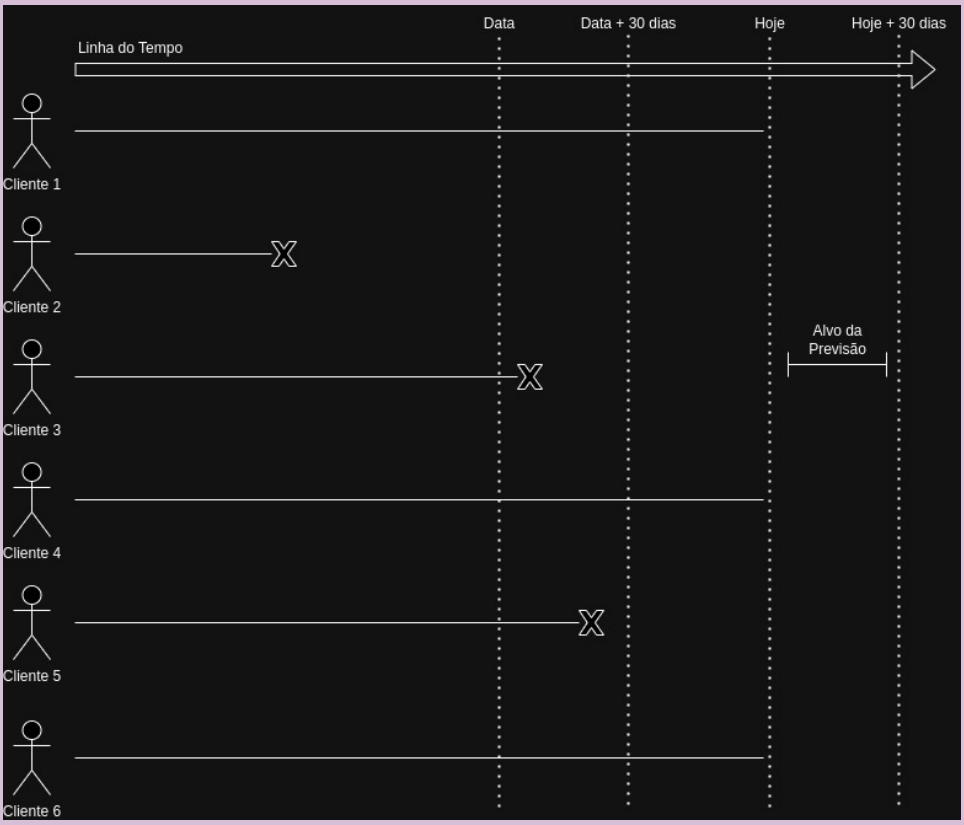

Figura 4.2: Alvo da previsão do regressor: período entre hoje e hoje + *dt* dias.

Após o treinamento realizado no Cut Off Date, é preciso novamente recalcular as features considerando o momento presente e realizar a inferência no momento presente. A Seção seguinte destaca detalhes sobre o pipeline,

enfatizando a divisão entre Dataframe de treino e Dataframe de inferência.

4.2 Pipeline Geral

O pipeline do regressor é ilustrado na Figura 4.3. Nela, as entradas do modelo estão em verde e compreendem os seguintes itens:

- Data: data atual, necessária para o estabelecimento do Cut Off date.
- Filepaths: caminhos das Tabelas Poli Júnior e Total Pass da máquina na qual o código será executado.
- Intervalo dt: compreende as datas para as quais deseja-se uma previsão. Por exemplo, dt = [100,200,300] realizará a previsão do cliente cancelar o plano em 100, 200 e 300 dias.

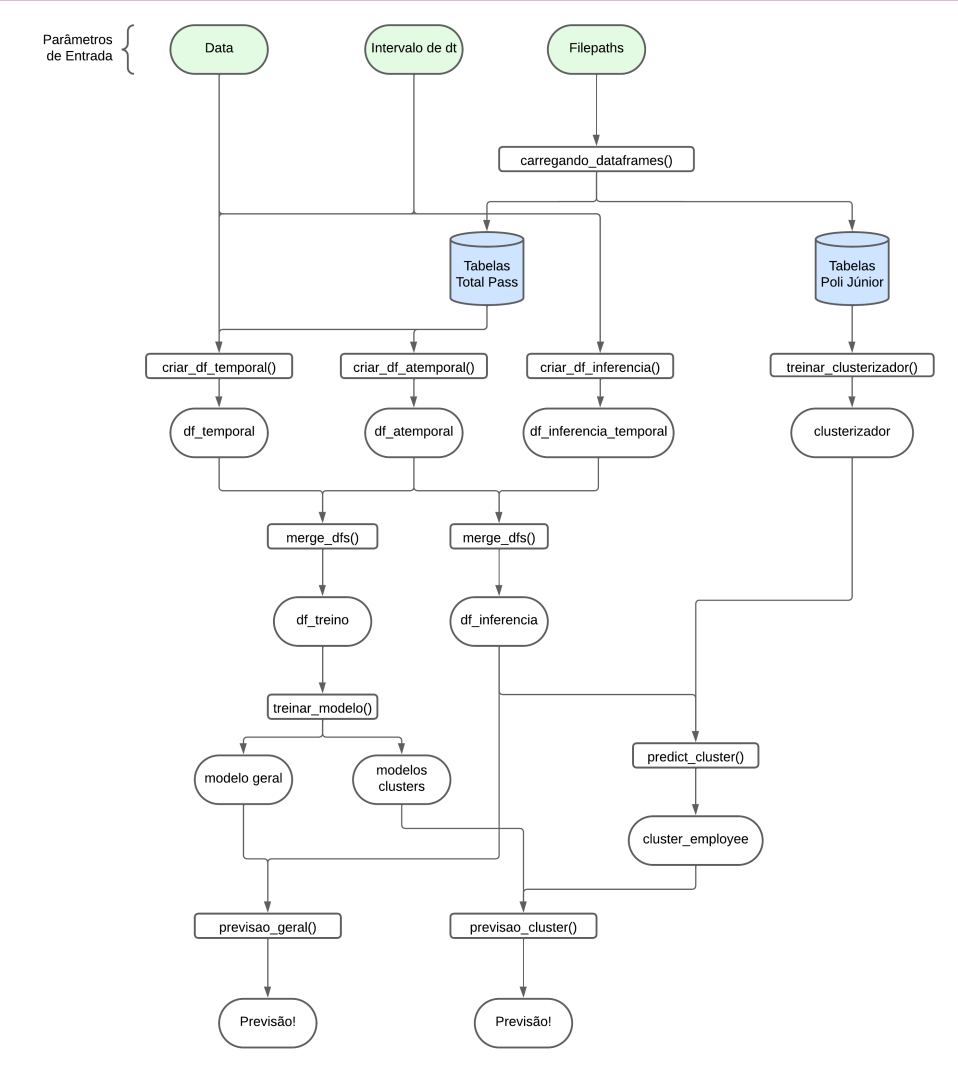

Figura 4.3: Pipeline do Regressor. As entradas do modelo estão em verde e os conjuntos de dados em azul.

A partir dos dados de entrada, três dataframes são criados:

- Df Temporal: abrange todas as features temporais, isto é, que dependem da data atual para serem calculadas. É utilizado no treinamento dos modelos.
- Df Atemporal: abrange todas as features atemporais, isto é, que são imutáveis. É utilizado no treinamento dos modelos.
- Df Inferência: Diferentemente dos outros dataframes, este dataframe possui apenas os dados de teste, ou seja, possui também a feature de predição.

Os dataframes temporal e atemporal são unidos em um só, chamado Df Treino. Este é responsável por abastecer de dados a função "treinar modelo". Com isso, o modelo geral e o modelo clusters são criados.

Por fim, o Df Inferência pode ser utilizado na função "previsão geral"para realizar a previsão. Além disso, este mesmo dataframe também é utilizado na função "predict cluster", para predizer o cluster de cada instância e selecionar o modelo selecionado para prever os dados daquele cluster. Com isso, as duas previsões são geradas e exibidas no Notebook.

4.3 Modelo Geral

Um dos principais entraves para uma performance satisfatória do modelo é a assimetria de classes na base de dados. Isto é, para alguns períodos amostrais 'dt', o número de instâncias que não cancelaram o plano é significativamente maior do que as que cancelaram, dificultando a capacidade de reconhecer padrões do algoritmo de Machine Learning. As principais técnicas utilizados no mercado para lidar com esse problema são Oversampling, Undersampling e SMOTE:

- Oversampling (Sobreamostragem): No oversampling, a ideia é aumentar o número de instâncias da classe minoritária, ou seja, da classe que possui menos exemplos na base de dados. Isso é feito criando novas instâncias sintéticas da classe minoritária, por técnicas como interpolação ou duplicação.
- Undersampling (Subamostragem): No undersampling, o objetivo é reduzir o número de instâncias da classe majoritária, ou seja, da classe que possui mais exemplos na base de dados. Isso é feito removendo aleatoriamente exemplos da classe majoritária até que a distribuição das classes esteja mais equilibrada. Embora seja uma abordagem mais simples, ela pode levar à perda de informações importantes presentes nos dados removidos.
- SMOTE (Synthetic Minority Over-sampling Technique): O SMOTE é uma técnica de sobreamostragem que cria instâncias sintéticas da classe minoritária ao sintetizar exemplos que estão localizados entre instâncias já existentes da classe minoritária. Isso é feito calculando as diferenças entre instâncias minoritárias próximas e multiplicando-as por um valor aleatório entre 0 e 1, que é adicionado à instância minoritária original para criar uma nova instância.

Para o projeto, foram testadas as 3 técnicas, além do treinamento sem nenhuma técnica. As figuras 4.4 e 4.5 exibem os resultados para um intervalo de tempo de 90 dias e de 270 dias.

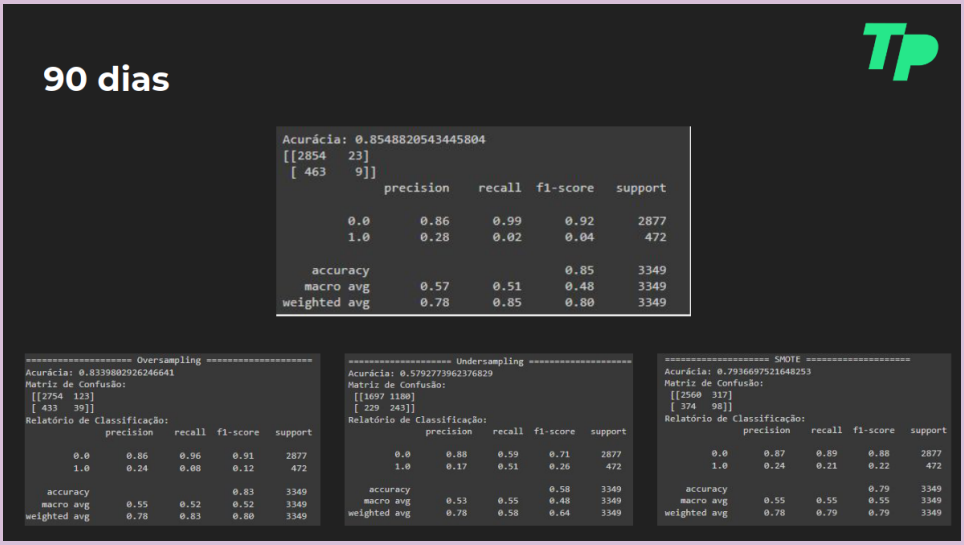

Figura 4.4: Report de Classificação para o Modelo Geral em um intervalo de 90 dias.

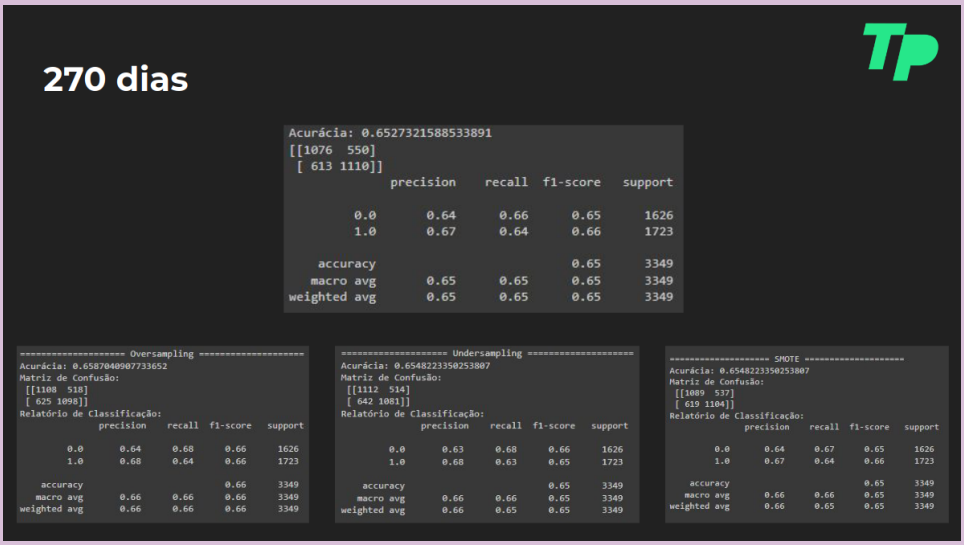

Figura 4.5: Report de Classificação para o Modelo Geral em um intervalo de 270 dias.

Pelos reports de classificação para esses e outros intervalos de tempo, concluiu-se que o modelo proposto não se adequou bem à assimetria de classe presente para baixos 'dt'. Isso pode ser evidenciado pelo baixíssimo recall positivo exibido na Figura 4.4. Esse valor de 0.02 indica que, sem métodos de desbalanceamento de classe, o modelo se torna extremamente conservador e só arrisca em afirmar que um cliente cancelou em 2% dos clientes totais que cancelaram. Portanto, observando um recall mais balanceado com o método Undersampling, este foi escolhido para ser a técnica utilizada para intervalos de tempo menor do que 100 dias.

Para os demais intervalos, os balanceamentos de classe não foram efetivos, aumentando de maneira irrisória a acurácia, vide Figura 4.5. Para garantir a robustez do modelo, nenhum método será implementado para intervalos de tempo maior do que 100 dias.

Além disso, a normalização dos dados é uma etapa necessária também para o regressor, uma vez que as diferentes escalas das features poderiam gerar um falso viés de distância entre os dados, isto é, variáveis de escalas maiores seriam consideradas mais distantes do que de escalas menores. Para tal, o algoritmo MinMaxScaler da biblioteca Scikit-Learn foi utilizado. Essa função realiza a seguinte transformação nos dados:

$$X_{scaled} = \frac{X - X_{min}}{X_{max} - X_{min}} \tag{4.1}$$

Por fim, detalhes do modelo de Machine Learning escolhidos serão explicitados. A principal exigência do modelo escolhido é a existência de uma função que preveja a probabilidade de uma classificação, e não apenas um valor booleano. Isso era necessário para o funcionamento do projeto, uma vez que o objetivo não era maximizar os acertos dentro dos 50% clientes mais suscetíveis ao cancelamento, mas era a obtenção de uma lista descendente com a probabilidade dos clientes cancelarem. Com isso, seria possível direcionar uma política de retenção diretamente aos clientes que possuem 90% ou 80% de chance de cancelamento.

Após um teste com muitas algoritmos, o *RandomForestClassifier* da biblioteca Scikit-Learn mostrou-se mais performático. Esse resultado era esperado, uma vez que esse classificador é o mais utilizado na indústria, especialmente em problemas complexos de muitas features, como este projeto.

No algoritmo RandomForestClassifier, o problema é tratado como uma árvore de decisão. A ideia é dividir os dados em diferentes grupos em cada "nó"da árvore, com base em certos critérios. Neste projeto, usamos o critério de Gini, que avalia a impureza dos grupos criados pela divisão dos dados. Durante o treinamento, várias árvores de decisão são criadas com diferentes partes dos dados, e cada uma delas contribui para a decisão final. Todos os outros parâmetros foram mantidos como 'default' e podem ser consultados na documentação da biblioteca Scikit Learn.

4.4 Modelo Clusterizado

Para o modelo clusterizado, os mesmos processos do modelo geral foram realizados. No entanto, o modelo é abastecido apenas com dados dos clientes que pertencem ao cluster treinado. Com isso, é possível realizar um treinamento mais específico. Vale ressaltar que, antes deste processo, é necessário realizar a inferência do clusterizador em todos os dados do dataframe de treinamento, a fim de criar uma tabela com os clientes e os respectivos clusters.

4.5 Análise dos Modelos

Dada a assimetria de classes inerente ao projeto, a análise do modelo deve ser feita por métricas específicas, como o recall e a precisão. Primeiramente, os clientes são classificados em 4 grupos, definidos em função de sua classe - churn ou não churn - e a previsão realizada pelo modelo:

- Verdadeiros positivos: Clientes que cancelaram e o modelo previu cancelamento. Nessa situação, a Total Pass teria a possibilidade de evitar o cancelamento.
- Falsos positivos: Clientes que não cancelaram e o modelo previu cancelamento. Nessa situação, a Total Pass perderia capital, atuando sobre um cliente que não cancelaria.
- Verdadeiros negativos: Clientes que não cancelaram e o modelo previu não cancelamento. Nessa situação, a Total Pass não gasta capital desnecessariamente.
- Falsos negativos: Clientes que cancelaram e o modelo previu não cancelamento. Nessa situação, a Total Pass perdeu a chance de investir na retenção desses clientes.

Portanto, o cenário ideal é maximizar os verdadeiros positivos e verdadeiros negativos, minimizando os falsos positivos e falsos negativos, para garantir uma abordagem eficaz na retenção de clientes e na otimização do capital da Total Pass. As métricas de recall e precisão são definidas em função dessas 4 categorias, e podem ser definidas por:

$$recall = \frac{verdadeiros\ positivos}{verdadeiros\ positivos + falsos\ negativos} \tag{4.2}$$

$$precisao = \frac{verdadeiros\ positivos}{verdadeiros\ positivos + falsos\ positivos} \tag{4.3}$$

Como a maior parte das instâncias é da classe negativa (não churn), é natural o algoritmo tender a classificar todas as instâncias como negativas. Para evitar esse viés, analisa-se o recall, que mede a capacidade do modelo

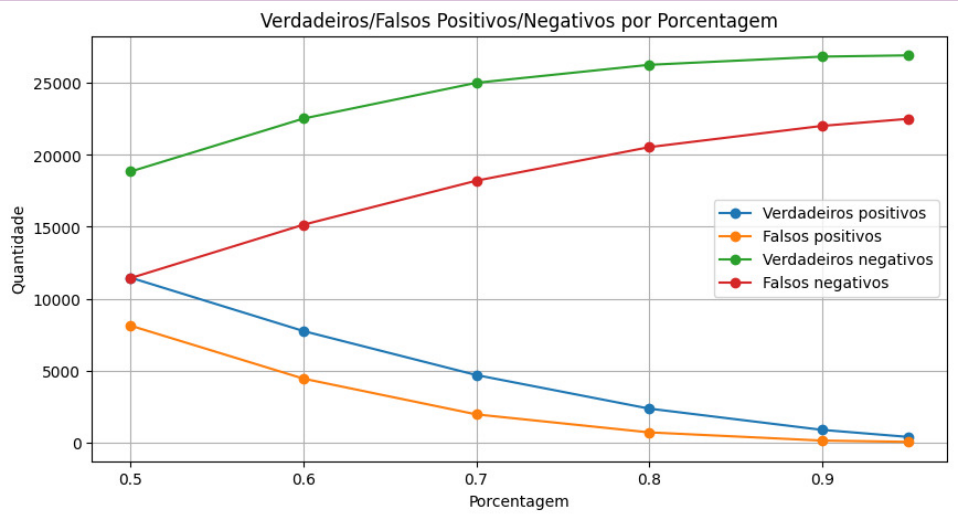

Figura 4.6: Quantidade de clientes por categoria para cada porcentagem de confiança em um período de 180 dias.

se arriscar a predizer uma instância positiva. Um recall alto significa que os clientes que cancelaram foram bem identificados, e que a quantidade de falsos negativos foi baixa.

Já a precisão é importante para analisar quão assertivo foi o modelo em prever corretamente as instâncias que ele classificou como positivas. Uma precisão alta indica que o número de falsos positivos foi baixo e que as instâncias classificadas como positiva eram majoritariamente positivas. Dessa forma, ficará evidente o constante tradeoff entre precisão e recall, haja vista que, na medida em que o modelo se arrisca mais, ele aumenta seu recall e diminui sua precisão.

A Figura 4.6 indica a quantidade de clientes por categoria para cada porcentagem de confiança em um período de 180 dias. Esse índice é responsável por evidenciar a certeza do modelo no momento de predição de uma instância e é utilizado de modo que clientes serão classificados como positivos para a porcentagem 0.8 se o modelo acredita que a instância possui 80% ou mais de chance de churn. Desse modo, a quantidade de predições positivas diminui a medida em que aumentamos a porcentagem de confiança do modelo.

Na medida em que se aumenta a porcentagem de confiança, nota-se a crescente métrica da precisão, evidenciada na Figura 4.7. No entanto, o tradeoff entre precisão e recall é representado na Figura 4.8, uma vez que o recall - quantidade de verdadeiros positivos em relação a quantidade de falsos negativos - diminui drasticamente quando aumenta-se o índice de confiança do modelo. Ou seja, quanto maior a certeza do modelo, maior a precisão e menor a quantidade de instâncias em que o modelo se arrisca a realizar uma predição positiva.

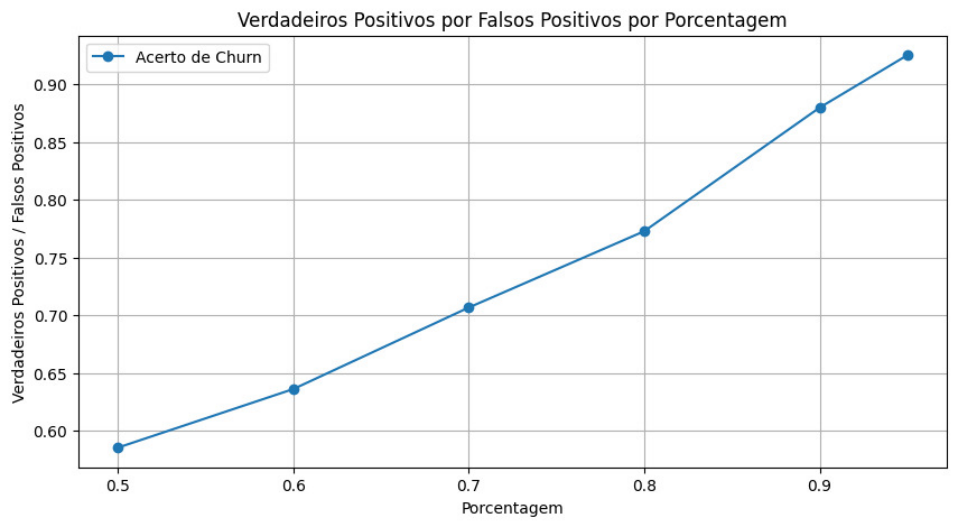

Figura 4.7: Precisão para cada porcentagem de confiança em um período de 180 dias.

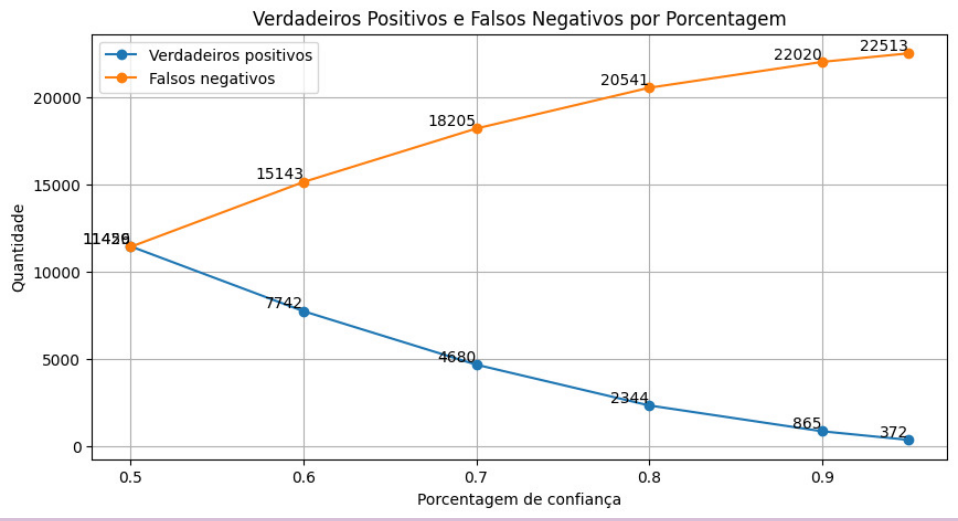

Figura 4.8: Relação entre Verdadeiros Positivos e Falsos Negativos por porcentagem de confiança em um período de 180 dias.

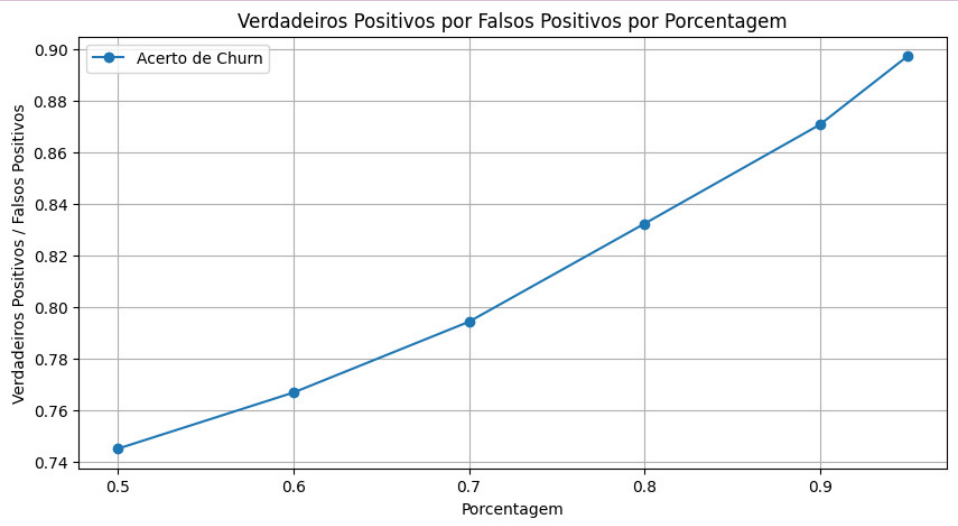

Figura 4.9: Precisão para cada porcentagem de confiança em um período de 365 dias.

Além disso, vale ressaltar que os modelos com intervalos de tempo maiores foram os mais bem sucedidos, haja vista a maior quantidade de dados para reconhecimento de padrões e menor assimetria entre as classes. Para um dt de 365, por exemplo, obteve-se uma precisão de aproximadamente 83% e um recall de mais de 70%! Assim, uma ação de retenção da Total Pass sobre esse público atingiria mais de 70% das pessoas que cancelariam e apenas 17% das pessoas contactadas não teriam cancelado (falsos positivos). Esses dados são explicitados nas Figuras 4.9 e 4.10.

Por fim, uma breve análise dos modelos clusterizados permite a conclusão de que os modelos não lidaram bem com a segmentação em uma amostra de dados já desbalanceada. Ao clusterizar os clientes, criam-se agrupamentos mais homogêneos, mas nem sempre mais balanceados. É natural que alguns clusters tornem-se mais balanceados e outros menos. A Figura 4.11 evidencia um cluster balanceado, com um recall alto e que poderia ser bem utilizado pela Total Pass, enquanto a Figura 4.12 mostra o oposto, uma vez que este cluster possui recall baixíssimo e não deve ser utilizado.

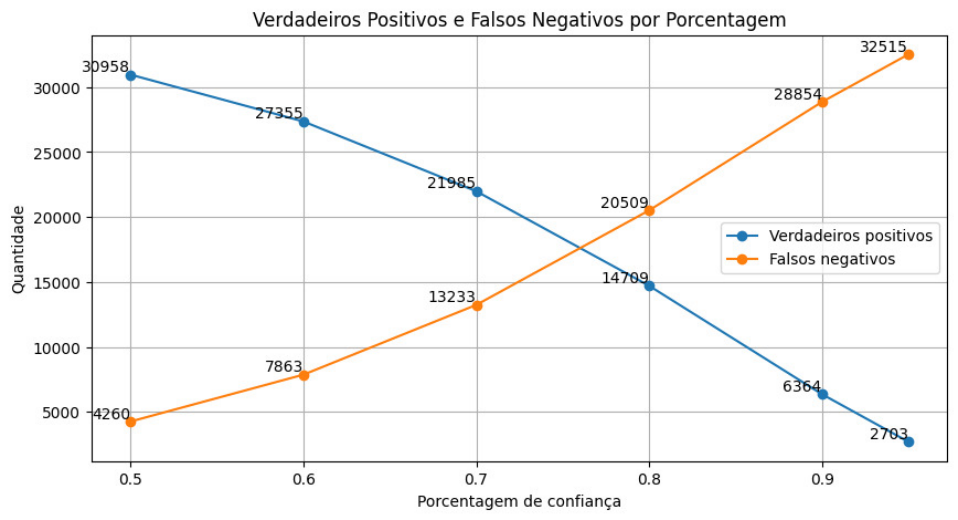

Figura 4.10: Relação entre Verdadeiros Positivos e Falsos Negativos por porcentagem de confiança em um período de 365 dias.

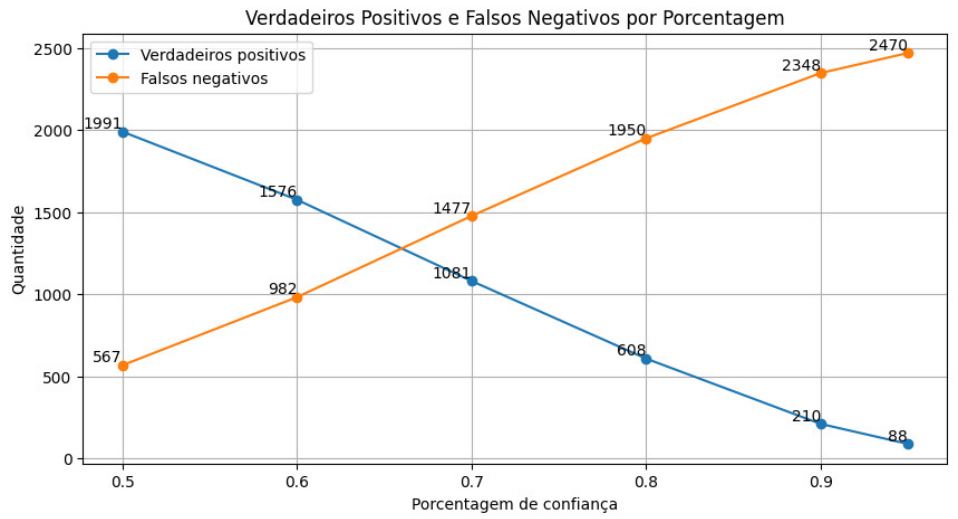

Figura 4.11: Exemplo de cluster com recall satisfatório.

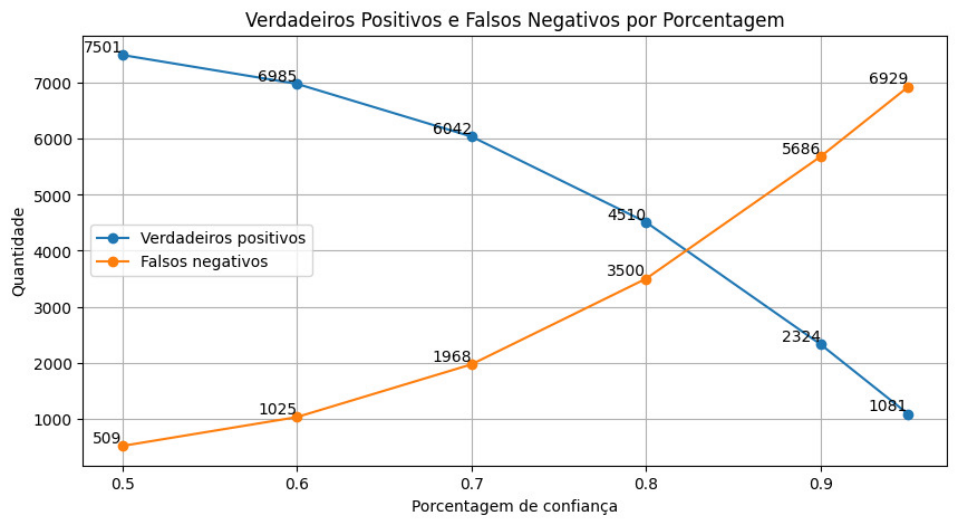

Figura 4.12: Exemplo de cluster com recall instatisfatório.

5. Conclusão

O resultado satisfatório do modelo geral para grandes intervalos de tempo evidenciam a coerência dos dados fornecidos e a capacidade do modelo Random Forest de realizar previsões sobre o padrão de churn dos clientes da Total Pass. No entanto, faz-se necessária uma análise posterior da equipe de retenção sobre os dados, de modo a entender o tradeoff entre precisão e recall e minimizar as perdas com a ação adotada. Isto é, iniciativas mais custosas devem ser direcionadas à clientes que o modelo previu com muita certeza, uma vez que busca-se a maior precisão possível para evitar custos com clientes que não cancelariam. No entanto, essa iniciativa selecionaria poucas pessoas, tornando necessárias ações mais baratas e abrangentes para clientes que o modelo classificou com pouca certeza.