



NULL - Churn

Rappi

Controle do Documento

Histórico de revisões

Data	Autor	Versão	Resumo da atividade
10/08/2022	Giovana Lisbôa Thomé	1.1	Atualização da seção 4.4.1
10/08/2022	Pedro Munhoz	1.2	Preenchimento da seção 4.4.1 Preenchimento da seção 4.1.4 Preenchimento da seção 4.2
10/08/2022	Rodrigo Martins	1.3	Preenchimento da seção 4.1.2 Preenchimento da seção 4.2
10/08/2022	Eric Tachdjian	1.4	Preenchimento da seção 4.1.5
14/08/2022	Rodrigo Martins	1.5	Preenchimento da seção 4.2
14/08/2022	Sergio Lucas	1.6	Preenchimento da seção 4.1.3.1 Preenchimento da seção 4.1.3.3 Preenchimento da seção 4.2.1 , 4.2.1.1
14/08/2022	Beatriz Hirasaki Leite	1.7	Preenchimento da seção 4.1.3.2 Preenchimento da seção 4.2.1.3
14/08/2022	Arthur Reis	1.8	Preenchimento da seção 4.1.3 Preenchimento da seção 4.2.3
21/08/2022	Rodrigo Martins	1.9	Preenchimento da seção 4.1.7
24/08/2022	Eric Tachdjian	2.1	Preenchimento da seção 4.1.6
25/08/2022	Giovana Thomé	2.2	Preenchimento da seção 1 e revisão geral do documento
28/08/2022	Eric Tachdjian	2.3	Preenchimento da seção 4.3
28/08/2022	Sergio Lucas	2.4	Atualização da seção 4.1.6
28/08/2022	Beatriz Hirasaki e Pedro Munhoz	2.5	Revisão geral do documento
29/08/2022	Rodrigo Martins	2.6	Preenchimento da seção 4.1.7
29/08/2022	Giovana Thomé	2.7	Atualização da seção 4.3

12/09/2022	Giovana Thomé, Rodrigo Martins e Pedro Munhoz	3.1	Preenchimento da seção 4.4
12/09/2022	Pedro Munhoz	3.2	Preenchimento da seção 4.5
13/09/2022	Giovana Thomé e Pedro Munhoz	3.3	Atualização da seção 4.4
24/09/2022	Giovana Thomé	4.1	Atualização da seção 4.4
25/09/2022	Arthur Reis	4.2	Atualização da seção 4.4
28/09/2022	Giovana Thomé e Pedro Munhoz	4.3	Atualização das seções 4.4 e 4.5
05/10/2022	Eric Tachdjian e Giovana Thomé	4.4	Atualização e revisão geral do documento
06/10/2022	Eric Tachdjian, Giovana Thomé e Pedro Munhoz	4.5	Conclusão do documento

Sumário

1. Introdução	8
2. Objetivos e Justificativa	9
2.1. Objetivos	9
2.2. Justificativa	9
3. Metodologia	10
3.1. CRISP-DM	10
3.2. Ferramentas	11
3.3. Principais técnicas empregadas	11
4. Desenvolvimento e Resultados	12
4.1. Compreensão do Problema	12
4.1.1. Contexto da indústria	12
4.1.2. Análise SWOT	14
4.1.3. Planejamento Geral da Solução	14
4.1.3.1. Qual é o problema a ser resolvido	14
4.1.3.2. Quais os dados disponíveis	15
4.1.3.3. Qual a solução proposta	15
4.1.3.4. Qual o tipo de tarefa	15
4.1.3.5. Como a solução proposta deverá ser utilizada	16
4.1.3.6. Quais os benefícios trazidos pela solução proposta	16
4.1.3.7. Qual será o critério de sucesso e qual medida será utilizada para o avaliar	16
4.1.4. Value Proposition Canvas	17
4.1.5. Matriz de Riscos	17
4.1.6. Personas	18
4.1.7. Jornadas do Usuário	20
4.2. Compreensão dos Dados	21
4.2.1. Dados a serem utilizados	21
4.2.1.1. Descrição de agregação e mesclagem de dados	26

4.2.1.2. Descrição dos riscos e contingências relacionados a esses dados	Recebemos uma ampla variedade de dados, entretanto, cabem algumas ressalvas: 27
4.2.1.3. Descrição de como será selecionado o subconjunto para análises iniciais	27
4.2.1.4. Descrição das restrições de segurança.	29
4.2.2. Descrição estatística básica dos dados	29
4.2.3. Descrição da predição desejada	30
4.3. Preparação dos Dados	32
4.3.1. Descrição de manipulações nos registros e suas respectivas features.	32
4.3.2. Agregação de registros e/ou derivação de novos atributos.	35
4.3.3. Remoção ou substituição de valores ausentes/em branco.	35
4.3.4. Identificação das features selecionadas, com descrição dos motivos de seleção.	36
4.4. Modelagem	37
4.4.1. KNN(K-Nearest Neighbors)	38
4.4.1.1. Explicação do modelo	38
4.4.1.2. Motivo de escolha do modelo	38
4.4.1.3. Resultados obtidos	39
4.4.2 Decision Tree	39
4.4.2.1. Explicação do modelo	39
4.4.2.2. Motivo de escolha do modelo	40
4.4.2.3. Resultados obtidos	40
4.4.3 Naive Bayes	41
4.4.3.1. Explicação do modelo	41
4.4.3.2. Motivo de escolha do modelo	41
4.4.3.3. Resultados obtidos	42
4.4.4 Random Forest	42
4.4.4.2. Motivo de escolha do modelo	43
4.4.4.3. Resultados obtidos	43
4.4.5 Regressão Logística	44
4.4.5.1. Explicação do modelo	44
4.4.5.2. Motivo de escolha do modelo	45
4.4.5.3. Resultados obtidos	45

4.4.6 Support Vector Machine - SVM	46
4.4.6.1. Explicação do modelo	46
4.4.6.2. Motivo de escolha do modelo	46
4.4.6.3. Resultados obtidos	46
4.4.7. AdaBoost	47
4.4.7.1. Explicação do modelo	47
4.4.7.2. Motivo de escolha do modelo	47
4.4.7.3. Resultados obtidos	47
4.4.8. Extra Trees	48
4.4.8.1. Explicação do modelo	48
4.4.8.2. Motivo de escolha do modelo	48
4.4.8.3. Resultados obtidos	48
4.4.9. KNN com hiperparametrização	49
4.4.9.1. Motivo de escolha do modelo e conjunto de dados	49
4.4.9.2. Configurações de hiperparâmetros	50
4.4.9.2. Resultados obtidos	50
4.4.10. Extra Trees com hiperparametrização	52
4.4.10.1. Motivo de escolha do modelo e conjunto de dados	52
4.4.10.2. Configurações de hiperparâmetros	52
4.4.10.3. Resultados obtidos	53
4.4.11. Catboost	54
4.4.11.1. Explicação do modelo	54
4.4.11.2. Motivo de escolha do modelo	54
4.4.11.3. Resultados obtidos	54
4.4.12. XGBoost	55
4.4.12.1. Explicação do modelo	55
4.4.12.2. Motivo de escolha do modelo	55
4.4.12.3. Resultados obtidos	55
4.5. Avaliação	57
4.6 Comparação de Modelos	61
5. Conclusões e Recomendações	63

6. Referências	64
7. Anexos	66

1. Introdução

O parceiro de negócios no módulo 3 é a empresa Rappi, que atua principalmente no setor de entregas, contendo escopo bastante diversificado incluindo restaurantes, farmácias, supermercados, shoppings, entre outros serviços como viagens e entregas express. A empresa nasceu na Colômbia, atualmente com sede em Bogotá, e é presente em grande parte da América Latina.

Atualmente, no Brasil, a Rappi enfrenta um problema de uma alta taxa de rotatividade (*churn rate*) de seus entregadores em seu aplicativo, sem conseguir reter um grande número deles. São trabalhadores – muitas vezes referidos como RTs – que, em sua maioria, trabalham na Rappi para complementar sua renda. Além da baixa retenção de entregadores, a empresa não tem conhecimento das principais causas de churn em sua plataforma e dos entregadores que são mais propensos a saírem.

Ao redor do mundo, empresas fazem o uso de inteligências artificiais para melhora de seus modelos de negócios, com base em previsões orientadas a dados. O presente documento registra o processo de criação de um modelo preditivo de machine learning de estudantes do Instituto de Tecnologia e Liderança (Inteli) em parceria com a Rappi.

2. Objetivos e Justificativa

2.1. Objetivos

O principal objetivo da Rappi é diminuir a quantidade de entregadores que dão churn, uma vez que a porcentagem de saída do aplicativo não é saudável para o negócio.

Outro objetivo deles é a visualização de quais entregadores teriam a maior chance de sair da plataforma. Isso também possibilita, de forma humana, visualizar o que mais causa algum entregador parar de entregar pela Rappi.

2.2. Justificativa

A solução Null-Churn construída pelo grupo Rappitenderos apresenta uma inteligência artificial supervisionada de confiança, que passa do 92% de acurácia, ou seja, consegue dizer se algum entregador tende a deixar a plataforma com 92% de certeza.

Os benefícios que a solução traz para a Rappi é a visualização de quais RTs tem o risco de abandonar o aplicativo, enquanto também, facilita o trabalho de algum funcionário Rappi na identificação de quais pontos precisam ser melhorados na relação Rappi-entregador, para que eles permaneçam na plataforma por mais tempo.

A solução tem um grande potencial para auxiliar a Rappi na relação dela com os entregadores e pode ajudar a diminuir significativamente a porcentagem daqueles que deixam a plataforma.

3. Metodologia

As metodologias utilizadas como referências para o desenvolvimento do projeto de modelos preditivos foram Scrum e CRISP-DM. Essas ferramentas de gestão de projetos estão relacionadas em um certo período de tempo para incrementar partes do projeto e validar com o cliente se a construção do produto ocorre conforme os pré-requisitos das regras de negócios estipulados pelo próprio cliente.

Ambas as metodologias se complementam na duração do projeto inteiro, onde a organização de tarefas orientadas pelo Scrum orientou no desenvolvimento dos requisitos estipulados pela CRISP-DM. A partir disso, dividiu-se o projeto de modelos preditivos em 5 grandes etapas (análise e reconhecimento dos dados, escolha de features adequadas, construção de modelos, adição de hiperparâmetros para o modelo e o teste desses modelos prontos) e cada etapa dessa foi incrementada com base nas anteriores.

Na Sprint 1, foi feita a análise e a compreensão dos dados. Essa parte da metodologia orientou-se no entendimento de cada informação presente e como elas se relacionam com o nosso público-alvo, os entregadores da Rappi. Já na segunda Sprint, foi feito o reconhecimento de dados mais importantes e relevantes que solucionam o problema de abandono dos trabalhadores que prestam serviço para a empresa. Na terceira parte do projeto, construiu-se os modelos preditivos baseados nos dados que foram escolhidos. Analisamos a assertividade desse modelo com relação ao nosso target (informação responsável pela identificação do problema). Através dos dados que foram inseridos, o algoritmo prevê, através de padrões, o que ocorreu no target. Por exemplo, com base nas informações escolhidas, o modelo preditivo prevê se o entregador deu ou não Churn (abandonou a prestação de serviços para a empresa). Na Sprint 4, colocou-se o hiperparâmetro, ou seja, variáveis capazes de especificarem o que um modelo preditivo deve fazer para funcionar (esses hiperparâmetros são incrementais e servem única e exclusivamente para melhorar o modelo, ou seja torná-lo mais próximo da realidade. Na quinta Sprint foram feitos alguns ajustes nos modelos, o teste e o deploy desses mesmos algoritmos.

3.1. CRISP-DM

A metodologia CRISP-DM, que em português significa padrão de indústria em informação cruzada para a mineração de dados, é capaz de transformar os dados de uma empresa em conhecimento e informações úteis para a corporação. Essa metodologia possui 6 etapas bem definidas sobre processo de ciência de dados. Essas etapas foram divididas e adaptadas para cinco sprints, sendo elas entendimento do negócio, entendimento dos dados, preparação dos dados, modelagem, avaliação e implementação/deployment.

A organização dos dados envolve o entendimento do negócio que é uma parte muito importante do CRISP-DM, porque essa etapa é específica às necessidades da empresa e qual o problema será resolvido com aquele conjunto de dados. A partir disso, entendemos a

importância da ação do entendimento de dados para o modelo de negócio da empresa. A segunda etapa que temos nessa parte é o entendimento dos dados, ou seja, é de suma importância que a análise seja feita a fim de compreender o que aquele conjunto de dados diz, observar padrões que são simples para um estudo prévio do que essas informações realmente significam. A terceira etapa é a de preparação dos dados, ou seja, os dados adquiridos e coletados precisam ser manipulados para que se possa enxergar ainda mais o que aqueles dados dizem, escolher aquelas informações que são mais importantes ou ainda descartar aquelas que são menos importantes.

As seguintes etapas correspondem a modelagem (elaboração de diversos modelos), avaliação e implementação/deployment dos modelos (escolha e disponibilização do modelo escolhido para utilização no ambiente da empresa).

3.2. Ferramentas

O Google Collaboratory, também chamado de Colab, é uma ferramenta similar ao Jupyter Notebook utilizada principalmente para a análise de dados que permite ao usuário, por meio do navegador, executar um código em Python. Ela utiliza programação em células e os registros são salvos na nuvem.

As principais bibliotecas em Python utilizadas foram Pandas e Numpy para manipulação de dados e dataframes, Scikit-learn para ferramentas de aprendizado de máquina e Plotly para plotagem de gráficos.

Foi utilizado também o Pycaret que auxilia na construção de modelos e agiliza o processo de decisão de quais algoritmos devem ser utilizados.

3.3. Principais técnicas empregadas

Dentre as técnicas utilizadas na confecção do modelo de predição, destacam-se, após a limpeza dos dados nulos, transformação do target com **label encoding** e remoção dos outliers, o balanceamento dos dados de treino e teste com **Oversampling, Undersampling ou os dois juntos** e o refinamento dos algoritmos pré-prontos **KNN, Decision Tree, Random Forest, Naive Bayes, Regressão Logística, SVM, AdaBoost, Extra Trees, CatBoost, Bagging Classifier e XGBoost** com a **hiperparametrização**. Todos os tópicos em negrito abordados neste parágrafo são de suma importância para a compreensão da solução, então todos serão explicados com profundidade, sobretudo na seção [4.4](#).

Com o emprego das técnicas citadas anteriormente, a resolução do problema da descoberta de quais entregadores têm mais probabilidade de dar Churn é possível, uma vez que essas são etapas fundamentais para o modelo funcionar, seja a definição do Churn como um número com o **Label Encoding**, seja com o ponto de partida dos modelos expostos com a melhora nas métricas de avaliação pela **hiperparametrização**.

4. Desenvolvimento e Resultados

4.1. Compreensão do Problema

4.1.1. Contexto da indústria

Cinco Forças de Porter

Dado que o modelo preditivo a ser desenvolvido é destinado à Rappi, a análise de mercado em que a empresa está inserida foi feita utilizando o modelo das Cinco Forças de Porter, criado por Michael Porter. O modelo de análise de competição entre as empresas engloba cinco tópicos principais: rivalidade entre os concorrentes, ameaça de novos concorrentes, ameaça de produtos substitutos, poder de negociação dos clientes e poder de negociação dos fornecedores, todos aprofundados a seguir, respectivamente.

No setor de delivery, a rivalidade é alta por conta dos fortes concorrentes já consolidados no mercado, tanto de entrega de comida quanto outros produtos, como compras de supermercado, farmácia, etc. Apesar da empresa dispor de tecnologia, assim como seus competidores, a tarefa de captar parceiros de negócio (entregadores, restaurantes, farmácias e supermercados) não é tão exitosa quanto a de seus concorrentes, criando um cenário de concorrência alta.

Já o cenário de entrada de novos concorrentes é bastante chamativo por ser um setor altamente lucrativo e com espaço para inovações. Porém, por conta da rivalidade elevada já mencionada, competidores já consolidados no mercado dificultam bastante a entrada.

Ao mesmo tempo que a utilização de aplicativos de delivery é um substituto para a ida do cliente às lojas físicas, aplicativos de comunicação também podem representar um meio substituto para comunicação remota restaurante cliente, como já é observado atualmente, principalmente em cidades não tão populosas, sem serviços como iFood e Rappi. Um dos maiores exemplos é o Whatsapp, aplicativo de mensagens utilizado em larga escala no Brasil.

Apesar da informação do perfil dos clientes da Rappi não ser aberta ao público, é possível realizar uma análise e apontamentos das características mais generalistas de seus clientes. Pessoas que prezam pela praticidade de recebimento de produtos em casa e pela economia de tempo, além de terem poder de compra e disposição para gastos do gênero são o principal público do aplicativo. Porém, para uma conquista e retenção mais eficaz de usuários, a empresa deve deixar claro seu diferencial das múltiplas opções de produtos dentro do app, caso contrário seus clientes tem certo “poder de negociação” por conseguir facilmente migrar de um serviço de delivery para outro.

Finalmente, os principais fornecedores para esse modelo de negócio são restaurantes, farmácias e mercados (para produtos comercializados) e também o serviço de entregas prestados por terceiros. Como a análise de Porter diz, o aumento do poder de negociação dos

fornecedores causa uma diminuição na lucratividade da empresa, o que pode ser observado atualmente com a rotatividade alta dos entregadores, que demanda maior incentivo por parte da Rappi para mantê-los.

Principais players, tendências e modelo de negócios

Por representar um cenário de mar vermelho, o mercado de vendas online de produtos consumíveis, principalmente alimentos, tem grandes nomes espalhados pelo globo. No Brasil, o principal nome e dono de mais de 80% do market share é a empresa iFood, [lançada em 2011](#). Por outro lado, considerando a América Latina (excluindo o Brasil), o contexto muda e Rappi entra em cena com mais de 60% dos clientes de aplicativos do tipo, tendo o serviço do Uber Eats como seu concorrente em potencial, apesar de não representar grande ameaça por não possuir grande clientela. Já para compras de supermercado online, outro player também entra em jogo, o Cornershop, também da rede Uber. O serviço não oferece entrega de restaurantes ou farmácias e possui um sistema de delivery bastante semelhante a de seus competidores, porém não representa uma ameaça por possuir baixo índice de market share.

Ao falar dos principais players do ramo, todos tendem a manter um avanço tecnológico crescente para sempre melhorar a experiência de seus usuários e fornecedores. Além de seus sistemas internos e o próprio aplicativo, inteligências artificiais têm grande potencial para extração de conhecimento a partir da análise dos dados coletados em larga escala. A tecnologia pode ser utilizada também para melhor atendimento ao cliente, visto que negócios totalmente online tendem a acumular numerosas reclamações. Por fim, uma abordagem levando em conta os princípios ESG (Environmental, Social and Governance) podem trazer maior lucratividade, visto que a corporação trabalhará em cima de assuntos que vão além do seu papel proposto.

Adentrando o modelo de negócios da Rappi, vê-se uma plataforma digital que almeja conectar estabelecimentos aos clientes, fazendo a ponte por meio das entregas com motoristas próprios da Rappi. Esse modelo nasceu com a finalidade de ser a referência da compra online de qualquer produto na América Latina, em que a compra deve chegar o mais rápido possível nas mãos dos clientes no conforto de seus lares. Ou seja, o diferencial da Rappi é a abrangência de sua cobertura, envolvendo não só o serviço de delivery de alimentos, mas também a entrega de itens de farmácia, locação de carros, reserva de hotéis, voos e muito mais!

4.1.2. Análise SWOT

Forças	Fraquezas
<ul style="list-style-type: none">• <i>Serviços de entregas diversos (Farmácias, Restaurantes, Mercados, etc)</i>• <i>Possuí um tempo de resposta rápido para entregas</i>• <i>Ampla presença no mercado Latino-Americano</i>• <i>Proporciona incentivos financeiros aos clientes regularmente</i>	<ul style="list-style-type: none">• <i>O procedimento de cancelamento para o RT é mal estruturado</i>• <i>Ineficiência na comunicação de diferenciais</i>• <i>Suporte técnico ineficiente</i>• <i>Base de dados de complexo entendimento</i>
Oportunidades	Ameaças
<ul style="list-style-type: none">• <i>Aumento da demanda para delivery</i>• <i>Ascensão da cultura ESG</i>• <i>Ascensão do Home Office / Híbrido</i>	<ul style="list-style-type: none">• <i>Recessão econômica dos países latino-americanos</i>• <i>Forte concorrência; melhor experiência do usuário, maior área de atuação, consolidação, etc.</i>• <i>Alta rotatividade de entregadores</i>• <i>Legislação para entregadores</i>

4.1.3. Planejamento Geral da Solução

4.1.3.1. Qual é o problema a ser resolvido

Atualmente, não existe um vínculo empregatício entre a Rappi e seus entregadores, logo, não há nenhuma razão para os entregadores questionarem o abandono do uso do aplicativo. Estas saídas influenciam na taxa de rotatividade (“Churn Rate”, em inglês) da empresa, visto que, sem os entregadores, não há como o aplicativo oferecer seus diversos serviços ao usuário final e a empresa não gera receita. O problema é, ao mesmo tempo, objetivo, é manter os entregadores satisfeitos ao utilizar a plataforma.

Este é um problema difícil de enfrentar, com diversas maneiras possíveis de ser abordado, porém, toda solução começa com a compreensão; Os problemas dos entregadores são causados por agentes externos ou pela própria empresa? A comunicação entre eles e a empresa está trazendo resultados tangíveis para serem utilizados nesta compreensão? O que é necessário para a empresa conseguir diminuir a taxa de rotatividade? Responder estas questões é o objetivo de nosso trabalho, o problema a ser resolvido.

4.1.3.2. Quais os dados disponíveis

[Link do .ipynb de análise de dados \(Colab\)](#)

[Link do .ipynb de análise de dados \(GitHub\)](#)

4.1.3.3. Qual a solução proposta

Após análise do documento TAPI, é possível montar uma lista de características que a entrega terá, seja por necessidade acadêmica, seja por requisição do cliente.

Como o projeto é realizado também como uma forma de aprendizado, existem requisições acadêmicas de padronizações entre todos os projetos. Como já mencionado, a linguagem de programação utilizada será Python, fazendo uso do Google Collaboratory, onde todos os arquivos provenientes da ferramenta tem extensão *.ipynb*.

Já pelas requisições do parceiro de projetos ou cliente, foi descrita uma saída de dados, onde se destacam os tópicos:

- **Sistema de Pontuação:** Ranquear o RT mais propenso a dar churn para o menos propenso a dar churn.
- **Escala percentual de Churn:** De 0 à 100%, o quão propenso o RT está a dar churn

Após discussões internas do grupo e com o corpo docente, existem alguns detalhes que valem a pena ser ressaltados sobre análise de viabilidade:

- **Dados disponibilizados:** Com os dados disponíveis, não é possível realizar a produção de uma inteligência artificial supervisionada de característica regressiva; ou seja, não há como ranquear os funcionários com a chance percentual de churn. A solução desenvolvida será de classificação, ou seja, os entregadores serão descritos como churn ou não churn.
- **Integração da Solução:** Como o produto do nosso projeto será entregue em como um notebook no Google Colab, não haverá uma integração imediata.

Juntando todas as análises, é possível dizer que a solução pode ser descrita como:

- Código em Python, entregue por meio do Google Colab (extensão *.ipynb*)
- Inteligência Artificial Supervisionada por Classificação

4.1.3.4. Qual o tipo de tarefa

A tarefa realizada pelo grupo foi do tipo classificatório. Como já descrito anteriormente, a inteligência artificial dará uma predição binária, onde o entregador receberá uma label de *churn* ou *não churn*.

4.1.3.5. Como a solução proposta deverá ser utilizada

O modelo construído pelo grupo tem potencial de auxiliar um agente humano para detectar os entregadores mais propensos a dar churn, ajudando a reduzir o descontentamento com a plataforma Rappi e chegar a uma solução do que fazer com esses entregadores. O sistema utilizará um input de tabelas (datasets) e, como saída, retornará o label se o entregador dará churn ou não.

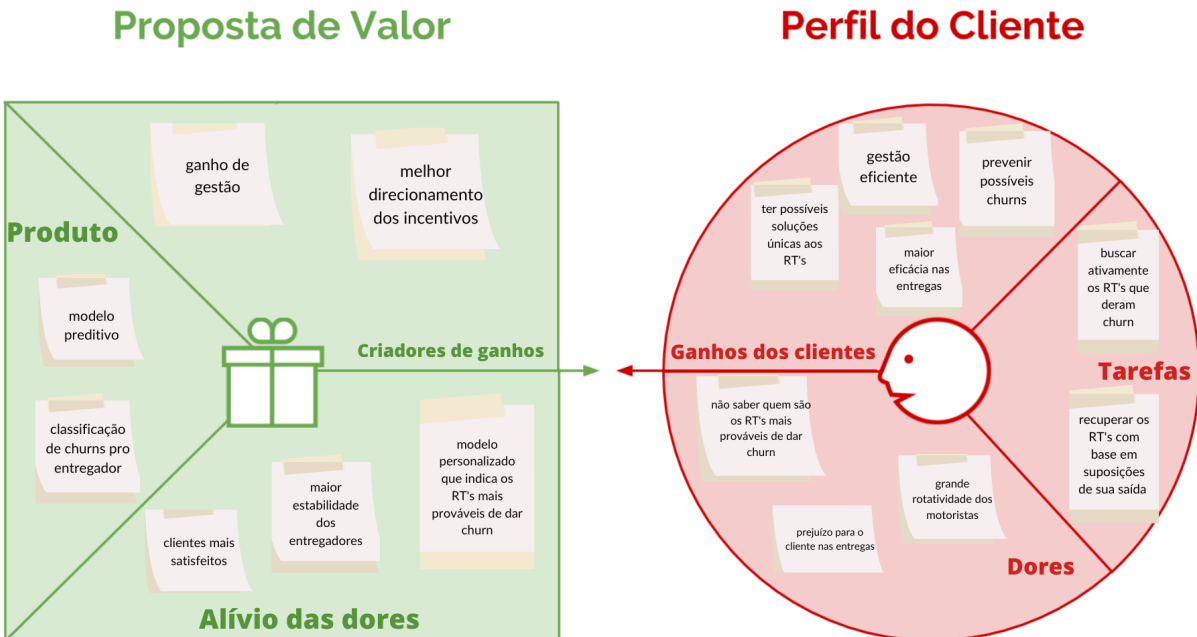
4.1.3.6. Quais os benefícios trazidos pela solução proposta

Os benefícios que a identificação de possíveis churns trará para a Rappi serão, diretamente, a maior retenção de entregadores por meio dos incentivos que serão dados a partir da classificação do modelo. Indiretamente, valores que já foram gerados durante o desenvolvimento do projeto e poderão continuar gerando valor é um maior entendimento dos próprios dados, por meio das análises feitas.

4.1.3.7. Qual será o critério de sucesso e qual medida será utilizada para o avaliar

O sucesso do modelo preditivo será pautado, principalmente, na busca de uma classificação para os entregadores, entre dar churn ou não. Para medir o sucesso da inteligência artificial, serão utilizadas métricas de avaliação de modelos classificatórios, como acurácia, revocação, precisão e F1 score, explicadas com mais profundidade futuramente, na seção 4.5 da documentação, correspondente à avaliação dos modelos.

4.1.4. Value Proposition Canvas



4.1.5. Matriz de Riscos

		Ameaças				Oportunidades				
P r o b a b i l i d a d e	90%						Grupo com capacidades diversas e com muito potencial para ajudar um ao	Conhecer um mercado com potencial enorme, enquanto trabalhamos	Conhecer e compreender a realidade de muitos RTs	
	70%			Falta de conhecimento prévio atrapalhar o andamento do projeto			Muitas possibilidades de se aprimorar tecnicamente			
	50%				Escopo ser muito grande e não possibilitar a entrega					
	30%					Ajudar no dia a dia da Rappi				
	10%		Problemas pessoais		Trabalho ser abaixo do nível desejado tanto pelos integrantes quanto pelo cliente					
	Muito baixo	Baixo	Moderado	Alto	Muito alto	Muito alto	Alto	Moderado	Baixo	Muito baixo
Impacto										

4.1.6. Personas

Acesso ao arquivo com as personas:

[Personas \(Canvas\)](#)



Persona 01 - Profissional de Operações Rappi

Daniel Lopes foi construído com alguns entendimentos sobre a empresa e o problema, entre eles estão:

- **Estrutura Técnica** :

A estrutura técnica apresentada pela Rappi se demonstra inadequada para realizar o trabalho eficientemente, com muitos dados sendo de difícil acesso e falta de normalização nas tabelas extraídas.

- **Decisões Difíceis** :

O time de operações se encontra encarregado de entender os problemas e punir apenas aqueles que tentam fraudar o sistema, o que prova ser uma tarefa de difícil execução.

- **Expectativas** :

Apesar das dificuldades, a expectativa é um modelo preditivo com a resposta de quais entregadores darão churn

Robson Teixeira

Esta pessoa não existe, imagem gerada com auxílio de uma inteligência artificial no site: <https://thispersondoesnotexist.com>



Biografia

Pai, solteiro, desempregado, afetado pelo cenário econômico brasileiro. Formação técnica em mecânica automotiva, ama automóveis e possui moto própria.



Desejos

Dar oportunidades para seus filhos, comprar um automóvel esportivo, aumentar seu padrão de vida, ter uma renda estável, obter ensino superior.

Objetivos

Garantir segurança financeira, garantir qualidade de vida, principalmente de ensino, para seus filhos, tempo de descanso, lazer com sua família, continuar seus estudos

Dores

Instabilidade financeira, ausência de plano de saúde, ausência de direitos, ausência de previdência social, ausência de folga Débito com a empresa

Impedimentos

Trabalhar horas além do saudável
Remuneração baixa
Dependente da empresa e do cliente que realizou o pedido.
Blocks injustos

Motivações

Sobrevivência
Prover financeiramente para seus dependentes
Esforço para melhora de vida

Necessidades

Segurança financeira, saúde, lazer.
Prover para seus dependentes
Ser ouvido

Influências

Thiago Nigro - Autor
Ronaldinho Gaúcho - Jogador de Futebol
Ayrton Senna - Piloto de corrida F1
Will Smith - Ator

PERSONA - RAPPITENDERO

Persona 02 - Entregador Rappi (Rappitendero)

Robson Teixeira foi construído com base em entregadores reais que compartilharam suas opiniões em redes sociais. Com nosso entendimento do problema, os tópicos levantados foram:

- **Cenário Econômico** :

O Brasil está passando por uma fase de recessão econômica e o desemprego é um problema que já se tornou comum nas vidas dos cidadãos. Os entregadores, tanto da Rappi quanto de outros aplicativos, nascem na necessidade de obter renda neste período.

- **Dificuldades**:

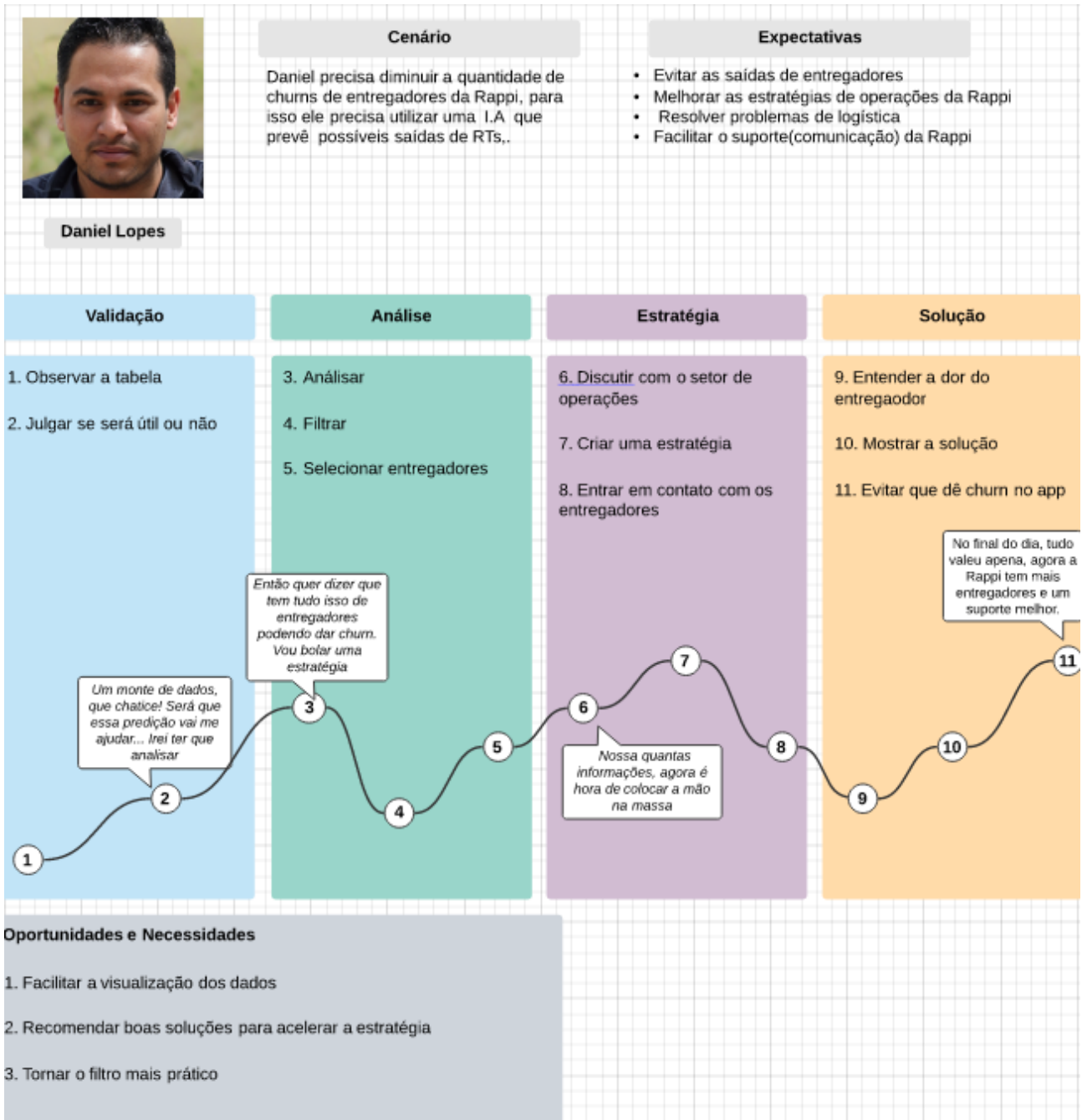
O trabalho em si se apresenta difícil; o pagamento por entrega é inconsistente, o entregador não tem benefícios, direitos trabalhistas, ou plano de saúde, e não tem vínculo empregatício com a empresa, o que gera grande incerteza sobre seu futuro.

- **Imprevisibilidade**:

O churn pode vir de várias formas; insatisfação com a Rappi, uma oferta melhor das empresas competidoras, ou até mesmo a própria saída da profissão de entregador.

4.1.7. Jornadas do Usuário

[Jornada do Usuário \(Lucid.app\)](#)



4.2. Compreensão dos Dados

4.2.1. Dados a serem utilizados

Até o momento, nos foram disponibilizadas 11 tabelas. O método `df.info()` foi utilizado para obter o tipo de dado de cada coluna, como pode ser visto a seguir na seção `Dtype` das capturas de tela feitas de todas as tabelas:

- **2022-08-10 10_40am.csv**
 - Conteúdo: Quilometragem
 - Tamanho: 31.382.215 linhas x 4 colunas - 1,29 GB

11. 2022-08-10 - Quilometragem

```
import pandas as pd
df = pd.read_csv('/content/drive/Shareddrives/Grupo Rappitendeiros/2022-08-10 10_40am.csv')
df.info()
```

/usr/local/lib/python3.7/dist-packages/IPython/core/interactiveshell.py:3326: DtypeWarning: C
exec(code_obj, self.user_global_ns, self.user_ns)
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 31382215 entries, 0 to 31382214
Data columns (total 4 columns):
Column Dtype
--- ---
0 ORDER_ID int64
1 STOREKEEPER_ID float64
2 DISTANCE_TO_USER float64
3 BUNDLE_ID object
dtypes: float64(2), int64(1), object(1)
memory usage: 957.7+ MB

- **attendance_rate.csv**
 - Conteúdo: Aceitação dos pedidos
 - Tamanho: 653.167 linhas x 2 colunas - 7 MB

4. Attendance rate

```
[6] import pandas as pd

df = pd.read_csv('/content/drive/Shareddrives/Grupo Rappitendeiros/attendance_rate.csv')
df.info()
```

<class 'pandas.core.frame.DataFrame'>
RangeIndex: 653167 entries, 0 to 653166
Data columns (total 2 columns):
Column Non-Null Count Dtype
--- ---
0 STOREKEEPER_ID 653166 non-null float64
1 ACCEPTANCE_RATE 263051 non-null float64
dtypes: float64(2)
memory usage: 10.0 MB

- **comp defects.csv**

- Conteúdo: Reclamações de Pedidos
- Tamanho: 6.783.958 linhas x 10 colunas - 388,9 MB

5. Comp Defects

```
import pandas as pd

df = pd.read_csv('/content/drive/SharedDrives/Grupo Rappitendeiros/comp defects.csv')
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 6783958 entries, 0 to 6783957
Data columns (total 10 columns):
#   Column              Dtype
---  -
0   STOREKEEPER_ID      float64
1   WEEK                object
2   CITY                object
3   LEVEL_ID            float64
4   LEVEL_NAME          object
5   ORDERS              int64
6   GMV_TOTAL           float64
7   COMPENSATIONS       float64
8   DEFECT_COMPENSATIONS float64
9   DEFECT_ORDER        float64
dtypes: float64(6), int64(1), object(3)
memory usage: 517.6+ MB
```

- **criacao contas churn-002.csv**

- Conteúdo: Entregadores que deram churn
- Tamanho: 32.568.384 linhas x 9 colunas - 2,63 GB

6. Criação Contas Churn

```
import pandas as pd

df = pd.read_csv('/content/drive/SharedDrives/Grupo Rappitendeiros/criacao contas churn-002.csv')
df.info()
```

```
/usr/local/lib/python3.7/dist-packages/IPython/core/interactiveshell.py:3326: DtypeWarning: Columns (6) have mixed
  exec(code_obj, self.user_global_ns, self.user_ns)
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 32568384 entries, 0 to 32568383
Data columns (total 9 columns):
#   Column              Dtype
---  -
0   ID                  int64
1   FIRST_NAME          object
2   GENDER              object
3   CITY               object
4   SK.CREATED_AT::DATE object
5   TRANSPORT_MEDIA_TYPE object
6   CARTAO             object
7   LEVEL_NAME          object
8   FECHA_ULT           object
dtypes: int64(1), object(8)
memory usage: 2.2+ GB
```

- **earnings.csv**

- Conteúdo: Receita de cada entregador

- Tamanho: 566.099 linhas x 4 colunas - 22,2 MB

7. Earnings

```
import pandas as pd
df = pd.read_csv('/content/drive/Shareddrives/Grupo Rappitendeiros/earnings.csv')
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 566099 entries, 0 to 566098
Data columns (total 4 columns):
#   Column          Non-Null Count  Dtype
---  -
0   MONTH            566099 non-null object
1   STOREKEEPER_ID   566099 non-null int64
2   EARNINGS          566099 non-null float64
3   TIPS              566099 non-null float64
dtypes: float64(2), int64(1), object(1)
memory usage: 17.3+ MB
```

- **Incidentes_Regras RT.csv**
 - Conteúdo: Punições
 - Tamanho: 2.405.601 linhas x 9 colunas - 234,2 MB

1. Incidentes_Regras RT

```
[3] import pandas as pd

df = pd.read_csv('/content/drive/Shareddrives/Grupo Rappitendeiros/Incidentes_Regras RT.csv')
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 2405601 entries, 0 to 2405600
Data columns (total 9 columns):
#   Column              Dtype
---  -
0   DATE                 object
1   NAME                 object
2   INCIDENT_ID          int64
3   STOREKEEPER_ID       int64
4   PUNISHMENT_MINUTES    int64
5   PUNISHMENT_TYPE       object
6   DISCIPLINE_RULE_BUCKET object
7   CATEGORY_RULE         object
8   ORDER_ID             float64
dtypes: float64(1), int64(3), object(5)
memory usage: 165.2+ MB
```

- **infos gerais.csv**

- Conteúdo: Informações gerais sobre entregadores ativos
- Tamanho: 180.178 linhas x 25 colunas - 29,9 MB

8. Infos Gerais

```
import pandas as pd
df = pd.read_csv('/content/drive/SharedDrives/Grupo Rappitendeiros/infos gerais.csv')
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 180178 entries, 0 to 180177
Data columns (total 25 columns):
#   Column                                Non-Null Count  Dtype
---  -
0   ID                                     180178 non-null int64
1   NOME                                  180158 non-null object
2   SOBRENOME                             180164 non-null object
3   GENERO                                180172 non-null object
4   DATA_NASCIMENTO                      180178 non-null object
5   CIDADE                                180178 non-null object
6   IS_ACTIVE                             180178 non-null bool
7   TRANSPORTE                            180178 non-null object
8   AUTO_ACEITE                           180178 non-null bool
9   COUNT_ORDERS_LAST_7D                  180178 non-null int64
10  COUNT_ORDERS_LAST_30D                 180178 non-null int64
11  COUNT_ORDERS_CANCELED_LAST_7D         180178 non-null int64
12  COUNT_ORDERS_CANCELED_LAST_30D        180178 non-null int64
13  GORJETA                                180178 non-null float64
14  PRIMEIRO_PEDIDO                       180178 non-null object
15  ULTIMO_PEDIDO                         180178 non-null object
16  COUNT_ORDERS_RESTAURANTES              180178 non-null int64
17  COUNT_ORDERS_MERCADO                   180178 non-null int64
18  COUNT_ORDERS_FARMACIA                  180178 non-null int64
19  COUNT_ORDERS_EXPRESS                   180178 non-null int64
20  COUNT_ORDERS_ECOMMERCE                 180178 non-null int64
21  COUNT_ORDERS_ANTOJO                    180178 non-null int64
22  FRETE_MEDIO                           180176 non-null float64
23  COOKING_TIME_MEDIO                     161449 non-null float64
24  ITENS_MEDIO                            179826 non-null float64
dtypes: bool(2), float64(4), int64(11), object(8)
memory usage: 32.0+ MB
```

- Orders Done e Cancel.csv

- Conteúdo: Pedidos atendidos e cancelados
- Tamanho: 653.166 linhas x 4 colunas - 9,1 MB

2. Ordens Done e Cancel

```
import pandas as pd

df = pd.read_csv('/content/drive/SharedDrives/Grupo Rappitendeiros/Ordens Done e Cancel.csv')
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 653166 entries, 0 to 653165
Data columns (total 4 columns):
#   Column                Non-Null Count  Dtype
---  -
0   STOREKEEPER_ID        653166 non-null int64
1   ORDERS_DONE            653166 non-null int64
2   ORDERS_CANCEL          653166 non-null int64
3   CANCELS_OPS_RT        653166 non-null int64
dtypes: int64(4)
memory usage: 19.9 MB
```

- Product return.csv

- Conteúdo: Retorno de pedidos cancelados
- Tamanho: 41.535 linhas x 11 colunas - 4,6 MB

3. Product Return

```
[5] import pandas as pd

df = pd.read_csv('/content/drive/Shareddrives/Grupo Rappitendeiros/Product return.csv')
df.info()

<class 'pandas.core.frame.DataFrame'>
RangeIndex: 41535 entries, 0 to 41534
Data columns (total 11 columns):
#   Column                Non-Null Count  Dtype
---  -
0   ID_ENTREGADOR          41535 non-null  int64
1   LEVEL_NAME             41535 non-null  object
2   MODAL                  41535 non-null  object
3   CITY                   41535 non-null  object
4   CREATED_AT             41535 non-null  object
5   ORDER_ID               41535 non-null  int64
6   PRODUCT_RETURNS        41535 non-null  float64
7   VERTICAL_SUB_GROUP     41535 non-null  object
8   COUNT_TO_GMV           41535 non-null  bool
9   GMV                    41535 non-null  float64
10  STORE_ID               41535 non-null  int64
dtypes: bool(1), float64(2), int64(3), object(5)
memory usage: 3.2+ MB
```

- **supply.csv**
 - Conteúdo: Tempo de conexão do RT
 - Tamanho: 124.526 linhas x 10 colunas - 10,1 MB

9. Supply

```
[10] import pandas as pd

df = pd.read_csv('/content/drive/Shareddrives/Grupo Rappitendeiros/supply.csv')
df.info()

<class 'pandas.core.frame.DataFrame'>
RangeIndex: 124526 entries, 0 to 124525
Data columns (total 10 columns):
#   Column                Non-Null Count  Dtype
---  -
0   CITY                   124526 non-null  object
1   DATE                   124526 non-null  object
2   WEEK                   124526 non-null  object
3   CREATED_CARD           106279 non-null  object
4   STOREKEEPER_ID         124526 non-null  int64
5   LEVEL_NAME_2           113497 non-null  object
6   HAVE_CARD              124526 non-null  int64
7   TRANSPORT_MEDIA_TYPE   124526 non-null  object
8   NUM_ORDERS             104034 non-null  float64
9   SUPPLY_HOURS           124526 non-null  float64
dtypes: float64(2), int64(2), object(6)
memory usage: 9.5+ MB
```

- **tempo resolucao e modal-001.csv**
 - Conteúdo: Tempo de espera pelo suporte
 - Tamanho: 32.568.384 linhas x 9 colunas - 2,63 GB

10. Tempo Resolução e Modal

```
import pandas as pd
df = pd.read_csv('/content/drive/SharedDrives/Grupo Rappitendeiros/tempo resolutcao e modal-001.csv')
df.info()
```

```
/usr/local/lib/python3.7/dist-packages/IPython/core/interactiveshell.py:3326: DtypeWarning: Columns (6)
  exec(code_obj, self.user_global_ns, self.user_ns)
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 32568384 entries, 0 to 32568383
Data columns (total 9 columns):
#   Column                Dtype
---  ----
0   ID                    int64
1   FIRST_NAME            object
2   GENDER                object
3   CITY                  object
4   SK.CREATED_AT::DATE   object
5   TRANSPORT_MEDIA_TYPE object
6   CARTAO                object
7   LEVEL_NAME            object
8   FECHA_ULT             object
dtypes: int64(1), object(8)
memory usage: 2.2+ GB
```

○

Todos estes dados são de fonte da própria empresa e de manipulação interna

4.2.1.1. Descrição de agregação e mesclagem de dados

Utilizando dados que são comuns em diferentes tabelas como o “STOREKEEPER ID”, é possível realizar agregações, que são fundamentais para a criação de novas features que estão documentadas na seção 4.3.

Exemplos de agregações :

```
df_dados = df_attendance.merge(df_earnings_earnings, how='inner', left_on='STOREKEEPER_ID', right_on='STOREKEEPER_ID') # Earnings
df_dados2 = df_dados.merge(df_compDef_level, how='inner', left_on='STOREKEEPER_ID', right_on='STOREKEEPER_ID') # Level
df_dados3 = df_dados.merge(df_orders_done_cancel, how='inner', left_on='STOREKEEPER_ID', right_on='STOREKEEPER_ID') # Orders done e cancel
df_dados1 = df_dados.merge(df_productReturn_return, how='inner', left_on='STOREKEEPER_ID', right_on='STOREKEEPER_ID') # Product Return
df_kill_distance = df_kill[["DISTANCE_TO_USER", "STOREKEEPER_ID"]] # Distância
```

Mais visualizações da compreensão dos dados estão presentes nas seções “Entendimento dos dados” e “Preparação dos dados” no colab

4.2.1.2. Descrição dos riscos e contingências relacionados a esses dados

Recebemos uma ampla variedade de dados, entretanto, cabem algumas ressalvas:

Sobre qualidade: as tabelas enviadas apresentam fatores que complicam a análise, como haver várias colunas com nomes diferentes representando o mesmo conceito – algo contornável com uma análise mais apurada e compreensão da biblioteca “Pandas”. Ademais, há algumas colunas sem valor, o que prejudica na predição, contudo com o balanceamento dos dados e técnicas para padronização, também não é uma falha crítica

Já no que se refere à cobertura dos dados, o objetivo do modelo preditivo proposto pela Rappi – a classificação de 0 a 100 da probabilidade de um entregador deixar a plataforma – não será atingido com os dados apresentados, uma vez que as informações fornecidas são binárias, isto é, evidenciam com “sim” ou “não” se um entregador deu churn, não havendo uma gradação que permita prever um valor numérico entre os extremos.

O acesso aos dados ocorreu de maneira prevista, sem intercorrências.

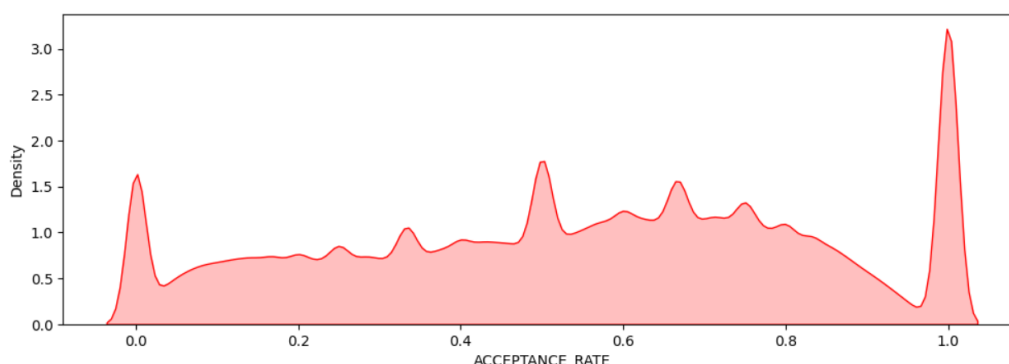
A solução encontrada pelo grupo é, desenvolver um sistema que explicita quais entregadores são prováveis de dar churn e quais não, mostrando ao usuário da aplicação a relação do nome do entregador com uma marcação “sim” ou “não” para possibilidade de churn.

4.2.1.3. Descrição de como será selecionado o subconjunto para análises iniciais

Fizemos 5 análises:

- *Taxa de aceitação do entregador:*

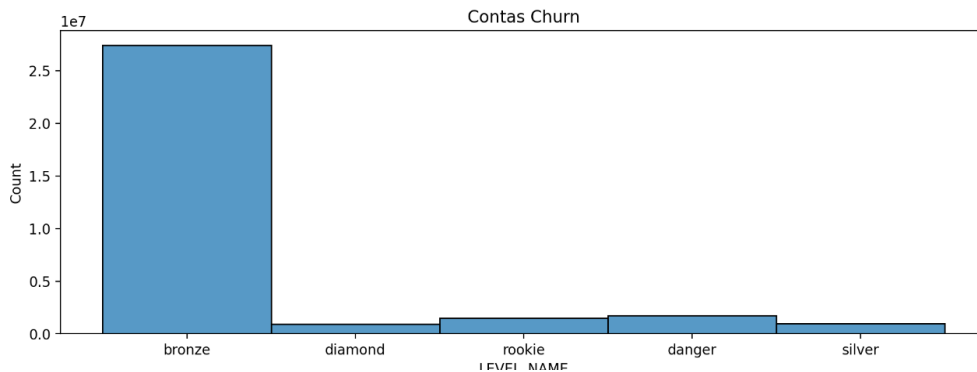
Esse gráfico demonstra o nível de aceitação pela densidade. Grande parte dos entregadores aceitam todos os pedidos, mas a variação presente no gráfico se dá devido aos entregadores que não aceitam todos os pedidos. Mas, mesmo com essa variação, o gráfico continua bem distribuído.



- *Taxa de level por conta churn:*

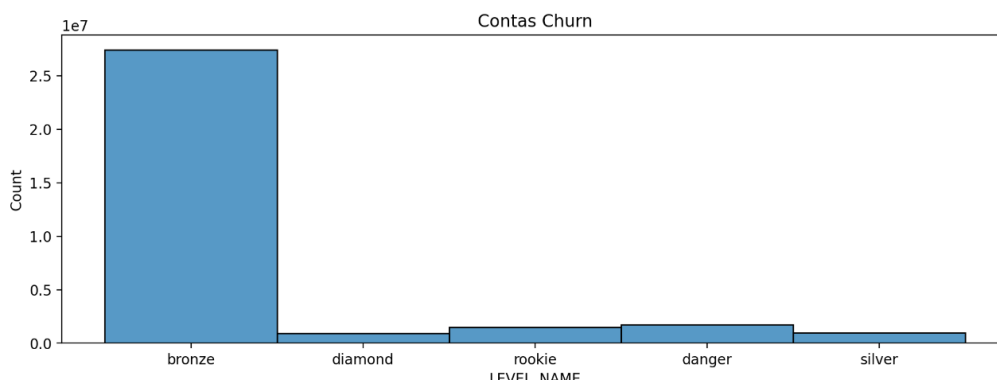
Esse gráfico diz respeito ao nível do entregador com relação às contas churns, ou seja, as contas que o entregador abandonou. Aqui há uma discrepância muito alta, pois

podemos ver entregadores de nível alto (bronze) tendo uma taxa de churn muito alta. Temos duas interpretações para esse gráfico: as contas bronze são maiores (em número), então o número de churn é maior; os entregadores aumentam o rendimento, aumentam o nível, e voltam a deixar o rendimento cair, fazendo com que o nível diminua também. Com isso, eles se frustram e acabam dando churn.



- *Auto aceite entre contas churn:*

Esse gráfico demonstra os entregadores que têm o auto aceite ativo e têm a conta churn, ou seja, estão saindo ou pensam em sair da empresa. 60% mantêm o auto aceite e possuem conta churn, e 40% possuem auto aceite e não deram churn. Com isso, podemos concluir que a maioria dos entregadores que tem o auto aceite ativo deram churn, possivelmente por haver muitos pedidos e a carga de trabalho ser muito pesada, fazendo com que eles se canssem e decidam deixar a empresa pelo baixo retorno.

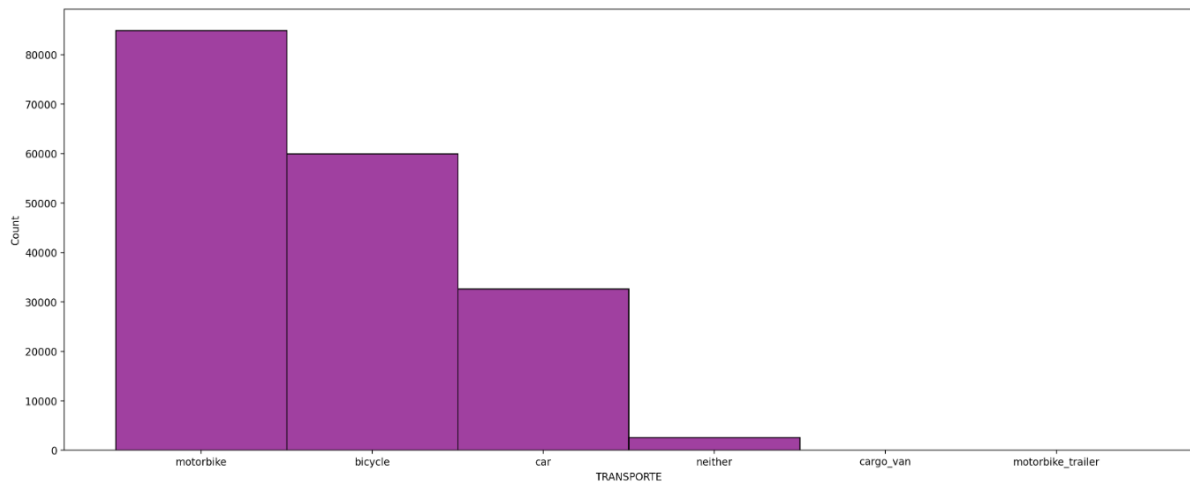


- *Meios de transporte usados pelos entregadores:*

Esse gráfico demonstra que os entregadores preferem veículos econômicos, em que eles possam gastar menos dinheiro com combustível. Por isso, a maioria utiliza bicicleta

e

moto.



4.2.1.4. Descrição das restrições de segurança.

Uma vez que o modelo elaborado envolve a interpretação de bancos de dados com informações extremamente sensíveis à Rappi, há uma grande preocupação com a segurança dos dados encaminhados e analisados. Sendo assim, o grupo redator dessa documentação se compromete a manter em total confidencialidade qualquer informação contida neste projeto de modelo preditivo, a fim de prezar nosso compromisso para com a empresa.

4.2.2. Descrição estatística básica dos dados

[Link do .ipynb de descrição estatística básica dos dados \(Colab\)](#)

[Link do .ipynb de descrição estatística básica dos dados \(GitHub\)](#)

Para a descrição estatística básica dos dados foram utilizados os gráficos descritos e mostrados na seção 4.2.1.3. em conjunto com as seguintes análises:

- *Taxa de aceitação do entregador:* É visível que existem muitos RTs com notas 0,0 ou 1,0 que pensamos ser entregadores com muito poucas entregas e seus dados podem não ser tão benéficos para o desenvolvimento do produto. Outro ponto relevante é a quantidade de RTs próximos de 0,5, que dialoga diretamente com a média obtida (0,537). O restante do gráfico é relativamente constante, com uma maior concentração na região acima de 0,5

- *Taxa de level por conta churn:* É perceptível que o número do rank bronze, é extremamente superior ao resto dos Levels, portanto, provando que o maior número de churn são Bronzes(depois de atingir 20 entregas) e não rookies(antes de atingir 20 entregas)
- *Auto aceite entre contas churn:* Diferente do que o grupo esperava, a opção de autoaceite não tem um grande impacto na taxa de churn, tendo um aumento muito pequeno no resultado. Não é um resultado muito útil para a equipe
- *Meios de transporte usados pelos entregadores:* A maioria dos entregadores utilizam motos, ou seja, está dentro da hipótese que a gasolina pode influenciar o churn.

4.2.3.Descrição da predição desejada

Nosso modelo preditivo terá foco na análise da satisfação dos entregadores e realizará seu monitoramento, por meio de referenciais, como:

- TOTAL_RESPONSE_TIME;
- YEARLY_EARNINGS;
- DISTANCE_PER_ORDER;
- TOTAL_ORDERS;
- RT_PUNISHED_ORDERS;
- ORDERS_DONE_RATE;
- CHURN;

Esses dados revelam quem será nosso alvo para a elaboração do modelo preditivo. O algoritmo terá como base os dados do RT(mencionados anteriormente), em que será possível estimar a probabilidade de churn deste indivíduo. Dessa forma, o resultado será uma inteligência artificial supervisionada que consegue responder, com mais de 90% de precisão, se determinado entregador vai dar churn.

Diante disso, podemos observar que essas informações serão muito úteis para a construção de nosso algoritmo, visto que nosso produto se baseará nos ganhos do RT (YEARLY_EARNINGS), seu rendimento (ORDERS_DONE_RATE) e outras features mencionadas acima. O modelo criado pelo grupo irá, a partir dessas informações, determinar se determinado entregador está tendendo ao churn ou não, possibilitando intervenção por parte da Rappi.

Um modelo preditivo de inteligência artificial tem como característica a renovação de seus resultados com a inserção de novos parâmetros; possibilitando uma fácil atualização quando for necessário.

Devido a natureza dos dados fornecidos pelo parceiro, não será possível o rankeamentos dos entregadores (de 0% de chance de churn até 100%). Então o tipo de modelo utilizado pelo grupo foi um modelo de classificação capaz de dizer que aquele RT tem maior probabilidade de dar ou não o churn.

4.3. Preparação dos Dados

Código IPYNB referente à primeira análise e preparação dos dados dos itens 4.3.1, 4.3.3 e 4.3.4 da feature engineering:

[Link do .ipynb de feature engineering - preparação dos dados \(Colab\)](#)

[Link do .ipynb de feature engineering - preparação dos dados \(GitHub\)](#)

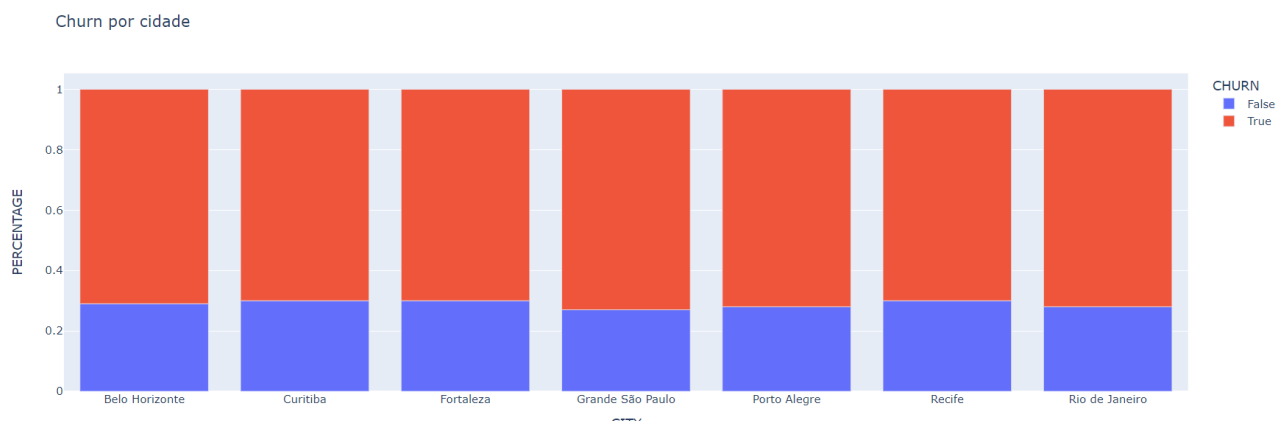
4.3.1. Descrição de manipulações nos registros e suas respectivas features.

Target - Utilizando a tabela “*criação contas churn*”, separamos as contas que deram churn e eliminamos as duplicatas. Ao realizar esse processo, obtivemos os IDs necessários e que serão utilizados nas outras features.

```
df_churn.head()
```

	ID	CHURN
0	1286316	True
1	1110698	True
2	284886	True
3	1036587	True
4	106854	True

Infos gerais - Utilizando a informação obtida no “Target”, cruzamos uma das tabelas que contém informações das cidades nas quais os RTs trabalham (tabela “*info gerais*”), obtendo os números de churn por cidade. Como a taxa de churn por cidade é muito semelhante, como pode-se ver no gráfico abaixo, é irrelevante utilizar os dados de cidade dos entregadores como feature.



Product return, Cartão, GMV e Punishment type - Foram alteradas apenas os nomes das colunas das tabelas para padronização.

Em segunda análise, pela segunda vez que a metodologia CRISP-DM era rodada, uma preparação de dados mais aprofundada foi feita. Essa preparação de dados pode ser consultada nos seguintes links (as novas agregações e derivações feitas podem ser encontradas na seção “Preparação dos Dados”):

[Link do .ipynb do template – preparação dos dados \(Colab\)](#)

[Link do .ipynb do template – preparação dos dados \(GitHub\)](#)

	ID	CHURN
0	1286316	True
1	1110698	True
2	284886	True
3	1036587	True
4	106854	True
...
204184	33249	False
204185	33203	False
204186	33194	False
204187	33161	False
204188	33052	False
204189 rows × 2 columns		

Assim como antes, foi criada a feature target, utilizando as tabelas “*criacao contas churn-002.csv*” e “*infos gerais.csv*”. A primeira tabela mencionada reúne todas contas que já deram churn registradas no banco de dados da Rappi, já a segunda tabela tem dados de todos os entregadores já registrados no aplicativo. Com a junção dos dados das duas tabelas, temos informações dos RTs que estão classificados como “true” e “false”, ou seja, o target do projeto:

A partir do target e das outras tabelas disponibilizadas pela Rappi, foi possível a criação/derivação das seguintes features:

- **PERMANENT_BLOCK**: feature de caráter booleano que pega os dados *permanent_block*, da coluna *PUNISHMENT_TYPE*, da tabela *Incidentes_Regras RT*, e torna como *true* todos os RTs que já tomaram block permanente.
- **FRAUD**: feature de caráter booleano que pega os dados *Fraud*, da coluna *PUNISHMENT_TYPE*, da tabela *Incidentes_Regras RT*, e torna como *true* todos os RTs que foram classificados como fraude.
- **ORDER_ID_PUNISH**: feature de caráter numérico que pega os dados *ORDER_ID_PUNISH*, da coluna *PUNISHMENT_TYPE*, da tabela *Incidentes_Regras RT*, e retorna o ID das punições que o RT já tomou (relacionado ao tipo de punição).
- **RT_PUNISHED_ORDERS**: feature de caráter numérico que pega os dados da coluna *ORDER_ID*, da tabela *Incidentes_Regras RT*, e soma o total de punições tomados pelo RT.

- **ORDERS_DONE_RATE**: feature de carácter numérico que soma todos os pedidos do RT, incluindo cancelados pelo cliente e pelo time de operações da Rappi, utilizando dados das colunas *ORDERS_DONE*, *ORDERS_CANCEL* e *CANCELS_OPS_RT*, da tabela *Orders done e Cancel*.
- **TOTAL_ORDERS**: feature de carácter numérico que faz a média aritmética entre os dados das seguintes colunas: $\frac{TOTAL_ORDERS + CANCELS_OPS_RT + ORDERS_CANCEL}{TOTAL_ORDERS}$, todos obtidos da tabela *Orders done e Cancel*.
- **YEARLY_HOURS**: feature de carácter numérico que pega os dados da coluna *HOURS_CONNECTED*, da tabela *Supply (New)*, referente a todas as horas que o RT passou conectado à plataforma.
- **YEARLY_ORDERS_DONE**: feature de carácter numérico que pega os dados da coluna *ORDERS*, da tabela *Supply (New)*, referente ao número de entregas realizadas pelo RT em por ano.
- **YEARLY_EARNINGS**: feature de carácter numérico que soma os dados da coluna *EARNINGS* e *TIPS*, da tabela *Earnings*.
- **YEARLY_TIPS**: feature de carácter numérico que pega os dados de gorjetas da coluna *TIPS*, da tabela *Earnings*.
- **YEARLY_GMV**: feature de carácter numérico que representa a soma dos dados das colunas *PRODUCT_RETURNS* e *GMV*.
- **YEARLY_PRODUCT_RETURNS**: feature de carácter numérico que representa o total de GMV acumulado pelo RT por ano, retirado da coluna *GMV*, da tabela *Product return*.
- **DEFECT_LAST_WEEK**: feature de carácter numérico (representado uma data), que utiliza os dados das colunas *WEEK* e *GMV_TOTAL*, da tabela *Comp defects*, que representa a última semana que o RT recebeu reclamações de defeitos.
- **TOTAL_GMV**: dado de carácter numérico que utiliza o dado *GMV_TOTAL* da tabela *Comp defects*.
- **YEARLY_ORDERS** e **YEARLY_COMPENSATIONS**: features de carácter numérico que representam, respectivamente, o número de entregas feitas anualmente pelo RT e a soma do valor em pago (devolvido) ao usuário por ano.
- **DEFECTS_LEVEL_NAME** e **DEFECTS_LEVEL_ID**: features de carácter categórico e numérico, respectivamente, onde o ID é apenas uma label para o nível do RT. Utilizam os dados das colunas *LEVEL_NAME* e *LEVEL_ID*, da tabela *Comp defects*.
- **TOTAL_RESPONSE_TIME**: feature de carácter numérico, que utiliza os dados das colunas *RESPONSE_TIME* e *TICKET_ID*, da tabela *Tempo de resolução e modal*, que representa o tempo de resposta, em minutos, do time da Rappi quanto ao ticket de reclamação aberto.
- **TOTAL_RESOLUTION_TIME**: feature de carácter numérico, que utiliza os dados das colunas *RESOLUTION_TIME* e *TICKET_ID*, da tabela *Tempo de resolução e modal*, que representa o tempo de resolução do ticket, em minutos, do time da Rappi quanto ao ticket de reclamação aberto.

- **TOTAL_DISTANCE** e **DISTANCE_PER_ORDER**: features de caráter numérico, utilizando dados da coluna *DISTANCE_TO_USER*, da tabela *2002-08-10(Quilometragem)*, que representam, respectivamente, a distância total percorrida pelo entregador e a média da distância de todos os pedidos.

4.3.2. Agregação de registros e/ou derivação de novos atributos.

Código das agregações e derivações:

[Link do .ipynb de agregações e derivações \(Colab\)](#)

[Link do .ipynb de agregações e derivações \(GitHub\)](#)

- Transformamos o dado "DISTANCE_TO_USER", o qual antes da mudança continha a distância por pedidos do entregador, para a soma das distâncias por pedidos, assim criando um dado único para o entregador.
- Transformamos o dado "SUPPLY_HOURS", o qual antes era um conjunto de dados desagregados das horas ativas no aplicativo por ID, na soma destes dados, indicando apenas um único dado total por ID.
- Mesclamos o dado "ORDERS_DONE" com "ORDERS_CANCEL" para calcular o rendimento total do entregador ("ORDERS_DONE_RATE") (tabela: *Ordens Done e Cancel.csv*).
- Modificamos o dado "EARNINGS", o qual se referia às várias quantias por mês do entregador, pela média dessas quantias, transformando em apenas um dado total por ID.
- O cálculo de "EARNINGS_PER_ORDER" e "EARNINGS_PER_DISTANCE" foi feito utilizando o quanto o entregador ganhou dividido pela quantidade de entregas realizadas e km andados para a entrega, respectivamente (tabela: *earnings.csv* e *2022-08-10 10_40am.csv*).
- O "STOREKEEPER_YEAR" foi calculado utilizando o dia, o mês e o ano de nascimento do RT (tabela: *infos gerais.csv*).
- O "ID" foi colocado em nosso modelo para identificar o entregador e com isso identificar se as informações que estão presentes no modelo (outras features) levam o trabalhador ao Churn ou não.
- O "TOTAL_RESPONSE_TIME" é a soma de quanto tempo A Rappi levou para responder às reclamações do entregador por ano.
- A "YEARLY_EARNINGS" é quanto de receita o entregador obteve durante as entregas. Ela é uma feature que foi derivada somando todo o dinheiro que o entregador recebeu em uma ano (365 dias).
- A "DISTANCE_PER_ORDER" é quantos pedidos o entregador realizou por entrega. Ela foi obtida a partir da soma de todos os pedidos feitos em um ano, dividido pela distância total percorrida pelo entregador durante um ano.
- A "ORDERS_DONE_RATE" é uma outra derivação e significa a taxa de aceitação de pedido feito por um entregador.

4.3.3. Remoção ou substituição de valores ausentes/em branco.

- Na tabela "*eliminação de contas churn*", eliminamos os IDs duplicados.

- Na tabela “*infos gerais*”, selecionamos as cidades com o maior número de contas e desconsideramos algumas cidades com nomes que não a representavam, como por exemplo a cidade que tinha o nome “10”.
- Removemos as duplicatas para a criação da feature “*Cartão*” na tabela “*criação de contas churn*”.
- Em linhas gerais, os valores nulos/ausentes foram preenchidos com 0.

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

- Feature “Cidades” - Criamos essa feature com o objetivo de verificar quais cidades possuíam uma maior quantidade de churns para possíveis usos no futuro. O resultado dessa feature foi diferente do esperado e todas as cidades com quantias consideráveis de entregadores tinham uma taxa de churn parecida, por isso não utilizamos no modelo.
- Feature “Cartão” - Criamos essa feature para verificar se o RT que deu churn possuía ou não o cartão da Rappi. Nós a criamos pois esperávamos que a posse do cartão teria impacto direto com a decisão do motoqueiro de deixar o app pelas supostas dívidas contraídas.
- Feature “STOREKEEPER_YEAR” - Transformamos o ano de nascimento do RT na idade real dele. Essa feature foi criada para verificar se a idade faz ou não diferença na decisão do churn e caso faça, no futuro seja de fácil reconhecimento.
- Feature “ORDERS_DONE_RATE” - Criado para verificar a porcentagem de sucesso na entrega de cada entregador. Caso o entregador esteja com uma taxa de entrega baixa, pode ser um sinal de fragilidade quanto ao churn. Um entregador pode “falhar” na entrega quando ele, o cliente ou a operação da Rappi cancele o pedido.
- Feature “EARNINGS_PER_ORDER” e “EARNINGS_PER_DISTANCE” - Como diferentes entregadores trabalham um período diferente de tempo, o valor bruto acumulado não é um fator decisivo para descobrir se aquele motoqueiro pode dar churn, por isso criamos essas duas features, que têm a função de dividir o dinheiro ganho pela quantidade de pedidos e pela distância percorrida pelo RT, respectivamente. Ela serve para mostrar qual o real ganho do entregador.

4.4. Modelagem

[Link do .ipynb de modelagem \(Colab\)](#)

[Link do .ipynb de modelagem \(GitHub\)](#)

Em primeira análise, vale constatar que, dentre as inúmeras possibilidades de algoritmos pré-prontos que embasam a solução de Inteligência Artificial proposta, nosso grupo escolheu 8 algoritmos principais. A escolha de cada modelo será desenvolvida logo nos próximos tópicos, sendo eles: K Nearest Neighbour - KNN, Decision Tree, Naive Bayes, Random Forest, Regressão logística, Support Vector Machines - SVM, Ada Boost e Extra Trees.

Com uma análise mais apurada, modelos como o XGBoost e Catboost também foram escolhidos para o desenvolvimento.

Como o objetivo da Rappi é obter uma predição com Inteligência Artificial que evidencie quais entregadores tendem a dar Churn e, após alinhamento com o representante da empresa, foi constatado que é preferível que o modelo inclua em sua predição trabalhadores que não tendem a deixar a plataforma do que deixar de fora aqueles que podem dar Churn. O parâmetro de avaliação da predição foi, principalmente, a acurácia do modelo, isto é, qual a porcentagem de predições corretas dentre todas realizadas, e a revocação, que indica a porcentagem de Churn que o modelo considerou dentre todos que, de fato, deram Churn, deixando nas entrelinhas uma pequena taxa se houver Churns que o modelo não considerou, ou seja, falsos negativos, sendo essa uma métrica fundamental de ser próxima de 0 por requisição do cliente. Foram também apresentadas as métricas de precisão e F1 score, explicadas com maior profundidade futuramente.

Uma vez que foram fornecidas distintas tabelas com informações separadas que se relacionavam somente pelo ID do entregador, foi constatado que não havia uma relação perfeita, isto é, há dados como o salário do entregador que estavam nulos para alguns ID's. Então, ao fazer a junção das tabelas, existiam linhas com valores nulos em alguns campos importantes. Como solução temporária, foi decidido remover essas linhas, a fim de que sejam mantidos somente valores coletados. Na mesma linha de pensamento, foi observado que, após a limpeza, a quantidade de entregadores que deram Churn estava bem desproporcional em relação àqueles que não deram, sendo esse um grande problema para a Inteligência Artificial, por enviesá-la a prever o valor mais recorrente (churn positivo). A partir disso, foi necessário igualar as amostras e o caminho escolhido foi descartar valores da maior amostra, até que ela se igualasse com a menos presente. Esse balanceamento, undersampling, pode ser observado na imagem abaixo, e foi escolhido para não criar valores artificiais:

```
CHURN
1      78024
0      21115
dtype: int64
```

```
CHURN
0      21115
1      21115
dtype: int64
```

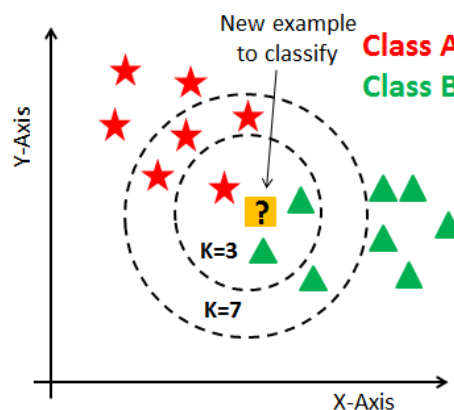
Para cada modelo incluído na documentação, foram apresentadas as respectivas matrizes de confusão, indicando os resultados verdadeiros positivos (entregadores que deram churn e o modelo previu corretamente), verdadeiros negativos (entregadores que não deram churn e foram previstos corretamente), falsos positivos (entregadores que não deram churn mas o modelo previu como churn) e falsos negativos (entregadores que deram churn e o modelo previu como não churn). Segue a explicação da matriz de confusão em imagem para melhor entendimento:

		Valor Predito	
		Sim	Não
Real	Sim	Verdadeiro Positivo (TP)	Falso Negativo (FN)
	Não	Falso Positivo (FP)	Verdadeiro Negativo (TN)

4.4.1. KNN(K-Nearest Neighbors)

4.4.1.1. Explicação do modelo

O modelo K-Nearest Neighbours foi importado da biblioteca *sklearn.neighbours* e funciona pela classificação de padrões que prediz o resultado a partir do cálculo da distância entre o novo dado de treinamento inserido ainda não classificado e seus vizinhos dentro de um raio específico, como é ilustrado na imagem ao lado.



Idealmente, o número K de vizinhos deve ser ímpar para que suas previsões não fiquem divididas entre 50% de decisão. Além disso, quanto maior o número de vizinhos selecionados, maior será a tendência da classificação ser a mesma, uma vez que a previsão é baseada nos valores mais próximos e não pela quantidade absoluta.

4.4.1.2. Motivo de escolha do modelo

O modelo KNN foi um dos escolhidos porque utiliza um modelo de classificação por padrões de características semelhantes entre todas as ocorrências das features, sendo interessante por possuímos uma grande amostra, que possibilita incluir valores dentro de uma

faixa já existente, ou seja, teoricamente sempre haverá algum dado próximo para servir de referência. Como foram escolhidas mais de três features, não é possível fazer um mapeamento visual de proximidade de ocorrências para melhor análise da classificação, uma vez que cada feature corresponde a um eixo do gráfico, sendo um modelo com mais de 3 dimensões.

4.4.1.3. Resultados obtidos

Ao rodar as features selecionadas com dados rebalanceados utilizando a técnica de undersampling, os seguintes resultados foram obtidos:

Acuracidade (treino): 0.8984

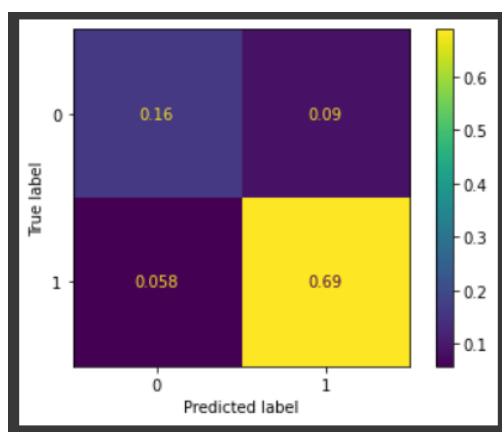
Acuracidade (teste): 0.8514

Revocação: 0.9217

Precisão: 0.8841

F1_score: 0.9025

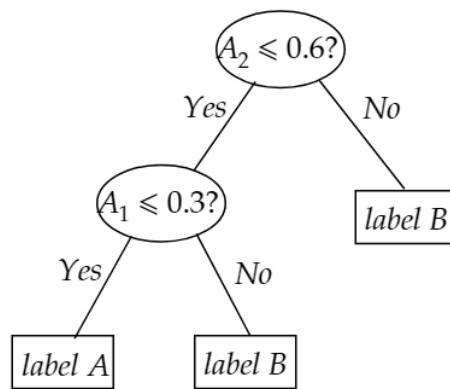
Matriz de confusão:



4.4.2 Decision Tree

4.4.2.1. Explicação do modelo

O modelo de classificação de árvore de decisão consiste em um conjunto de escolhas hierárquicas, separadas entre nós internos e externos (folhas). Ao utilizar esse modelo, é especificada a profundidade de decisões feitas pela árvore, isto é, quantos níveis de escolhas serão feitas. Para não ocorrer overfitting na classificação, os níveis de decisão não devem ser tão profundos na fase de treino do modelo.



4.4.2.2. Motivo de escolha do modelo

O modelo foi escolhido por analisar os padrões de forma binária, onde cada nó da árvore é destrinchado em dois outros nós por meio de uma condição. As condições são feitas com base em padrões encontrados nos dados e, assim, os dados novos inseridos são classificados. Por conta dessa análise de padrões por similaridade dos dados, a árvore de decisão é um dos modelos escolhidos para classificação do projeto, por apresentar uma lógica que pode ser válida.

4.4.2.3. Resultados obtidos

Acuracidade (treino): 0.8896

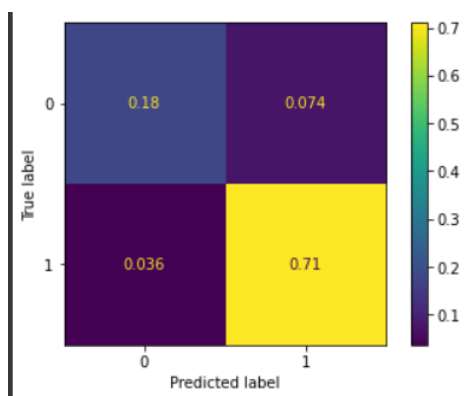
Acuracidade (teste): 0.8902

Revocação: 0.9515

Precisão: 0.9062

F1_score: 0.9283

Matriz de confusão:



4.4.3 Naive Bayes

4.4.3.1. Explicação do modelo

The diagram illustrates the Naive Bayes formula with the following components and labels:

- Likelihood of the Evidence given that the Hypothesis is True** (orange text, points to $P(E|H)$)
- Prior Probability of the Hypothesis** (red text, points to $P(H)$)
- Posterior Probability of the Hypothesis given that the Evidence is True** (blue text, points to $P(H|E)$)
- Prior Probability that the evidence is True** (green text, points to $P(E)$)

$$P(H|E) = \frac{P(E|H) * P(H)}{P(E)}$$

O modelo Naive Bayes foi importado da biblioteca `sklearn.naive_bayes` e funciona a partir da suposição da independência entre variáveis do problema, realizando uma classificação probabilística de observações, caracterizando-as em classes pré-definidas. Sendo um modelo para a classificação de atributos discretos, o Naive Bayes tem aplicações na análise de crédito, diagnósticos médicos ou busca por falhas em sistemas mecânicos.

É interessante saber que o Naive Bayes é um dos modelos mais conhecidos a aplicar o conceito de probabilidade. Esse modelo, como o nome indica, faz uso do teorema de Bayes como princípio fundamental ($P(A/B) = P(B/A)P(A)/P(B)$). Essa fórmula indica que a probabilidade de um evento A ocorrer é o produto da probabilidade de um evento B ocorrer dado que A ocorreu pela a probabilidade de um evento A ocorrer, dividido pela probabilidade de um evento B ocorrer. Desse modo, o algoritmo calcula a chance de um fenômeno ocorrer novamente, sendo bem aplicável a lógica de indicar se uma pessoa dará Churn com base em um histórico.

4.4.3.2. Motivo de escolha do modelo

O modelo Naive Bayes foi escolhido pela facilidade que o algoritmo possui em calcular a probabilidade de um evento ocorrer no futuro dado que ocorreu no passado. Sendo assim, o Naive Bayes nos proporciona analisar quais entregadores possuem a maior chance de dar Churn, baseado em seu rendimento de trabalho e quanto essa produtividade corroborou no abandono do trabalhador do aplicativo. O modelo usa um método efetivo para prever a probabilidade de classe diferente com base em vários atributos, assim ele realiza a leitura de todas as nossas features escolhidas e como elas influenciaram nas contas Churn e calcula chance de ocorrência desse fenômeno de saída dos entregadores do aplicativo da Rappi para o novo dataset.

4.4.3.3. Resultados obtidos

Acuracidade (treino): 0.8046

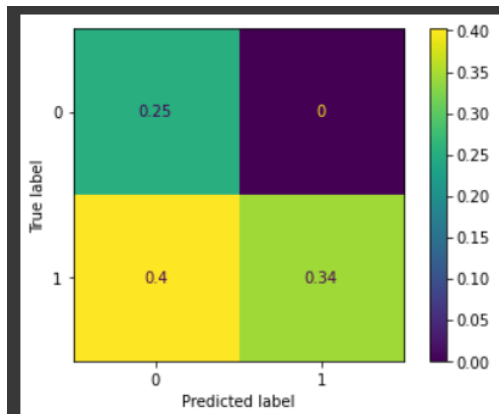
Acuracidade (teste): 0.5864

Revocação: 0.4457

Precisão: 1.0000

F1_score: 0.6166

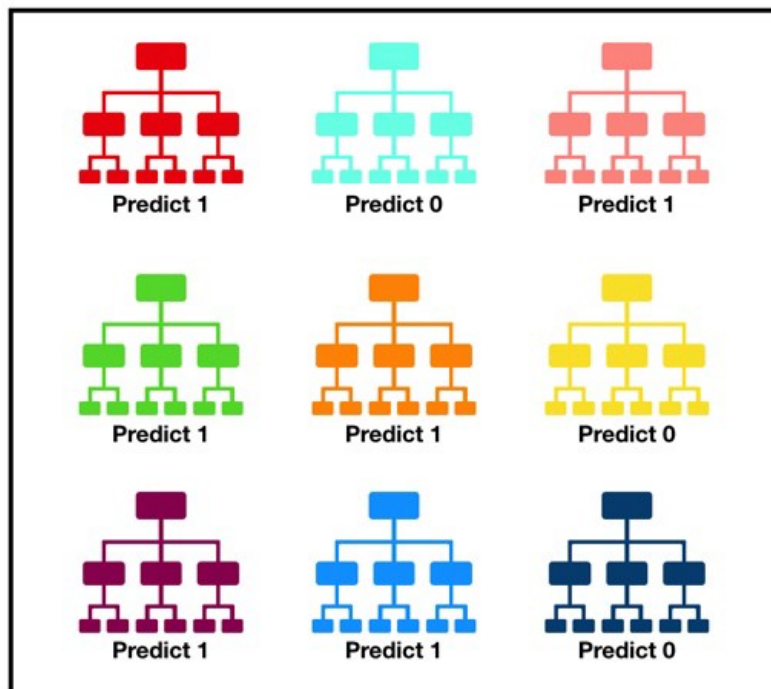
Matriz de confusão:



4.4.4 Random Forest

4.4.4.1. Explicação do modelo

O modelo de Random Forest foi importado da biblioteca sklearn.ensemble e funciona como um conjunto de árvores de decisões (decision trees) randômicas, criando uma gama de escolhas hierárquicas. Portanto, o output desse algoritmo vem da quantidade majoritária da classificação de todas as árvores, ou seja, o resultado da predição é fruto daquele resultado que mais aparece em árvores separadas.



Tally: Six 1s and Three 0s
Prediction: 1

4.4.4.2. Motivo de escolha do modelo

O modelo foi escolhido pela métrica da precisão e por ter a possibilidade de ser melhorado devido a quantidade de hiperparâmetros, além de que, se esse modelo se mostrar o melhor, é possível indicar a porcentagem de certeza que o modelo tem de que o entregador dará Churn ou não, a partir de quantas vezes o resultado em questão apareceu em cada árvore separadamente.

4.4.4.3. Resultados obtidos

Acuracidade(treino): 0.9047

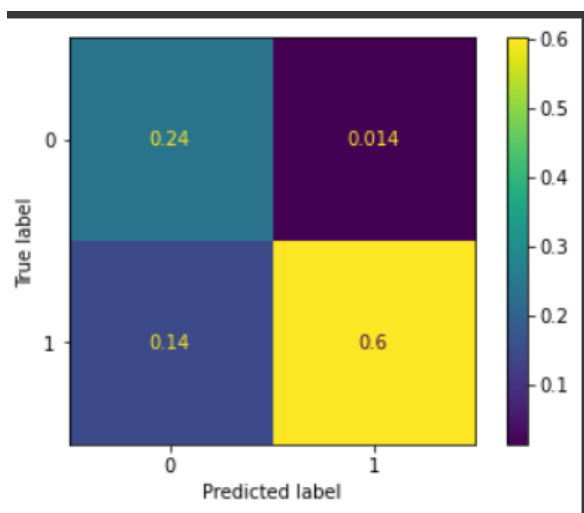
Acuracidade(teste): 0.8651

Revocação: 0.8426

Precisão: 0.9730

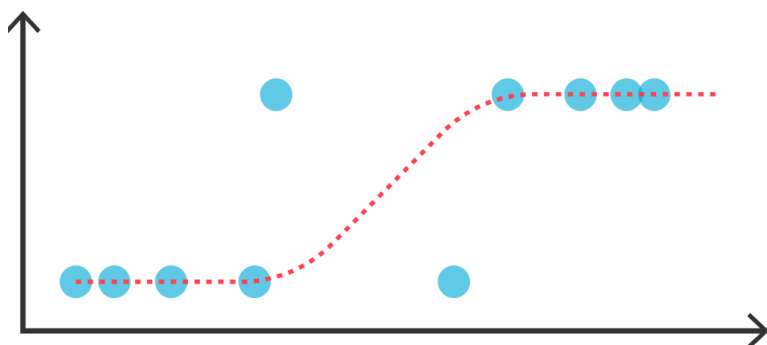
F1_score: 0.9031

Matriz de confusão:



4.4.5 Regressão Logística

4.4.5.1. Explicação do modelo



Em linhas gerais, os pontos colocados no espaço dimensional, como visto na imagem acima, são binários, isto é, representam 0 ou 1 e a partir da proximidade entre os pontos de valor 1 é traçada uma curva logarítmica que servirá de referência para a predição. Quanto mais próximo o valor a ser predito estiver dessa curva, maior é a probabilidade desse valor ser 1, ou seja, corresponder ao Churn em nosso modelo.

De forma aprofundada, o modelo de regressão logística foi importado da biblioteca `sklearn.linear_model` e modela a probabilidade de um evento de dois possíveis resultados ocorrer, fazendo com que as probabilidades logarítmicas para o evento sejam uma combinação linear de uma ou mais variáveis independentes ("preditores"). Na análise de regressão, a regressão logística (ou regressão logit) está estimando os parâmetros de um modelo logístico. Formalmente, na regressão logística binária existe uma única variável dependente binária, codificada por uma variável indicadora, onde os dois valores são rotulados como "0" e "1", enquanto as variáveis independentes podem ser cada uma uma variável binária (duas classes, codificadas por uma variável indicadora) ou uma variável contínua (qualquer valor real). A probabilidade correspondente do valor rotulado "1" pode variar entre 0 (certamente o valor "0") e 1 (certamente o valor "1"), daí a rotulagem; a função que converte log-odds em probabilidade é

a função logística, daí o nome. para a escala log-odds é chamado de logit , de logistic unit, sendo esses, nomes alternativos.

4.4.5.2. Motivo de escolha do modelo

O modelo de regressão logística foi escolhido porque ele é capaz de prever um fenômeno ou resultado (se o entregador dará churn ou não), por meio das características e comportamentos dos RT 's (experiência, tempo conectado no aplicativo, quilômetros rodados e etc.). Analisando essas informações, o modelo de regressão logística calcula a probabilidade do evento estudado ocorrer, ou seja a chance de nosso target em questão acontecer (nesse caso, o abandono dos entregadores). Esse processo se baseia nas informações descritas anteriormente, e, assim, analisa como as variáveis explanatórias (preditoras) se relacionam com a nossa variável dependente(target).

4.4.5.3. Resultados obtidos

Acuracidade(treino): 0.8446

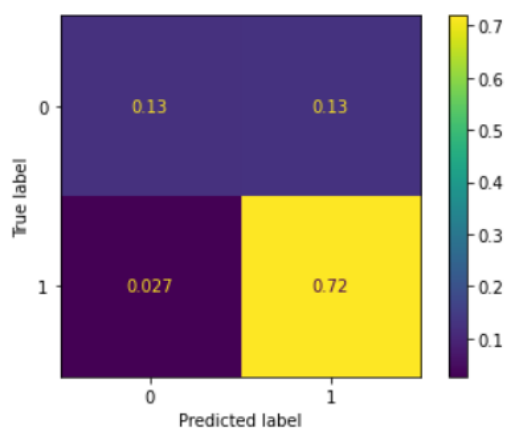
Acuracidade(teste): 0.8457

Revocação: 0.9643

Precisão: 0.8493

F1_score: 0.9032

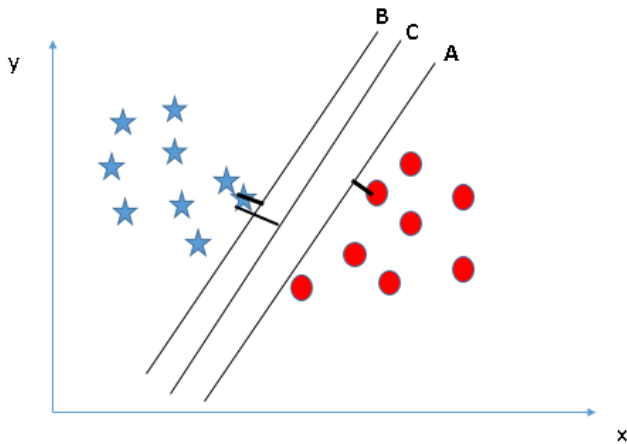
Matriz de confusão:



4.4.6 Support Vector Machine - SVM

4.4.6.1. Explicação do modelo

O modelo SVM foi importado da biblioteca `sklearn.svm` e tem a funcionalidade de traçar uma reta que separa os tipos de classificação, tentando maximizar a distância entre os dados de diferentes classificações e esta reta que separá-los.



4.4.6.2. Motivo de escolha do modelo

O algoritmo é excelente com uma quantidade grande de features e com dados não regulares. Além disso, o modelo possui uma vasta quantidade de parâmetros que podem ser utilizados para melhorar o desempenho do modelo, adaptando-se ao possível as features.

4.4.6.3. Resultados obtidos

Acuracidade (treino): 0.7452

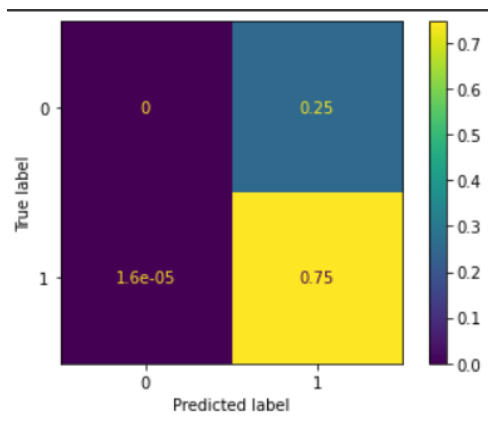
Acuracidade (teste): 0.7462

Revocação: 1.0000

Precisão: 0.7462

F1_score: 0.8546

Matriz de confusão:



4.4.7. AdaBoost

4.4.7.1. Explicação do modelo

Adapting Boosting é um modelo realizado em fases. Primeiramente, um modelo considerado “fraco” em questões de predição é utilizado e, com base nos resultados obtidos, ele aprende os padrões de erros progressivamente para realizar novos treinamentos em novos modelos até chegar em um resultado que o algoritmo acha satisfatório. A partir disso, é criada uma adaptação que, na maioria dos casos, não terá underfitting nos resultados classificadores e alcançará um resultado personalizado após várias melhorias.

4.4.7.2. Motivo de escolha do modelo

Por aprender com os próprios erros progressivamente, o Adaboost combina diversos modelos preditivos para retornar resultados e é uma boa opção para classificação por evidenciar uma combinação de modelos que dificilmente seria alcançada na tentativa e no erro.

4.4.7.3. Resultados obtidos

Acuracidade (treino): 0.9197

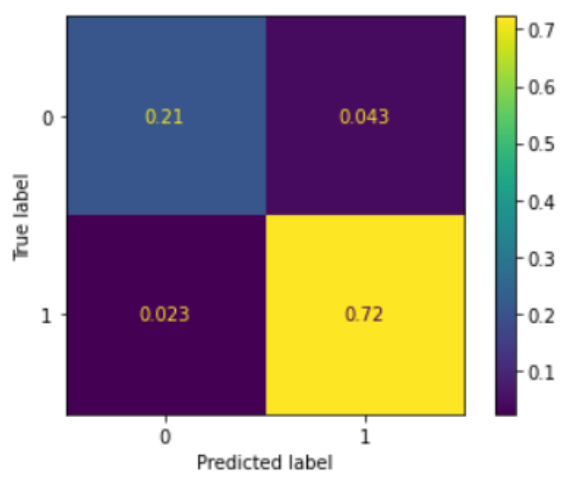
Acuracidade (teste): 0.9198

Revocação: 0.9227

Precisão: 0.9683

F1_score: 0.9449

Matriz de confusão:



4.4.8. Extra Trees

4.4.8.1. Explicação do modelo

Extra Trees é um modelo bastante semelhante ao Random Forest, com a diferença de adicionar um fator de aleatoriedade a mais na separação de dados. Após a seleção aleatória das variáveis candidatas para o nó inicial, os dados existentes em cada uma destas variáveis serão separados (split dos dados) também de maneira aleatória. Em seguida, os cálculos necessários para a otimização da árvore podem começar.

4.4.8.2. Motivo de escolha do modelo

O modelo foi utilizado para comparação de resultados com seus dois modelos semelhantes (Decision Tree e Random Forest), tomando conhecimento se o fator de aleatoriedade a mais do Extra Trees tem um diferencial grande nos resultados, porém houve um overfitting nos dados de treinamento do modelo.

4.4.8.3. Resultados obtidos

Acuracidade (treino): 1.0000

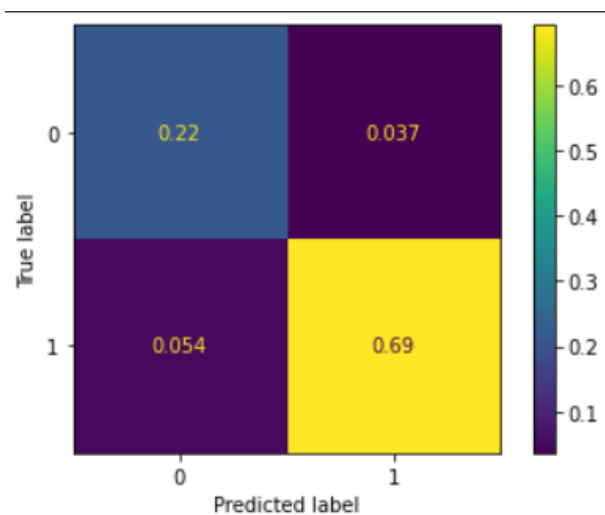
Acuracidade (teste): 0.9088

Revocação: 0.9270

Precisão: 0.9496

F1_score: 0.9382

Matriz de confusão:



Os seguintes modelos foram testados utilizando hiperparametrização, ou seja, foram testadas inserções de parâmetros diferentes dentro de cada modelo de classificação. Os hiperparâmetros servem para especificar como o modelo deve ser treinado, a fim de que refinamentos sejam feitos para que o modelo possa se tornar mais próximo da realidade. Em cada modelo, será especificado o motivo de escolha de suas respectivas features e hiperparâmetros, assim como os resultados obtidos.

[Link do .ipynb de modelagem com hiperparametrização \(Colab\)](#)

[Link do .ipynb de modelagem com hiperparametrização \(GitHub\)](#)

4.4.9. KNN com hiperparametrização

4.4.9.1. Motivo de escolha do modelo e conjunto de dados

O modelo KNN foi escolhido porque é um algoritmo muito utilizado para resolver problemas de classificação. Ou seja, podemos testar variáveis numéricas e, a partir da interpretação dos números, é retornada uma classe (valor nominal). Além disso, decidimos escolher esse modelo como uma solução compatível para resolver o problema da Rappi por outro motivo: o KNN analisa quais características dos vizinhos mais próximos são mais visíveis para relacionar com o target. Assim sendo, ele traça uma reta que mais se aproxima no conjunto de pontos e características desses vizinhos, que nesse caso são os próprios entregadores.

Os principais conjuntos de dados que foram escolhidos pelo nosso grupo para a construção do modelo KNN foram o 'CHURN' (target do problema, indica se o RT deixou de realizar entregas pelo aplicativo da Rappi), 'EARNINGS PER DISTANCE' (quanto que o entregador ganhou no ano por quilômetro rodado nas entregas), 'DISTANCE TO USER' (quanto o entregador percorreu no ano fazendo entregas) e 'Orders Cancel' (quantos pedidos cancelados foram sofridos pelo entregador, contando os cancelamentos feitos pela equipe de operações da Rappi, os pedidos cancelados por ele e os cancelados pelo próprio cliente).

```
[ ] # definição de x e y para treinamento e testagem dos modelos
x = df_final[['EARNINGS_PER_DISTANCE', 'DISTANCE_TO_USER', 'ORDERS_CANCEL']]
y = df_final['CHURN']
```

4.4.9.2. Configurações de hiperparâmetros

O hiperparâmetro de refinamento julgado como principal foi a quantidade de valores de comparação que serão utilizados para definir se o entregador dará Churn, 'n_neighbors', uma vez que a mudança do valor padrão pode impactar significativamente o resultado final. Além disso, também foi julgado importante evidenciar a importância que cada um desses vizinhos terá para a classificação do novo valor em Churn ou não Churn, sendo que foram testado todos os pontos vizinhos com mesma importância e os pontos mais próximos do valor a ser classificado como mais relevantes, no parâmetro 'weights'.

Também é importante configurar o KNN com diferentes formas de encontrar os vizinhos próximos, com algoritmos pré-prontos, a fim de que seja testada mais de uma lógica além da automática, no parâmetro 'algorithm'. Como consequência dessa escolha anterior, alguns algoritmos como o 'ball_tree' e o 'kd_tree' podem ser mais especificados no que diz respeito ao tempo de ação e memória que deve ser gasta, parâmetro 'leaf-size'. Todas as escolhas de hiperparâmetro estão entrelaçadas entre si, gerando um efeito positivo na personalização e uma consequentemente melhora na acurácia e revocação.

```
# definir o espaço possível dos hiperparametros a serem testados no modelo
parametros = {'n_neighbors': [3, 5, 7, 9, 13, 17, 21, 29],
              'weights': ['uniform', 'distance'],
              'algorithm': ['auto', 'ball_tree', 'kd_tree', 'brute'],
              'leaf_size': [15, 30, 45, 60]}

# achando os hiperparâmetros e treinando o modelo
knn_grid_search = GridSearchCV(estimator = KNeighborsClassifier(), param_grid=parametros)
knn_grid_search.fit(x_train, y_train.squeeze())
print(knn_grid_search.best_score_)
print(knn_grid_search.best_params_)
```

hiperparâmetros escolhidos para o modelo KNN

4.4.9.2. Resultados obtidos

Antes de analisar os resultados obtidos com os hiperparâmetros, é necessário ver os resultados do modelo KNN sem a construção desses mesmos hiperparâmetros:

```
# modelo de classificação K-Nearest Neighbours
from sklearn.neighbors import KNeighborsClassifier

## instânciação do objeto e treinamento do modelo KNN
knn = KNeighborsClassifier(n_neighbors=3).fit(x_train, y_train.squeeze())
# predição do modelo
y_pred = knn.predict(x_test)
metricas(knn)

Acuracidade (treino): 0.9578113750899928
Acuracidade (teste): 0.9074246892381201
Revocação: 0.884833981453784
Precisão: 0.9264015032884435
F1_score: 0.9051407588739289
```

Partindo desse ponto, pode-se analisar os resultados obtidos do algoritmo de vizinhança com os hiperparâmetros inclusos:

```
## instânciação do objeto e treinamento do modelo KNN
knn = KNeighborsClassifier(n_neighbors=3, leaf_size=15, weights='distance').fit(x_train, y_train.squeeze())
# predição do modelo
y_pred = knn.predict(x_test)
metricas(knn)

Acuracidade (treino): 0.9998880089592832
Acuracidade (teste): 0.9337787898017843
Revocação: 0.8925366437331738
Precisão: 0.972539113428944
F1_score: 0.9308220246451412
```

Resultados com hiperparâmetros, considerando os 3 vizinhos mais próximos do algoritmo

```
## instânciação do objeto e treinamento do modelo KNN
knn = KNeighborsClassifier(n_neighbors=9, leaf_size=15, weights='distance').fit(x_train, y_train.squeeze())
# predição do modelo
y_pred = knn.predict(x_test)
metricas(knn)

Acuracidade (treino): 0.9998880089592832
Acuracidade (teste): 0.9500541266937922
Revocação: 0.9262638348788513
Precisão: 0.9723661485319517
F1_score: 0.9487552661815396
```

Resultados com hiperparâmetros, considerando os 9 vizinhos mais próximos do algoritmo

```
## instânciação do objeto e treinamento do modelo KNN
knn = KNeighborsClassifier(n_neighbors=9, leaf_size=15, weights='distance').fit(x_train, y_train.squeeze())
# predição do modelo
y_pred = knn.predict(x_test)
metricas(knn)

Acuracidade (treino): 0.9998880089592832
Acuracidade (teste): 0.9500541266937922
Revocação: 0.9262638348788513
Precisão: 0.9723661485319517
F1_score: 0.9487552661815396
```

Resultados com hiperparâmetros, considerando os 17 vizinhos mais próximos do algoritmo:

```
[ ] ## instanciación do objeto e treinamento do modelo KNN
knn = KNeighborsClassifier(n_neighbors=17, leaf_size=15, weights='distance').fit(x_train, y_train.squeeze())
# predição do modelo
y_pred = knn.predict(x_test)
metricas(knn)

Acuracidade (treino): 0.9998880089592832
Acuracidade (teste): 0.9610661092239352
Revocação: 0.9486239904277595
Precisão: 0.9727014799478567
F1_score: 0.9605118691553401
```

A partir desses resultados, vê-se que a diferença entre os vizinhos mais próximos impacta no resultado final, então, após os testes com diferentes vizinhos, conclui-se que as menores diferenças entre as acuracidades e a maior revocação foi encontrada com 17 vizinhos próximos, conservando-se a métrica de 15 folhas.

4.4.10. Extra Trees com hiperparametrização

4.4.10.1. Motivo de escolha do modelo e conjunto de dados

Escolhemos o modelo Extra Trees porque o algoritmo cria diversas relações com as features que são inseridas nela. Essas mesmas relações formam árvores. Essas árvores procuram entender quais dos dados inseridos e as features escolhidas para o treino e teste do modelo possuem mais relação com o target (Churn). Quando o algoritmo chega a conclusão de qual relação é a melhor, ele aponta a certeza desse resultado obtido (eficácia do modelo). Outro motivo foram os bons resultados obtidos sem a hiperparametrização, havendo espaço para refinação e possível melhora dos resultados.

O conjunto de dados que foram escolhidos para a construção do modelo Extra Trees foram 'DAILY HOURS' (quantidade de horas que o entregador manteve conectado no aplicativo), 'DAILY ORDERS' (quantidade de pedidos feitos diariamente), 'MONTHLY EARNINGS' (receita mensal do entregador), 'MONTHLY TIPS' (quantidade de gorjetas recebidas pelo entregador em um mês), 'DISTANCE PER ORDER' (quantos pedidos o entregador fez por quilômetro) e 'DAILY DISTANCE' (quantos quilômetros os entregadores percorreram por dia) e o Churn (target do modelo que mostra se o entregador deixou de realizar entregas pelo aplicativo da Rappi).

```
# definição de x e y para treinamento e testagem dos modelos
x = df_final[['DAILY_HOURS', 'DAILY_ORDERS', 'MONTHLY_EARNINGS', 'MONTHLY_TIPS', 'DISTANCE_PER_ORDER', 'DAILY_DISTANCE']]
y = df_final['CHURN']

# rebalanceamento oversampling dos dados x y de treino
from imblearn.over_sampling import RandomOverSampler
rus = RandomOverSampler(random_state=0)
x_resampled, y_resampled = rus.fit_resample(x, y)

print(y_resampled.value_counts())
```

features que foram escolhidas em nosso código.

4.4.10.2. Configurações de hiperparâmetros

A escolha dos hiperparâmetros explicitada abaixo é fruto de um estudo da documentação oficial da biblioteca sklearn, em que foi concluído que o 'criterion', métrica responsável por avaliar a qualidade da divisão de cada árvore - ou seja, a fórmula matemática que o modelo se

baseará -, pode entregar resultados mais personalizados se o modelo for testado com cada um de seus parâmetros, seja ele 'gini', seja 'entropy' ou 'log_loss'. Em adição, foi escolhido 'max_features' pois essa métrica é um controlador de overfitting, uma vez que o valor passado para esse hiperparâmetro indica quantas features podem ser levadas em consideração para ramificar cada nó de cada árvore do Extra Trees. Por padrão, é selecionado aleatoriamente um número de features, mas, manualmente, pode-se controlar a lógica que o modelo usará para realizar o modelo final. Somente esses dois hiperparâmetros foram escolhidos, pois o tempo de execução do código do GridSearch se torna exponencial com o aumento de parâmetros e esses dois foram os mais relevantes.

```
from sklearn.model_selection import GridSearchCV

parameters={'criterion':['gini','entropy','log_loss'],
            'max_features':['int,float','auto','sqrt','log2']}

modelET = GridSearchCV(estimator=ExtraTreesClassifier(), param_grid=parameters)

modelET.fit(x_train, y_train)
print(modelET.best_score_)
print(modelET.best_params_)
```

hiperparâmetros escolhidos para o modelo Extra Trees.

4.4.10.3. Resultados obtidos

Antes de analisar os resultados obtidos, precisamos analisar também os resultados obtidos antes da construção desses hiperparâmetros:

```
# modelo de classificação Extra Trees
from sklearn.ensemble import ExtraTreesClassifier

# instanciação do objeto e treinamento do modelo Extra Trees
et = ExtraTreesClassifier().fit(x_train, y_train)
# predição do modelo
y_pred = et.predict(x_test)
metricas(et)
```

```
Acuracidade (treino): 1.0
Acuracidade (teste): 0.9145469052158979
Revocação: 0.9003633996854152
Precisão: 0.9262100711396289
F1_score: 0.9131038655647081
```

E agora analisaremos os resultados obtidos com os hiperparâmetros:

```
# instanciação do objeto e treinamento do modelo Extra Trees
et = ExtraTreesClassifier(criterion='gini', max_features='auto').fit(x_train, y_train)
# predição do modelo
y_pred = et.predict(x_test)
metricas(et)
```

```
Acuracidade (treino): 1.0
Acuracidade (teste): 0.9138977916615955
Revocação: 0.8993328632640886
Precisão: 0.9258731887092722
F1_score: 0.9124100645196791
```

Assim sendo, podemos concluir que houve uma leve queda de resultado do modelo, o que torna essa hiperparametrização não relevante, uma vez que a diferença nos resultados de acuracidade indicam que o modelo está praticamente em overfitting, pois a diferença se aproxima de 10%. Então essas configurações de hiperparâmetros não foram suficientes para melhorar a proximidade do modelo Extra Trees com a realidade.

Os próximos modelos foram criados durante a sprint 5:

4.4.11. Catboost

4.4.11.1. Explicação do modelo

Catboost é um modelo bastante semelhante ao Random Forest, com a diferença de adicionar um fator mais avançado de desenvolvimento após a criação das árvores. Cada árvore criada possui um índice de erro significativamente menor que a sua antecessora, capaz de criar um modelo extremamente poderoso. O número de árvores é determinado por um parâmetro definido no início

4.4.11.2. Motivo de escolha do modelo

O modelo é uma versão consideravelmente mais complexa do que o Random Forest e Extra Trees e mostrou um resultado consideravelmente melhor. Esse foi um dos modelos mais considerados pelo grupo e quase se tornou o escolhido, apresentando resultados quase confiáveis.

4.4.11.3. Resultados obtidos

Acuracidade (treino): 0.9613

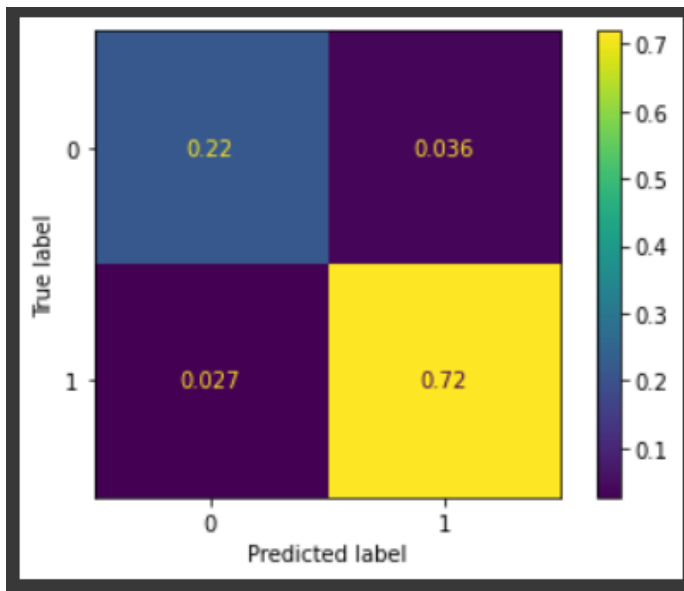
Acuracidade (teste): 0.9364

Revocação: 0.9635

Precisão: 0.9519

F1_score: 0.9576

Matriz de confusão:



4.4.12. XGBoost

4.4.12.1. Explicação do modelo

XGBoost é um que também utiliza o sistema de árvores de decisão. Ele utiliza do Gradient Boosting que minimiza o número de erros e amplifica seus resultados. Ele combina técnicas de otimização de software e hardware para produzir resultados superiores. Um modelo extremamente poderoso

4.4.12.2. Motivo de escolha do modelo

XGBoost é um modelo renomado e conhecido pelos seus resultados excelentes. ele é um algoritmo extremamente poderoso e possuía um histórico extremamente positivo e quando o grupo o testou (diversas vezes) ele apresentou um resultado satisfatório e posteriormente foi o modelo escolhido pelo grupo apresentado consistência e qualidade.

4.4.12.3. Resultados obtidos

Acuracidade (treino): 0.9220

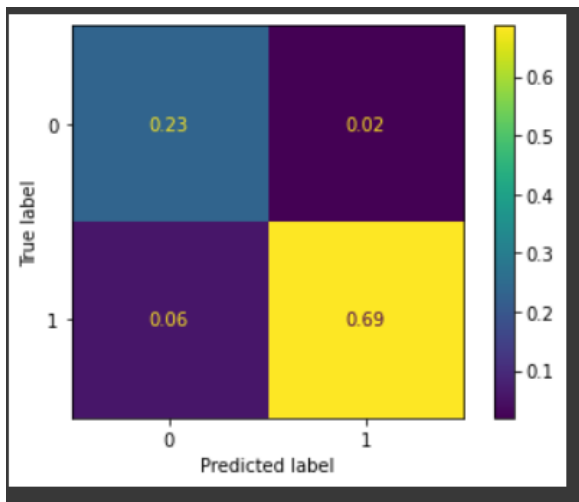
Acuracidade (teste): 0.9203

Revocação: 0.9198

Precisão: 0.9719

F1_score: 0.9451

Matriz de confusão:



4.5. Avaliação

Para melhor visualização dos resultados dos modelos, foi feita a tabela abaixo sobre métricas de avaliação, sendo resultados próximos de 100% os ideais (embora esse alvo possa indicar um overfitting, que indica que o modelo acertou todas as vezes que tentou prever aquela métrica específica, se houver uma diferença próxima a 10% do treino do teste):

Modelo	ACC (treino)	ACC (teste)	Revocação	Precisão	F1 Score
Modelos com rebalanceamento					
K Nearest Neighbor - KNN (Desbalanceado)	89,84%	85,14%	92,17%	88,41%	90,25%
Decision Tree (Desbalanceado)	88,96%	89,02%	95,15%	90,62%	92,83%
Naive Bayes (Oversampling + Undersampling)	80,46%	58,64%	44,57%	100%	61,66%
Random Forest (Oversampling)	90,47%	86,51%	84,26%	97,30%	90,31%
Regressão Logística (Desbalanceado)	84,46%	84,57%	96,43%	84,93%	90,32%
Support Vector Machines - SVM (Desbalanceado)	74,52%	74,62%	100%	74,62%	85,46%
AdaBoost (Undersampling)	91,97%	91,97%	92,27%	96,83%	94,49%
Extra Trees (Oversampling + Undersampling)	100%	90,88%	92,70%	94,96%	93,82%
Catboost (Oversampling)	96,13%	93,64%	96,35%	95,19%	95,76%
Bagging Classifier (Desbalanceado)	99,58%	92,89	95,31%	95,17%	95,24%
XGBoost (Desbalanceado)	93,71%	93,57%	97,22%	94,33%	95,76%

XGBoost (Oversampling)	93,64%	91,85%	92,24%	96,68%	94,41%
XGBoost (Oversampling + Undersampling)	97,93%	90,36%	89,19%	97,69%	93,25%
Modelo Escolhido					
XGBoost (Undersampling)	92,20%	92,03%	91,98%	97,19%	94,51%

As métricas de avaliação, como já citado anteriormente, são: acuracidade do treino, acuracidade do teste, revocação, precisão e F1 score, cada uma explicada com maior profundidade a seguir.

Primeiramente, cabe ressaltar as definições dadas anteriormente:

VP = entregadores que deram churn e o modelo previu corretamente

VN = entregadores que não deram churn e foram previstos corretamente

FP = entregadores que não deram churn mas o modelo previu como churn

FN = entregadores que deram churn e o modelo previu como não churn

A partir dessas informações, vê-se que a acurácia (ACC, do inglês *accuracy*) mostra quantas das classificações feitas pelo modelo foram de fato corretas, com sua fórmula sendo a soma de todos os resultados verdadeiros (verdadeiros positivos e verdadeiros negativos – VP e VN) sobre a soma de todas as classificações (verdadeiros positivos, verdadeiros negativos, falsos positivos e falsos negativos – VP, VN, FP e FN):

$$Acurácia = \frac{VP+VN}{VP+FN+VN+FP}$$

Como em cada modelo os dados foram separados entre teste e treino, temos os valores de acurácia de teste e de treino.

Revocação, também chamada de sensibilidade, é caracterizada pela razão da quantidade de classificações positivas verdadeiras (VP) e todos os dados que são, de fato, positivos (VP e FN). No presente projeto, é dada grande ênfase na análise da métrica de revocação, como já citado anteriormente, pois a Rappi especificou que é preferível acertar todos os entregadores que deram churn, embora haja uma taxa de erro um pouco maior nas classificações FP, do que deixar churns positivos preditos erroneamente.

$$Revocação = \frac{VP}{VP+FN}$$

Precisão, ao contrário de revocação, é definida pela razão entre os resultados VP e a soma dos resultados preditos como positivos, sendo eles corretos ou não. A métrica dá uma

ênfase maior para os erros por resultados FP e, como foi especificado pelo parceiro de projeto que é preferível que a ênfase seja nos resultados FN, precisão não foi um grande parâmetro para escolha de modelos.

$$Precisão = \frac{VP}{VP+FP}$$

F1 score, também chamado de F-measure, consiste na média harmônica entre as métricas de precisão e revocação. Como a métrica de precisão não é prioritária na avaliação dos modelos, consequentemente a métrica F1 score também fica em segundo plano:

$$F1 = 2 * \frac{precisão_{Teste} * revocação}{precisão_{Teste} + revocação}$$

Assim, podemos categorizar os resultados obtidos com base em todos esses indicadores, dando ênfase às acurácias do conjunto de dados de treino e de teste e revocação.

Desse modo, torna-se mister lembrar o objetivo da solução proposta, que é apontar, no mundo real, quais entregadores tendem a deixar de serem entregadores ativos na plataforma da Rappi. Então, como a Inteligência leva a uma tomada de decisão que impacta diretamente no setor financeiro da empresa, já que investimentos serão dados a partir de indicações específicas, o modelo deve ser o mais preciso possível. Por isso, tomando como exemplo uma taxa média de 75% de acerto obtida no conjunto dos testes, é indicado que em 25% das vezes o modelo deve errar a predição. Esse valor, obtido em testes de modelos iniciais, foi, a priori, visto como passível de ser melhorado, pois erros na solução não necessariamente são frutos de modelos improdutivos, mas, como citado anteriormente, podem ter ocorrido por escolhas de features ineficazes ou até mesmo um balanceamento que compromete a predição por descartar informações preciosas.

Em primeiro momento de escolha, durante a sprint 3 do projeto, referente à parte de modelagem na metodologia CRISP-DM, o modelo AdaBoost apresentou a maior média tanto na acurácia, quanto na revocação e demais parâmetros:

Modelo (rebalanceado)	ACC (treino)	ACC (teste)	Revocação	Precisão	F1 Score
AdaBoost	80,15%	79,81%	97,10%	81,00%	88,32%

É válido lembrar que, no momento de modelagem do AdaBoost acima, os dados estavam apenas rebalanceados utilizando a técnica de undersampling já explicada anteriormente, porém não foram testados hiperparâmetros para melhora do modelo.

Durante a sprint 4 do projeto, os modelos KNN e Extra Trees foram testados com hiperparametrização. A partir daí, o modelo KNN não apresentou overfitting ou underfitting, apresentando os melhores resultados até o momento:

Modelo	ACC (treino)	ACC (teste)	Revocação	Precisão	F1 Score
KNN (17 vizinhos)	99,98%	96,10%	94,86%	72,23%	96,05%

Porém, por conta das features utilizadas (tanto em quantidade quanto em qualidade), o modelo não pode ser utilizado como definitivo.

Durante a Sprint 5, ao refazer o Crisp-DM, o **modelo escolhido foi o XGBoost(Undersampling)**, muito baseado na premissa de que o modelo ideal é aquele que se adequa ao mundo real, e isso pode ser constatado com a menor diferença possível entre a acurácia de teste e de treino, então o Undersampling do XGBoost se adapta melhor ao pedido da Rappi e pode ser considerado o melhor modelo encontrado pelo grupo de desenvolvimento Rappitenderos.

Modelo Escolhido					
XGBoost (Undersampling)	92,20%	92,03%	91,98%	97,19%	94,51%

Cabe a observação de que esses resultados foram alcançados após decisões em cadeia, ou seja, se uma etapa for feita diferente os resultados também mudam. Com isso, é de extrema importância que, no momento da Rappi colocar os novos dados e o modelo em produção, coloque-se as mesmas tabelas nos lugares respectivos a elas, a fim de que o modelo não falhe. Mas, caso haja dúvidas, basta consultar o Manual do usuário, explicitado no fim deste documento na seção [7](#)

4.6 Comparação de Modelos

Utilizando a tabela criada na seção 4.5 e inserida abaixo, é visível que, dentre os diversos modelos criados pelo grupo, o modelo XGBoost(Undersampling) é o melhor modelo, portanto foi o escolhido.

A tabela abaixo também apresenta todo o processo de criação até chegarmos no modelo que julgamos ser o mais correto e preciso possível e facilita na compreensão das qualidades e fraquezas dos algoritmos que testamos com as features criadas pelo grupo.

Modelo	ACC (treino)	ACC (teste)	Revocação	Precisão	F1 Score
Modelos com rebalanceamento					
K Nearest Neighbor - KNN (Desbalanceado)	89,84%	85,14%	92,17%	88,41%	90,25%
Decision Tree (Desbalanceado)	88,96%	89,02%	95,15%	90,62%	92,83%
Naive Bayes (Oversampling + Undersampling)	80,46%	58,64%	44,57%	100%	61,66%
Random Forest (Oversampling)	90,47%	86,51%	84,26%	97,30%	90,31%
Regressão Logística (Desbalanceado)	84,46%	84,57%	96,43%	84,93%	90,32%
Support Vector Machines - SVM (Desbalanceado)	74,52%	74,62%	100%	74,62%	85,46%
AdaBoost (Undersampling)	91,97%	91,97%	92,27%	96,83%	94,49%
Extra Trees (Oversampling + Undersampling)	100%	90,88%	92,70%	94,96%	93,82%
CatBoost (Oversampling)	96,13%	93,64%	96,35%	95,19%	95,76%
Bagging Classifier (Desbalanceado)	99,58%	92,89	95,31%	95,17%	95,24%

XGBoost (Desbalanceado)	93,71%	93,57%	97,22%	94,33%	95,76%
XGBoost (Oversampling)	93,64%	91,85%	92,24%	96,68%	94,41%
XGBoost (Oversampling + Undersampling)	97,93%	90,36%	89,19%	97,69%	93,25%
Modelo Escolhido					
XGBoost (Undersampling)	92,20%	92,03%	91,98%	97,19%	94,51%

5. Conclusões e Recomendações

Para concluir a documentação do Null-Churn existem alguns pontos de atenção.

Em primeira análise, cabe ressaltar que o modelo escolhido foi o XGBoost e que as features envolvidas na seleção são 'TOTAL_RESPONSE_TIME', 'YEARLY_EARNINGS', 'DISTANCE_PER_ORDER', 'ORDERS_DONE_RATE', 'TOTAL_ORDERS', 'RT_PUNISHED_ORDERS', 'PERMANENT_BLOCK'.

Entrando em nosso raciocínio: segundo análises realizadas sobre os dados fornecidos, concluímos que a demora para a resolução de tickets criados pelos RTs foram um ponto crucial na decisão do churn, assim como a baixa remuneração por quilometragem que alguns recebem. Acreditamos que em caso da equipe Rappi chegar à mesma conclusão que nós, e esses pontos forem levados em consideração, a taxa de churn pode ser reduzida de forma significativa, já que grande parte dos entregadores que não possuem seus chamados atendidos no suporte tende a ficar insatisfeito a ponto de dar Churn, sendo esse o foco principal do conceito por trás do modelo, assim como aqueles com baixas remunerações.

Na sequência, gostaríamos de agradecer ao time Rappi pela parceria e predisposição para realizar nossos pedidos, sejam eles no app ou na sala de aula, sempre mostrando muito interesse nos projetos construídos pela turma, por meio de perguntas relevantes e reconhecimento dos alunos do Inteli como alunos, não funcionários!

Um ponto de atenção: a vontade de ser ouvido é presente na grande maioria dos entregadores! Então, mesmo com a predição realizada com a inteligência artificial fornecida pelo grupo Rappitenderos, não deixem de validar essas previsões e direcionar as forças na melhora da condição dos entregadores.

Agora um desejo que não é apenas do grupo Rappitenderos, mas sim de toda a turma: caso decidam seguir em frente com o uso de algum modelo, nos dêem algum tipo de atualização para sabermos da continuidade do nosso projeto!

MUITO OBRIGADO PELA PARCERIA! Esperamos vocês mais vezes no Inteli! 😊

6. Referências

Measurable AI, Insight.**2021 Brazil Food Delivery: iFood continues to lead with over 80% Market Share:** Brazilian Food Delivery. Edição.São Paulo: Nome da editora,07/10/2020, disponível em:

<https://medium.com/measurable-ai/2021-brazil-food-delivery-ifoood-continues-to-lead-with-over-80-market-share-9eaa8b3cb954>

Autor desconhecido.**Case de Sucesso: Conheça a História do iFood.** Edição. São Paulo:Salesforce.com,15/08/2020, disponível em:

<https://www.salesforce.com/br/customer-success-stories/ifoood/#:~:text=Desde%20o%20lan%C3%A7amento%20do%20Grupo,para%20clientes%20em%20diferentes%20localidades>

Nogare, Diego.**Performance de Machine Learning:Matriz de Confusão.**São Paulo:Diego Nogare 18/02/2020,disponível em:

<https://diegonogare.net/2020/04/performance-de-machine-learning-matriz-de-confusao/>

Kanumi.**Métricas de Avaliação em Machine Learning: Classificação.**São Paulo:Medium,10/06/2022, disponível em:

<https://medium.com/kunumi/m%C3%A9tricas-de-avalia%C3%A7%C3%A3o-em-machine-learning-classifica%C3%A7%C3%A3o-49340dcdb198>

Official, site Sklearn.**scikit-learn: machine learning in Python – scikit-learn 1.1.2 documentation.**Estados Unidos:Site oficial da biblioteca Sklearn,sem data de publicação, disponível em:

<https://scikit-learn.org/stable/>

Lucid.**Outdated browser | Lucid.**São Paulo:Lucid,data de publicação desconhecida, disponível em:

https://lucid.app/lucidchart/461347e2-da68-4e35-9bed-2548665629f0/edit?viewport_loc=-138%2C136%2C3328%2C1606%2C0_0&invitationId=inv_136bd04f-0ad1-44f1-9c25-9b919dcb8cd3#

Official, site Canvas.**Site do Canvas-Construção de Persona.**São Paulo:Site da construção de nossa Persona,20/08/2022, disponível em:

https://www.canva.com/design/DAFJfATtpbw/SpTRiFrzaZhMJV_lqFHVZA/edit?utm_content=DAFJfATtpbw&utm_campaign=designshare&utm_medium=link2&utm_source=sharebutton

Official, site Matplotlib.**Site da biblioteca Matplotlib.**Estados Unidos:Site Matplotlib,data desconhecida,disponível em:

[matplotlib.pyplot](https://matplotlib.org/pyplot)

Official, datageeks.**XGBoost | Algoritmo de Machine Learning que está dominando!**.São Paulo:Site data geeks,15/03/2021, disponível em:

: <https://www.datageeks.com.br/xgboost/>

7. Anexos

Colabs que representam tanto o Manual do Usuário, quanto um guia por todo trabalho desenvolvido nesse projeto:

[Metodologia](#)

[Manual do Usuário](#)