Faculdade de Informática e Administração Paulista - FIAP

Challenge ClickBus - Data Trip

Bruno Marcelo de Rosa	563779
Danilo Oliveira Alves	564109
Fred Henrique Ferreira de Sousa	565725
Luana Fernandes Silva Ferreira	560474
Vinicius de Sousa Macedo	561911

São Paulo

Setembro de 2025

Relatório Técnico – Projeto ClickBus

Tratamento do dataset e Análise Exploratória dos Dados

Utilizamos a biblioteca “gdown” para trazer os dados fornecidos pela ClickBus para o Colab. Após carregar a base, identificamos dados faltantes e o período inicial (12-09-2013) e final (01-04-2024) do dataset, para posteriormente fazer uma análise e entender os tipos de dados e as colunas que tínhamos, além da quantidade.

Identificamos que os dados estavam com hash em sua composição e para que pudéssemos deixá-los mais “amigáveis”, aplicamos um mapeamento nas colunas que julgamos necessárias, fazendo uma concatenação das variáveis de “lugar” para que os mesmos hash’s em diferentes colunas, no momento da transformação pudessem refletir os mesmos lugares sem que houvesse perda de dados. Fizemos o mesmo procedimento para a companhia de ônibus.

Guardamos os dados originais (em hash) em dicionários, para consultas futuras e os removemos do df, para facilitar a visualização, por fim, realizamos a renomeação das variáveis para facilitar a análise exploratória dos dados.

Foi feita uma limpeza inicial dos dados, removendo algumas linhas que consideramos como erros, a fim de retirar possíveis outliers ou dados ruidosos que pudessem afetar nossa análise posteriormente, sendo assim, foram removidos:

- valores negativos em valor do ticket;
- quantidade de linhas onde a origem na ida é igual ao destino na ida;
- linhas onde a origem na ida é diferente do destino na volta;
- linhas que mostraram que 1 ticket gerou passagem de ida e volta.

Além disso, assumimos que quando o cliente compra ida e volta, o destino da volta precisa, necessariamente, ser igual a origem da ida.

Julgamos necessário acrescentar uma variável referente ao COVID-19 antes de toda análise, pois entendemos que a pandemia foi um "choque exógeno" que alterou drasticamente os padrões de viagem. Sem essa variável, o modelo poderia tentar encontrar um padrão linear em um comportamento que foi tudo, menos linear. Ao criar essa feature, nós damos ao modelo um contexto crucial, onde ele poderá aprender que:

- Pré-COVID: Havia um padrão de normalidade com forte queda ao final desse período.
- Durante-COVID: A demanda aumentou.
- Pós-COVID: Houve uma retomada forte, atingindo níveis jamais vistos.

Essa variável não só justifica as quedas, mas também ajuda o modelo a entender a intensidade da retomada e a não penalizar clientes que "sumiram" durante o período. Sendo uma variável bastante importante para segmentação e padrão de comportamento. Assim, utilizamos dados históricos da Organização Pan-Americana da Saúde (OPAS) e da Organização Mundial da Saúde (OMS), para definir a data de início do período “durante-covid”: 11 de março de 2020, quando a COVID-19 foi caracterizada pela OMS como uma pandemia e a de fim: 5 de maio de 2023, quando a OMS declarou o fim da Emergência de Saúde Pública de Importância Internacional (ESPII) referente à COVID-19.

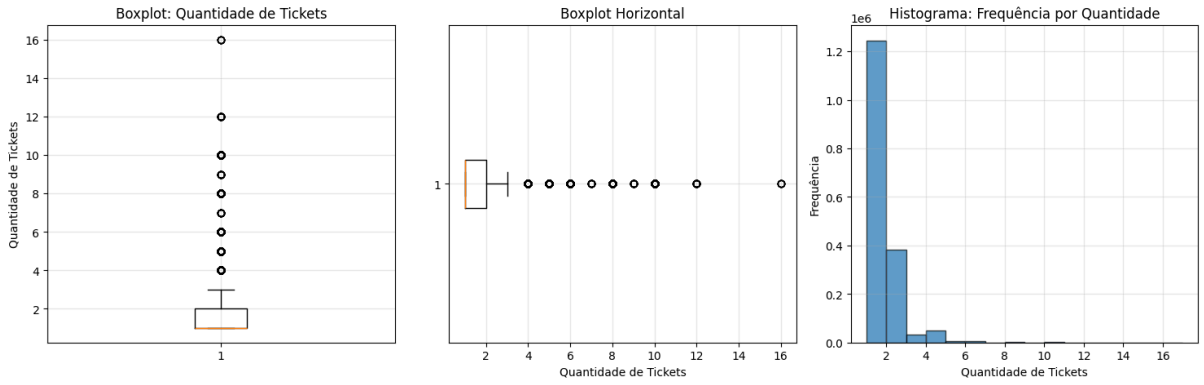
Tomamos a decisão, nesse momento, de já fazer uma divisão dos dados de forma temporal. Essa decisão foi baseada na análise inicial e no conhecimento de negócios de entender que o comportamento de compra é temporal Padrões de consumo, popularidade de trechos e até o efeito de campanhas de marketing mudam ao longo do tempo. Se usássemos uma divisão aleatória (como train_test_split faria por padrão), nosso modelo de treino aprenderia com dados futuros para prever o passado (data leakage ou vazamento de dados). Isso poderia inflar artificialmente as métricas de performance, mas o modelo falharia miseravelmente na produção, pois nunca viu dados tão recentes. Após isso, acrescentamos a variável “split” para puxarmos, quando fosse necessário, a divisão dos dados no modelo.

Para a análise exploratória dos dados e a criação de novas variáveis para os modelos, baseados em: Cliente e Ticket; Feriado; Variáveis temporais; Categorização do período; Trecho; Empresa; Cliente.

Podendo fazer uma analise de distribuição, de outliers e entendendo o comportamento dos clientes, assim,

- Q1 (25%): 1.0
- Q3 (75%): 4.0
- Outliers (acima de 8.5 tickets): 56448 compras

- Percentual de outliers: 9.81%
- Distribuição de quantidade de tickets:
- 1 tickets: 197,940 compras (34.40%)
- 2 tickets: 142,496 compras (24.77%)
- 3 tickets: 51,729 compras (8.99%)
- 4 tickets: 55,728 compras (9.69%)
- 5 tickets: 22,844 compras (3.97%)
- 6 tickets: 22,875 compras (3.98%)
- 7 tickets: 12,282 compras (2.13%)
- 8 tickets: 13,043 compras (2.27%)
- 9 tickets: 7,744 compras (1.35%)
- 10 tickets: 7,486 compras (1.30%)



Porém por ser uma variável bastante importante principalmente para o nosso processo de clusterização, posteriormente, decidimos não realizar limpeza neles, mas sim, realizar a criação de uma nova variável que pudesse refletir e agrupar todos os outliers em um determinado grupo, criando uma faixa de tickets de 1 até +4, assim, pudemos reduzir a cardinalidade e todos que fizeram compras acima de 4 tickets, entram nessa classe.

Realizamos o input da variável feriado em nosso dataset, separando as informações que tínhamos em ano, mês, dia, dia da semana, fim de semana e realizamos uma análise se a compra havia sido feita em feriado ou não. Tomamos como hipótese que compras feitas em um delta de 5 dias, poderiam estar fortemente relacionado com uma compra para o feriado.

A ideia de escolher um delta = 5 parte de uma hipótese inicial forte, baseada no comportamento típico do viajante. É um ponto de partida padrão na indústria, que

equilibra bem a captura do sinal do evento sem se contaminar com o ruído de dias normais.

A título de conhecimento, realizamos o gasto médio anual de alguns feriados, para que pudéssemos entender uma possível sazonalidade dentre os principais feriados e a quantidade de anos que ocorreram dentro de nossa base (a base disponibilizada para análise apresenta uma janela de dados “quebrada”).

--- 10 feriados por GMV médio anual ---

nome_feriado	gmv_medio_anual	compras_medias_anual	tickets_medios_anual	anos_ocorreu
Universal Fraternization Day	114470.746364	769.636364	1108.363636	11
Christmas Day	113492.540909	744.636364	1071.363636	11
Republic Proclamation Day	73144.582727	487.818182	677.000000	11
Independence Day	71371.887778	477.444444	657.444444	9
Our Lady of Aparecida	61922.174545	430.090909	581.545455	11
Good Friday	60300.246364	429.363636	573.909091	11
All Souls' Day	59294.754545	394.363636	549.818182	11
Tiradentes' Day	50766.876000	357.600000	478.500000	10
Worker's Day	46545.236000	362.500000	459.100000	10

Pudemos observar alguns dados bastante interessantes nesse processo referente a compras realizadas em feriados e em dias normais, apesar da janela de comparação em relação a quantidade de dias.

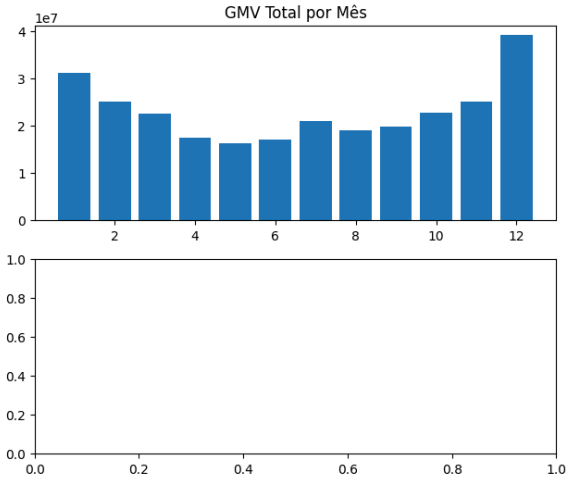
compra_ate_5_dias_feriado	Gasto total	Número de compras	Tickets totais	Dias unicos	Compras por dia	Tickets por dia
Compra dias normais	228,748,089	1,419,105	1,963,591	3205	442.78	612.66
Compra até 5 dias antes do feriado	47,554,121	304,931	429,889	560	544.52	767.66
	Compras em %	Tickets em %	Dias em %	Gasto por dia R$	Gasto em %	
Compra dias normais	82,30	82,00	85,10	71,372.26	82,80	
Compra até 5 dias antes do feriado	17,70	18,00	14,90	84,918.07	17,20	

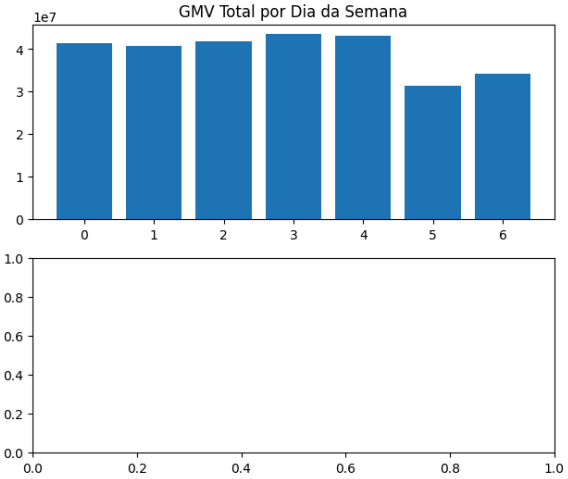
E uma análise em dias comuns, feriado e dentro de uma janela de 5 dias antes do feriado, refletem essas informações.

Resumo Comparativo Detalhado:

tipo_dia	Dia Comum	Janela Pré-Feriado (5d)	Feriado
gmv_total	228,748,088.60	40,629,777.26	6,924,343.61
gmv_medio	161.19	157.71	146.35
tickets_total	1,963,591.00	364,414.00	65,475.00
tickets_medio	1.38	1.41	1.38
n_compras	1,419,105.00	257,618.00	47,313.00

A criação de variáveis mensais e dias da semana foi bastante importante para conseguirmos entender uma sazonalidade e possíveis padrões de compra.





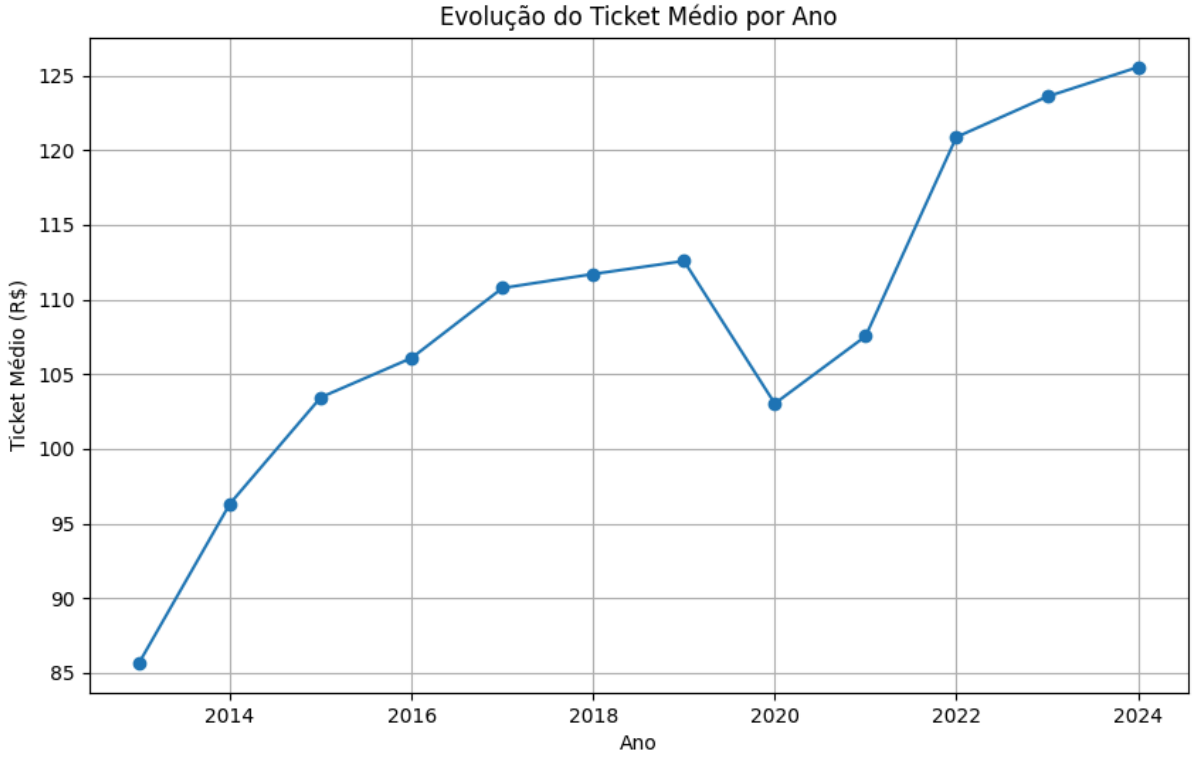
mes	count	mean	std	min	25%	50%	75%	max
1	184389.0	169.327097	174.491246	4.19	63.8400	118.57	209.52	4779.00
2	158719.0	157.969945	161.094573	5.00	61.3000	111.99	194.26	4365.84
3	144946.0	155.817299	159.978945	5.00	59.1825	110.80	192.76	5088.10
4	113201.0	153.859090	156.112478	5.00	59.7000	111.04	189.06	5076.60
5	109744.0	149.084122	152.246618	5.00	57.4700	106.21	184.07	4769.04
6	113582.0	150.535284	151.291536	1.35	58.0400	108.21	186.28	3733.66
7	128178.0	163.295009	164.027174	5.00	64.0025	117.94	201.36	4602.44
8	121457.0	156.797510	157.221704	5.00	61.1800	112.69	193.83	4786.61
9	125480.0	157.302231	159.704839	5.00	61.0200	112.72	193.39	3734.26
10	142357.0	159.262565	161.940113	2.06	61.6700	114.08	195.50	3865.89
11	153490.0	163.298374	174.060549	5.00	60.5300	114.09	199.79	6635.55
12	228493.0	171.103557	180.735861	5.00	64.4400	119.09	210.09	6723.93

dia_semana	count	mean	std	min	25%	50%	75%	max
0	244226.0	169.762900	171.710465	5.00	65.79	121.47	210.4175	4365.84
1	232875.0	174.734634	175.879363	5.00	68.44	125.52	216.6700	6723.93
2	243036.0	172.054857	172.889660	2.06	67.67	123.31	212.4000	5088.10
3	266133.0	163.532724	166.406806	5.00	63.84	116.89	200.6000	4377.17
4	279945.0	154.330780	163.004778	5.00	58.12	107.86	188.8500	6635.55
5	209775.0	149.533745	155.064044	1.35	57.53	106.03	183.5700	5076.60
6	248046.0	138.042208	143.936657	4.19	53.03	95.93	168.5400	5203.21

Estendendo essa análise e integrando com a variável de feriado

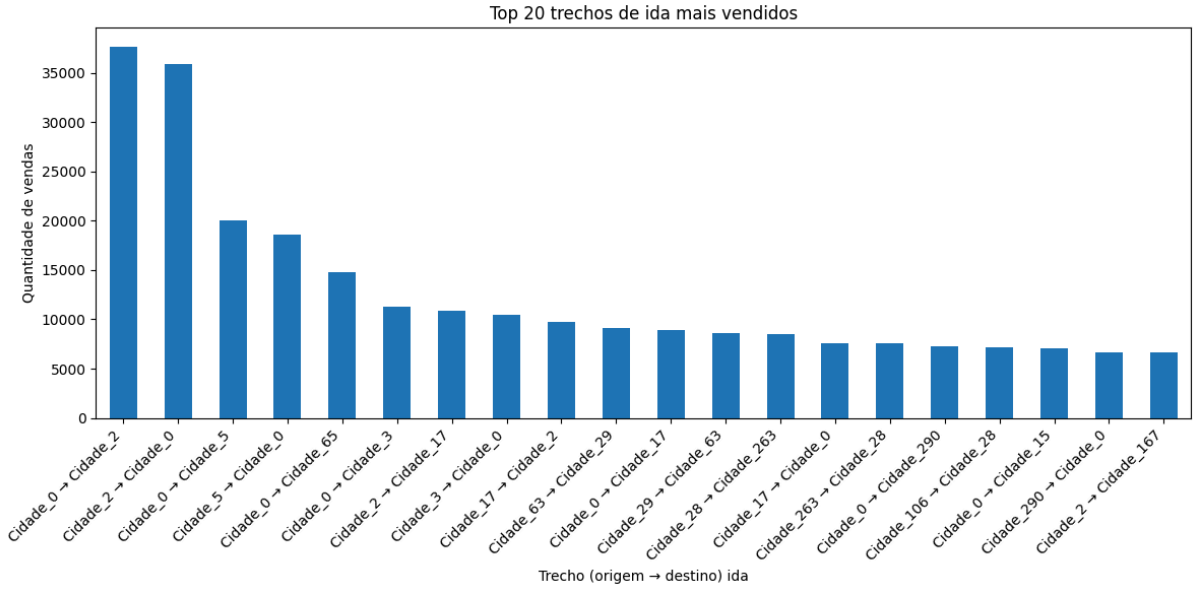
	gmv_success			quantidade_tickets		data_feriado
	sum	mean	std	count	sum	mean	
mes							
1	31,222,054.14	169.33	174.49	184,389.00	262,988.00	1.43	8,466.00
2	25,072,831.65	157.97	161.09	158,719.00	221,790.00	1.40	0.00
3	22,585,094.20	155.82	159.98	144,946.00	196,009.00	1.35	1,380.00
4	17,417,002.84	153.86	156.11	113,201.00	153,082.00	1.35	6,919.00
5	16,361,087.91	149.08	152.25	109,744.00	146,177.00	1.33	3,625.00
6	17,098,098.58	150.54	151.29	113,582.00	153,594.00	1.35	0.00
7	20,930,827.63	163.30	164.03	128,178.00	176,788.00	1.38	0.00
8	19,044,155.12	156.80	157.22	121,457.00	164,707.00	1.36	0.00
9	19,738,283.97	157.30	159.70	125,480.00	172,340.00	1.37	4,297.00
10	22,672,141.03	159.26	161.94	142,357.00	197,342.00	1.39	4,731.00
11	25,064,667.35	163.30	174.06	153,490.00	215,898.00	1.41	9,704.00
12	39,095,965.05	171.10	180.74	228,493.00	332,765.00	1.46	8,191.00

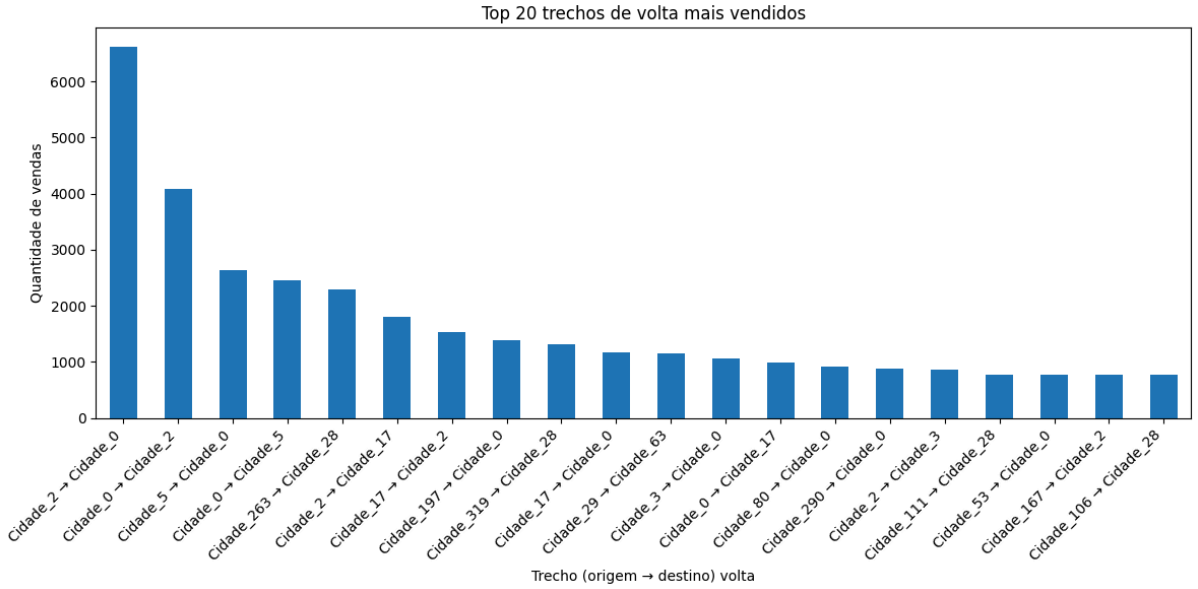
Foi analisado o comportamento do Ticket Médio por ano



Para reduzirmos a cardinalidade da variável de tempo, em relação às horas, aplicamos a estratégia de categorizar em manhã, tarde, noite ou madrugada, assim deixando janelas mais generalizadas de compra.

Para analisar a variável trecho, decidimos agrupar para entender os principais trechos (par ida-volta)





Porcentagem de vendas por trecho de ida (top 10) [%]:

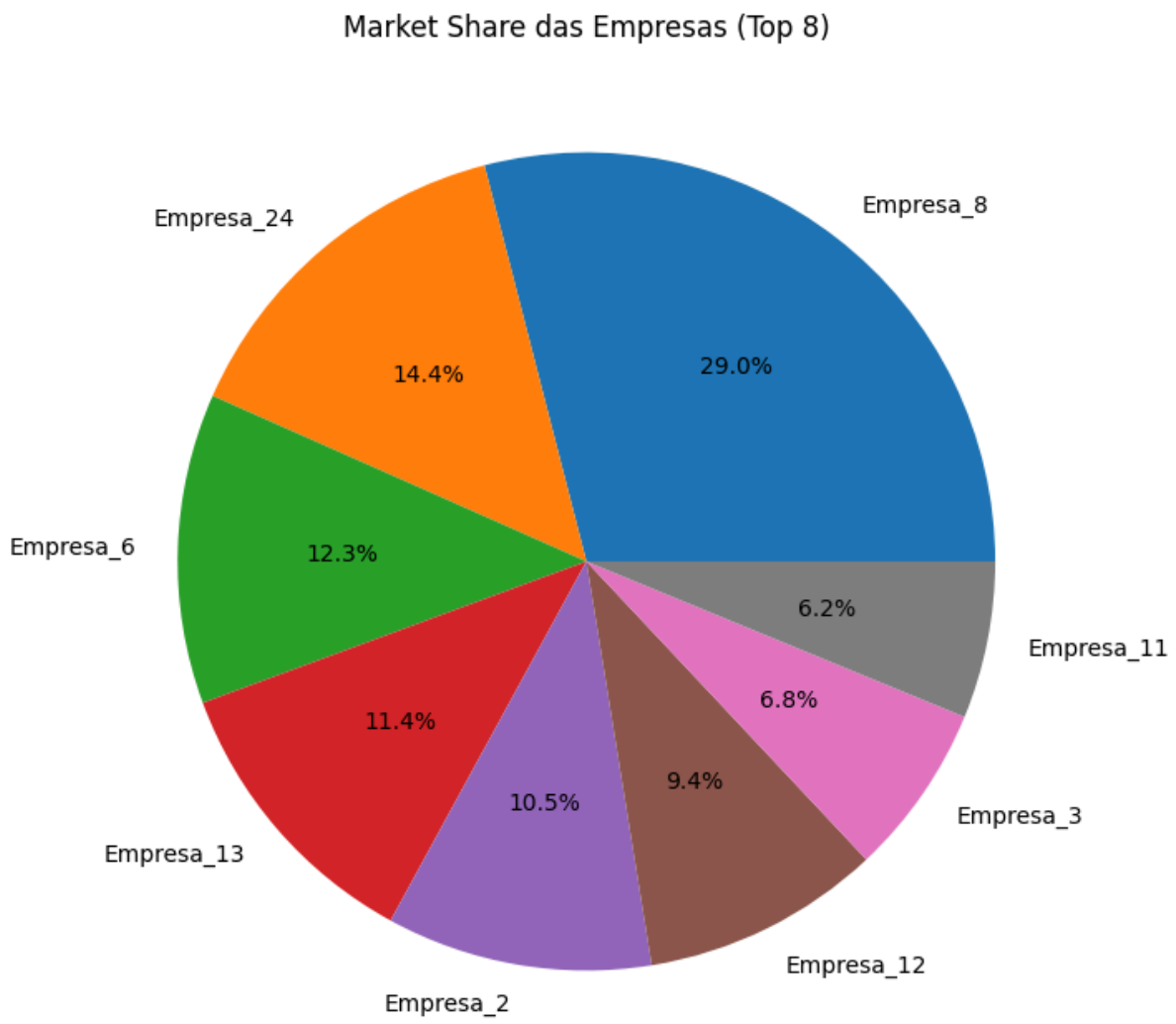
- Cidade_0 → Cidade_2 2.19
- Cidade_2 → Cidade_0 2.08
- Cidade_0 → Cidade_5 1.16

- Cidade_5 → Cidade_0 1.08
- Cidade_0 → Cidade_65 0.86
- Cidade_0 → Cidade_3 0.66
- Cidade_2 → Cidade_17 0.63
- Cidade_3 → Cidade_0 0.61
- Cidade_17 → Cidade_2 0.57
- Cidade_63 → Cidade_29 0.53

Porcentagem de vendas por trecho de volta (top 10) [%]:

- Cidade_2 → Cidade_0 3.79
- Cidade_0 → Cidade_2 2.34
- Cidade_5 → Cidade_0 1.50
- Cidade_0 → Cidade_5 1.41
- Cidade_263 → Cidade_28 1.31
- Cidade_2 → Cidade_17 1.03
- Cidade_17 → Cidade_2 0.87
- Cidade_197 → Cidade_0 0.79
- Cidade_319 → Cidade_28 0.76
- Cidade_17 → Cidade_0 0.6

Para as empresas, criamos associação entre os principais trechos e as empresas dominantes, com isso, também fizemos a criação de uma variável para entender quando o cliente escolhe a mesma empresa para ida e volta e quando ele usa diferentes empresas para esse processo, essa variável é importante para criarmos um score de fidelidade e de sazonalidade.



Para o cliente, o foco foi transformar seu histórico de compras em métricas de lealdade e engajamento, de forma análoga a como no mercado financeiro analisamos o histórico de um investidor para definir seu perfil. Criamos variáveis de fidelidade e recorrência de compras, com destaque para dois indicadores principais:

- cliente_recorrente: Uma flag binária (1 para clientes com 2 ou mais compras) que segmenta de forma imediata a nossa base entre "clientes ativos" e "clientes em experimentação".
- score_fidelidade: Uma métrica mais sofisticada que mede a concentração de gastos de um cliente em sua companhia aérea preferida. Funciona como um índice de concentração de portfólio no mundo dos investimentos: um score próximo de 1.0 indica um cliente "maximalista", com altíssima lealdade a uma marca, enquanto scores mais baixos revelam um cliente com um portfólio de

voos mais diversificado, provavelmente mais sensível a ofertas de concorrentes.

Em conjunto, estas novas variáveis transformam dados transacionais brutos em sinais claros sobre o valor e o comportamento do cliente, servindo para alimentar o modelo de Clusterização.

Clusterização

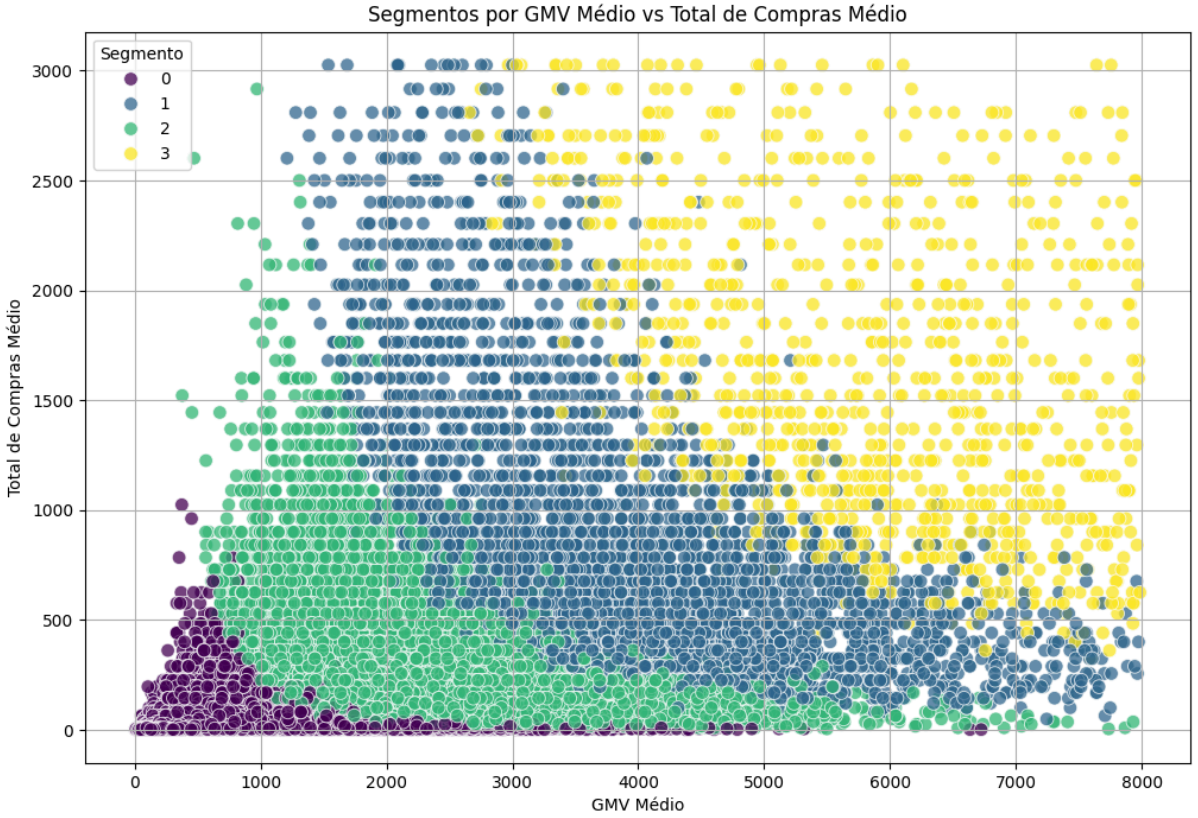
Agregação de Dados: A primeira etapa foi consolidar o histórico de transações em uma visão única por cliente. Para isso, agrupamos os dados por id_cliente e agregamos as features que criamos, utilizando médias para características de comportamento (como score_fidelidade_mean) e somas para métricas de volume (como gmv_success_sum). Isso nos deu um "DNA" financeiro e comportamental para cada cliente.

Pré-processamento e Escalonamento: Algoritmos de clusterização baseados em distância, como o K-Means, são sensíveis à escala das variáveis. Uma feature com valores na casa dos milhares (como valor_acumulado) dominaria uma feature na casa das dezenas (como quantidade_tickets). Para garantir que todas as variáveis tivessem o mesmo peso na definição dos segmentos, aplicamos a padronização (StandardScaler), normalizando os dados para que tivessem média zero e desvio padrão um.

Definição do Número de Clusters: Para determinar o número ideal de segmentos, utilizamos o Método do Cotovelo (Elbow Method). Esta técnica analisa a inércia do modelo (a soma das distâncias quadráticas das amostras ao centro do seu cluster mais próximo) para diferentes quantidades de clusters. O "cotovelo" no gráfico indica o ponto onde adicionar mais um cluster não traz um ganho significativo na explicação da variância, sugerindo o número ótimo de grupos.

Modelagem com K-Means e Análise de Perfil: Com o número ideal de 4 clusters definido, aplicamos o algoritmo K-Means. O modelo então atribuiu a cada cliente um segmento específico, criando a nossa variável final segmento. Para dar vida e significado a cada um desses segmentos, criamos o perfil_segmentos, uma tabela agregada que calcula a média das principais características para cada cluster. Isso nos permitiu comparar os grupos e atribuir "personas" a eles, como

"Clientes de Alto Valor", "Viajantes Recorrentes", "Clientes Econômicos" e "Novos Clientes", transformando os segmentos numéricos em insights acionáveis para o negócio.



Análise e Descrição dos Perfis de Clientes: com base na análise dos centróides de cada cluster, identificamos 4 grupos distintos com comportamentos e características muito claras. A seguir, descrevemos cada uma dessas "personas":

Perfil Médio de Cada Segmento de Cliente

segmento	0	1	2	3
gmv_success_mean	179.83	173.49	197.42	160.86
gmv_success_sum	343.97	3926.23	2201.59	5801.67
total_compras_sum	8.43	817.37	233.57	1591.70
quantidade_tickets_mean	1.49	1.33	1.42	1.29
valor_acumulado_sum	717.27	49604.02	15189.30	111178.40
cliente_recorrente_mean	0.44	1.00	1.00	1.00
score_fidelidade_mean	0.87	0.60	0.60	0.56
compras_anteriores_mean	0.56	13.02	6.53	18.96
faixa_tickets_int_mean	1.47	1.31	1.39	1.28
sazonalidade_score_mean	316.30	393.07	372.74	448.10
diversidade_destinos_max	1.56	5.36	4.25	6.63
diversidade_empresas_max	1.41	5.06	3.89	6.43
compra_ate_5_dias_feriado_sum	0.39	4.35	2.38	6.10
periodo_covid_int_max	1.00	1.48	1.38	1.54
periodo_covid_int_min	0.77	0.16	0.21	0.12
tipo_dia_int_mean	0.35	0.30	0.31	0.29
empresa_ida_<lambda>_enc	73.16	74.62	73.29	73.25
origem_ida_<lambda>_enc	303.00	286.79	283.54	276.15
destino_ida_<lambda>_enc	481.64	456.72	450.90	449.63
segmento	0.00	1.00	2.00	3.00

Segmento 0: Clientes Ocasionais ou Novos

Este é o nosso maior grupo em número de clientes, mas com o menor valor agregado. Caracterizam-se por terem o menor GMV médio e total, o menor número de compras e o menor score de fidelidade. São clientes que fizeram poucas viagens, provavelmente experimentando o serviço ou viajando por necessidade pontual.

Segmento 1: Viajantes Frequentes (Médio Valor)

Este grupo, embora não tenha o maior gasto total, destaca-se pelo altíssimo número de compras e pela alta frequência. Possuem um score_fidelidade moderado, mas uma alta diversidade de empresas e destinos, indicando que viajam muito, provavelmente a trabalho ou lazer, mas não são necessariamente clientes com altos gastos.

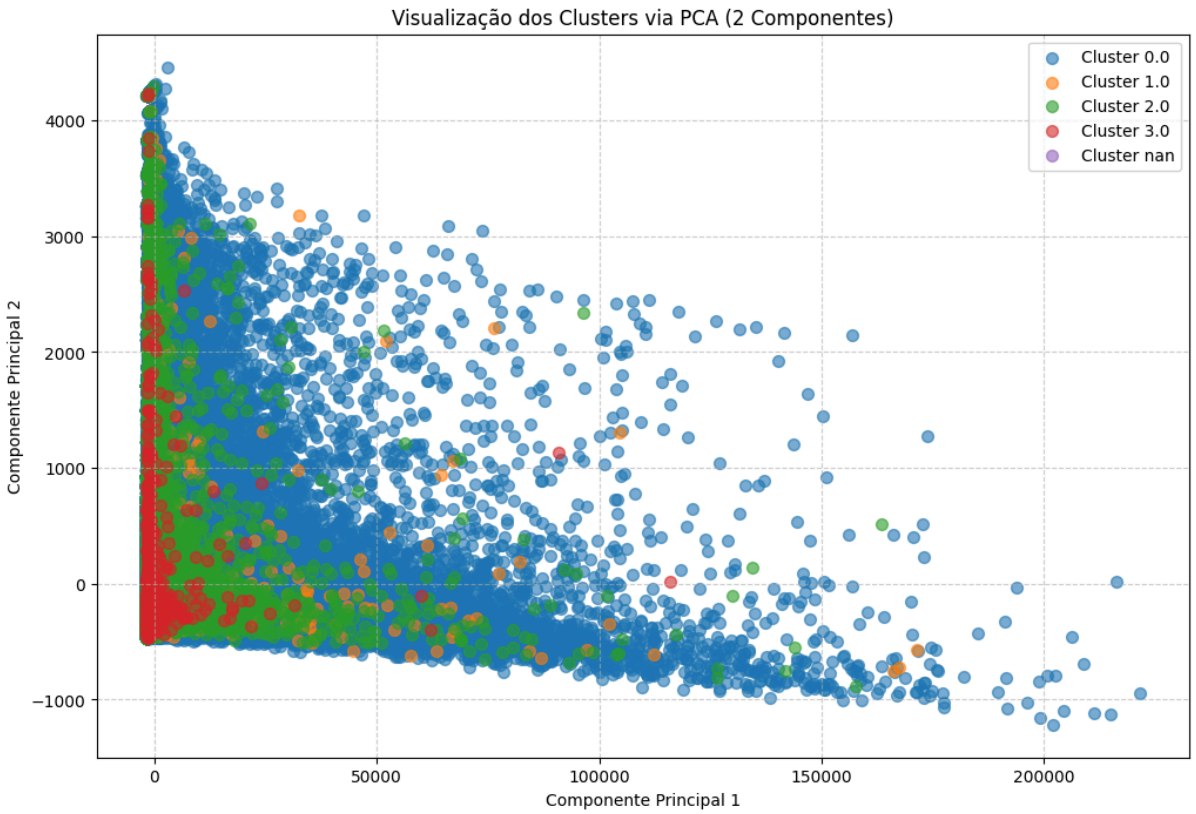
Segmento 2: Clientes Recorrentes de Valor Consistente

Este grupo representa um meio-termo saudável. São 100% recorrentes, têm um bom volume de compras e um gmv_success_sum considerável, superior ao Segmento 1. Têm um score de fidelidade um pouco maior e viajam com menos frequência, mas gastam mais quando o fazem.

Segmento 3: Clientes de Altíssimo Valor (VIPs)

Este é o nosso segmento de elite. Embora não sejam o maior grupo, eles possuem, de longe, o maior gmv_success_sum e valor_acumulado_sum. São 100% recorrentes, têm a maior diversidade de destinos e empresas, e uma alta propensão a viajar perto de feriados. São os nossos "heavy users" e os mais lucrativos.

Para validar a separação dos clusters e facilitar a interpretação, utilizamos a Análise de Componentes Principais (PCA). Esta técnica de redução de dimensionalidade nos permitiu condensar as múltiplas features de comportamento do cliente em dois "super-eixos" (Componente Principal 1 e 2), visualizando a distribuição dos segmentos num gráfico 2D. A clara separação dos grupos no gráfico de dispersão validou visualmente a eficácia da nossa segmentação.



Finalmente, para operacionalizar esta análise, geramos cinco arquivos CSV que servem como outputs principais do projeto, cada um com um propósito específico para o time de BI e Analytics:

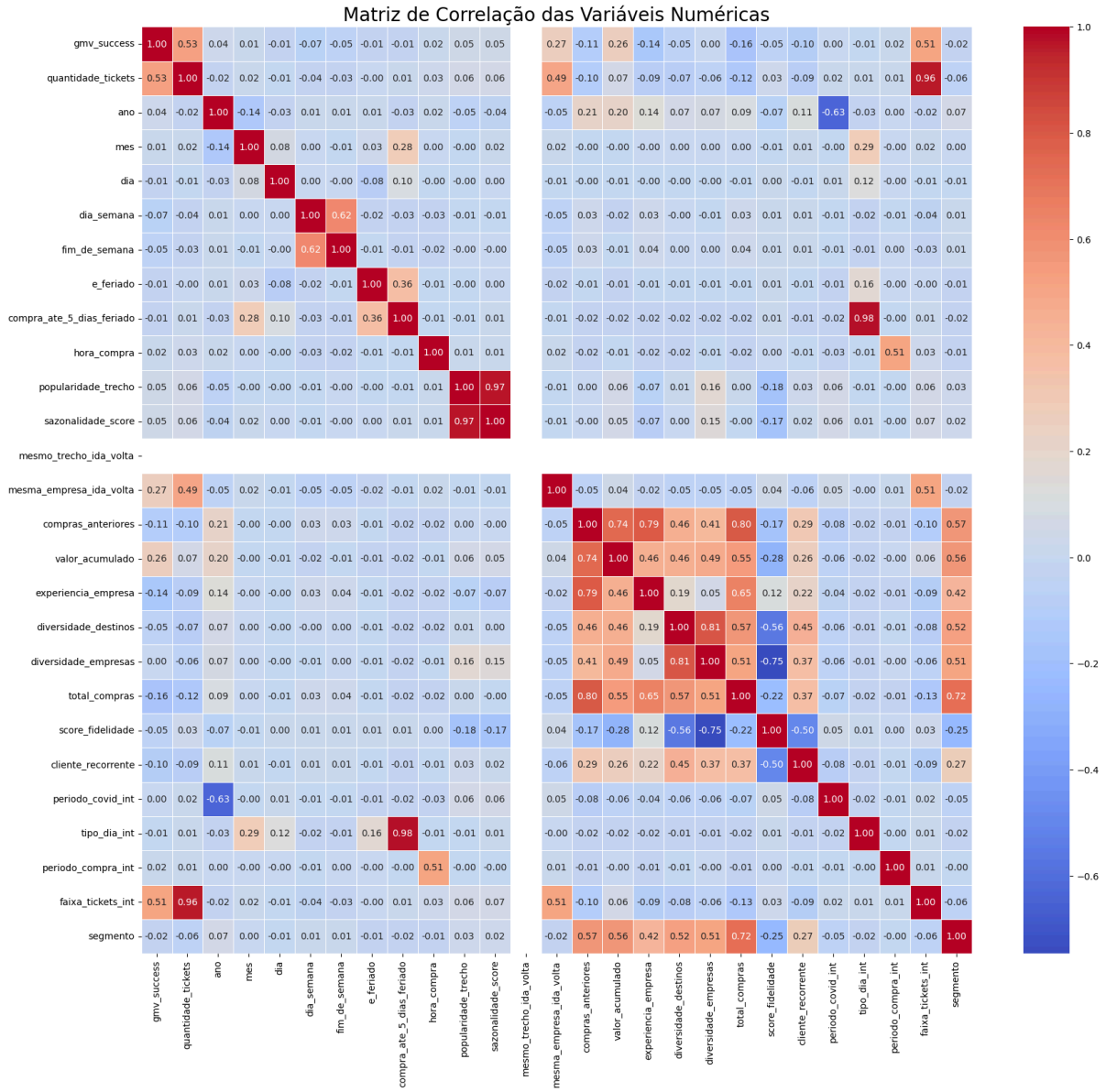
- df_completo_com_clusters.csv: É o nosso dataset mestre. Contém todos os dados de transações originais, enriquecidos com a coluna segmento. É o arquivo ideal para análises detalhadas e dashboards dinâmicos no Power BI, permitindo cruzar os perfis de cliente com qualquer outra dimensão do negócio.
- perfil_medio_segmentos.csv: Uma tabela agregada que serve como um "resumo executivo" de cada segmento. Apresenta o perfil médio de cada grupo com base nas principais features, ideal para relatórios gerenciais e apresentações.
- cluster_analise_final.csv: Contém o DataFrame principal utilizado na análise de clustering, incluindo as colunas originais e as features preparadas. Permite a replicação da análise e a construção de novas visualizações.
- cluster_medias.csv: Apresenta as médias das variáveis principais para cada cluster. É um ficheiro útil para gráficos comparativos e para identificar rapidamente as características mais marcantes de cada segmento.

- centroides clusters.csv: Contém os valores dos centróides de cada cluster no mesmo espaço de dados padronizados. É um ficheiro mais técnico, útil para análises avançadas e para compreender a localização de cada cluster em relação às variáveis normalizadas.

Timing é tudo

Para o desafio referente à previsão em uma janela de compra nos próximos 7 ou 30 dias, decidimos realizar somente para 30 dias, buscando um modelo mais performático e assertivo. Para isso, aplicamos um merge referente aos resultados obtidos com a segmentação, pois é uma variável bastante importante para o comportamento do cliente e a previsão de compra. O dataset com a engenharia de features aplicado, ficou em 45 colunas e 1.724.036 dados.

Aplicamos uma matriz de correlação nos dados numéricos para entender melhor a relação entre eles.



Assim, fizemos escolha das features para modelagem baseado em variáveis de valor e frequência, fidelidade e recorrência, comportamento e contexto e variáveis chave. Fizemos a divisão dos dados em

Treino: (1076778, 45) | Validação: (170000, 45) | Teste: (340000, 45)

Com o período obtido de:

- Período do treino: 2013-09-12 até 2022-11-04
- Período da validação: 2022-11-04 até 2023-04-22
- Período do teste: 2023-04-22 até 2024-04-01

Utilizamos o Pipeline com SMOTE para equalizar melhor os dados, devido ao desbalanceamento de clientes que compraram dentro da janela estabelecida de 30 dias.

O XGBoost foi utilizado para esse desafio e para métricas de performance, nos baseamos em acurácia, precisão, recall, F1-Score e ROC AUC Score.

Para a primeira rodada de treino, tivemos uma validação com tais resultados:

--- Métricas de Performance (Validação) ---

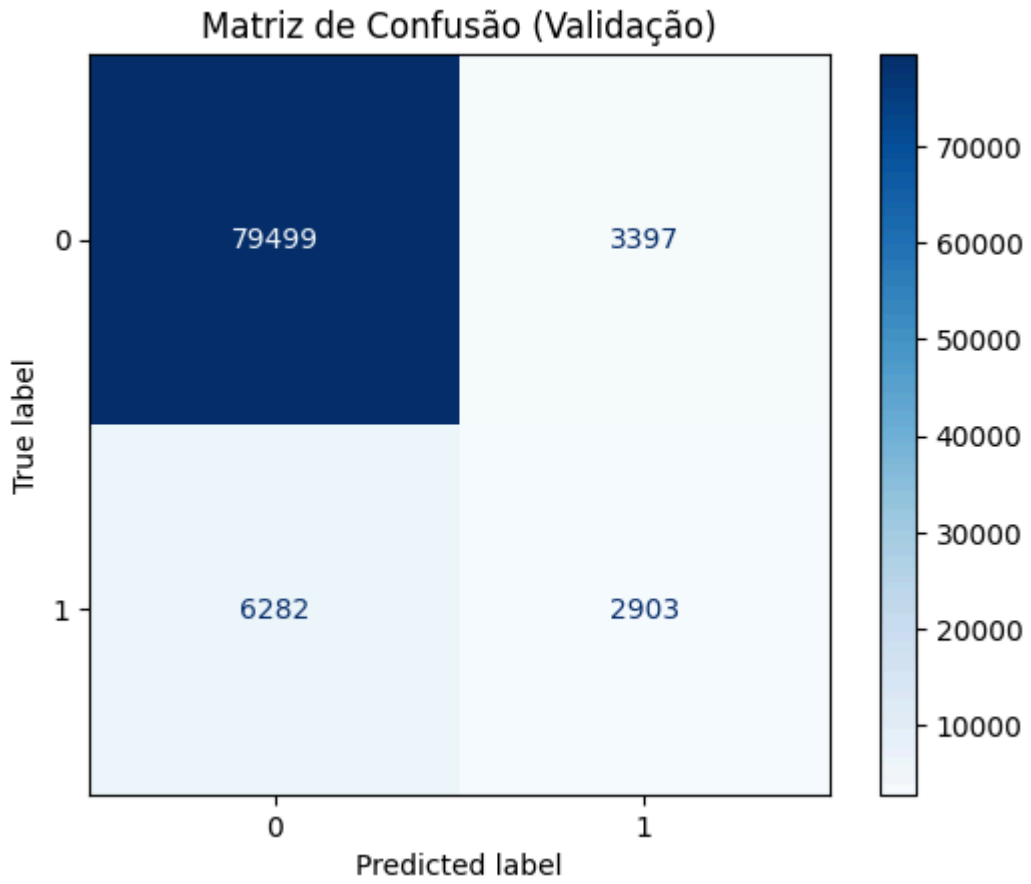
Acurácia: 0.8949

Precisão (para classe 1): 0.4608

Recall (para classe 1): 0.3161

F1-Score (para classe 1): 0.3749

ROC AUC Score: 0.8848



Interpretamos esses resultados que com um ROC AUC de 0.8848, o modelo apresentou um alto poder de separação, o que é uma base sólida para qualquer estratégia. A Precisão que está em 0.4608 indica que, de todas as vezes que o modelo previu que um cliente iria comprar, ele estava correto em cerca de 46% delas.

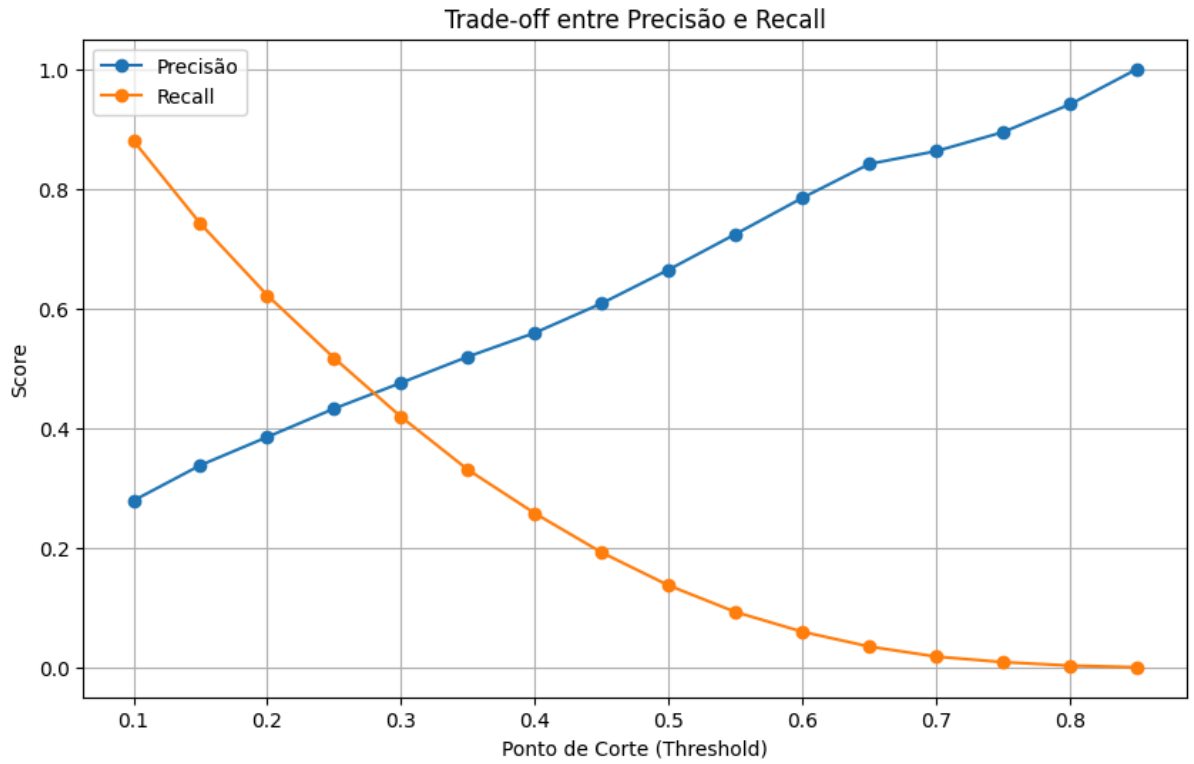
Falsos Positivos (FP): 3.397 - Estes são os clientes que o modelo previu que comprariam, mas não o fizeram. Este é o nosso "custo" de erro, pois poderíamos ter gasto recursos com eles.

Falsos Negativos (FN): 6.282 - Estes são os clientes que o modelo não previu, mas que realmente compraram. Estas são as nossas "oportunidades perdidas".

Assim, começamos os ajustes de hiperparâmetros utilizando o Randomized search CV, utilizamos também a validação cruzada estratificada para lidar com o desbalanceamento, com um número de iter = 20 e utilizando o ROC AUC como métrica de otimização. Nesse processo, também testamos diferentes thresholds para encontrar o melhor balanço, para isso obtivemos:

Melhores parâmetros encontrados: {'classifier__subsample': 1.0, 'classifier__n_estimators': 300, 'classifier__max_depth': 9, 'classifier__learning_rate': 0.2, 'classifier__colsample_bytree': 0.8}

Melhor ROC AUC score: 0.9448



O gráfico de Trade-off entre Precisão e Recall mostra visualmente que não é possível ter 100% de precisão e 100% de recall ao mesmo tempo.

Uma alta precisão significa menos falsos positivos (menos previsões de compra erradas). Se o custo de um "falso alarme" for alto, deve-se escolher um threshold mais elevado (por exemplo, 0.7 ou 0.8).

Um alto recall significa menos falsos negativos (menos clientes que compram mas que o modelo não previu). Se o custo de perder uma previsão for alto, deve-se escolher um threshold mais baixo (por exemplo, 0.2 ou 0.3).

Assim, rodando o modelo com os melhores parâmetros e com Threshold definido em 0.67, obtivemos:

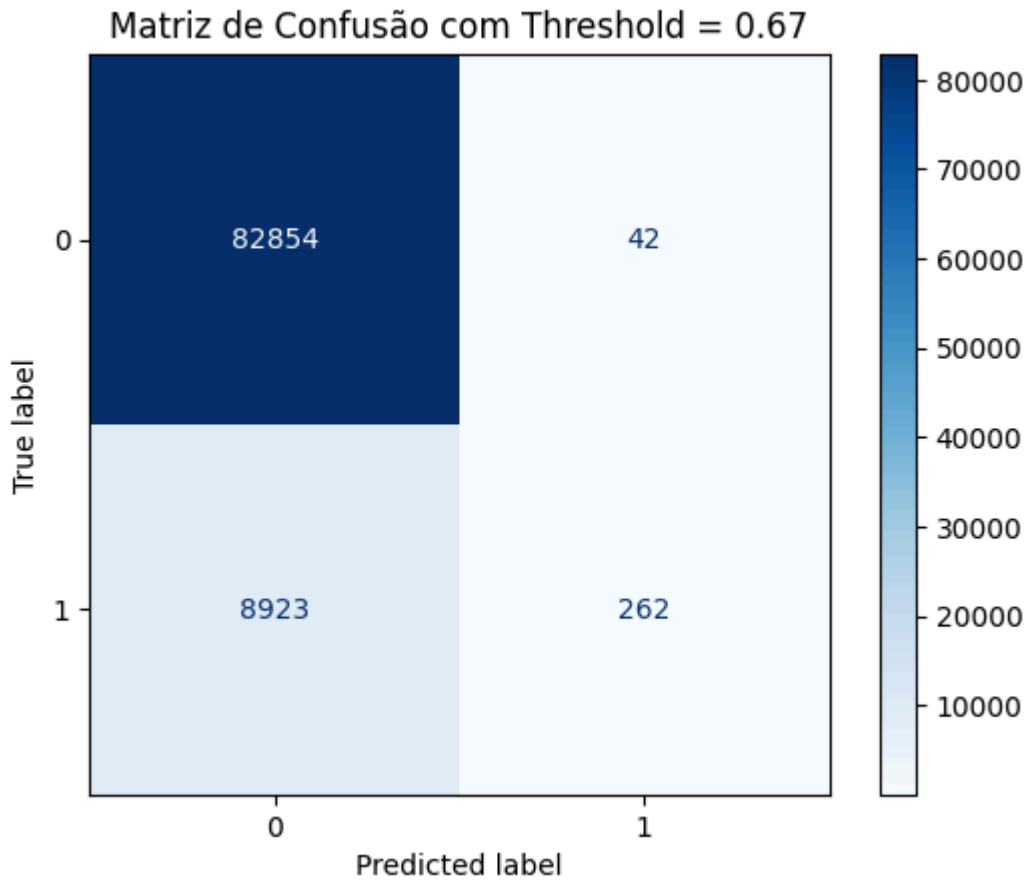
--- Métricas do Modelo Final com Threshold = 0.67 ---

Precisão: 0.8618

Recall: 0.0285

F1-Score: 0.0552

ROC AUC Score: 0.8962



Após os resultados, tentamos uma nova abordagem utilizando Scale_pos_Wight, para ter outra forma balancear as classes. Nessa nova rodada de testes, obtivemos o seguinte resultado:

--- Métricas de Performance (Validação com scale_pos_weight) ---

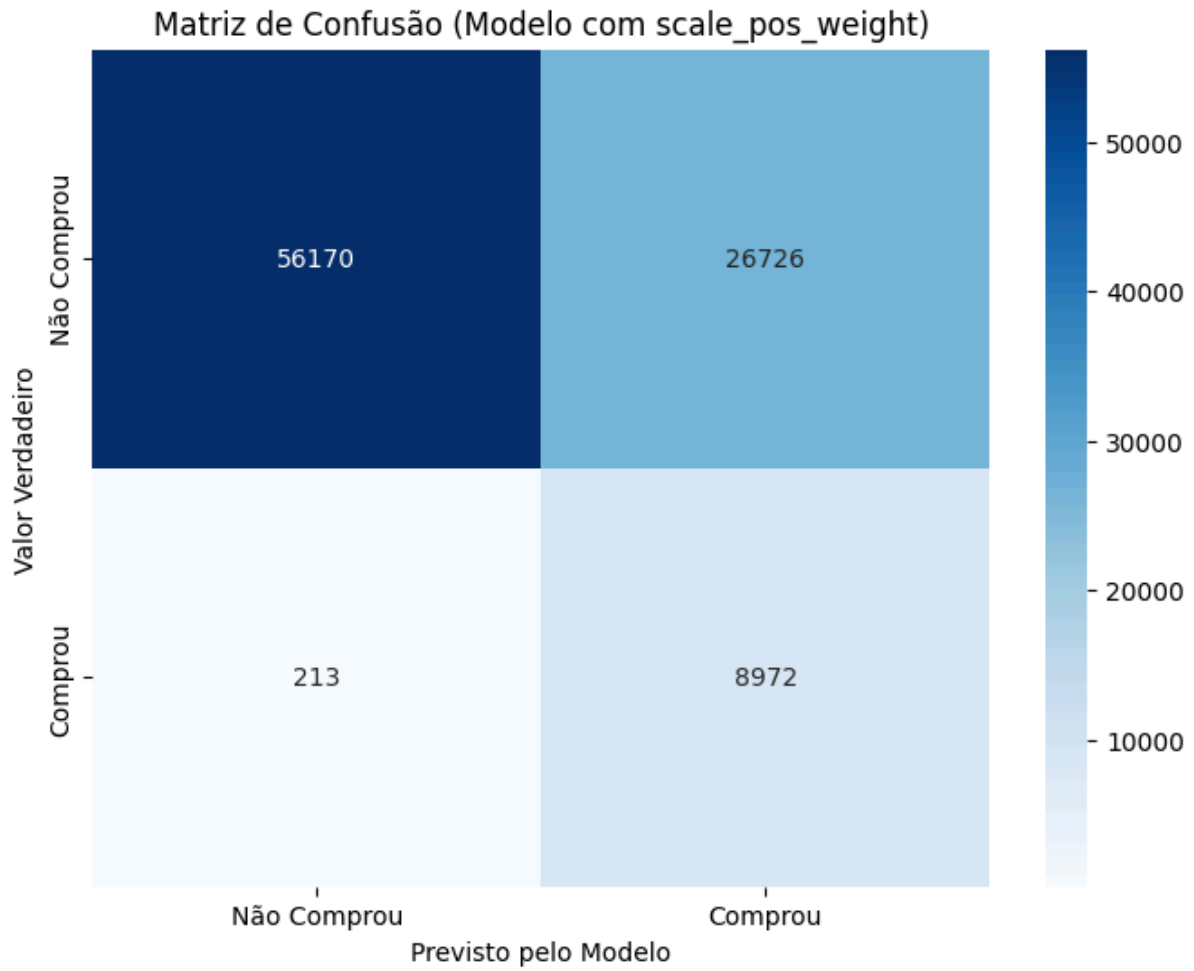
Acurácia (SPW): 0.7074

Precisão (SPW): 0.2513

Recall (SPW): 0.9768

F1-Score (SPW): 0.3998

ROC AUC Score (SPW): 0.8973




```
============================================================
        TABELA DE COMPARAÇÃO DOS MODELOS
============================================================
          Modelo com SMOTE  Modelo com Scale_Pos_Weight
Métrica
Acurácia            0.8949                       0.7074
Precisão            0.4608                       0.2513
Recall              0.3161                       0.9768
F1-Score            0.3749                       0.3998
ROC AUC             0.8848                       0.8973
```

Após a comparação, seguimos com a abordagem de modelo com SMOTE, utilizando mais parâmetros e aplicando com RandomizedSearchCV

--- Métricas de Performance (Validação) ---

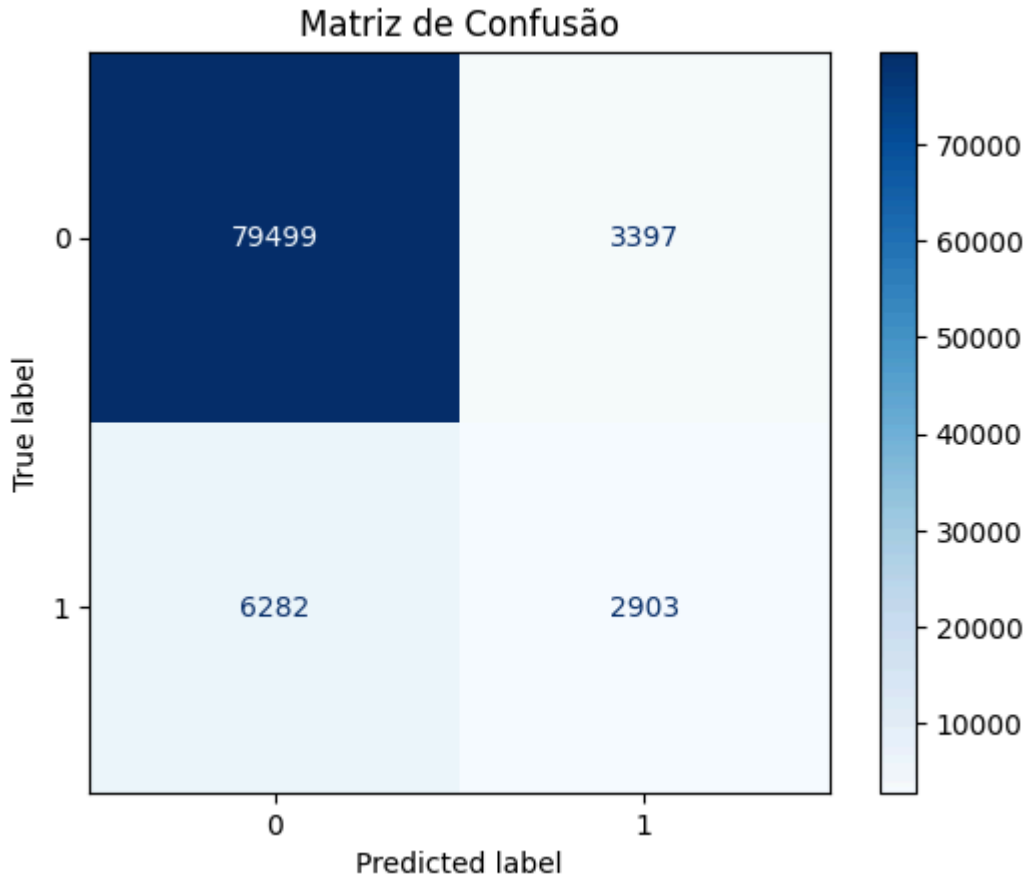
Acurácia: 0.8949

Precisão: 0.4608

Recall: 0.3161

F1-Score: 0.3749

ROC AUC Score: 0.8848



Rodamos o RandomizedSearchCV para F1-Score

--- Métricas de Performance (Teste Final) ---

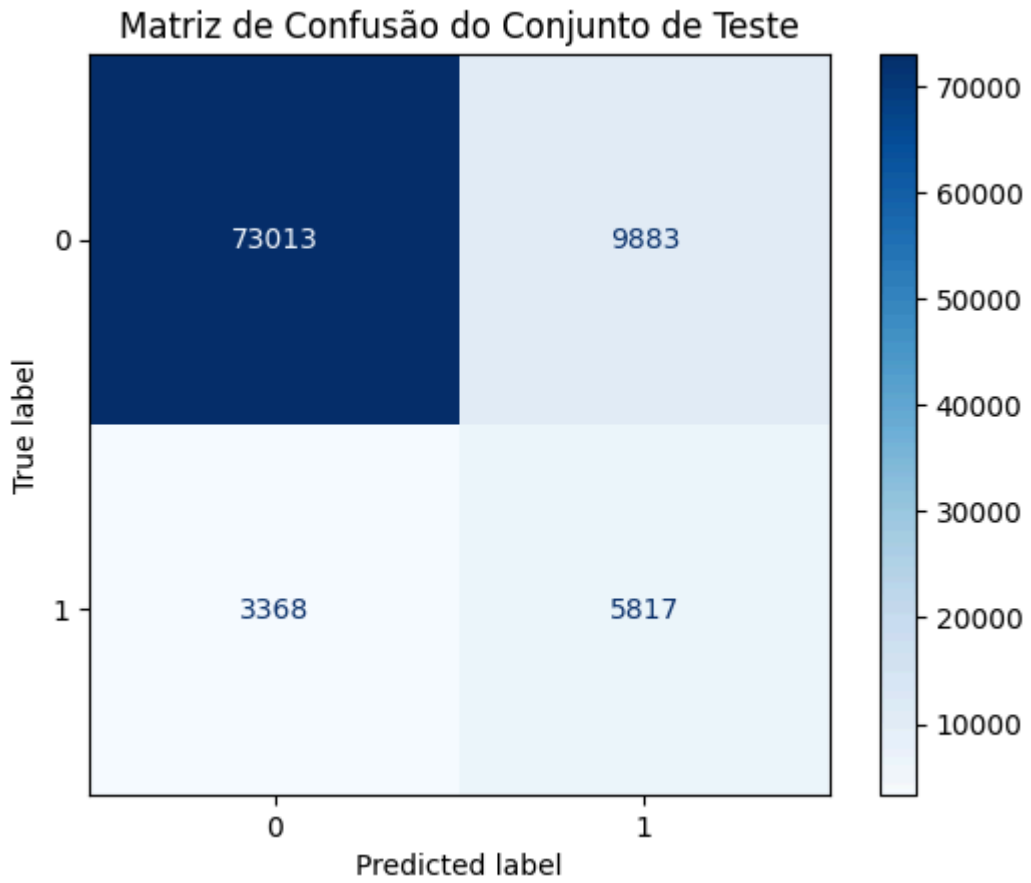
Acurácia: 0.8561

Precisão: 0.3705

Recall: 0.6333

F1-Score: 0.4675

ROC AUC Score: 0.8903



Com todos os testes e modelo rodado, chegamos a conclusão que os melhores parâmetros é o modelo Otimizado para F1-Score: Este é o portfólio otimizado. Ele encontrou o "ponto ótimo" na curva de risco-retorno:

Dobrou as Oportunidades: O Recall saltou de 0.31 para 0.63, capturando mais do que o dobro dos clientes que iriam comprar.

- Custo Controlado: A Precisão de 0.37 é um custo aceitável para um ganho tão expressivo em recall. Significa que, a cada 100 clientes que acionamos, 37 realmente compram.
- Melhor Balanço Geral: O F1-Score de 0.4675 é o maior entre todas as estratégias equilibradas, provando que este modelo tem a melhor harmonia entre não perder oportunidades e não gastar recursos à toa.

Previsão de trechos

Para a realização da previsão de trechos, precisamos reduzir a cardinalidade das features, pois é fundamental para que o modelo possa ser mais assertivo, assim, definimos uma meta de cobertura de 97%, dessa forma, reduzimos a cardinalidade e o que não está dentro dessa cobertura, é classificado como “OUTROS”.

Fizemos a escolha das variáveis para a modelagem, e utilizamos o modelo LightGBM para prever o trecho específico (origem-destino) que um cliente tem maior probabilidade de comprar em sua próxima viagem. Em resultados preliminares, tivemos o resultado:

--- Métricas de Performance Gerais (Validação) ---

Acurácia: 0.4128

Acurácia Balanceada: 0.3583

F1-Score (Ponderado): 0.4941

F1-Score (Micro Avg): 0.4128

F1-Score (Macro Avg): 0.2181

Onde o modelo tenta gerar 51 tipos de trechos diferentes mais a categoria “Outro” e quando verificamos o support de dados para cada classe, verificamos que é extremamente desproporcional, dificultando muito o modelo de conseguir realizar as devidas análises ou treinos, por conta disso, a Precisão média acaba sendo abaixo de 0,20.

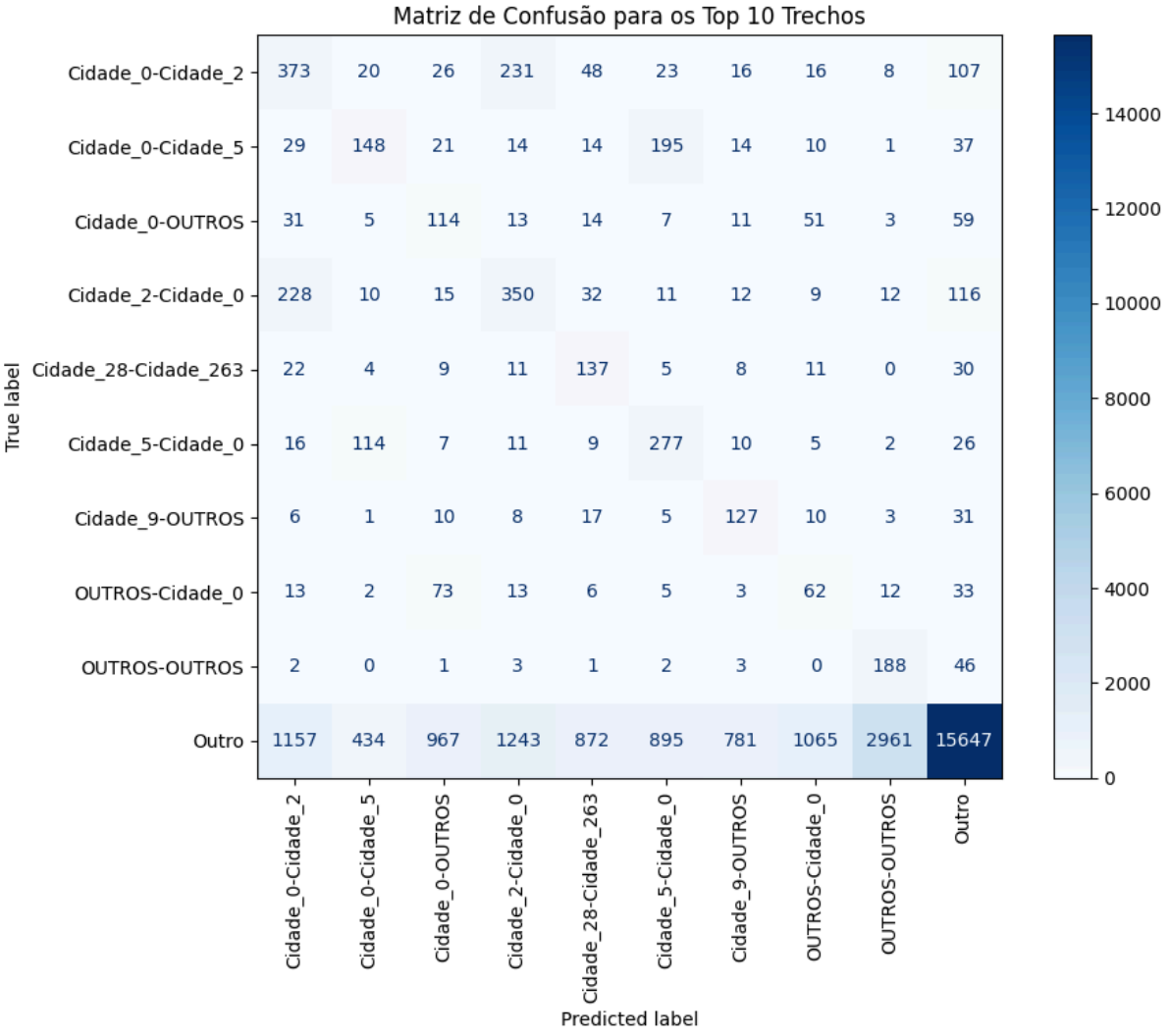
Em uma nova rodada de testes e interpretação dos dados e análise de cardinalidade das Features Categóricas, consolidamos para os 10 principais trechos e após treino do modelo, os resultados mostram-se promissores com aumento nas principais métricas de performance. Entretanto, o modelo ainda tem uma precisão bastante elevada para a classe “Outro”.

```
--- Métricas de Performance (Validação - Top 10 Trechos) ---
Acurácia: 0.5832
ROC AUC Score (Ponderado): 0.8622
F1-Score (Micro Avg): 0.5832
F1-Score (Macro Avg): 0.2558

--- Relatório de Classificação Detalhado (por trecho) ---
                     precision    recall  f1-score   support

   Cidade_0-Cidade_2       0.20      0.43      0.27       868
   Cidade_0-Cidade_5       0.20      0.31      0.24       483
     Cidade_0-OUTROS       0.09      0.37      0.15       308
   Cidade_2-Cidade_0       0.18      0.44      0.26       795
Cidade_28-Cidade_263       0.12      0.58      0.20       237
   Cidade_5-Cidade_0       0.19      0.58      0.29       477
     Cidade_9-OUTROS       0.13      0.58      0.21       218
     OUTROS-Cidade_0       0.05      0.28      0.08       222
       OUTROS-OUTROS       0.06      0.76      0.11       246
               Outro       0.97      0.60      0.74     26022

            accuracy                           0.58     29876
           macro avg       0.22      0.49      0.26     29876
        weighted avg       0.87      0.58      0.68     29876
```



Como estratégia de exploração, ainda nessa abordagem, aplicamos o RandomizedSearchCV com algumas iterações, tentando otimizar a métrica de “f1_macro”, além do teste em diversos parâmetros.

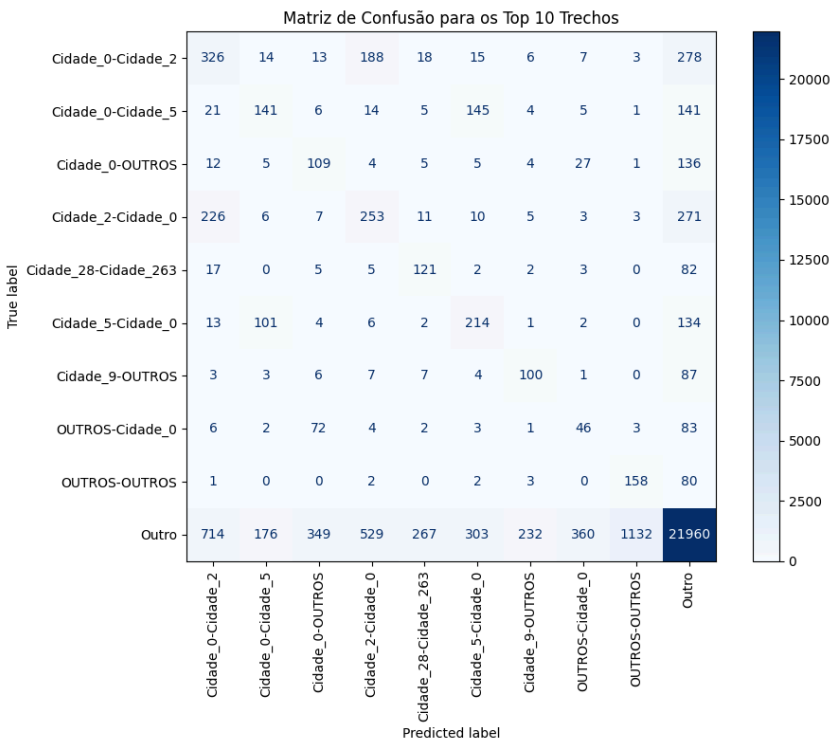
Assim, novamente temos um aumento em todas as métricas, com Acurácia saltando de 0.58 para 0.78. Apesar disso, ainda está sendo classificado e com bastante precisão a categoria “Outro”.

```
--- Métricas de Performance do Modelo Otimizado (Validação - Top 10 Trechos) ---
Acurácia: 0.7842
ROC AUC Score (Ponderado): 0.8571
F1-Score (Micro Avg): 0.7842
F1-Score (Macro Avg): 0.3427

--- Relatório de Classificação Detalhado (por trecho) ---
                    precision    recall  f1-score   support

   Cidade_0-Cidade_2       0.24      0.38      0.30       868
   Cidade_0-Cidade_5       0.31      0.29      0.30       483
     Cidade_0-OUTROS       0.19      0.35      0.25       308
   Cidade_2-Cidade_0       0.25      0.32      0.28       795
Cidade_28-Cidade_263       0.28      0.51      0.36       237
   Cidade_5-Cidade_0       0.30      0.45      0.36       477
     Cidade_9-OUTROS       0.28      0.46      0.35       218
     OUTROS-Cidade_0       0.10      0.21      0.14       222
       OUTROS-OUTROS       0.12      0.64      0.20       246
               Outro       0.94      0.84      0.89     26022

            accuracy                           0.78     29876
           macro avg       0.30      0.45      0.34     29876
        weighted avg       0.85      0.78      0.81     29876
```



Dado que a quantidade de dados para essa previsão de trechos diminuiu consideravelmente em relação ao que tínhamos disponível para o primeiro desafio, juntamos o conjunto de treino e de validação, assim o modelo tem mais dados para realizar o treinamento. Nessa rodada de teste, tivemos como hipótese a criação da feature “ultimo_destino_ida”, essa informação serve para saber onde o cliente “estava por último”, pois assim o modelo tem uma previsão maior de onde ele está e foca apenas em descobrir o destino.

Rodamos novamente com a busca por melhores parâmetros em RandomizedSearchCV, obtivemos os resultados abaixo:

```
--- Métricas de Performance Finais (Conjunto de Teste) ---
Acurácia: 0.5736
ROC AUC (Ponderado): 0.8776
F1-Score (Macro): 0.5488

--- Relatório de Classificação Detalhado ---
              precision    recall  f1-score   support

    Cidade_0       0.71      0.45      0.55      4467
  Cidade_106       0.34      0.42      0.38       413
   Cidade_17       0.48      0.64      0.55       855
    Cidade_2       0.59      0.63      0.61      2488
   Cidade_28       0.47      0.69      0.56       781
    Cidade_3       0.50      0.63      0.56      1171
    Cidade_5       0.54      0.66      0.60      1357
   Cidade_80       0.43      0.68      0.52       447
    Cidade_9       0.45      0.60      0.52       869
      OUTROS       0.70      0.60      0.65      2853

    accuracy                           0.57     15701
   macro avg       0.52      0.60      0.55     15701
weighted avg       0.60      0.57      0.57     15701
```

Apesar de uma queda bastante significativa de acurácia, o modelo que trás qual a cidade destino, apresenta Precisão maior para os destinos do que propriamente para a classe “Outro”, então é considerado uma melhora.

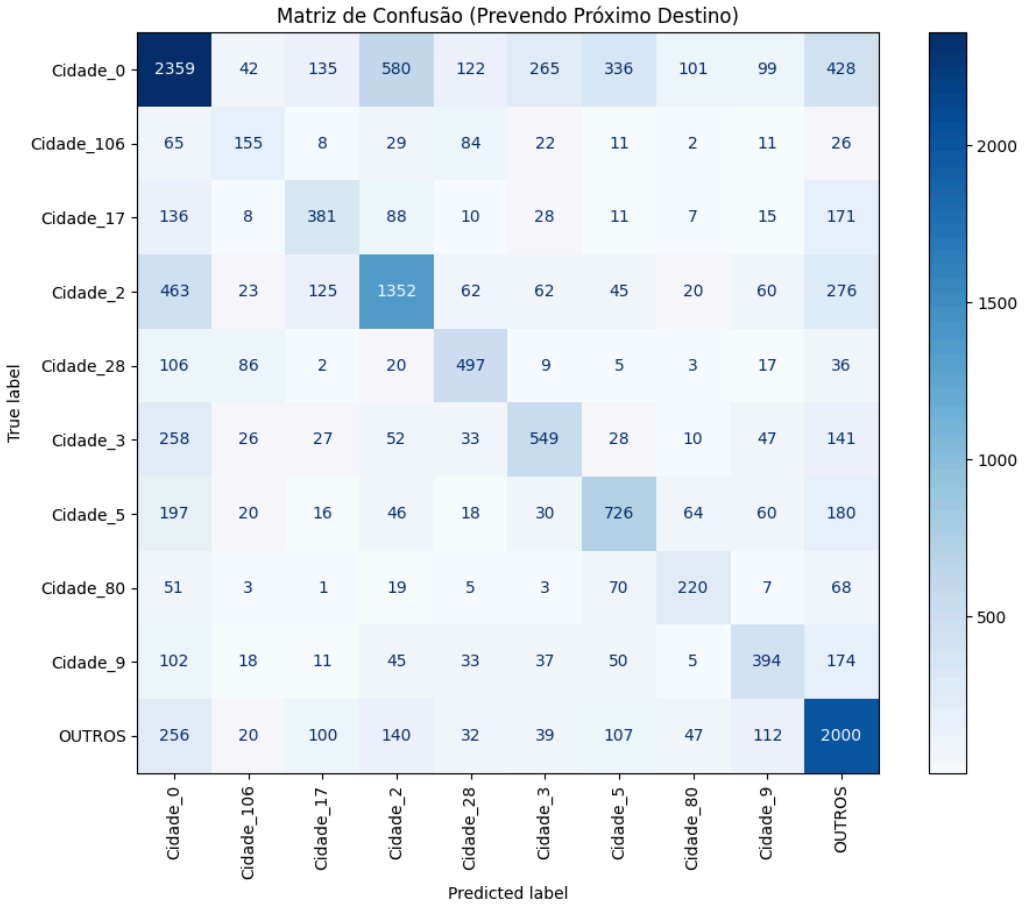
Porém, devido os dados estarem desbalanceados, utilizamos o pipeline SMOTE com as mesmas hipóteses do anterior. Assim, tivemos os resultados abaixo:

```
--- Métricas de Performance Finais (Conjunto de Teste) -
Acurácia: 0.5498
ROC AUC (Ponderado): 0.8776
F1-Score (Macro): 0.5142

--- Relatório de Classificação Detalhado ---
              precision    recall  f1-score   support

    Cidade_0       0.59      0.53      0.56      4467
  Cidade_106       0.39      0.38      0.38       413
   Cidade_17       0.47      0.45      0.46       855
    Cidade_2       0.57      0.54      0.56      2488
   Cidade_28       0.55      0.64      0.59       781
    Cidade_3       0.53      0.47      0.50      1171
    Cidade_5       0.52      0.54      0.53      1357
   Cidade_80       0.46      0.49      0.48       447
    Cidade_9       0.48      0.45      0.47       869
      OUTROS       0.57      0.70      0.63      2853

    accuracy                           0.55     15701
   macro avg       0.51      0.52      0.51     15701
weighted avg       0.55      0.55      0.55     15701
```



Apresentaram resultados bastante semelhantes.

Por fim, para consolidar toda rodada de testes e hipóteses, aplicamos a estratégia que combina:

1. Dados de treino + Validação;
2. Feature de Estado (“ultimo_destino”);
3. Balanceamento com SMOTE;
4. Foco nos Top 10 pares de trechos;
5. Aplicação de RandomizedSearchCV para otimização de hiperparâmetros.

Para isso

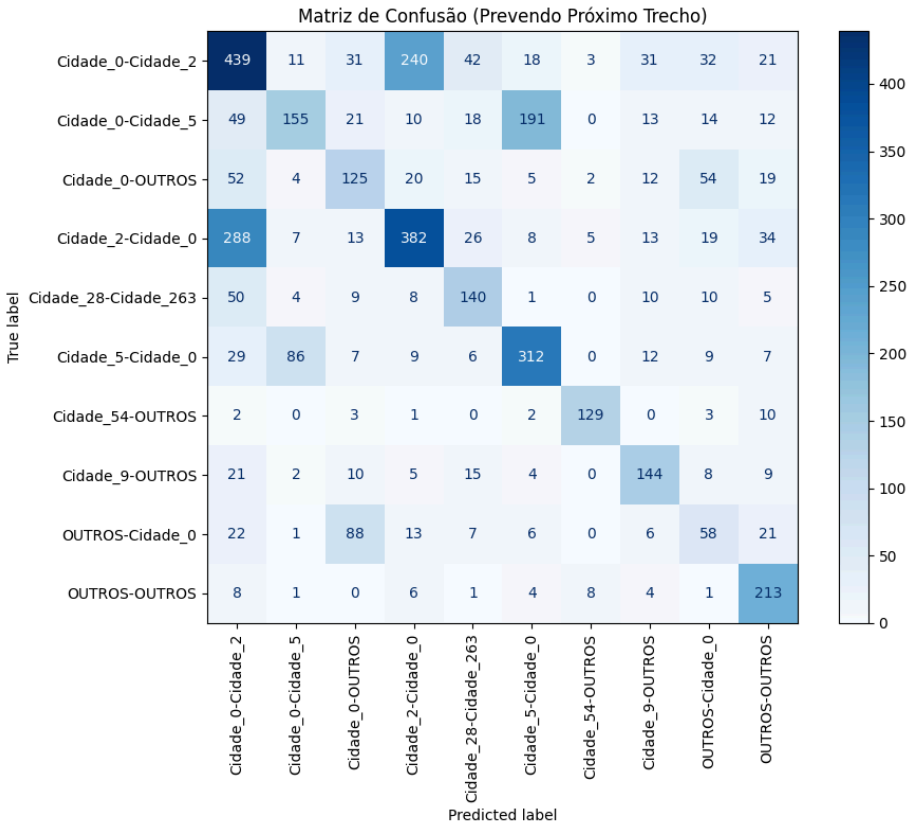
```
--- Métricas de Performance Finais (Conjunto de Teste) ---
Acurácia: 0.5237
ROC AUC (Ponderado): 0.8756
F1-Score (Macro): 0.5444

--- Relatório de Classificação Detalhado ---
                     precision    recall  f1-score   support

   Cidade_0-Cidade_2       0.46      0.51      0.48       868
   Cidade_0-Cidade_5       0.57      0.32      0.41       483
     Cidade_0-OUTROS       0.41      0.41      0.41       308
   Cidade_2-Cidade_0       0.55      0.48      0.51       795
Cidade_28-Cidade_263       0.52      0.59      0.55       237
   Cidade_5-Cidade_0       0.57      0.65      0.61       477
    Cidade_54-OUTROS       0.88      0.86      0.87       150
     Cidade_9-OUTROS       0.59      0.66      0.62       218
     OUTROS-Cidade_0       0.28      0.26      0.27       222
       OUTROS-OUTROS       0.61      0.87      0.71       246

            accuracy                           0.52      4004
           macro avg       0.54      0.56      0.54      4004
        weighted avg       0.52      0.52      0.52      4004
```

Tivemos o resultado a seguir, com support mais equilibrado, precisão das classes mais equilibradas e com o modelo com menos precisão de acertos referente a classe “OUTROS-OUTROS”.



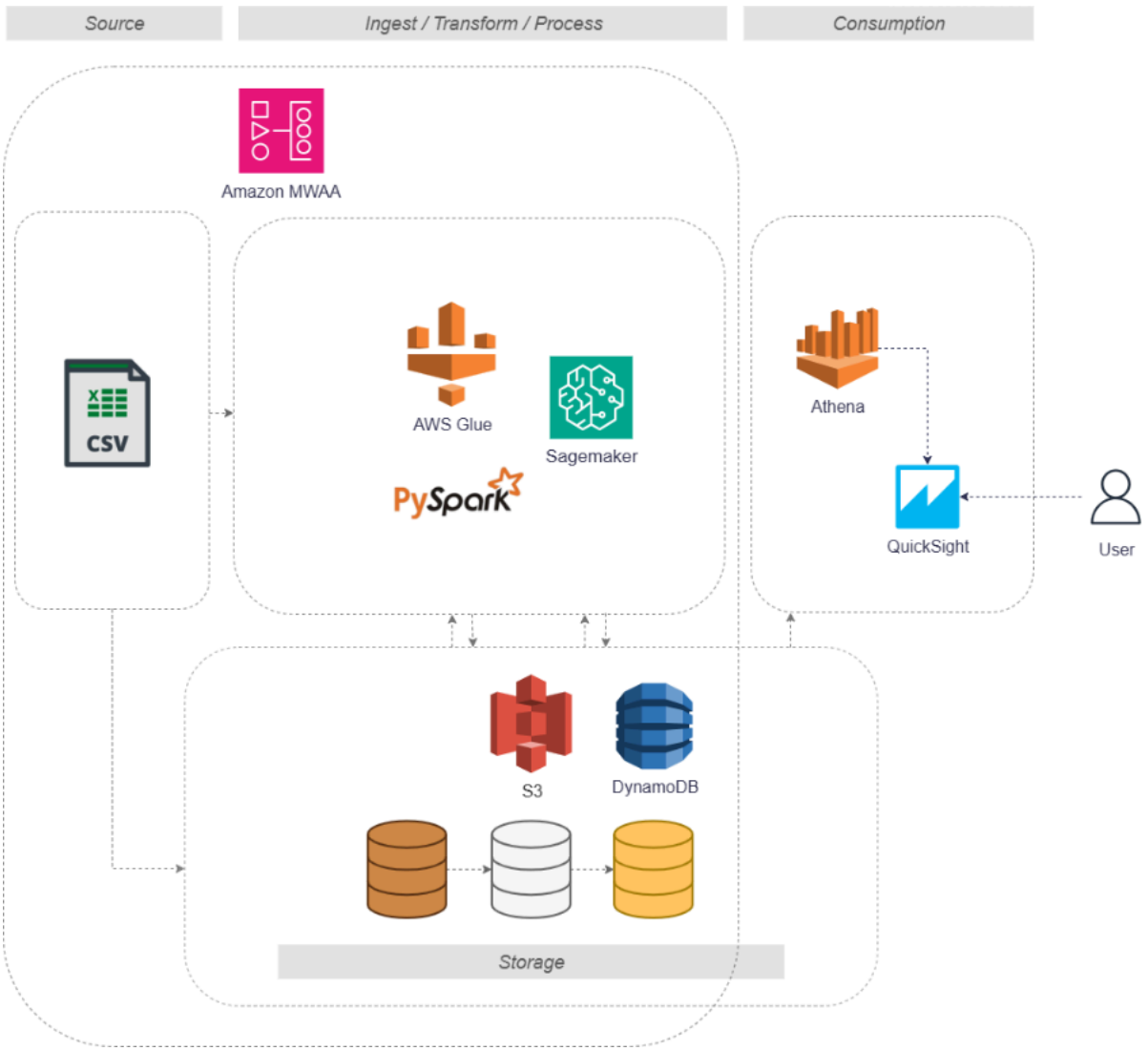
A complexidade de prever o par completo origem-destino nos levou a uma abordagem iterativa e estratégica. Inicialmente, o modelo demonstrou resultados promissores ao prever apenas o destino final do cliente, com uma acurácia e precisão significativamente maiores do que a previsão completa do trecho. Essa etapa preliminar validou nossa hipótese de que o modelo tinha um bom entendimento das preferências de viagem, o que nos deu a confiança necessária para evoluir para o problema de maior complexidade. A partir dessa base sólida, aprimoramos o modelo para prever o par completo de trechos, o que nos permitiu ter um ganho progressivo de assertividade e gerar insights ainda mais valiosos para o negócio.

O desafio de prever o próximo trecho do cliente foi abordado de forma iterativa, com foco em superar os desafios de alta cardinalidade e desbalanceamento de classes. As análises preliminares confirmaram a dificuldade do modelo em lidar com a categoria "Outro" e em manter uma precisão elevada. Para resolver isso, implementamos uma estratégia combinada e robusta: unimos os conjuntos de treino e validação para aumentar a quantidade de dados, criamos a feature ultimo_destino_ida para contextualizar o modelo, e aplicamos o SMOTE

para mitigar o desbalanceamento. A otimização final, realizada com RandomizedSearchCV e foco na métrica f1_macro, direcionou o modelo para uma performance mais equilibrada. Apesar de uma queda na acurácia geral, o resultado mais promissor foi a melhora significativa na precisão para trechos específicos em detrimento da classe "Outro". Isso demonstra que o modelo, após todo o refinamento, tornou-se capaz de gerar previsões mais assertivas e, consequentemente, mais valiosas para a tomada de decisões de negócio da ClickBus.

Arquitetura

Diagrama da Arquitetura de Dados Proposta – Projeto ClickBus



Contexto

A ClickBus forneceu um dataset em formato CSV, contendo aproximadamente 1.7 milhões de linhas com dados anonimizados. Nesta etapa, o objetivo do grupo foi propor uma solução de arquitetura capaz de atender aos três desafios propostos.

Diante dos objetivos, o grupo propôs uma arquitetura moderna, escalável e orientada a camadas de dados (Bronze, Prata e Ouro) utilizando serviços AWS.

Arquitetura Proposta

A arquitetura segue o seguinte fluxo:
Ingestão → **Processamento** → **Armazenamento** → **Modelagem/Análise** → **Consumo**, utilizando as ferramentas a seguir:

- Amazon MWAA (Managed Workflows for Apache Airflow)
- AWS Glue
- PySpark
- Amazon S3 (camadas Bronze, Prata e Gold)
- Amazon DynamoDB
- Amazon SageMaker
- Amazon Athena
- Amazon QuickSight

Uso das Ferramentas

Amazon S3 (Data Lake – Bronze, Prata e Gold)

- Uso:
    - Bronze: Armazena os dados crus (CSV original da ClickBus).

    - Prata: Dados limpos, tratados e padronizados (via AWS Glue e PySpark).
    - Gold: Dados enriquecidos, prontos para consumo analítico, dashboards e modelos preditivos.
- Justificativa:
    1. Armazenamento elástico, seguro e de baixo custo.
    2. Suporte nativo às integrações com Athena, SageMaker, Glue e QuickSight.
    3. Mantém a governança e versionamento dos dados.

AWS Glue

- Uso:
    - Criação de ETL jobs para limpeza, padronização e enriquecimento dos dados.
    - Criação de Glue Data Catalog para catalogar tabelas e permitir consultas SQL pelo Athena.
- Justificativa:
    1. Serviço serverless de ETL altamente integrado ao S3.
    2. Automatiza tarefas de schema discovery (inferência de colunas, tipos).
    3. Facilita governança e permite padronizar o dado na camada Prata.

PySpark

- Uso:
    - Processamento distribuído de alto volume de dados (1.7M linhas) para:
    - Feature engineering (criação de variáveis derivadas).

        - Normalização e agregação dos históricos de compra.
        - Treinamento de modelos em larga escala.
- Justificativa:
    1. Biblioteca robusta para processamento em larga escala.
    2. Suporte direto no Glue e SageMaker.
    3. Escalável para cenários futuros com volumes maiores.

Amazon MWAA (Apache Airflow)

- Uso:

    Orquestração do pipeline de dados de ponta a ponta.
    - DAGs definidas para:
    - Ingestão de novos CSVs no S3.
    - Execução dos jobs de ETL no Glue.
    - Atualização das camadas Bronze, Prata e Gold.
    - Treinamento e deploy dos modelos no SageMaker.
    - Atualização de dashboards no QuickSight.
- Justificativa:
    1. Permite agendamento, monitoramento e reprocessamento automatizado.
    2. Garante rastreabilidade e governança do fluxo de dados.

Amazon SageMaker

Uso:

Treinamento, validação e deploy dos modelos preditivos para os desafios:

- Segmentação de clientes: Clustering (KMeans, DBSCAN).
- Previsão de compra: Classificação binária (XGBoost, RandomForest).

- Previsão do próximo trecho: Classificação multiclasse / Recomendação (Modelos de Deep Learning).

Justificativa:

1. Ambiente gerenciado para ML, com escalabilidade e suporte a múltiplos algoritmos (incluindo XGBoost, já otimizado e integrado nativamente).
2. Permite integração direta com S3 (datasets) e DynamoDB (armazenamento de predições em tempo real).
3. Reduz tempo de setup de infraestrutura para ciência de dados, permitindo foco no treinamento e experimentação.
4. Facilita tanto experimentos rápidos (baseline com XGBoost) quanto modelos mais avançados (deep learning para recomendação).

Amazon DynamoDB

- Uso:
    - Armazenamento em tempo real das predições geradas pelos modelos.
    - Consulta rápida dos resultados por aplicações downstream (marketing, CRM, Growth).

- Justificativa:
    1. Banco NoSQL de baixa latência.
    2. Ideal para consultas rápidas por chave de cliente.
    3. Escalável e serverless, reduzindo custos operacionais.

Amazon Athena

- Uso:
    - Consultas SQL diretamente sobre dados no S3 (camadas Prata e Gold).
    - Suporte a análises exploratórias para os times de dados.
- Justificativa:
    1. Não exige provisionamento de infraestrutura.
    2. Integração nativa com Glue Data Catalog.
    3. Custo baseado apenas no volume escaneado.

Amazon QuickSight

- Uso:

  Construção de dashboards dinâmicos para:
    - Monitoramento do comportamento de compra dos clusters de clientes.
    - Visualização das previsões de compra (probabilidade de compra em 7/30 dias).
    - Acompanhamento dos trechos mais prováveis de compra.
- Justificativa:
    1. Ferramenta de BI serverless e integrada com S3, Athena e DynamoDB.
    2. Possibilita compartilhamento rápido de insights com áreas de negócio.

Fluxo Arquitetural

1. Ingestão: CSV é carregado no S3 Bronze.
2. Orquestração: MWAA dispara pipelines de ETL no Glue.

3. Processamento: Dados tratados com PySpark → S3 Prata.
4. Enriquecimento: Criação de features → S3 Gold.
5. Modelagem: SageMaker consome dados Gold, treina modelos e publica resultados.
6. Armazenamento de Previsões: DynamoDB recebe outputs para consumo em tempo real.
7. Análise e Consumo: Athena + QuickSight oferecem camadas analíticas e dashboards para o negócio.

Conclusão

A arquitetura proposta utiliza de forma estratégica os serviços da AWS, combinando data lake em camadas (S3), ETL e processamento distribuído (Glue + PySpark), orquestração (MWAA), modelagem de machine learning (SageMaker) e camadas de consumo (Athena, DynamoDB, QuickSight).

Essa estrutura garante:

- Escalabilidade para lidar com milhões de registros.
- Governança e versionamento de dados em múltiplas camadas.
- Automação e confiabilidade via Airflow.
- Insights acionáveis para áreas de marketing e Growth.
- Baixo tempo de resposta para recomendações e previsões

Referências bibliográficas

- Histórico da emergência internacional de COVID-19 - OPAS/OMS | Organização Pan-Americana da Saúde acessado em 15/09/2025.