



INSTITUTO

NOME DO PROJETO
Nome do Parceiro



Controle do Documento

Histórico de revisões

Data	Autor	Versão	Resumo da atividade
01/08/2022	Sophia Mello Dias	1.1.1	Criação do Documento
02/08/2022	Rafael Alves Cabral Yves Levi Paixão Lapa Sophia Mello Dias	1.1.2	Preenchimento da seção 2 Contexto da Indústria e 5 Forças de Potter Preenchimento da introdução
11/08/2022	Daniel Cunha Lyorrei Quintao Yves Lapa Sophia Dias Cristiane Coutinho Rafael Cabral	1.2.1	Value Proposition Canvas Matriz de Riscos Compreensão dos Dados Hipóteses Análise SWOT Planejamento Geral da Solução
16/08/2022	Sophia Dias Lyorrei Quintao Rafael Cabral	2.1	Jornadas do Usuário Preparação dos Dados
18/08/2022	Sophia Dias Cristiane Coutinho Lyorrei Quintao Yves Lapa Rafael Cabral	2.1.2	Proposta de Solução Planejamento Geral da Solução Compreensão dos Dados
19/08/2022	Lyorrei Quintao Cristiane Coutinho Daniel Cunha Yves Lapa	2.1.3	
23/09/2022	Sophia Mello Dias	4.2.9	Ajuste de imagens, legenda e paragrafação

Sumário

1. Introdução.....	4
2. Objetivos e Justificativa.....	5
2.1. Objetivos.....	5
2.2. Proposta de Solução.....	5
2.3. Justificativa.....	5
3. Metodologia.....	6
3.1. CRISP-DM.....	6
3.2. Ferramentas.....	6
3.3. Principais técnicas empregadas.....	6
4. Desenvolvimento e Resultados.....	7
4.1. Compreensão do Problema.....	7
4.1.1. Contexto da indústria.....	7
4.1.2. Análise SWOT.....	7
4.1.3. Planejamento Geral da Solução.....	7
4.1.4. Value Proposition Canvas.....	7
4.1.5. Matriz de Riscos.....	7
4.1.6. Personas.....	8
4.1.7. Jornadas do Usuário.....	8
4.2. Compreensão dos Dados.....	9
4.3. Preparação dos Dados.....	10
4.4. Modelagem.....	11
4.5. Avaliação.....	12
5. Conclusões e Recomendações.....	13
6. Referências.....	14
Anexos.....	15

1. Introdução

O stakeholder Banco Pan atua na área de mercado financeiro com destaque para as áreas de cartões de crédito, crédito consignado, financiamento de veículos, investimentos de renda fixa e banco digital. O banco tem foco nas classes C, D e E da população e conta com mais de 17 milhões de clientes.

Atualmente o Banco Pan possui índices que devem ser melhorados acerca da relação com o cliente, pois está em 3º lugar no índice BACEN do Ranking de Reclamações e conta com diversas reclamações no Procon. Ademais, o banco após se digitalizar aumentou sua cartela de produtos à oferecer e busca atingir um maior número de usuários.

2. Objetivos e Justificativa

2.1. Objetivos

O Banco Pan tem a necessidade de melhorar a sua reputação com o mercado e para isso ele precisa de uma solução que classifique melhor os clientes e não clientes do banco junto da sua necessidade de atendimento.

Sua nota no Reclame Aqui é de 7,5 e ele é o terceiro pior banco segundo o BaCen, sendo assim, o banco precisa classificar os seus clientes entre atritado, engajado e novo cliente para dar o procedimento correto ao atendimento.

2.2. Proposta de Solução

O grupo PanDevs se propõe a desenvolver um software com o uso de Aprendizado de Máquina e Inteligência Artificial que seja capaz de analisar o banco de dados de clientes, segmentar grupos (atritado, engajado e novo cliente) e identificar qual a necessidade do atendimento de cada cliente ou não cliente e com isso retornar ao Banco Pan. Dessa forma, será utilizado uma análise preditiva de classificação, utilizando correlações entre os dados fornecidos e o status do cliente, retornando 1 para indicar se é ou 0 para indicar quando não for.

Portanto, a classificação dos clientes auxiliará a área de atendimento do Banco Pan, direcionando e agilizando - o.

2.3. Justificativa

A computação tem algumas vantagens na hora de analisar dados em grandes escalas, como é o caso do nosso cliente, pois ela consegue processar as informações de forma sistemática em grande escala em uma velocidade que nenhum ser humano conseguiria replicar. Além disso, o algoritmo consegue armazenar e comparar uma quantidade simultânea de dados que garante uma maior confiabilidade dos resultados obtidos.

3. Metodologia

Descreva as etapas metodológicas que foram utilizadas para o desenvolvimento, citando o referencial teórico. Você deve apenas enunciar os métodos, sem dizer ainda como ele foi aplicado e quais resultados obtidos.

3.1. CRISP-DM

Descreva brevemente a metodologia CRISP-DM e suas etapas de processo

3.2. Ferramentas

Descreva brevemente as ferramentas utilizadas e seus papéis (Google Colaboratory)

3.3. Principais técnicas empregadas

Descreva brevemente as principais técnicas empregadas, algoritmos e seus benefícios

4. Desenvolvimento e Resultados

De maneira geral, você deve descrever nesta seção a aplicação dos métodos aprendidos e os resultados obtidos por seu grupo em seu projeto

4.1. Compreensão do Problema

Link do projeto:

<https://colab.research.google.com/drive/1aV2Xo-Fjdsofvhl8HM3eyEvm0XzEilo?usp=sharing>

4.1.1. Contexto da indústria

Em uma perspectiva macroeconômica, é importante frisar que o país passa por um momento de retomada das atividades habituais e trabalhos presenciais após um período em que foram concedidos diversos auxílios. Portanto, em uma análise top-down é possível de se observar o constante aumento da política de retração da oferta monetária no mercado, ou seja, com o aumento da taxa de juros – definida pelo Bacen – que chega a 13,75% ao ano, o dinheiro acaba sendo mais custoso, afetando empréstimos, tanto para novas empresas que querem captação para a abertura e expansão de novos negócios, mas também de pessoas físicas, podendo aumentar o risco de inadimplência. Porém, com a retomada da economia, o desemprego no Brasil já recua para 9,3%, sendo a menor taxa para o segundo trimestre desde 2015, de acordo com o jornal Folha de São Paulo. Apontando que, apesar do aumento do custo do dinheiro, as pessoas, principalmente de classes sociais mais baixas – C, D, E – conseguem captar algum dinheiro mediante a volta dos seus empregos.

Tendo em vista os drivers do setor bancário, estes são, em resumo: as taxas de desemprego, o ciclo de crédito e o custo do dinheiro mediante a inflação. Desse modo, a indústria apresenta como principais players o Itaú, o BTG Pactual, o Santander, Bradesco, Caixa Econômica Federal, Nubank, Inter, sendo os primeiros mais voltados ao nicho de alto varejo, enquanto as novas fintechs foram no crédito barato para as classes mais baixas financeiramente.

Sob uma perspectiva administrativa desenvolvida por Michael Porter, tal setor apresenta, contemporaneamente, baixíssimas barreiras de entrada, dado o desenvolvimento das tecnologias e a facilidade de se abrir uma fintech atualmente, pois um dos fatores que confere desenvolvimento ao banco é justamente o seu menor custo de captação com depósitos varejistas, maior receita com juros, facilidade de acesso no meio digital, e, ainda, a presença de agências físicas, ainda mais no ambiente C, D, E em que atua o Banco Pan, o que aumenta a ameaça de produtos substitutos. Assim, aumenta-se a rivalidade com concorrentes

que acabam focando em uma estratégia de isenção de taxas, nichamento em poucos produtos financeiros que operam bem e melhor experiência do usuário.

Sob a perspectiva do poder de barganha com os clientes, tende-se a observar um alto custo de aquisição do cliente e fraca aderência ao produto, tendo em vista a diversidade de fintechs no mercado com os mesmos atrativos e mesmas funções. Ao passo que, numa relação de poder de compra com fornecedores, o setor bancário - e o Banco Pan em si - estaria mais focado em negociar com credores o financiamento mais barato para a sua expansão e aquisição de mercado, mesmo com os sucessivos aumentos das taxas de juros, o que dificulta esse movimento. Quanto ao modelo de negócios do Banco Pan, assim como dos seus principais concorrentes, estes focam em desburocratizar os serviços financeiros e facilitar a vida do consumidor por meio da tecnologia por meio de um melhor atendimento, isto é, personalizado, e inovar nos produtos fornecidos. Por fim, dentro do modelo de negócios bancário atual é evidente o uso de modelos preditivos, para definirem o tipo do cliente, o seu perfil de investimento, o local onde estão mais propensos a serem adquiridos, entre outros fatores em que a inteligência artificial consegue ser atribuída.

4.1.2. Análise SWOT



Imagen 1 - Matriz SWOT

4.1.3. Planejamento Geral da Solução

Dados Disponíveis

- anomes: Ano e mês dos dados referentes à pessoa

- num_cpf_hash: número de CPF hashado de clientes do Banco Pan e retirados também do Open Banking
- vlr_credito: Crédito que o cliente tem disponível em todo o mercado, retirado do Open Banking
- vlr_saldo: valor que o cliente deve ao Banco Pan (saldo + crédito), os identificados com NaN não são clientes, fornecidos pelo Banco Pan
- num_atend_atrs: Número de atendimentos que estão fora do prazo estipulado pelo Banco Pan de 1 ano, fornecidos pelo Banco Pan
- vlr_score: Número de score do mercado inteiro de cada pessoa, fornecido pelo Open Banking
- num_produtos: Quantidade de produtos relacionados à crédito contratados por um cliente dentro do Banco Pan e fornecidos pelo Banco Pan
- qtd_oper: Quantidade de operações realizadas (relacionadas à crédito) com os produtos contratados referentes ao Banco Pan
- qtd_reclm: Quantidade de reclamações realizadas pelo cliente referente ao Banco Pan
- qtd_restr: Quantidade de restritivos (pendências) que uma pessoa tem no mercado, fornecido pelo Open Banking
- vlr_renda: Valor total de renda que uma pessoa tem no mercado, fornecido pelo Open Banking
- cod_rating: Classificação do cliente no Banco Pan, indica o quanto bom cliente é (se não tem, não é cliente do Pan)
- ind_atrito: indica se o cliente do Banco Pan é (1) ou não (0) atritado
- ind_engaj: indica se o cliente do Banco Pan é (1) ou não (0) engajado
- ind_novo_cli: indica se uma pessoa é (1) ou não (0) um possível novo cliente do Banco Pan

Solução Proposta

Aprendizado de Máquina e Inteligência Artificial que seja capaz de analisar o banco de dados de clientes, classificá-los em atritado, engajado e novo cliente e identificar qual a necessidade do atendimento de cada cliente ou não cliente e com isso retornar ao Banco Pan.

Tipo de tarefa

O modelo preditivo a ser desenvolvido será de classificação, onde irá indicar se um cliente é engajado, atritado ou um possível novo cliente.

Modo de utilização da solução proposta

A solução proposta deverá ser utilizada na área de atendimento do Banco Pan, para que estes tenham um conhecimento prévio sobre o tipo de cliente em que estão lidando e a possibilidade de uma melhoria no atendimento.

Benefícios da solução proposta

A partir da classificação do cliente pelo modelo, de acordo com as características do cliente, auxiliará no atendimento a estes, fornecendo um melhor caminho e agilidade ao tratar cada cliente.

Critério de sucesso e métrica de avaliação

Classificar o cliente como atritado, engajado ou novo cliente e como critério de sucesso será avaliado se a porcentagem de acertos em relação ao modelo dado for maior do que 80%.

4.1.4. Value Proposition Canvas

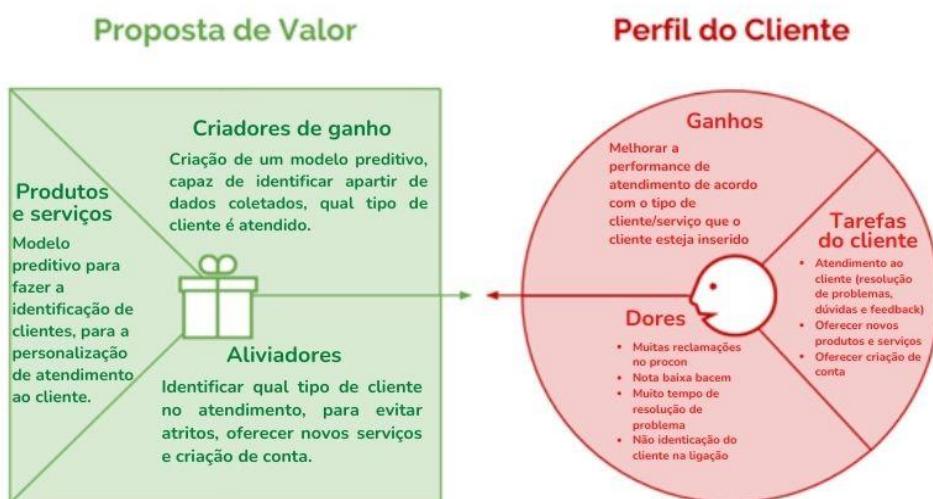


Imagen 2 - Canvas Value Proposition

4.1.5. Matriz de Riscos

Matriz de Risco						
Probabilidade	Ameaça			Oportunidade		
Alta	Conhecimento limitado do assunto			Automatização no atendimento	Numero alto de usuários	
Médio		Código oferecer direcionamento errado para o atendimento	Dataset incompleto	Fidelização do usuário	Muitas reclamações / clientes atritados	
Baixa	Faltar com ética no código	Vazamento de dados	Não concluir o projeto	Índice Bacen piorar		
	Baixo	Médio	Alta	Alta	Médio	Baixo
	Impacto					

Imagen 3 - Matriz de Risco

4.1.6. Personas

Afetadas pelo modelo:

Nós desenvolvemos todos os dias produtos que atendem a necessidades de higiene, cuidados pessoais e nutrição, com marcas que ajudem as pessoas a se sentir bem, parecer bem e aproveitar melhor a vida.

Sophia

Sophia quer resolver seus problemas com o banco.



Descrição

Sophia tem 23 anos e abriu a sua conta há 2 anos, ela ainda mora com os seus pais e sempre gostou da sua experiência com o Pan, mas nos últimos 3 meses ela está com dificuldade em mudar a sua senha do cartão.

Maiores objetivos

Conseguir trocar a senha do seu cartão de crédito
Comprar o seu carro
Estabilidade financeira
Ser efetivada no seu estágio

Personalidade

- Gosta de Funk
- Muito paciente
- Adora conhecer bares novos

Interesses

- Academia
- Livros
- Podcasts
- TikTok

Apps usados pela Sophia

- Banco Pan
- Uber
- Instagram
- WhatsApp

Ganhos

- Facilidade de atendimento via app

Dores

- Muito tempo gasto para solucionar problemas

Imagen 4 - Persona

Sophia tem 23 anos e sua primeira conta em uma instituição financeira foi com o Banco Pan. Infelizmente ela está com problemas em mudar a sua senha do cartão, as ligações demoram muito e não resolvem seu problema, que já persiste a 3 meses.

Ivone



Ivone quer adquirir produtos com o banco.

💡 Brief description

Ivone tem 63 anos, está trabalhando na área de engenharia civil e em processo de aposentadoria e precisa de um banco para guardar sua renda de forma segura.

😊 Personality

- Introvertida
- Gosta de gatos

👀 Interests

- Tecnologia
- Conhecimento
- Natureza
- Animais
- Cozinhar

📲 Apps used by Ivone

- Petlove
- Petz
- Facebook
- Pinterest

💰 Ganhos

- Conta rende mais que a poupança
- Segurança contra roubos na rua
- Confabilidade no nome do banco

😔 Dores

- Insegurança em investir o seu dinheiro
- Precisa de um banco de forma rápida
- Não tem muito tempo para sair de casa e ir resolver as coisas do banco

Imagen 5 - Persona

Ivone está em processo de aposentadoria e já é cliente do Banco Pan, por isso deseja contratar novos produtos que condizem com sua atual condição financeira e de vida, atendendo de forma fácil suas necessidades.

Lucca

Lucca é um possível novo cliente.



Brief description

Lucca é um jovem entusiasmado com a tendência atual do mundo tech, então ele quer abrir uma conta num banco digital que supra suas principais necessidades

Personality

- Geek
- Extrovertido
- Curioso

Interests

- Tecnologia
- Festas
- Sair com os amigos
- Redes sociais

Apps used by Lucca

- Discord
- Instagram
- Twitter
- Reddit

Maiores objetivos

- Lucca tem como principal objetivo se tornar CEO de uma empresa.

Ganhos

- Facilidade em encontrar suas soluções bancárias
- Rapidez em abertura de conta

Dores

- Não tem cartão de crédito ainda
- Foi negado nos demais bancos

Imagen 6 - Persona

Lucca já tem contas em outros bancos, mas não está satisfeito com nenhum deles. Como é jovem, quer um banco que esteja conectado com o mundo atual, seja tecnológico e tenha tudo o que precisa na palma de sua mão.

Que usará o modelo:

Beatriz

Beatriz é uma atendente do banco Pan.



👉 Descrição

Beatriz possui 22 anos, trabalha numa empresa terceirizada que presta serviço para o Banco Pan e tem sonhos de cursar enfermagem.

😊 Personalidade

- Muito amigável
- Muito paciente
- Aplicada

👀 Interesses

- Gosta de comunicação
- Gosta de jogar vôlei

📲 Apps usados pela Sophia

- Banco Pan
- Uber
- Instagram
- WhatsApp

😊 Ganhos

- Conseguirá identificar qual tipo de cliente está lidando e saberá qual postura deve ter.

😢 Dores

- Não sabe se um cliente está procurando um novo produto ou quer fazer reclamações.

Imagen 7 - Persona

4.1.7. Jornadas do Usuário

Mapas de usuários que são afetados pelo modelo:

Sophia: Atritado

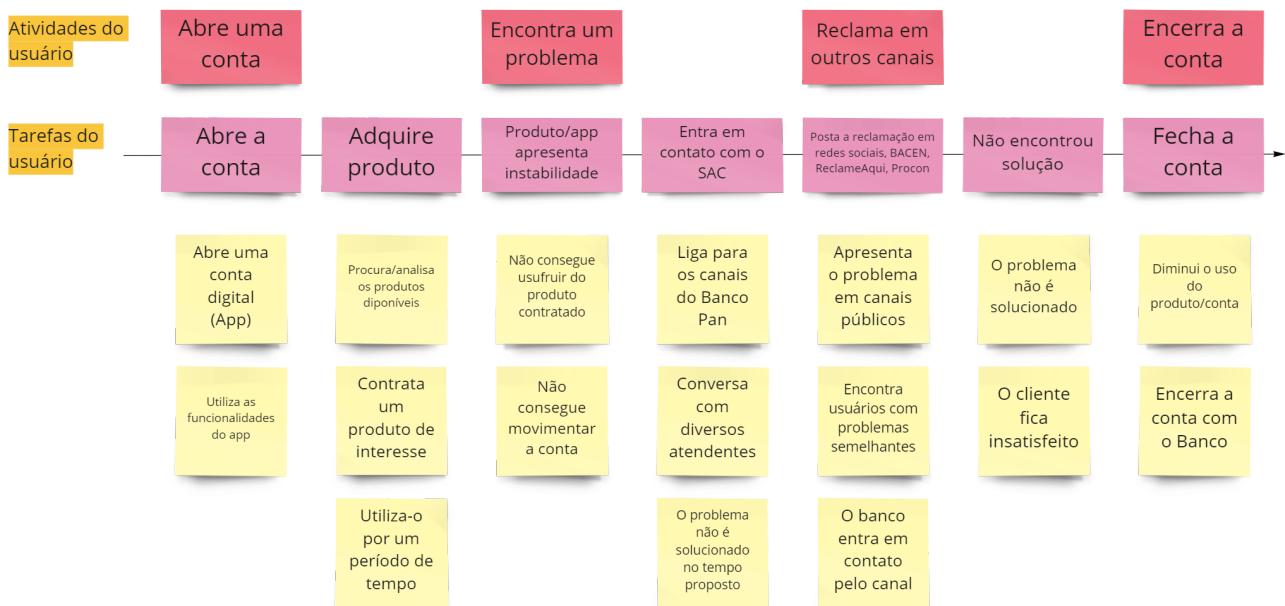


Imagen 8 - Jornada do usuário 1

Ivone: adquirir novos produtos

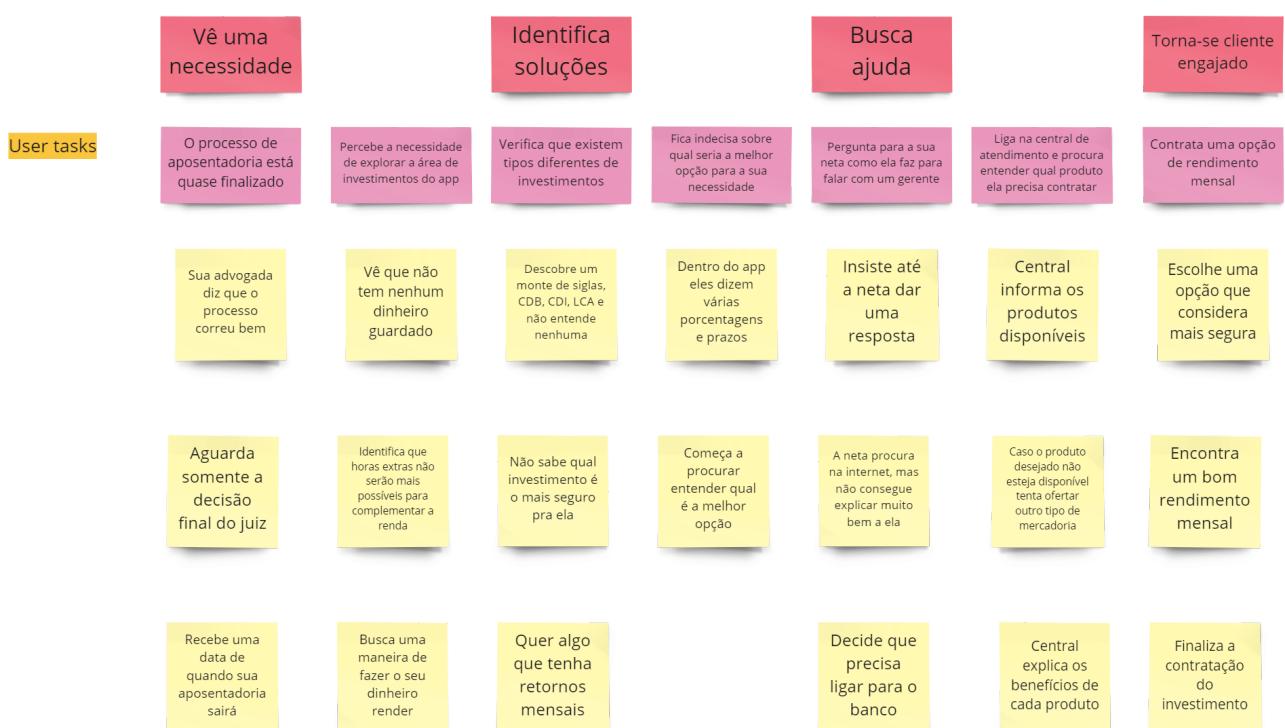


Imagen 9 - Jornada do Usuário 2

Lucca: Novo cliente

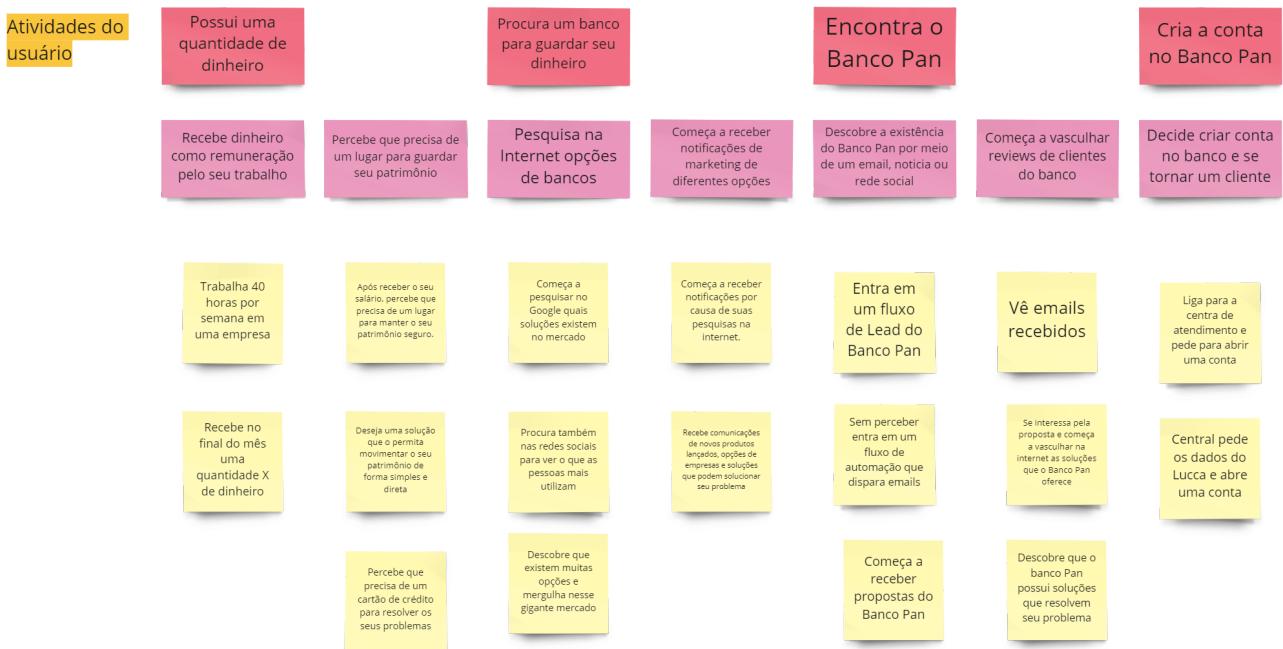


Imagen 10 - Jornada do Usuário 3

Jornada do usuário que utiliza o modelo:

Beatriz: Atendente

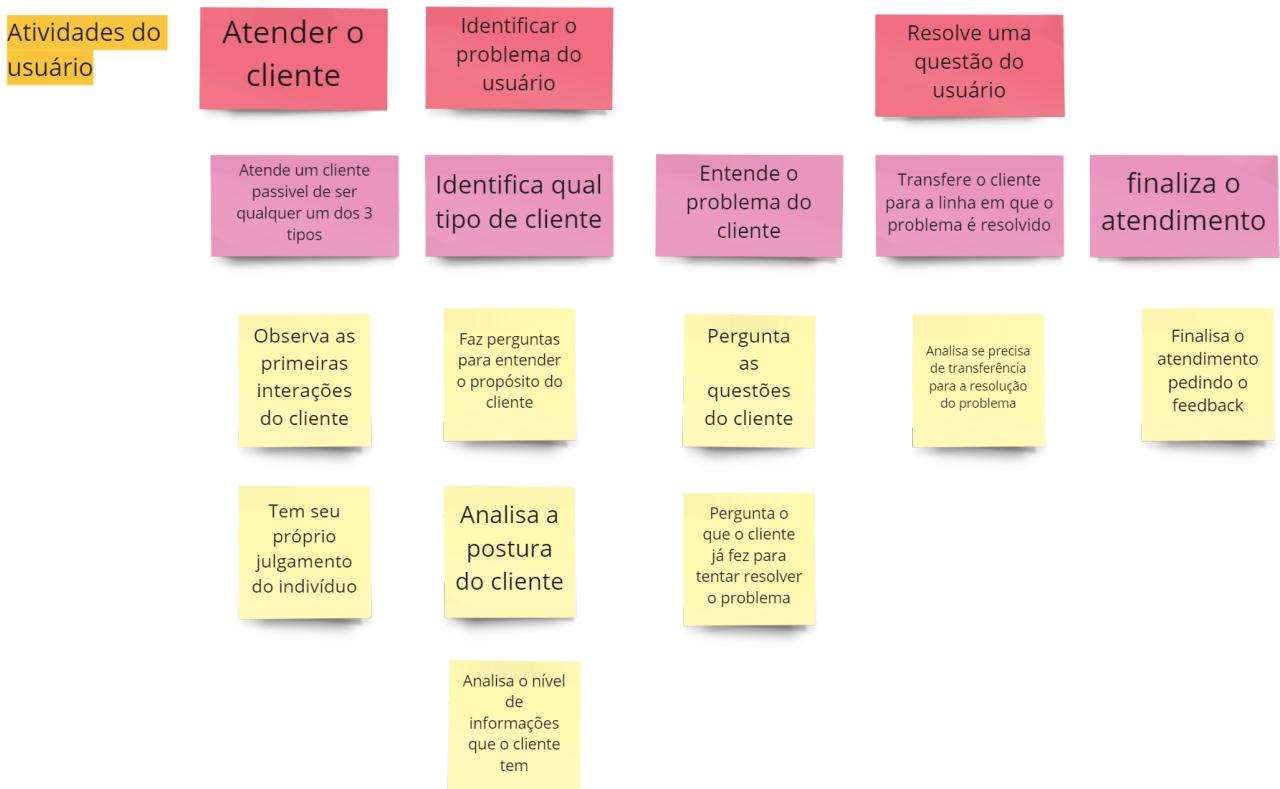


Imagen 11 - Jornada do usuário 4

4.2. Compreensão dos Dados

Descrição dos dados

Conforme enviado pelo stakeholder (Banco Pan), os dados utilizados são no formato CSV, com mais de 12 milhões de linhas (1.424 GB), divididas por safras mensais, contendo informações sobre: Valor de Crédito, Valor de Saldo, Número de Atendimentos Atrasados, Valor de Score, Número de Produtos, Número de Atendimentos, Número de CPF, Quantidade de Operações, Quantidade de Reclamações, Quantidade de Restritivos, Valor de Renda e Cod Rating. Desses dados primeiramente fornecidos, os dados referentes ao Valor de Crédito, Valor de Score, Valor de Renda e Cod Rating referem-se ao mercado, e não restritivamente ao Banco Pan.

Informações importantes a respeito dos dados

O Banco Pan forneceu apenas um conjunto de dados.

Apesar de possuirmos uma quantidade considerável de dados para serem analisados e contribuírem para a inteligência a ser criada (mais de 12 milhões de linhas), algumas das colunas possuem um número extremamente elevado de linhas com valor nulo. São elas, o número de atendimentos atrasados (“num_atend_atrs”), o número de atendimentos totais (“num_atend”), quantidade de reclamações (“qtd_reclm”), indicador de índice atritado (“ind_atritado”), indicador de cliente engajado (“ind_engajado”), indicador novo cliente (“ind_novo_cliente”), Valor de saldo (“vlr_saldo”), número de produtos (“num_produtos”), quantidade de restrições (“qtd_restr”), quantidade de operações (“qtd_oper”).

Foi confirmado com o cliente que os dados nulos podem ser considerados pela nossa inteligência com um valor default. Além disso, analisamos a estruturação da base de dados que recebemos e tentamos realmente compreender como poderemos utilizar os valores de cada coluna para a criação do nosso sistema de predição. Também definimos quais outras informações poderiam ser utilizadas para o nosso modelo de inteligência para que possamos fazer uma requisição de novos dados ao nosso cliente (Banco Pan). Para valores Nan em vlr_score, o grupo optou por fazer as médias de valores para os valores vazios.

Como mostrado na seção 4.3, selecionamos cada coluna que iremos utilizar para a criação de nosso modelo e listamos o motivo da escolha de cada uma. Nessa mesma seção, descrevemos melhor como foi feita a manipulação dos dados, a agregação de linhas e a remoção dos valores que não seriam utilizados. Esperamos que, a partir das features que escolhemos, possamos alcançar uma precisão acima de 90% de sucesso.

Por se tratar dos dados de um banco, é necessário que nós tenhamos um cuidado ainda mais elevado para manter a base de dados o mais privada possível. Dito isso, não podemos, de forma alguma, copiar essas informações para alguma forma de armazenamento que seja externa ao ambiente do Inteli.

Descrição estatística básica dos dados

Abaixo há a descrição estatística básica dos dados, priorizando os atributos de interesse. Segue imagens das tabelas contendo a descrição estatística básica de cada coluna em nossa base de dados

Base de dados normal:

	anomes	vlr_credito	vlr_saldo	num_atend_atrs	vlr_score	num_produtos	num_atend	qtd_oper	qtd_reclm	qtd_restr	vlr_renda	ind_atrito	ind_engaj	ind_novo_cli
count	860680.0	8.60680e+05	4.447680e+05	860680.000000	851123.000000	477651.000000	860680.000000	468603.000000	860680.000000	604604.000000	0.0	860680.000000	860680.000000	860680.000000
mean	202204.0	1.865143e+04	5.754732e+03		0.000675	440.868065	1.777785	0.002469	12.580972	0.000066	3.044331	NaN	0.000251	0.172231
std	0.0	5.497118e+04	1.033775e+04		0.029453	205.964544	1.188577	0.064618	11.026851	0.008138	3.792886	NaN	0.015840	0.377582
min	202204.0	0.000000e+00	1.000000e-02		0.000000	0.000000	1.000000	0.000000	0.000000	0.000000	1.000000	NaN	0.000000	0.000000
25%	202204.0	0.000000e+00	9.997700e+02		0.000000	304.000000	1.000000	0.000000	5.000000	0.000000	1.000000	NaN	0.000000	0.000000
50%	202204.0	2.225550e+02	2.435460e+03		0.000000	402.000000	1.000000	0.000000	10.000000	0.000000	2.000000	NaN	0.000000	0.000000
75%	202204.0	1.746405e+04	6.724953e+03		0.000000	575.000000	2.000000	0.000000	17.000000	0.000000	4.000000	NaN	0.000000	0.000000
max	202204.0	1.034811e+07	1.666399e+06		6.000000	1000.000000	14.000000	9.000000	265.000000	1.000000	217.000000	NaN	1.000000	1.000000

Imagen 12 - Tabela

Base de dados de clientes:

	anomes	vlr_credito	vlr_saldo	num_atend_atrs	vlr_score	num_produtos	num_atend	qtd_oper	qtd_reclm	qtd_restr	vlr_renda	ind_atrito	ind_engaj	ind_novo_cli
count	444768.0	4.447680e+05	4.447680e+05	444768.000000	435423.000000	402667.000000	444768.000000	388815.000000	444768.000000	210194.000000	0.0	444768.000000	444768.000000	444768.0
mean	202204.0	3.409334e+04	5.754732e+03		0.001248	527.216582	1.862276	0.004609	14.113447	0.000124	3.348830	NaN	0.000463	0.333288
std	0.0	7.034999e+04	1.033775e+04		0.040187	221.863178	1.238613	0.088564	11.115636	0.011120	3.815523	NaN	0.021516	0.471389
min	202204.0	0.000000e+00	1.000000e-02		0.000000	0.000000	1.000000	0.000000	0.000000	1.000000	NaN	0.000000	0.000000	0.0
25%	202204.0	3.040588e+03	9.997700e+02		0.000000	374.000000	1.000000	0.000000	7.000000	0.000000	1.000000	NaN	0.000000	0.000000
50%	202204.0	1.546866e+04	2.435460e+03		0.000000	504.000000	1.000000	0.000000	12.000000	0.000000	2.000000	NaN	0.000000	0.000000
75%	202204.0	3.630122e+04	6.724953e+03		0.000000	682.000000	2.000000	0.000000	18.000000	0.000000	4.000000	NaN	0.000000	1.000000
max	202204.0	1.034811e+07	1.666399e+06		6.000000	1000.000000	14.000000	9.000000	265.000000	1.000000	110.000000	NaN	1.000000	1.000000

Imagen 13 - Tabela clientes Banco Pan

Base de dados de não clientes:

	anomes	vlr_credito	vlr_saldo	num_atend_atrs	vlr_score	num_produtos	num_atend	qtd_oper	qtd_reclm	qtd_restr	vlr_renda	ind_atrito	ind_engaj	ind_novo_cli
count	415912.0	4.159120e+05	0.0	415912.000000	415700.000000	74984.000000	415912.000000	79788.000000	415912.000000	394410.000000	0.0	415912.000000	415912.0	415912.000000
mean	202204.0	2.138149e+03	NaN	0.00063	350.422718	1.324069	0.000180	5.113062	0.000005	2.882054	NaN	0.000024	0.0	0.410902
std	0.0	2.081214e+04	NaN	0.008205	138.948483	0.718572	0.015582	6.693122	0.002193	3.770741	NaN	0.004903	0.0	0.491998
min	202204.0	0.000000e+00	NaN	0.000000	0.000000	1.000000	0.000000	0.000000	0.000000	1.000000	NaN	0.000000	0.0	0.000000
25%	202204.0	0.000000e+00	NaN	0.000000	246.000000	1.000000	0.000000	1.000000	0.000000	1.000000	NaN	0.000000	0.0	0.000000
50%	202204.0	0.000000e+00	NaN	0.000000	353.000000	1.000000	0.000000	3.000000	0.000000	2.000000	NaN	0.000000	0.0	0.000000
75%	202204.0	0.000000e+00	NaN	0.000000	428.000000	1.000000	0.000000	7.000000	0.000000	3.000000	NaN	0.000000	0.0	1.000000
max	202204.0	7.638867e+06	NaN	2.000000	989.000000	6.000000	4.000000	115.000000	1.000000	217.000000	NaN	1.000000	0.0	1.000000

Imagen 14 - Tabela não clientes Banco Pan

Hipótese 1

Clientes com o score mais alto possuem um valor de crédito no banco pan maior. Hipótese não comprovada.

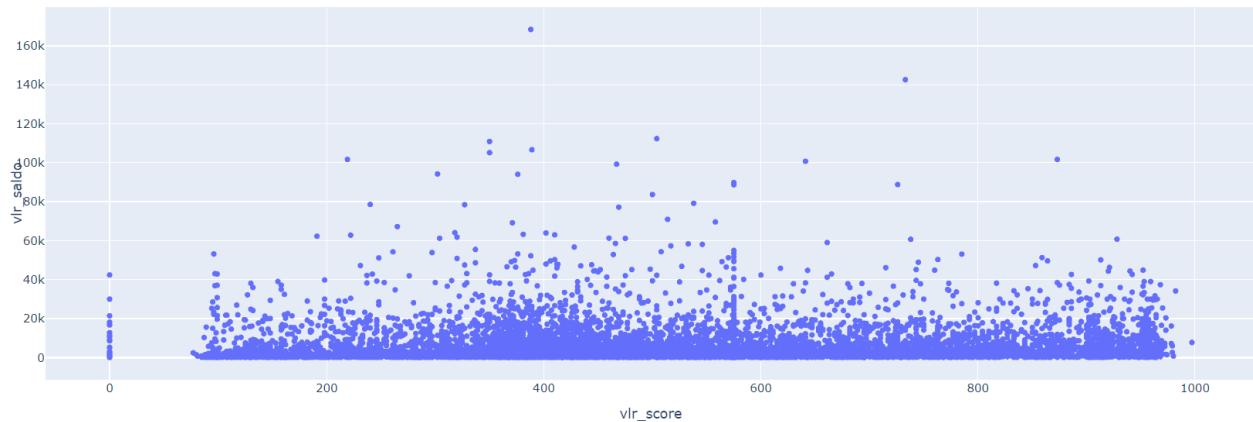


Imagen 15 - Gráfico

Hipótese 2

O banco pan possui mais clientes com classificação A do que H. Hipótese comprovada.

```
client.cod_rating.describe()
```

	count	unique	top	freq
Name:	435627	10	A	330229
dtype:	object			

Imagen 16 - Descrição da tabela de clientes

Hipótese 3

Como o banco atende a clientes de baixa renda, acreditamos que eles contratem vários produtos com baixos valores. Hipótese comprovada.

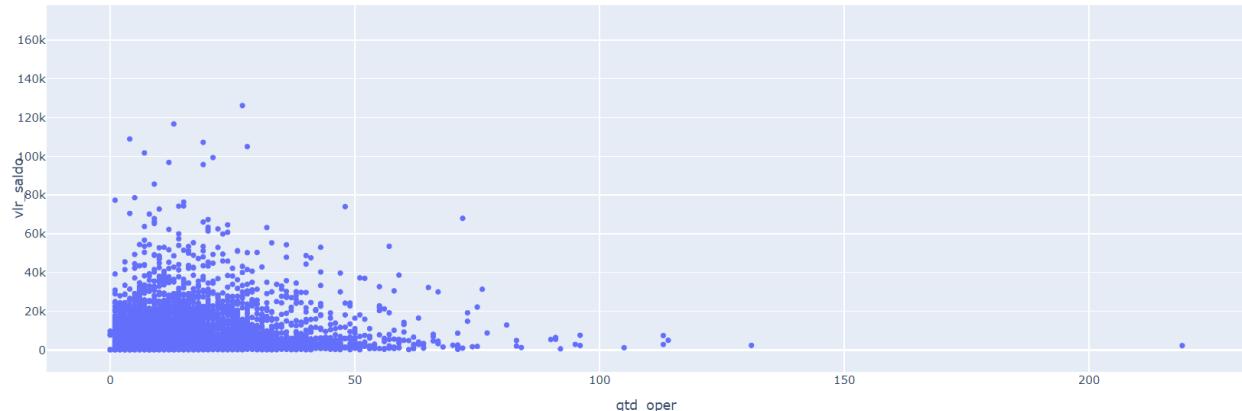


Imagen 17 - Gráfico

Target do modelo

O modelo de predição tem como objetivo (“target”), determinar se o cliente que está entrando em contato com o banco está pendendo para **Atritado, Engajado ou Novo cliente**. A natureza dessa predição será discreta, pois os dados que utilizaremos para o modelo também serão discretos. Além disso, utilizar a predição discreta se encaixa melhor na solução implementada, pois a predição irá categorizar o cliente.

4.3. Preparação dos Dados

Descrição das etapas realizadas para definir os dados e os atributos descritivos dos dados (“features”) a serem utilizados. Essa descrição foi feita de modo a garantir uma futura reprodução do processo por outras pessoas.

Descrição de quaisquer manipulações necessárias nos registros e suas respectivas features.

Atualmente nossa base de dados conta com cerca de 13 milhões de linhas. De acordo com o Rafael (Banco Pan), cada milhão de linhas representa um mês do ano. Dito isso, pretendemos agrupar as linhas de acordo com o mês que representam. Para desenvolver o modelo preditivo, o grupo irá utilizar os dados de apenas um mês.

O parceiro também mencionou que a coluna “cod_rating” pode ser utilizada para separar a nossa base de dados entre clientes e não clientes, já que as linhas com os valores dessa coluna vazios se referem a não clientes e, por isso, dividimos os dados em que estamos analisando em duas tabelas.

Como mostrado na imagem a seguir, também identificamos que a coluna “vlr_renda” não estava formatada da maneira correta. Ela possuía a tipagem String sendo que deveria ser um Float. Por isso foi necessário que fizéssemos uma manipulação para atribuir a ela a tipagem que queríamos (Float).

df.describe(include="object")			
	num_cpf_hash	vlr_renda	cod_rating
count	860680	0	444768
unique	860680	0	10
top	ffffd54b45ec46113523184fc07185a0d5cbfa876a07ba...	NaN	A
freq	1	NaN	331447

Imagen 18 - Descrição dos objetos da tabela

Como deve ser feita a agregação de registros e/ou derivação de novos atributos?

Uma agregação possível será classificar o “cod_rating” através de rankings, como, por exemplo, de A-B de C-E e F-H e, com isso identificar padrões entre essa coluna e se o cliente é atritado ou não.

Como devem ser removidos ou substituídos valores ausentes/em branco.

Após analisarmos a base de dados, identificamos diversas colunas que possuem muitos valores vazios (NaN). Dito isso, analisamos cada coluna separadamente utilizando o método unique() e concluímos que os valores nulos das seguintes colunas podem ser preenchidos com valores padrões (“0”).

- num_atend_atrs
- qtd_reclm
- ind_atritado
- ind_engajado
- ind_novo_cliente
- num_atend

Identificação das features selecionadas, com descrição dos motivos de seleção.

Colunas a serem utilizadas para definir se um cliente é atritado:

```
'vlr_saldo', 'vlr_credito', 'num_atend_atrs', 'qtd_reclm', 'vlr_renda'
```

- qtd_reclm: Quantidade de reclamações desse cliente. É um indicador de cliente atritado, pois quanto mais reclamações um cliente possui, maior é a probabilidade dele ser atritado com o banco.
- num_atend_atrs: Número de atendimentos atrasados. É um indicador de cliente atritado, pois quanto mais atendimentos atrasados existem, maior a probabilidade daquele cliente estar irritado com o banco.
- vlr_credito: Quantidade de renda que possui no mercado.
- vlr_saldo: Indica quanto crédito o cliente possui no banco. Pode ser utilizado para checar se clientes com o saldo maior têm mais probabilidade de serem atritados ou não.
- vlr_renda: Valor declarado pelo cliente de quanto é a sua renda por mês.
- ind_atritado: Coluna com valores certos para treinar a inteligência.

Colunas a serem utilizadas segundo o KBest do SKLearn:

Após analisarmos a base de dados e identificarmos as features de modo manual, usamos a biblioteca SKLearn e a função KBest que identifica dentro do Dataframe quais não as colunas que mais se correlacionam com a resposta que precisamos.

Atritado:

- vlr_saldo: Indica quanto crédito o cliente possui no banco.
- num_atend_atrs: número de ligações atendidas pelo Pan e não resolvidas dentro de 1 mês.
- qtd_reclm: quantidade de reclamações.
- vlr_renda: Valor declarado pelo cliente de quanto é a sua renda por mês.

Engajado:

- vlr_credito: valor do crédito de mercado.
- vlr_saldo: Indica quanto crédito o cliente possui no banco.
- num_produtos: número de produtos de crédito contratados no banco Pan
- qtd_oper: Quantidade de renda que possui no mercado.

Novo cliente:

- vlr_credito: valor do crédito de mercado.
- vlr_score: Score no mercado (0 a 1000).

- qtd_restr: Restritivos (pendências no mercado).
- vlr_renda: Quantidade de renda que possui no mercado

Colunas a serem utilizadas para definir se um cliente é engajado:

- num_produtos: Quantidade de produtos contratados no banco. É um indicador de cliente engajado pois, quanto mais produtos o cliente contratar, maior será a chance dele contratar um novo produto, tendo em vista que já existe uma confiança do cliente no banco.
- qtd_oper: Quantidade de operações realizadas. Quanto maior o número de operações de um cliente no banco, mais ele é engajado dentro do banco.
- vlr_credito: valor do crédito de mercado. Um cliente com maior crédito no mercado tem maiores chances de também ser um cliente que queira contratar crédito do banco.
- vlr_saldo: Indica quanto crédito o cliente possui no banco. É um indicador de cliente engajado, pois quanto maior for o valor de crédito, maior vai ser a interação do cliente com o banco.
- ind_engajado: Coluna com valores certos para treinar a inteligência.

Colunas a serem utilizadas para definir se um registro pode ser um potencial novo cliente:

- vlr_renda: Valor declarado pelo cliente de quanto é a sua renda por mês.
- vlr_credito: Indica o crédito fora do banco que o cliente tem. É um indicador de novo cliente pois dependendo do valor pode ser um cliente em busca de mais crédito ou financiamento, a procura de novos serviços.
- vlr_score: Indica o Score do mercado. É um indicador de novo cliente, pois dependendo do valor o banco aceita fornecer crédito a ele, ou não.
- qtd_restr: Pendências do mercado. É um indicador, pois quanto menor o número de restrições maior a chance do banco querer fornecer crédito a ele.
- ind_novo_cliente: Indicador do potencial novo cliente (preditivo).

4.4. Modelagem

Foram realizados testes em 6 distintos modelos, para cada tipo de cliente: atritado, engajado e novo cliente.

Todos os modelos testados foram escolhidos com base nos estudos previstos pelos professores e com base no método de tentativa e erro.

4.4.1 Atritado

Primeiro, foram separados os dados para treinamento e teste, de acordo com as colunas selecionadas no Feature Engineering:

```
[55] from imblearn.over_sampling import SMOTE  
  
cols = ['vlr_saldo', 'vlr_credito', 'num_atend_atrs', 'qtd_reclm', 'vlr_renda']  
  
# Dividindo x e y  
x = client[cols]  
y = client['ind_atrito']  
  
sm = SMOTE(random_state=42, sampling_strategy="auto")  
X_res, y_res = sm.fit_resample(x, y)
```

Divisão entre treinamento e teste

```
[56] from sklearn.model_selection import train_test_split  
import matplotlib.pyplot as plt  
  
# Dividindo dados para treino e dados para teste  
x_train, x_test, y_train, y_test = train_test_split(X_res, y_res, test_size = 0.2, random_state = 42)
```

Imagen 19 - Código de importação de biblioteca

Após a separação, foram testados diferentes modelos de classificação, assim como métricas para avaliação:

A) KNN

```
from sklearn.neighbors import KNeighborsClassifier  
  
kn_n = KNeighborsClassifier(n_neighbors=3)  
kn_n.fit(x_train, y_train)
```

KNeighborsClassifier(n_neighbors=3)

```
y_pred_knn = kn_n.predict(x_test)  
y_pred_knn
```

array([0., 0., 0., ..., 0., 0., 0.])

Imagen 20 - Modelo KNN

```
from sklearn.metrics import accuracy_score  
accuracy_score(y_test, y_pred_knn)
```

```
0.9996948099743281
```

```
from sklearn.metrics import precision_score  
precision_score(y_test, y_pred_knn)
```

```
1.0
```

```
from sklearn.metrics import recall_score  
recall_score(y_test, y_pred_knn)
```

```
0.2916666666666667
```

Imagen 21 - Métricas de avaliação KNN

```
from sklearn.metrics import plot_confusion_matrix  
plot_confusion_matrix(kn_n,x_test,y_test, cmap='Blues')
```

```
<sklearn.metrics._plot.confusion_matrix.ConfusionMatrixDisplay at 0x7f86426edd90>
```

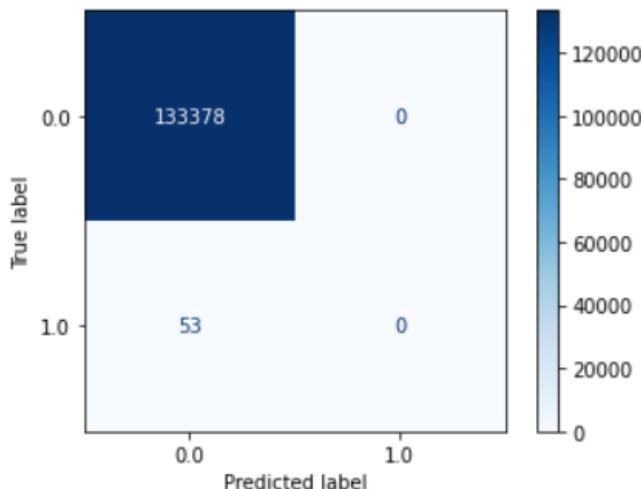


Imagen 22 - Matriz de confusão KNN

Concluindo em um confuso resultado, a princípio, dado que teve uma acurácia de 99%, juntamente com a precisão de 100%, porém com a métrica de recall foi 29%, com uma Matriz de Confusão um tanto tendenciosa. Uma vez que, devido ao baixo número de atritados, o modelo não consegue ter uma real noção desse número e chuta todos como não atritados.

B) Árvore de Decisão

```
from sklearn.model_selection import train_test_split
from sklearn.tree import DecisionTreeClassifier

avd = DecisionTreeClassifier(criterion='entropy', random_state=42)
avd.fit(x_train, y_train)

DecisionTreeClassifier(criterion='entropy', random_state=42)

y_pred_avd = avd.predict(x_test)
y_pred_avd

array([0., 0., 0., ..., 0., 0., 0.])
```

Imagen 23 - Modelo Árvore de Decisão

Acuracia

```
[24] from sklearn.metrics import accuracy_score
accuracy_score(y_test, y_pred_avd)

0.9992819058219485
```

Precisão

```
[25] from sklearn.metrics import precision_score
precision_score(y_test, y_pred_avd)

0.2333333333333334
```

Recall

```
[26] from sklearn.metrics import recall_score
recall_score(y_test, y_pred_avd)

0.2916666666666667
```

Imagen 24 - Métricas de avaliação Árvore de Decisão

```
plot_confusion_matrix(avd,x_test,y_test, cmap='Blues')
```

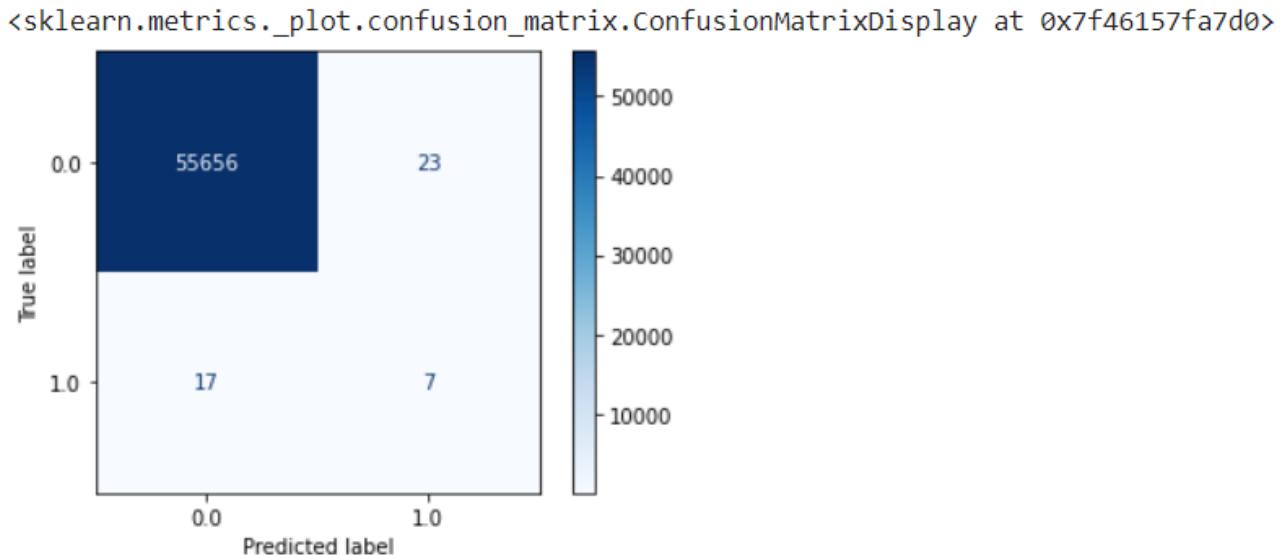


Imagen 25 - Matriz de Confusão Árvore de Decisão

Concluindo em um resultado confuso, a princípio, dado que teve uma acurácia de 99%, porém as métricas de precisão e recall foram de apenas 23% e 29%, respectivamente. Ainda nesse aspecto, manteve uma boa quantificação da Matriz de Confusão, porém com o mesmo problema do quesito apresentado pelo modelo KNN

c) SVM

```
from sklearn import svm
clf = svm.SVC()
clf.fit(x_train, y_train)
clf
```

SVC()

```
y_pred_clf =clf.predict(x_test)
```

y_pred_clf

```
array([0., 0., 0., ..., 0., 0., 0.])
```

Imagen 26 - Modelo SMV

Acuracia

```
[24] from sklearn.metrics import accuracy_score  
accuracy_score(y_test, y_pred_avd)  
  
0.9992819058219485
```

Precisão

```
[25] from sklearn.metrics import precision_score  
precision_score(y_test, y_pred_avd)  
  
0.2333333333333334
```

Recall

```
▶ from sklearn.metrics import recall_score  
recall_score(y_test, y_pred_avd)  
  
👤 0.2916666666666667
```

Imagen 27 - Métricas SMV

```
from sklearn.metrics import confusion_matrix  
plot_confusion_matrix(clf,x_test,y_test, cmap='Blues')
```

```
<sklearn.metrics._plot.confusion_matrix.ConfusionMatrixDisplay at 0x7f85cc8f7b50>
```

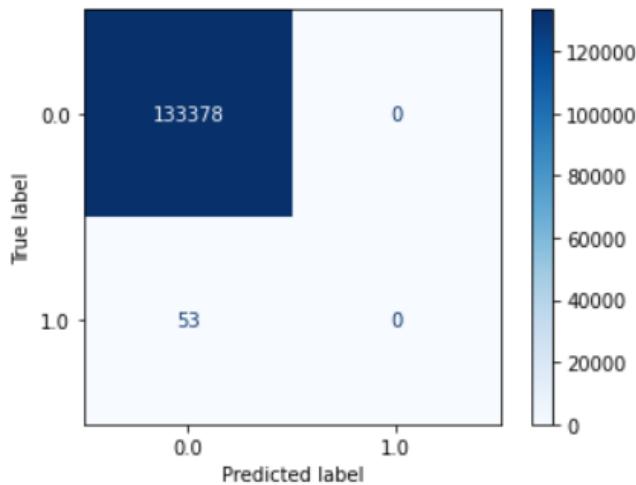


Imagen 28 - Matriz de Confusão SMV

Concluindo em um péssimo resultado, a princípio, dado que teve uma acurácia de 99%, porém as métricas de precisão e recall foram zeradas.

D) Random Forest

```
from sklearn.ensemble import RandomForestClassifier  
from sklearn.metrics import plot_confusion_matrix  
  
rdf = RandomForestClassifier()  
rdf.fit(x_train, y_train)  
  
RandomForestClassifier()  
  
y_pred_rdf = rdf.predict(x_test)
```

Acuracia

```
[37] from sklearn.metrics import accuracy_score  
accuracy_score(y_test, y_pred_rdf)  
  
0.9996948099743281
```

Precisão

```
[38] from sklearn.metrics import precision_score  
precision_score(y_test, y_pred_rdf)  
  
1.0
```

Recall

```
[39] from sklearn.metrics import recall_score  
recall_score(y_test, y_pred_rdf)  
  
0.2916666666666667
```

Imagen 29 - Modelo Random Forest

Imagen 30 - Métricas Random Forest

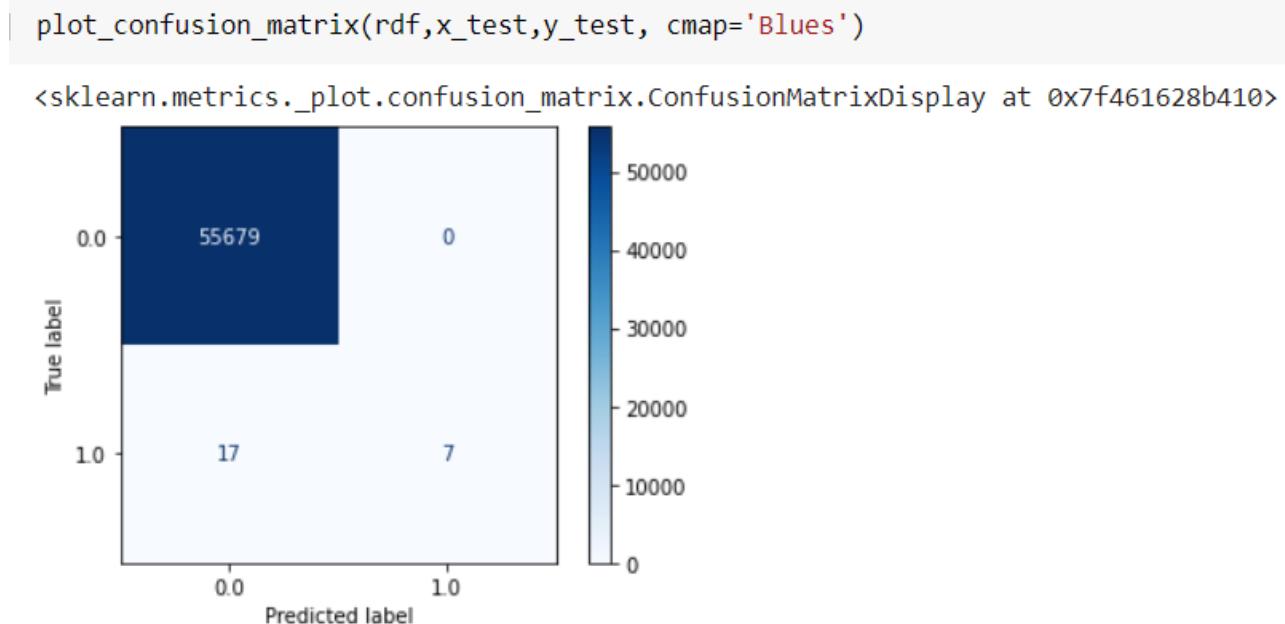


Imagen 31 - Matriz de Confusão Random Forest

Concluindo em um resultado, a princípio, razoável que teve uma acurácia de 99%, precisão de 100% e recall 29%.

E) Naive Bayes

```
from sklearn.naive_bayes import GaussianNB  
nbs = GaussianNB()  
nbs.fit(x_train, y_train)
```



```
GaussianNB()
```

```
y_pred_nbs = rdf.predict(x_test)
```

Imagen 32 - Modelo Naive Bayes

Acuracia

```
[43] from sklearn.metrics import accuracy_score  
accuracy_score(y_test, y_pred_nbs)
```

```
0.9996948099743281
```

Precisão

```
[44] from sklearn.metrics import precision_score  
precision_score(y_test, y_pred_nbs)
```

```
1.0
```

Recall

```
[45] from sklearn.metrics import recall_score  
recall_score(y_test, y_pred_nbs)
```

```
0.2916666666666667
```

Imagen 33 - Métricas Naive Bayes

```
plot_confusion_matrix(nbs,x_test,y_test, cmap='Blues')
```

```
<sklearn.metrics._plot.confusion_matrix.ConfusionMatrixDisplay at 0x7f46162847d0>
```

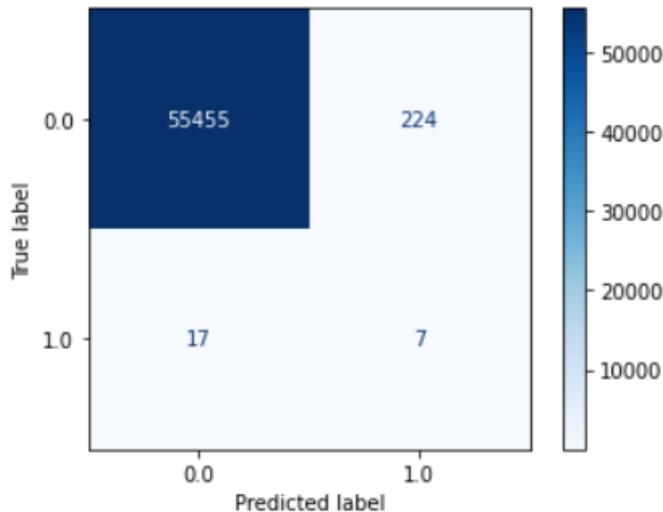


Imagen 34 - Matriz de Confusão Naive Bayes

Concluindo em um resultado, a princípio, razoável que teve uma acurácia de 99%, precisão de 100% e recall 29%.

F) Regressão Logística

```
from sklearn.linear_model import LogisticRegression  
rgl = LogisticRegression().fit(x_train, y_train)
```

```
y_pred_rgl = rgl.predict(x_test)
```

Imagen 35 - Modelo Regressão Logística

Acuracia

```
[ ] from sklearn.metrics import accuracy_score  
accuracy_score(y_test, y_pred_rg1)
```

0.9996948099743281

Precisão

```
[ ] from sklearn.metrics import precision_score  
precision_score(y_test, y_pred_rg1)
```

1.0

Recall

```
▶ from sklearn.metrics import recall_score  
recall_score(y_test, y_pred_rg1)
```

 0.2916666666666667

Imagen 36 - Métricas Regressão Logística

```
plot_confusion_matrix(rgl,x_test,y_test, cmap='Blues')
<sklearn.metrics._plot.confusion_matrix.ConfusionMatrixDisplay at 0x7f85cc4ed210>
```

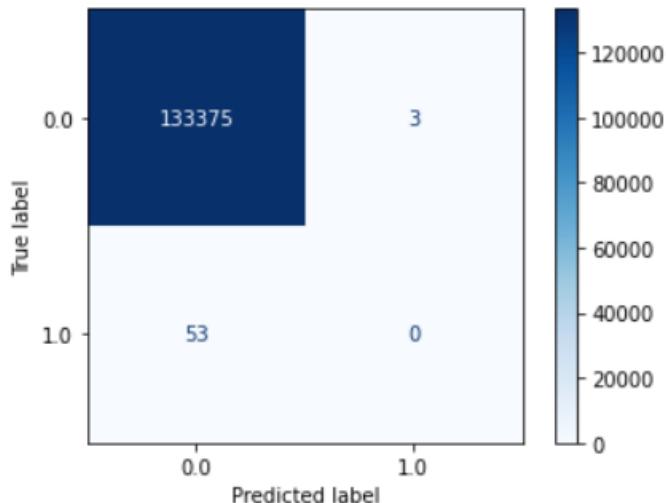


Imagen 37 - Matriz de Confusão Regressão Logística

Concluindo em um resultado, a princípio, razoável que teve uma acurácia de 99%, precisão de 100% e recall 29%.

Todos os resultados dos clientes atritados sofreram uma grande influência do número de clientes atritados no conjunto de dados disponibilizados para o projeto, uma vez que apenas uma pequena parte deles realmente eram atritados, viciando o modelo em alguns casos específicos.

Além disso, os algoritmos que apresentaram um melhor desempenho no conjunto, isto é, melhor precisão, acurácia e recall foram devidamente escolhidos

4.4.2 Novos clientes

Primeiro, foram separados os dados para treinamento e teste, de acordo com as colunas selecionadas no Feature Engineering:

```
from sklearn.model_selection import train_test_split

cols = ['vlr_score', 'vlr_credito', 'num_atend_atrs', 'qtd_restr']

# Dividindo x e y
x = not_client[cols]
y = not_client['ind_novo_cli']

# Dividindo dados para treino e dados para teste
x_train, x_test, y_train, y_test = train_test_split(x, y, test_size = 0.3, random_state = 42)
```

Imagen 38 - Código de importação de biblioteca

Após a separação, foram testados diferentes modelos de classificação, assim como métricas para avaliação:

G) KNN

```
from sklearn.neighbors import KNeighborsClassifier

kn_n = KNeighborsClassifier(n_neighbors=3)
kn_n.fit(x_train, y_train)

KNeighborsClassifier(n_neighbors=3)

y_pred_knn = kn_n.predict(x_test)
y_pred_knn

array([1., 0., 0., ..., 0., 0., 0.])
```

Imagen 39 - Modelo KNN

```
from sklearn.metrics import accuracy_score
accuracy_score(y_test, y_pred_knn)
```

0.9990431160643927

```
from sklearn.metrics import precision_score
precision_score(y_test, y_pred_knn)
```

0.9967266775777414

```
from sklearn.metrics import recall_score
recall_score(y_test, y_pred_knn)
```

0.9997947876051714

Imagen 40 - Métricas de avaliação KNN

```
from sklearn.metrics import plot_confusion_matrix
plot_confusion_matrix(kn_n,x_test,y_test, cmap='Blues')
```

<sklearn.metrics._plot.confusion_matrix.ConfusionMatrixDisplay at 0x7f5d4993c210>

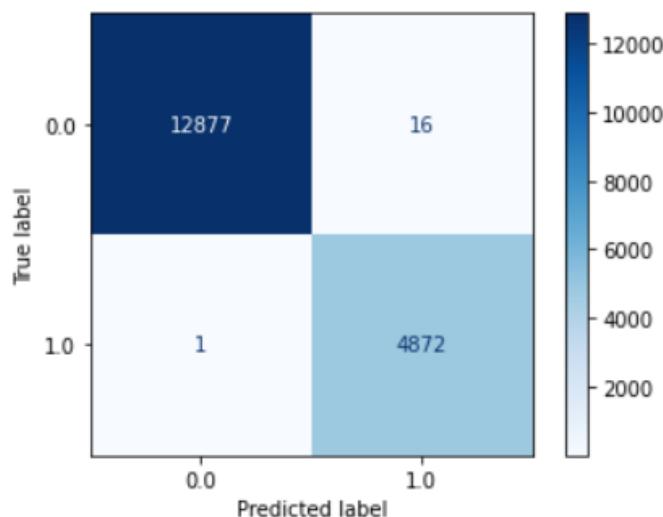


Imagen 41 - Matriz de confusão KNN

Concluindo em um ótimo resultado, a princípio, dado que teve uma acurácia de 99,9%, juntamente com a precisão de 99,9% e recall de 99,9%.

H) Árvore de Decisão

```
from sklearn.model_selection import train_test_split
from sklearn.tree import DecisionTreeClassifier

avd = DecisionTreeClassifier(criterion='entropy', random_state=42)
avd.fit(x_train, y_train)

DecisionTreeClassifier(criterion='entropy', random_state=42)

y_pred_avd = avd.predict(x_test)
y_pred_avd

array([1., 0., 0., ..., 0., 0., 0.])
```

Imagen 42 - Modelo Árvore de Decisão

Acuracia

```
[ ] from sklearn.metrics import accuracy_score  
accuracy_score(y_test, y_pred_avd)
```

1.0

Precisão

```
▶ from sklearn.metrics import precision_score  
precision_score(y_test, y_pred_avd)
```



1.0

Recall

```
[ ] from sklearn.metrics import recall_score  
recall_score(y_test, y_pred_avd)
```

1.0

Imagen 43 - Métricas de avaliação Árvore de Decisão

```
plot_confusion_matrix(avd,x_test,y_test, cmap='Blues')
```

```
<sklearn.metrics._plot.confusion_matrix.ConfusionMatrixDisplay at 0x7f9a083cbf10>
```

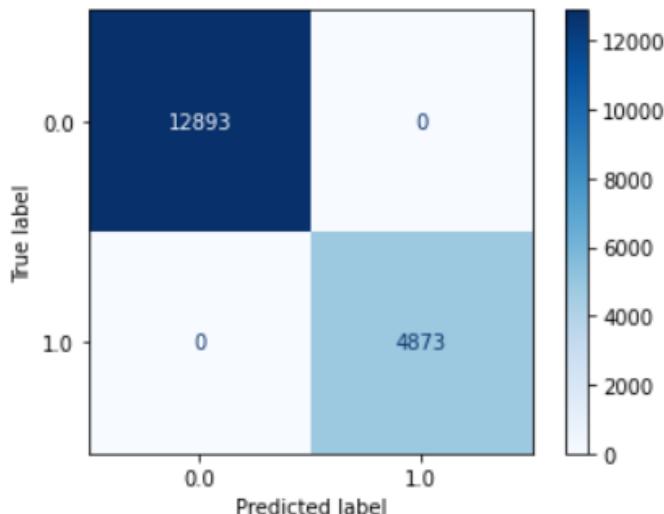


Imagen 44 - Matriz de Confusão Árvore de Decisão

Concluindo em um ótimo resultado, a princípio, dado que teve uma acurácia de 100%, juntamente com a precisão de 100% e recall de 100%.

I) Random Forest

```
from sklearn.ensemble import RandomForestClassifier
from sklearn.metrics import plot_confusion_matrix

rdf = RandomForestClassifier()
rdf.fit(x_train, y_train)

RandomForestClassifier()

y_pred_rdf = rdf.predict(x_test)
```

Imagen 45 - Modelo Random Forest

Acuracia

```
[22] from sklearn.metrics import accuracy_score  
accuracy_score(y_test, y_pred_rdf)
```

1.0

Precisão

```
[23] from sklearn.metrics import precision_score  
precision_score(y_test, y_pred_rdf)
```

1.0

Recall

```
▶ from sklearn.metrics import recall_score  
recall_score(y_test, y_pred_rdf)
```

⇨ 1.0

Imagen 46 - Métricas Random Forest

```
plot_confusion_matrix(rdf,x_test,y_test, cmap='Blues')  
<sklearn.metrics._plot.confusion_matrix.ConfusionMatrixDisplay at 0x7f9a07f9fa50>
```

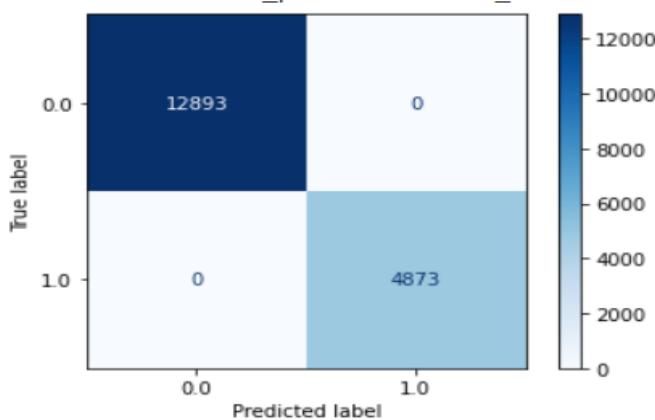


Imagen 47 - Matriz de Confusão Random Forest

Concluindo em um resultado, a princípio, razoável que teve uma acurácia de 99%, precisão de 100% e recall 29%.

J) Naive Bayes

```
from sklearn.naive_bayes import GaussianNB  
nbs = GaussianNB()  
nbs.fit(x_train, y_train)  
  
GaussianNB()  
  
y_pred_nbs = rdf.predict(x_test)
```

Imagen 48 - Modelo Naive Bayes

Acuracia

```
[28] from sklearn.metrics import accuracy_score  
accuracy_score(y_test, y_pred_nbs)  
[] 1.0
```

Precisão

```
[29] from sklearn.metrics import precision_score  
precision_score(y_test, y_pred_nbs)  
[] 1.0
```

Recall

```
[30] from sklearn.metrics import recall_score  
recall_score(y_test, y_pred_nbs)  
[] 1.0
```

Imagen 49 - Métricas Naive Bayes

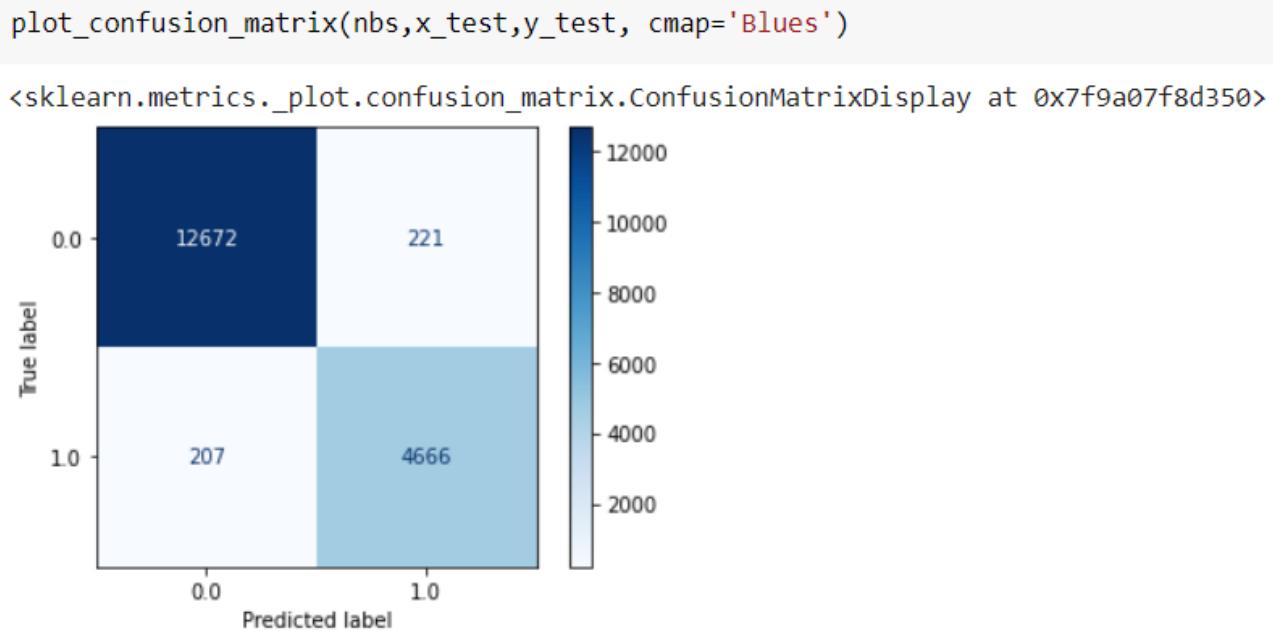


Imagen 50 - Matriz de Confusão Naive Bayes

Concluindo em um ótimo resultado, a princípio, dado que teve uma acurácia de 100%, juntamente com a precisão de 100% e recall de 100%.

K) Regressão Logística

```
from sklearn.linear_model import LogisticRegression  
rgl = LogisticRegression().fit(x_train, y_train)
```

```
y_pred_rgl = rgl.predict(x_test)
```

Imagen 51 - Modelo Regressão Logística

Acuracia

```
[ ] from sklearn.metrics import accuracy_score  
accuracy_score(y_test, y_pred_rgl)
```

0.8526398739164697

Precisão

```
[ ] from sklearn.metrics import precision_score  
precision_score(y_test, y_pred_rgl)
```

0.7830780818478533

Recall

```
[ ] from sklearn.metrics import recall_score  
recall_score(y_test, y_pred_rgl)
```

0.640057459470552

Imagen 52 - Métricas Regressão Logística

```
plot_confusion_matrix(rgl,x_test,y_test, cmap='Blues')
```

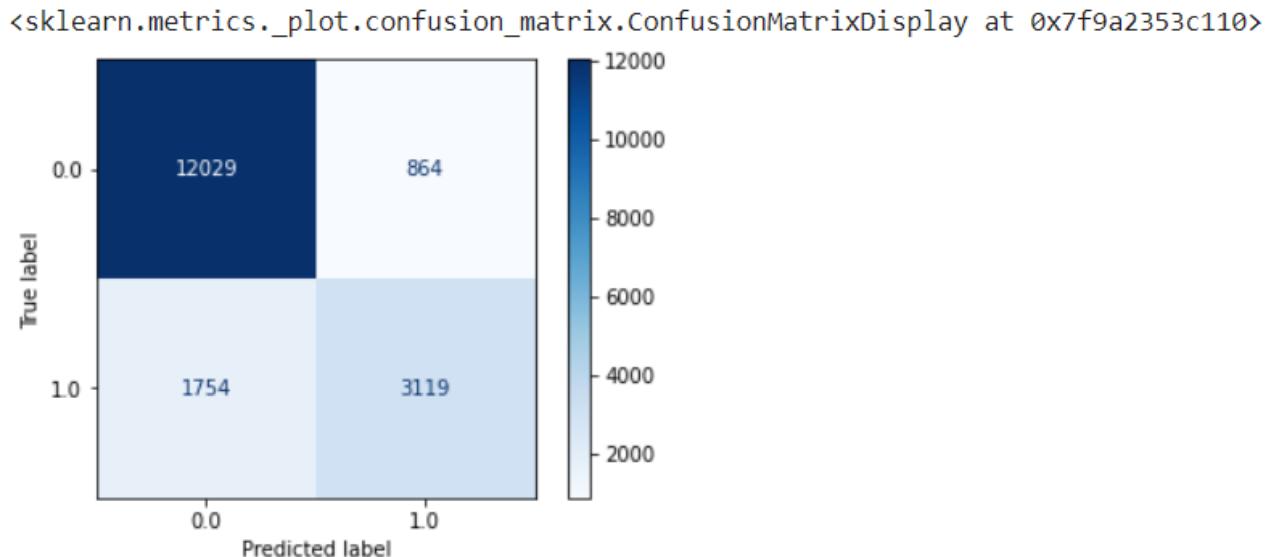


Imagen 53 - Matriz de Confusão Regressão Logística

Concluindo em um resultado, a princípio, muito bom, porém com uma desvantagem em relação a todos os outros, uma vez que as métricas da regressão logística foram acurácia de 85%, precisão de 78% e recall de 64%.

4.4.3 Engajado

Primeiro, foram separados os dados para treinamento e teste, de acordo com as colunas selecionadas no Feature Engineering:

```
from sklearn.model_selection import train_test_split
cols = ['vlr_saldo','vlr_credito', 'num_produtos', 'qtd_oper']
# Dividindo x e y
x = client[cols]
y = client['ind_engaj']
# Dividindo dados para treino e dados para teste
x_train, x_test, y_train, y_test = train_test_split(x, y, test_size = 0.2, random_state = 42)
```

Imagen 54 - Código de importação de biblioteca

Após a separação, foram testados diferentes modelos de classificação, assim como métricas para avaliação:

L) Árvore de Decisão

```
from sklearn.model_selection import train_test_split
from sklearn.tree import DecisionTreeClassifier

avd = DecisionTreeClassifier(criterion='entropy', random_state=42)
avd.fit(x_train, y_train)
```

```
DecisionTreeClassifier(criterion='entropy', random_state=42)
```

```
y_pred_avd = avd.predict(x_test)
y_pred_avd
```

```
array([0., 0., 0., ..., 1., 0., 1.])
```

Imagen 55 - Modelo Árvore de Decisão

Acurácia

```
[ ]  from sklearn.metrics import accuracy_score
      accuracy_score(y_test, y_pred_avd)

[ ]  0.7128315605224183
```

Precisão

```
[ ]  from sklearn.metrics import precision_score
      precision_score(y_test, y_pred_avd)

[ ]  0.5773719730051607
```

Recall

```
[ ]  from sklearn.metrics import recall_score
      recall_score(y_test, y_pred_avd)

[ ]  0.5765480060255292
```

Imagen 56 - Métricas de avaliação Árvore de Decisão

```
from sklearn.metrics import plot_confusion_matrix
plot_confusion_matrix(avd,x_test,y_test, cmap='Blues')

<sklearn.metrics._plot.confusion_matrix.ConfusionMatrixDisplay at 0x7f9d210da8d0>
```

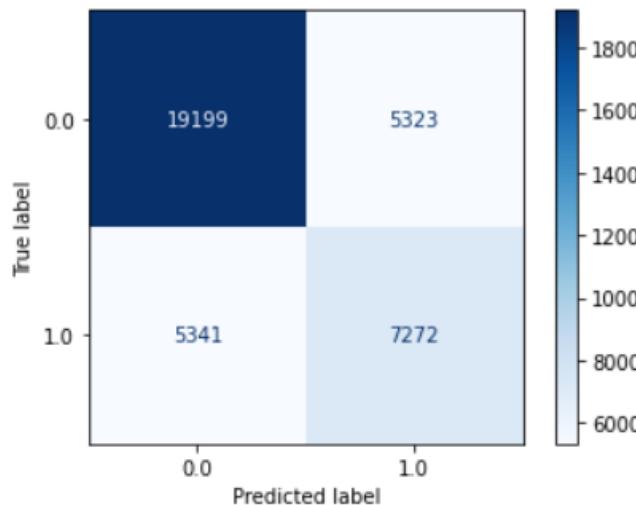


Imagen 57 - Matriz de Confusão Árvore de Decisão

Concluindo em um bom resultado, a princípio, dado que teve uma acurácia de 71%, juntamente com a precisão de 57% e recall de 57%.

M) Random Forest

```
from sklearn.ensemble import RandomForestClassifier
from sklearn.metrics import plot_confusion_matrix

rdf = RandomForestClassifier()
rdf.fit(x_train, y_train)

RandomForestClassifier()

y_pred_rdf = rdf.predict(x_test)
```

Imagen 58 - Modelo Random Forest

Acuracia

```
[ ] from sklearn.metrics import accuracy_score  
accuracy_score(y_test, y_pred_rdf)  
  
0.760899421031372
```

Precisão

```
[ ] from sklearn.metrics import precision_score  
precision_score(y_test, y_pred_rdf)  
  
0.6640597539543058
```

Recall

```
[ ] from sklearn.metrics import recall_score  
recall_score(y_test, y_pred_rdf)  
  
0.5991437405851106
```

Imagen 59 - Métricas Random Forest

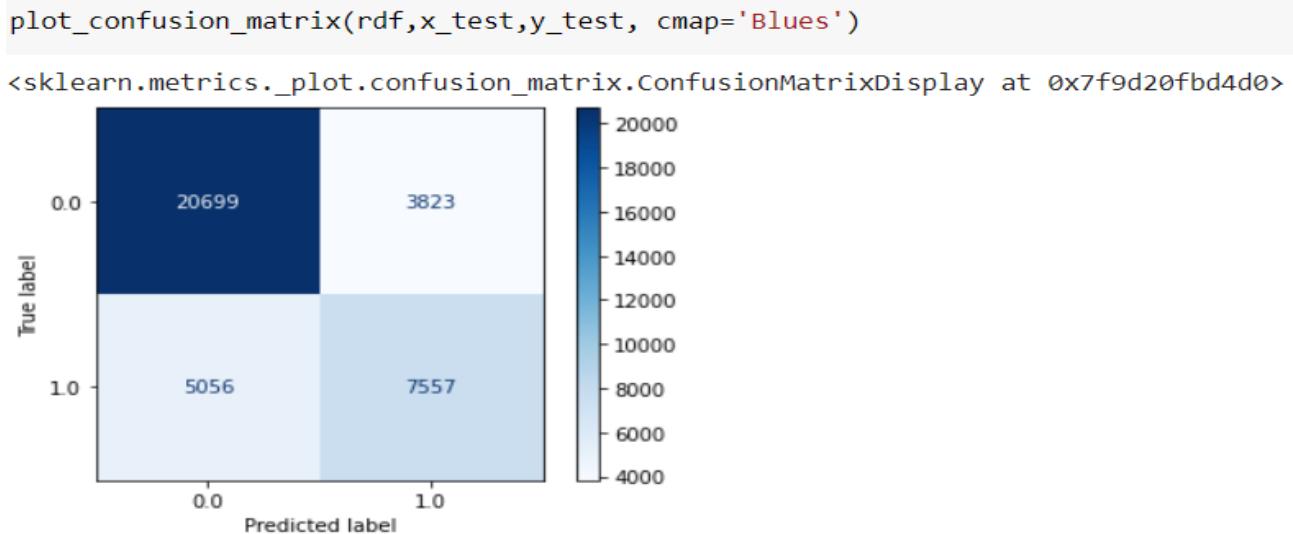


Imagen 59 - Matriz de Confusão Random Forest

Concluindo em um resultado, a princípio, razoável que teve uma acurácia de 76%, precisão de 66% e recall 59%.

Assim, percebe-se uma tendência para cada classe de modelos (Atritados, Novos Clientes e Engajados). Portanto, necessita-se de maiores estudos para a conclusão e devida avaliação dos modelos, uma vez que há vários dados conflitantes na base de dados e nos foi disponibilizada uma nova que será avaliada na sprint seguinte. Como citado no livro Machine Learning na Prática, em sua introdução sobre os modelos de Árvore de Decisão, que mostra que através de um conjunto definido de regras, um novo valor passará por uma série de perguntas, que seriam os ramos da árvore, para que a melhor decisão seja prevista, fazendo-a repetidamente até achar o melhor resultado. O mesmo livro, assim como a biblioteca oficial do sci-kit learning, trata dos demais tipos de modelagem, KNN, SVM, Árvore de Decisão, Naive Bayes, Regressão Logística, entre outros. Comprovando a robustez do modelo apresentado, dadas as devidas proporções e ponderações. Por fim, as métricas de avaliação seguiram conforme descrito pelos autoestudos, aulas e o seguinte artigo:

<https://vitorborbarodrigues.medium.com/m%C3%A9tricas-de-avalia%C3%A7%C3%A3o-acur%C3%A1cia-precis%C3%A3o-recall-quais-as-diferen%C3%A7as-c8f05e0a513c>

Em que a acurácia indica, em geral, como o modelo performou, a precisão, por sua vez, para detectar situações em que os Falsos Positivos são considerados mais prejudiciais que os Falsos Negativos, por exemplo. E o recall pode ser usado em uma situação em que os Falsos Negativos são considerados mais prejudiciais que os Falsos Positivos.

4.4.5 Hiperparâmetros

A partir das avaliações dos modelos, foram escolhidos os melhores algoritmos para a testagem dos diferentes hiperparâmetros. Foram feitas manipulações manuais com os hiperparâmetros e com GridSearch e RandomSearch.

1. Atritado

Link do notebook GoogleCollaboratory:

<https://colab.research.google.com/drive/1bQHMuWi8n3EoI95Om64d5oZYCdCvHSHA?usp=sharing>

- KNN
- Árvore de decisão
- Random Forest

2. Novos clientes

- Árvore de decisão
- Random Forest

3. Engajado

Link do notebook GoogleCollaboratory:

<https://colab.research.google.com/drive/1Ok9a8ao3Y5eKv5DY0-d0u2jW8ONLS-2h?usp=sharing>

- Árvore de decisão
- Random Forest

A seguir, vamos mostrar os resultados obtidos utilizando o random Search em cada um dos modelos.

Modelo atritado

Para atritado os modelos utilizados fora: KNN, Random Forest e Arvore de decisão.

KNN

Para Randomize Search, os hiperparametros utilizados foram; “n_neighbors”, “weights”, “algorithm” e “leaf_size”

- N_neighbors: Número de vizinhos a serem ultilizados;
- Weights: Função de peso usada na previsão. Valores possíveis.
- Algorithm: Algoritimos utilizados para calcular os vizinhos mais próximos.
- Leaf_size: Tamnho da leaf passando para BallTree ou KDTree

```
parametros = {'n_neighbors': [3, 5, 7, 9, 13, 15, 17, 21, 23, 25, 27, 29, 31, 33],
              'weights': ['uniform', 'distance'],
              'algorithm': ['auto', 'ball_tree', 'kd_tree', 'brute'],
              'leaf_size': [7, 15, 30, 45, 60, 75]}
random_search = RandomizedSearchCV(estimator = KNeighborsClassifier(), param_distributions=parametros)
random_search.fit(x_train, y_train.squeeze())
```

Imagen 60 - Parâmetros de KNN

Resultados obtidos:

```
y_pred = knn_best.predict(x_test)
print('Acc treino: ', knn_best.score(x_train, y_train ))
print('Acc teste: ', knn_best.score(x_test, y_test.squeeze() ))
print( 'Revocação: ', recall_score( y_test, y_pred ))
print( 'Precisão: ', precision_score( y_test, y_pred ))
print( 'F1_score: ', f1_score( y_test, y_pred ))
```

```
Acc treino:  0.926985195154778
Acc teste:  0.8494623655913979
Revocação:  0.9304347826086956
Precisão:  0.784841075794621
F1_score:  0.8514588859416446
```

Imagen 61 - Resultados KNN

Melhores parâmetros:

```
{'weights': 'distance', 'n_neighbors': 5, 'leaf_size': 7, 'algorithm': 'kd_tree'}
```

Imagen 62 - Melhores parâmetros KNN

Árvore de decisão

Para árvore de decisão, os hiperparametros utilizados foram: “Criterion” e “Splitter”

- Splitter: A estratégia utilizada para escolher a divisão em cada nó.
- Criterion: A função para medir a qualidade de uma divisão.

```
# Definir o espaço possível dos hiperparametros a serem testados no modelo
parametros = {'criterion': ['gini', 'entropy', 'log_loss'],
              'splitter': ['best', 'random']}
```



```
random_search = RandomizedSearchCV(estimator = DecisionTreeClassifier(), param_distributions=parametros)
random_search.fit(x_train, y_train.squeeze())
```

Imagen 63 - Parâmetros para Árvore de Decisão

Resultados obtidos:

```
y_pred = avd_best.predict(x_test)
print('Acc treino: ', avd_best.score(x_train, y_train ))
print('Acc teste: ', avd_best.score(x_test, y_test.squeeze() ))
print( 'Revocação: ', recall_score( y_test, y_pred ))
print( 'Precisão: ', precision_score( y_test, y_pred ))
print( 'F1_score: ', f1_score( y_test, y_pred ))
```



```
Acc treino:  1.0
Acc teste:  0.864247311827957
Revocação:  0.8782608695652174
Precisão:  0.8370165745856354
F1_score:  0.8571428571428571
```

Imagen 64 - Resultados Árvore de Decisão

Melhores parâmetros:

```
{'splitter': 'best', 'criterion': 'entropy'}
```

Imagen 65 - Melhores parâmetros Árvore de Decisão

Random Forest

Para o modelo atritado, o grupo, em comum acordo, decidiu como hiperparâmetros para o modelo de Random Forest que deveríamos utilizar o Random Search, uma vez que mostra apenas alguns valores, retornando o teste de alguns, e não usará da força bruta, dificuldade de processamento e alto consumo de memória do Grid Search. Nesse caso, foram utilizados o “n_estimators” para a estimar o número de árvores, o “criterion” para medir a qualidade da divisão dos dados, “max_depth” representa a profundidade máxima da árvore.

```
# Definir o espaço possível dos hiperparametros a serem testados no modelo
parametros = {'n_estimators': [10, 20, 40, 60, 80, 100],
              'criterion': ['gini', 'entropy','log loss'],
              'max_depth': [10,20,30,40,50],}
```



```
random_search = RandomizedSearchCV(estimator = RandomForestClassifier(), param_distributions=parametros)
random_search.fit(x_train, y_train.squeeze())
```

Imagen 66 - Parâmetros para RandomForest

Resultados obtidos:

```

y_pred = rdf_best.predict(x_test)
print('Acc treino: ', rdf_best.score(x_train, y_train ))
print('Acc teste: ', rdf_best.score(x_test, y_test.squeeze() ))
print( 'Revocação: ', recall_score( y_test, y_pred ))
print( 'Precisão: ', precision_score( y_test, y_pred ))

```

```

Acc treino: 1.0
Acc teste: 0.8844086021505376
Revocação: 0.9072463768115943
Precisão: 0.8528610354223434

```

Imagen 67 - Resultados RandomForest

Melhores parâmetros:

```
{'n_estimators': 80, 'max_depth': 20, 'criterion': 'gini'}
```

Imagen 68 - Melhores parâmetros RandomForest

Modelo engajado

Para cliente engajado, utilizamos dois modelos: Árvore de Decisão e Random Forest.

Árvore de decisão

No modelo de árvore de decisão, testamos diferentes valores para os parâmetros splitter e criterion.

Significado de cada parâmetro:

- splitter: A estratégia utilizada para escolher a divisão em cada nó.
- criterion: A função para medir a qualidade de uma divisão.

Resultados obtidos:

```

1 y_pred = knn_best.predict(x_test)
2 print('Acc treino: ', knn_best.score(x_train, y_train ))
3 print('Acc teste: ', knn_best.score(x_test, y_test.squeeze() ))
4 print( 'Revocação: ', recall_score( y_test, y_pred ))

Acc treino: 0.9999193027759845
Acc teste: 0.9778620615450828
Revocação: 0.9657652024386432

```

Imagen 69 - Melhores parâmetros para Árvores de Decisão

Melhores parâmetros:

```
{'splitter': 'random', 'criterion': 'gini'}
```

Imagen 70 - Melhores parâmetros Árvore de Decisão

Random Forest

No modelo de random forest, testamos diferentes valores para os parâmetros criterion, n_estimators e max_depth.

Significado de cada parâmetro:

- criterion: A função para medir a qualidade de uma divisão.
- n_estimators(10,20,30): O número de árvores na floresta.
- max_depth(10,20,30): A profundidade máxima da árvore.

Resultados obtidos:

```
[91] 1 y_pred = knn_best.predict(x_test)
    2 print('Acc treino: ', knn_best.score(x_train, y_train))
    3 print('Acc teste: ', knn_best.score(x_test, y_test.squeeze()))
    4 print('Revocação: ', recall_score(y_test, y_pred))

Acc treino:  0.988675489563159
Acc teste:  0.9883795997417689
Revocação:  0.9999218383617321
```

Imagen 71 - Resultados RandomForest

Melhores parâmetros:

```
{'criterion': 'gini', 'max_depth': 10, 'n_estimators': 20}
```

Imagen 72 - Melhores parâmetros RandomForest

Modelo novo cliente

Árvore de decisão

No modelo de árvore de decisão, testamos diferentes valores para os parâmetros splitter, criterion e max_depth.

Significado de cada parâmetro:

- splitter: A estratégia utilizada para escolher a divisão em cada nó.
- criterion: A função para medir a qualidade de uma divisão.
- max_depth: A profundidade máxima da árvore.

```
parametros = { 'splitter': ['best', 'random'],
               'criterion': ['gini', 'entropy', 'log loss'],
               'max_depth': [10,20,30,40,50],}

random_search = RandomizedSearchCV(estimator = DecisionTreeClassifier(), param_distributions=parametros)
random_search.fit(x_train, y_train.squeeze())
```

Imagen 73 - Parâmetros para Árvore de Decisão

Resultados obtidos:

```

y_pred = dt_best.predict(x_test)
print('Acc treino: ', dt_best.score(x_train, y_train ))
print('Acc teste: ', dt_best.score(x_test, y_test.squeeze() ))
print( 'Revocação: ', recall_score( y_test, y_pred ))
print( 'Precisão: ', precision_score( y_test, y_pred ))

```

```

Acc treino: 1.0
Acc teste: 1.0
Revocação: 1.0
Precisão: 1.0

```

Imagen 74 - Resultados Árvore de Decisão

Melhores parâmetros:

```
{'splitter': 'best', 'max_depth': 40, 'criterion': 'gini'}
```

Imagen 75 - Melhores parâmetros Árvore de Decisão

Random Forest

No modelo de Random Forest, testamos diferentes valores para os parâmetros n_estimators, criterion, max_depth.

Significado de cada parâmetro:

- n_estimators: O número de árvores na floresta.
- criterion: A função para medir a qualidade de uma divisão.
- max_depth: A profundidade máxima da árvore.

```

# Definir o espaço possível dos hiperparâmetros a serem testados no modelo
parametros = {'n_estimators': [10, 20, 40, 60, 80, 100],
              'criterion': ['gini', 'entropy','log loss'],
              'max_depth': [10,20,30,40,50],}

random_search = RandomizedSearchCV(estimator = RandomForestClassifier(), param_distributions=parametros)
random_search.fit([x_train, y_train.squeeze()])

```

Imagen 76 - Parâmetros para RandomForest

Resultados obtidos:

```

y_pred = rdf_best.predict(x_test)
print('Acc treino: ', rdf_best.score(x_train, y_train ))
print('Acc teste: ', rdf_best.score(x_test, y_test.squeeze() ))
print( 'Revocação: ', recall_score( y_test, y_pred ))
print( 'Precisão: ', precision_score( y_test, y_pred ))

Acc treino: 1.0
Acc teste: 1.0
Revocação: 1.0
Precisão: 1.0

```

Imagen 77 - Resultados RandomForest

Melhores parâmetros:

```
{'n_estimators': 20, 'max_depth': 10, 'criterion': 'entropy'}
```

Imagen 78 - Melhores parâmetros para RandomForest

4.5. Avaliação

Modelo atritado

Nossos primeiros modelos para a predição de um cliente atritado se mostraram extremamente falhos devido ao baixo número de clientes atritados em relação ao total de clientes de nossa base de dados. Os modelos estavam estimando que a maioria dos clientes não eram atritados, o que fez com que ele acertasse a maior parte das vezes. Isso pode ser comprovado pelas taxas de acurácia e precisão representadas a seguir:

```

from sklearn.metrics import accuracy_score
accuracy_score(y_test, y_pred_knn)

```

0.9996948099743281

```

from sklearn.metrics import precision_score
precision_score(y_test, y_pred_knn)

```

1.0

Imagen 79 - Acurácia de modelo

Porém, a taxa de recall foi extremamente baixa, o que mostrou que os modelos estavam errando a maioria dos clientes atritados:

```
from sklearn.metrics import recall_score  
recall_score(y_test, y_pred_knn)
```

0.2916666666666667

Imagen 80 - Taxa de recall

Após a criação desses modelos, percebemos a necessidade de fazer um undersampling de não atritados e um oversampling de atritados. Ou seja, tivemos que melhorar a proporção de clientes atritados e não atritados para treinar nosso modelo de forma mais eficiente. Essa é uma prática não ideal, visto que treina o modelo com dados artificiais que não são condizentes com a realidade. Porém, dada a nossa baixa quantidade de atritados, foi a única solução cabível.

Após esse processo, percebemos que o Random Forest obteve o melhor resultado:

```
Acuracia  
[71] from sklearn.metrics import accuracy_score  
accuracy_score(y_test, y_pred_rdf)  
0.8911290322580645  
  
Precisão  
[72] from sklearn.metrics import precision_score  
precision_score(y_test, y_pred_rdf)  
0.8646408839779005  
  
Recall  
[73] from sklearn.metrics import recall_score  
recall_score(y_test, y_pred_rdf)  
0.9072463768115943
```

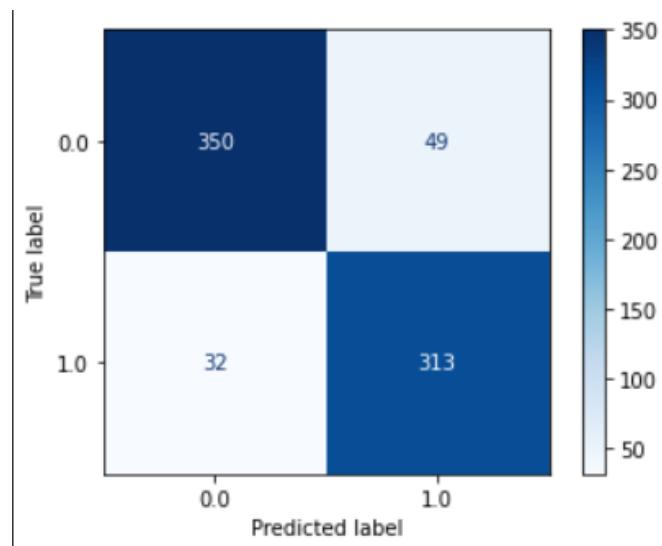


Imagen 81 - Métricas KNN

Modelo engajado

	ind_engaj
ind_engaj	1.000000
num_produtos	0.707495
qtd_oper	0.296881
vlr_saldo	0.170425
vlr_credito	0.070471
vlr_score	0.066338
vlr_renda	0.034249
num_atend	0.030704
num_atend_atrs	0.021248
ind_atrito	0.006447
qtd_reclm	0.005056
qtd_restr	0.002315
Unnamed: 0	0.002035

Imagen 82 - Métricas KNN

Observando os dados acima, podemos perceber que há uma grande correlação entre o número de produtos de um usuário e o seu índice de engajamento. Quando revisamos os dados fornecidos pelo cliente, percebemos que esta coluna faz parte da regra de negócio da definição de um cliente engajado.

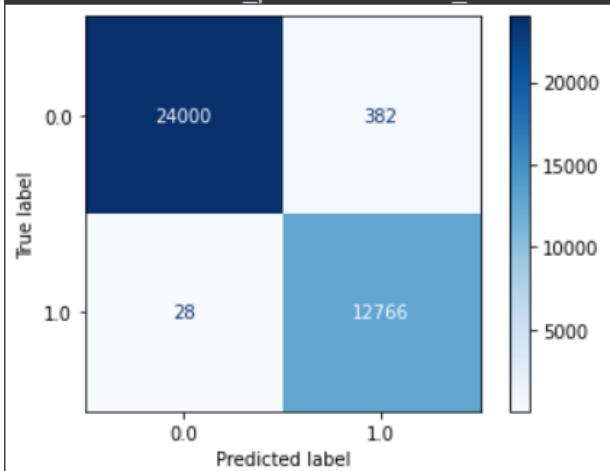
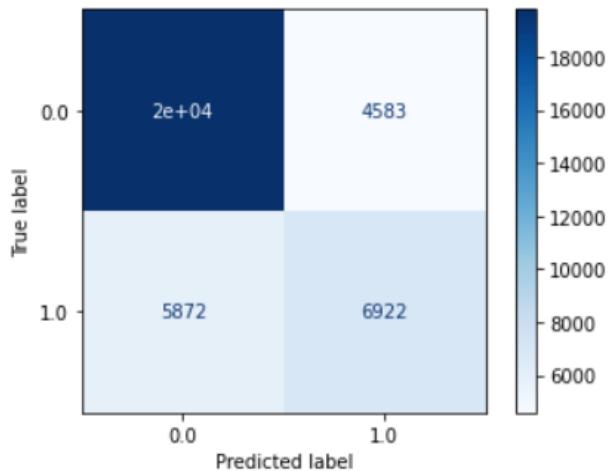
Modelo com a coluna num_produtos	Modelo sem a coluna num_produtos
Acurácia	Acurácia
<pre>[47] from sklearn.metrics import accuracy_score accuracy_score(y_test, y_pred_rdf)</pre> <p>0.9889713793845492</p>	<pre>[153] from sklearn.metrics import accuracy_score accuracy_score(y_test, y_pred_rdf)</pre> <p>0.7187701743060039</p>
Precisão	Precisão
<pre>[48] from sklearn.metrics import precision_score precision_score(y_test, y_pred_rdf)</pre> <p>0.9709461515059324</p>	<pre>[154] from sklearn.metrics import precision_score precision_score(y_test, y_pred_rdf)</pre> <p>0.601651455888744</p>
Recall	Recall
<pre>[49] from sklearn.metrics import recall_score recall_score(y_test, y_pred_rdf)</pre> <p>0.9978114741284977</p>	<pre>[155] from sklearn.metrics import recall_score recall_score(y_test, y_pred_rdf)</pre> <p>0.5410348600906675</p>
	

Imagen 83 - Comparaçāo de Métricas

Observando os dados acima, identificamos com precisão qual era a coluna que estava causando esse leak de dados. Após uma discussão do grupo, concluímos que, apesar desse

vazamento estar acontecendo, a coluna “num_produtos” estará sempre presente na base de dados do Banco Pan, visto que é uma característica que todos os clientes possuem. Por isso, decidimos criar dois modelos para engajados. Dessa forma, o Banco Pan terá a possibilidade de escolher qual utilizar dependendo da base de dados que possuir.

Modelo novo cliente

Nos modelos de novo cliente, obtivemos um resultado perfeito no algoritmo de árvore de decisão, o que nos fez perceber que existe um vazamento nos dados. Por isso, fizemos uma pesquisa criando uma tabela de correlação entre as colunas e descobrimos que a feature “vlr_score” tem uma grande correlação com o “ind_novo_cli”:

ind_novo_cli	
ind_novo_cli	1.000000
vlr_score	0.618704
qtd_restr	0.316344
vlr_renda	0.065169
vlr_credito	0.028897
Unnamed: 0	0.005114
num_atend_atrs	0.004194
qtd_reclm	0.002539
num_atend	0.001255

Imagen 84 - Tabela de Correlação

Após isso, adaptamos o modelo de árvore de decisão (algoritmo com o melhor resultado) para visualizar a diferença dos resultados utilizando ou não a coluna “vlr_score”:

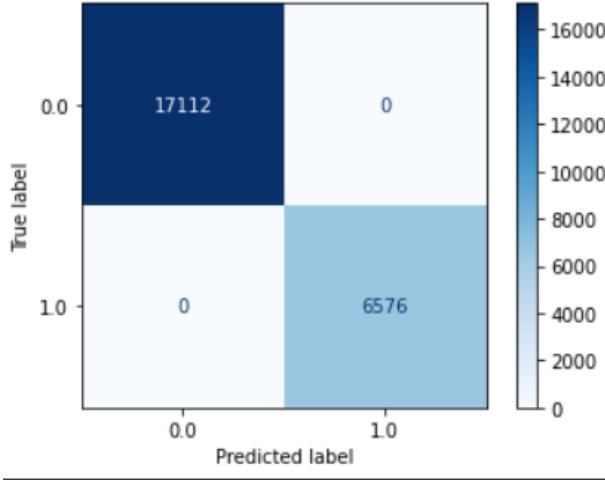
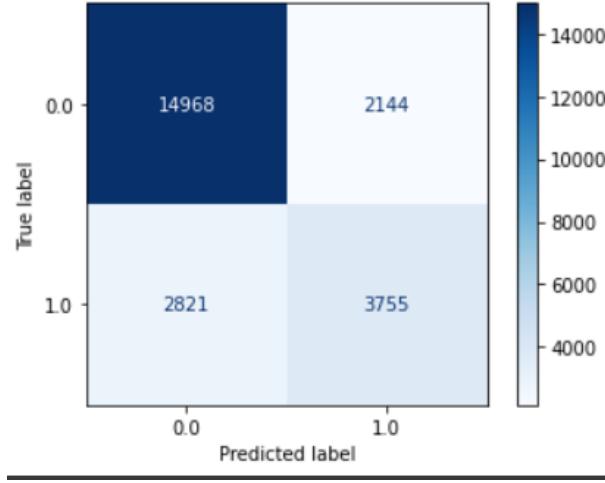
Modelo árvore de decisão com a coluna vlr_score	Modelo árvore de decisão sem a coluna vlr_score																		
<p>Acurácia</p> <pre>[16] from sklearn.metrics import accuracy_score accuracy_score(y_test, y_pred_avd)</pre> <p>1.0</p> <p>Precisão</p> <pre>[17] from sklearn.metrics import precision_score precision_score(y_test, y_pred_avd)</pre> <p>1.0</p> <p>Recall</p> <pre>[18] from sklearn.metrics import recall_score recall_score(y_test, y_pred_avd)</pre> <p>1.0</p>	<p>Acurácia</p> <pre>[114] from sklearn.metrics import accuracy_score accuracy_score(y_test, y_pred_avd)</pre> <p>0.7904002026342452</p> <p>Precisão</p> <pre>[115] from sklearn.metrics import precision_score precision_score(y_test, y_pred_avd)</pre> <p>0.6365485675538227</p> <p>Recall</p> <pre>[116] from sklearn.metrics import recall_score recall_score(y_test, y_pred_avd)</pre> <p>0.5710158150851582</p>																		
 <table border="1"> <thead> <tr> <th>True label \ Predicted label</th> <th>0.0</th> <th>1.0</th> </tr> </thead> <tbody> <tr> <th>0.0</th> <td>17112</td> <td>0</td> </tr> <tr> <th>1.0</th> <td>0</td> <td>6576</td> </tr> </tbody> </table>	True label \ Predicted label	0.0	1.0	0.0	17112	0	1.0	0	6576	 <table border="1"> <thead> <tr> <th>True label \ Predicted label</th> <th>0.0</th> <th>1.0</th> </tr> </thead> <tbody> <tr> <th>0.0</th> <td>14968</td> <td>2144</td> </tr> <tr> <th>1.0</th> <td>2821</td> <td>3755</td> </tr> </tbody> </table>	True label \ Predicted label	0.0	1.0	0.0	14968	2144	1.0	2821	3755
True label \ Predicted label	0.0	1.0																	
0.0	17112	0																	
1.0	0	6576																	
True label \ Predicted label	0.0	1.0																	
0.0	14968	2144																	
1.0	2821	3755																	

Imagen 85 - Comparação de Métricas

Observando os dados acima, identificamos com precisão qual era a coluna que estava causando esse leak de dados. Pensando no dia a dia da empresa e levando em consideração que a feature vlr_score é essencial para o reconhecimento de um novo cliente, tomamos a

decisão de criar dois modelos, com e sem a coluna vlr_score: um no qual o Banco Pan poderá prever com 100% de precisão os novos clientes e outro na qual ele conseguirá prever os usuários que possuem mais probabilidade de se tornarem novos clientes. Assim o parceiro terá em suas mãos a possibilidade de trabalhar com esses dois algoritmos e escolher aquele que fizer mais sentido para a empresa.a

5. Conclusões e Recomendações

Escreva, de forma resumida, sobre os principais resultados do seu projeto e faça recomendações formais ao seu parceiro de negócios em relação ao uso desse modelo. Você pode aproveitar este espaço para comentar sobre possíveis materiais extras, como um manual de usuário mais detalhado na seção “Anexos”.

Não se esqueça também das pessoas que serão potencialmente afetadas pelas decisões do modelo preditivo, e elabore recomendações que ajudem seu parceiro a tratá-las de maneira estratégica e ética.

6. Referências

Nesta seção você deve incluir as principais referências de seu projeto, para que seu parceiro possa consultar caso ele se interessar em aprofundar.

Utilize a norma ABNT NBR 6023 para regras específicas de referências. Um exemplo de referência de livro:

LaUCK, Heloisa. **Liderança em gestão escolar**. 4. ed. Petrópolis: Vozes, 2010.

SOBRENOME, Nome. **Título do livro**: subtítulo do livro. Edição. Cidade de publicação: Nome da editora, Ano de publicação.

Anexos

Utilize esta seção para anexar materiais como manuais de usuário, documentos complementares que ficaram grandes e não couberam no corpo do texto etc.