MODELO PREDITIVO DE CHURN DE ENTREGADORES

EMPRESA RAPPI

inteli
instituto
de tecnologia
e liderança

Controle do Documento

Histórico de revisões

Data	Autores	Versão	Resumo da atividade
08/08/2022	-Alysson Cordeiro; -Bruno Moitinho Leão; -Frederico Schur; -Israel Carvalho; -Luiz Carlos da Silva Júnior; -Stefano Tosi Butori.	1.0	Criação do documento
16/08/2022	Todos	1.1	Elaboração dos tópicos 4.1.1 a 4.1.5
17/08/2022	Todos	1.2	Elaboração do tópico 4.2
30/08/2022	Todos	2.1	Elaboração do tópico 4.3
01/09/2022	Todos	2.2	Elaboração do tópico 4.4
15/09/2022	Todos	3.1	Atualização do tópico 4.3
16/09/2022	Todos	3.2	Atualização do tópico 4.4
23/09/2022	Todos	4.1	Atualização dos tópicos 4.3 e 4.4
01/10/2022	Todos	5.1	Elaboração dos tópicos 1, 2 e 3
05/10/2022	Todos	5.2	Elaboração dos tópicos 5 e 6
06/10/2022	Todos	5.3	Revisão e conclusão do documento
17/03/2023	Escritório de Projetos	5.4	Ocultação de dados sensíveis.

Sumário

1. Introdução

Rappi é uma startup estabelecida na América Latina, com sede em Bogotá, Colômbia. O seu modelo de negócios é baseado em entregas sob demanda, via aplicativo. Fundada em 2015, a Rappi é atualmente uma das maiores empresas do setor de delivery em nível mundial, tendo uma participação significativa nesse mercado.

Segundo a companhia, mais entregadores ("RT"s) estão saindo da plataforma por mês do que seria aceitável. Nosso objetivo é desenvolver um modelo preditivo que possa classificar os entregadores mais propensos a sair da plataforma, para que a empresa possa entrar em contato com eles e adotar as medidas preventivas adequadas.

2. Objetivos e Justificativa

2.1. Objetivos

Objetivos gerais:

- Garantir a satisfação dos entregadores e dos clientes
- Melhorar a experiência dos usuários na plataforma
- Otimizar a operação diária da Rappi

Objetivos específicos:

- Diminuir a saída ("churn") de entregadores da plataforma
- Identificar os entregadores mais propensos a sair

2.2. Justificativa

Nossa proposta consiste em um modelo classificatório com o uso de machine learning, que será detalhado nos próximos tópicos.

Esse modelo poderá fornecer uma solução prática para atender os objetivos gerais e específicos expostos acima. Será uma solução inovadora, pois a empresa não tentou desenvolver internamente um sistema semelhante e que pode trazer "insights" valiosos.

3. Metodologia

3.1. CRISP-DM

A metodologia CRISP-DM (Cross Industry Standard Process for Data Mining) é atualmente o modelo de processo mais utilizado para trabalhar com mineração de dados. Ele envolve as seguintes etapas:

1. Entendimento do Negócio: Entender o negócio do cliente e quais os problemas enfrentados;
2. Entendimento dos Dados: Entender os dados existentes – como se relacionam, o que são, o que representam;
3. Preparação dos Dados: normalizar os dados, excluir "outliers", transformar features categóricas em features numéricas ("labeling");
4. Modelagem: aplicar um modelo de machine learning para os dados preparados;
5. Avaliação: verificar as métricas obtidas com o uso desse modelo;
6. Implantação ("Deployment"): apresentar os resultados obtidos de uma maneira otimizada.

O processo do CRISP-DM é cíclico e envolve retomar as etapas 1 a 5 continuamente, até que seja possível apresentar a conclusão final na etapa 6.

3.2. Ferramentas

Usamos neste projeto o Google Colaboratory ("Colab"), uma ferramenta do Google que permite desenvolver projetos de Ciência de Dados de forma colaborativa e interativa. O Colab possibilita a criação de projetos na nuvem em forma de caderno ("notebooks"), com código dividido em blocos que podem ser executados sequencialmente, em instâncias virtuais gratuitas. Esses notebooks já são criados com as principais bibliotecas usadas em projetos de Ciência de Dados, como pandas, sk-learn, matplotlib e numpy.

3.3. Principais técnicas empregadas

Para desenvolver o modelo, testamos diferentes algoritmos de machine learning, explicados detalhadamente nos tópicos 4.4. e 4.5. Para ajudar no teste, usamos a biblioteca Pycaret, para otimizar o processo de descoberta dos melhores algoritmos, uma técnica conhecida como AutoML.

4. Desenvolvimento e Resultados

4.1. Compreensão do Problema

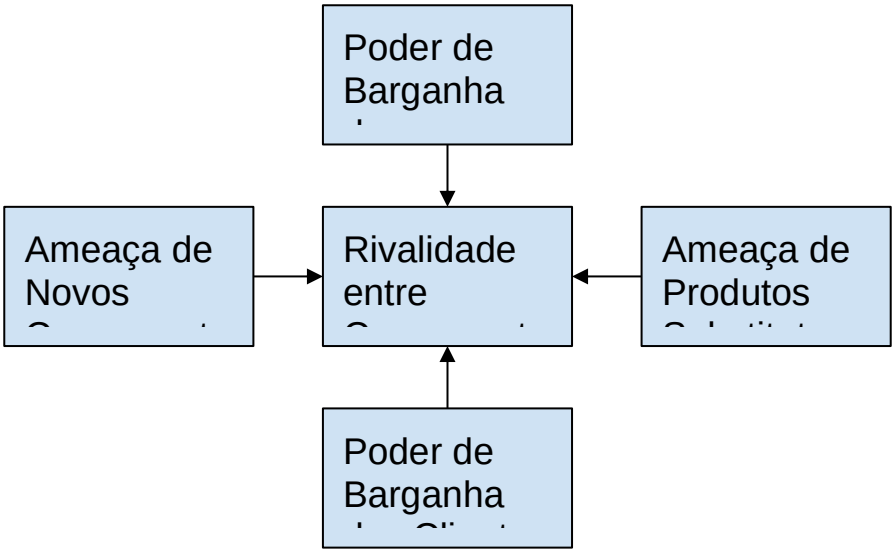
4.1.1. Contexto da indústria

O modelo de negócio da Rappi é do tipo plataforma multilateral, pois conecta estabelecimentos comerciais a entregadores e a clientes que precisam de serviços de entrega (delivery). A plataforma disponibiliza cardápios e menus das mais variadas instalações - restaurantes, lojas de todos os ramos, supermercados, entre outros.

Além do mais, a concorrência da companhia é bastante abrangente, porque, de certa forma, inclui desde toda e qualquer empresa de delivery (especialmente os aplicativos de entregas, como, por exemplo, IFood, Loggi, Glovo e Lalamove) e até os e-commerces gigantes, como a Amazon; esse de uma forma indireta.

Vale ressaltar, ademais, que o delivery é uma das principais tendências do mercado após o isolamento social da pandemia do COVID-19 em 2020. E ele tende apenas se fortalecer. No entanto, as entregas estão deixando de ser focadas apenas em comida para alcançar novos patamares, como medicamentos e outros produtos diversos, por exemplo. Além disso, esse tipo de mercado está optando por uma prática mais sustentável, como, por exemplo, ajudar o consumidor a identificar restaurantes que usam menos plásticos em seus produtos e tornar grande parte dos pedidos neutros em carbono (CO_2) no país.

Utilizaremos a análise das 5 Forças de Porter para entender melhor a indústria de delivery por aplicativo.



Ameaça de Novos Concorrentes:

Durante a pandemia, 89% dos estabelecimentos passaram a utilizar o delivery como uma solução para disponibilizar seus produtos (grande crescimento do mercado). Entretanto, o alto custo operacional gerado pela operação no mercado de delivery acaba dificultando a instalação de novas empresas, além da manutenção de empresas já estabelecidas, como por exemplo o recente caso do Uber Eats (alto custo operacional).

Poder de Barganha dos Clientes:

Por mais que houve um aumento expressivo no setor de delivery impulsionado principalmente pela necessidade das empresas de se adequarem às restrições impostas ao SARS-CoV-2 (coronavírus)e, também, considerando o cenário macroeconômico dos últimos meses, é possível notar um aumento da inflação e, consequentemente, uma diminuição no grau de consumo em diversas áreas da economia, incluindo a de delivery. Logo, é razoável inferir que o poder de barganha dos consumidores nesse cenário é relativamente baixo, visto que diante de um cenário em que há uma alta de preços em toda a economia, os consumidores não conseguirão "barganhar" por preços menores.

Ameaça de Produtos Substitutos:

Não há muitos produtos substitutos para o delivery por aplicativo. A opção alternativa mais evidente seria a entrega efetivada diretamente em contato com o restaurante, eliminando a intermediação via aplicativo. Essa opção não oferece a comodidade de ver o cardápio de opções pelo aplicativo e apresenta o incômodo de ter que pesquisar um restaurante e conversar com um atendente por telefone.

Poder de Barganha dos Fornecedores:

Tendo em vista todo o cenário descrito nos tópicos anteriores, é possível concluir que o poder de barganha dos fornecedores é expressivo, uma vez que eles controlam em qual marketplace estarão disponíveis. Por outro lado, contratos de exclusividade podem restringir a capacidade de escolha dos fornecedores e reduzir o poder de barganha destes.

Rivalidade entre Concorrentes:

A rivalidade entre os concorrentes nesse setor é extremamente alta em consequência do constante esforço para garantirem uma operação de qualidade, além de um grande market share. É comum que empresas desse setor gastem seus recursos de caixa para crescer, e, consequentemente, acabam não dando lucro.

4.1.2. Análise SWOT

Strengths (Forças)	Weakness (Fraquezas)
- Entrega turbo em 10 minutos - Alguns parceiros exclusivos - Parceria com players de outros setores (ex: HBOMax) - Presença substancial na América Latina - Atuação em outros setores, como o financeiro (RappiBank)	- Dependência de entregadores que não são CLT - A função de rastreio da entrega não é consistente - "Churn" elevado de entregadores
Opportunities (Oportunidades)	**Threats (Ameaças)**
- Saída recente de um player relevante, abrindo assim uma nova fatia do mercado - Aumento da demanda por delivery de diversos produtos	- Mercado bastante competitivo - Instabilidade de fatores externos que afetam os entregadores, como preço da gasolina, variações climáticas ou feriados

4.1.3. Planejamento Geral da Solução

Dados disponíveis: A Rappi disponibilizou até o momento uma série de arquivos CSV, com o conteúdo detalhado de forma mais extensiva no tópico 4.2.1 deste documento. Os dados desses arquivos, conforme informações repassadas pelo cliente em entrevista, foram obtidos por meio do aplicativo de delivery.

Esses arquivos incluem informações diversas sobre os entregadores, como a taxa de aceitação dos pedidos, receita mensal, reclamações sobre pedidos, data de criação das contas que sofreram "churn", registro de suspensões e avisos, número de ordens, tempo que o entregador ficou "online", ordens devolvidas e dados gerais.

Solução proposta: Pretendemos desenvolver um modelo preditivo para classificar se um entregador irá permanecer ou sair ("churn") da plataforma Rappi. Como efeito secundário, podemos verificar a probabilidade que o modelo entende que aquele resultado se concretizará e os fatores ("features") que o modelo considera para a classificação geral.

Tipo de tarefa (regressão ou classificação): a predição de "churn" é essencialmente uma tarefa de *classificação binária* - o objetivo é informar se os entregadores irão: (i) permanecer; ou (ii) sair ("churn") da plataforma dentro de um período determinado.

Forma de utilização da solução proposta: a Rappi poderá usar o modelo para classificar se um entregador está propenso a sair da plataforma e, se desejar, dedicar mais esforços para manter esse entregador na plataforma.

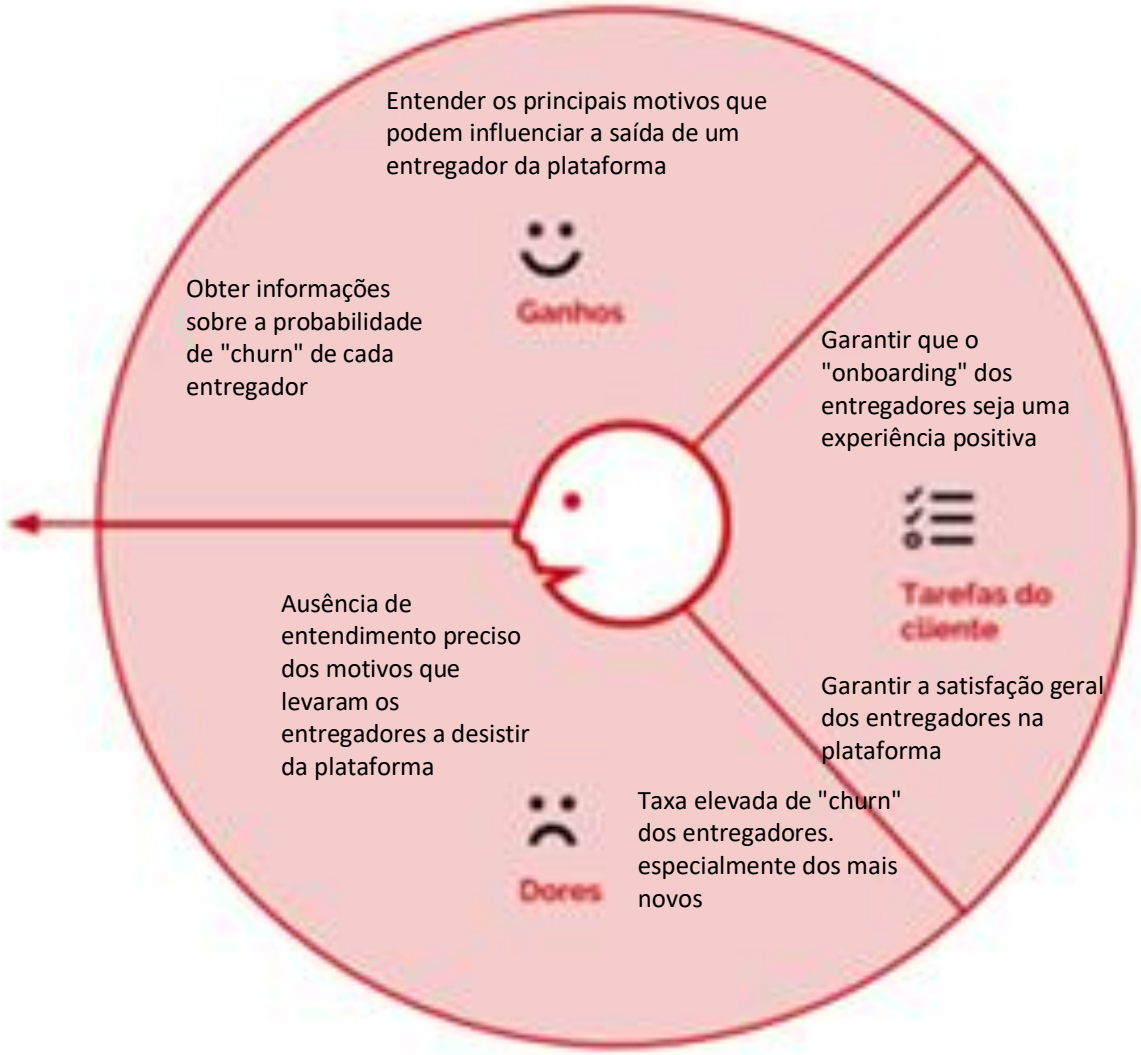
Benefícios trazidos pela solução proposta: o modelo pode permitir uma visualização rápida dos entregadores mais inclinados a parar de usar a plataforma em breve, levar a um maior entendimento dos fatores que podem causar esse evento, facilitar o diálogo e reduzir a taxa de "churn" para um patamar adequado.

Critério de sucesso e métricas de avaliação: entendemos que o modelo deve apresentar certa probabilidade de acerto em suas predições. Para tanto, utilizamos como métricas de avaliação:

A. acurácia ("accuracy"), a razão entre o número de predição corretas e o número total de predições, fornece um indicador de confiabilidade geral do modelo:
$$\frac{Positivos\ Verdadeiros + Negativos\ Verdadeiros}{Positivos\ Verdadeiros + Negativos\ Verdadeiros + Positivos\ Falsos + Negativos\ Falsos};$$
B. precisão ("precision"), a confiabilidade do modelo quando ele aponta que um resultado é positivo:
$$\frac{Positivos\ Verdadeiros}{Positivos\ Verdadeiros + Positivos\ Falsos};$$
C. revocação ("recall"), a confiabilidade do modelo em detectar *todos* os resultados positivos corretamente: $\frac{Positivos\ Verdadeiros}{Positivos\ Verdadeiros + Negativos\ Falsos}$;

4.1.4. Value Proposition Canvas

Perfil do Cliente



4.1.5. Matriz de Riscos

		Ameaças				
Probabilidade	90%			Bugs no código do modelo preditivo		
	70%					
	50%			Imprecisão / inconsistência dos dados usados para análise		
	30%			Não ter dados relevantes para solucionar o problema	Modelo ter uma precisão ou acurácia baixa	
	10%					Sistema exigir muita capacidade computacional
		Muito baixo	Baixo	Moderado	Alto	Muito alto
		Impacto				

		Oportunidades				
Probabilidade	90%					
	70%	Concorrência não possuir ferramenta que mapeie o churn	Sistema prevê corretamente a chance de saída dos RTs	Bom volume de dados para treinar o modelo		
	50%					
	30%					
	10%					
		Muito alto	Alto	Moderado	Baixo	Muito baixo
		Impacto				

4.1.6. Personas

José González

AGE	29
EDUCATION	Gestão da Tecnologia da Informação
STATUS	Solteiro
OCCUPATION	Diretor de Operações
LOCATION	São Paulo
TECH LITERTE	Alta

> Estou acostumado com atendimento ao cliente, principalmente entender o comportamento do cliente em relação ao produto.

Personalidade

Comunicativo Simpático

Biografia

Mora em São Paulo, na capital. Graduou-se em Sistemas da Informação, terminou sua pós em Gestão de TI em 2020.Se especializou em customer experience. Devido a sua especialização, foi promovido recentemente em uma empresa de Delivery . É um homem comunicativo, empreendedor e está sempre antenado para as novas tendências e oportunidades de mercado. González atualmente está solteiro e gosta de viajar sempre que pode.

Necessidades

- Entender o motivo do elevado nível da saída de entregadores de aplicativo da empresa.
- Pretende utilizar um modelo preditivo para classificar se um entregador permancerá ou sairá da companhia.

Frustração

- Baixo contato com entregadores de delivery.
- Possui dificuldade em analisar a probabilidade de “churn” dos entregadores.
- Não tem entendimento exato dos motivos que levaram os entregadores a desistir da plataforma.

Hobbies

Viajar; leitura; aprender novos idiomas; esportes.

Motivações

Liderança; conhecimento; Ter flexibilidade na carreira; Impactar os clientes com os resultados de seus projetos.

Cleyton Soares

AGE	21
EDUCATION	Ensino Médio completo
STATUS	Solteiro
OCCUPATION	Entregador de delivery
LOCATION	São Paulo
TECH LITERTE	Média

> Há 2 anos estou na empresa de de delivery como entregador. E com isso desenvolvi uma boa comunicação e relação com os clientes.

Personalidade

Extrovertido Responsável

Biografia

Cleyton mora na Zona Leste de São Paulo. Terminou seu Ensino Médio em 2019. Entrou para Rappi para trabalhar como entregador após ser demitido durante a pandemia. É um jovem bem extrovertido e comunicativo. Gosta de bater-papo com os amigos.

Necessidades

- Maximizar os ganhos na plataforma
- Ter mais pedidos de entregas concluídos
- Contratação de seguros de vida da empresa de delivery

Frustração

- Insastisfação quando o cliente cancela o pedido
- Aplicativo da empresa mostra a rota do GPS diferente do local desejado
- Sua remuneração é desproporcional ao valor da inflação da gasolina

Hobbies

Esportes; Assistir séries; jogar games eletrônicos; ir à praia.

Motivações

Iniciativa; determinação e pontualidade em entregar os produtos pedidos pelos clientes.

4.1.7. Jornada do Usuário

Jose González

Cenário

Jose é um diretor de operações da Rappi e percebe que hoje a empresa tem uma grande evasão de entregadores ("RTs"). Assim, deseja criar uma ferramenta de IA que consiga prever as chances de um RT sair da plataforma

Metas

- Prever e antecipar a saída dos RTs
- Entender os motivos da saída dos RTs
- Criar ações e medidas para a retenção dos RTs

Extrair	Popular	Carregar	Resultado
• Levantar dados e informações necessárias dos RTs na base de dados da Rappi	• Popular o notebook de processamento de features preparado pelo grupo	• Carregar os dados no modelo, usando o notebook de modelagem preparado pelo grupo	• Obter uma predição da possibilidade de saída dos RTs • Compreensão dos motivos de saída dos RTs • Ações para a permanência dos RTs

"Que dados devo obter para alimentar o sistema adequadamente?" (1)

"Vou inserir os dados em um notebook para processamento" (2)

"Agora vou carregar os dados no modelo" (3)

"Que interessante, esses entregadores tem maior probabilidade de sair da plataforma" (4)

Oportunidades

- Auxiliar na tomada de decisao dos gerentes
- Ajudar os RTs a terem seus problemas resolvidos

Métricas

- Reduzir o "churn" de RTs para um nível aceitável
- Aumentar o Net Promoter Score ("NPS") dos RTs

4.2. Compreensão dos Dados

4.2.1. Descrição Geral dos Dados

Os dados disponibilizados pelo cliente consistem em diversos arquivos "Comma-separated values" ("CSV"), elencados na tabela abaixo. Segundo informações repassadas pela Rappi, todos os dados foram obtidos através do aplicativo. Também descrevemos na tabela o que cada arquivo representa, em geral, e o seu tamanho, com número de linhas e de colunas.

ABAIXO, SERIA POSSÍVEL VISUALIZAR A TABELA DE ARQUIVO CONTENDO TODOS OS DADOS FORNECIDOS PELO PARCEIRO PARA A CONSTRUÇÃO DO MODELO PREDITIVO. NO ENTANTO, EM RESPOSTA AO PEDIDO DO PARCEIRO PARA PROTEGER DADOS SENSÍVEIS, A TABELA FOI OCULTADA.

Forma de agregação e mesclagem de dados:

Todos os arquivos CSV tem uma coluna com um ID único do entregador. Desse modo, podemos usar esse identificador único para juntar as diferentes tabelas (fazer um "join") e assim cruzar as informações de um CSV com o outro.

Riscos e contingências relacionados aos dados (qualidade, cobertura/diversidade e acesso):

Em relação a qualidade, temos o risco que os arquivos repassados possam ter dados que não são verossímeis por uma série de motivos (inserções errôneas, importações indevidas, "bugs" no aplicativo ou base de dados). Adicionalmente, podemos verificar um nível elevado de ruído em alguns arquivos CSV, como informações ausentes (campos nulos), muitos outliers inconsistentes (por exemplo, muitos entregadores com receitas além do esperado no arquivo "earnings") e valores repetidos (especialmente no arquivo "criacao contas churn").

Sobre cobertura e diversidade, temos alguns arquivos com muitas informações, como o "criacao contas churn" com mais de 32 milhões de registros. Isso pode levar a um excesso de informações para tratamento e análise, caso não seja feito um cruzamento e filtragem prévios.

Finalmente, no que concerne ao acesso, a Rappi não pode liberar alguns dados, como as regras automáticas usadas para aplicar suspensões e avisos na plataforma (por exemplo, quanto tempo um entregador pode ficar parado até receber um aviso). Assim, isso pode dificultar algumas análises.

Seleção do subconjunto para análises iniciais:

Temos algumas hipóteses iniciais sobre as causas do "churn", que irão embasar nossas análises iniciais. Identificamos essas hipóteses a seguir, junto com o conjunto de dados relevante:

- *Hipótese 1: Os entregadores ("RT"s) deixam a plataforma por muitas suspensões recorrentes.*
  - Subconjunto relevante: arquivo "Incidentes_Regras RT", com as suspensões. Verificar se existe uma relação entre número e duração de suspensões e "churn".

- *Hipótese 2: Os entregadores deixam a plataforma por devoluções excessivas de pedidos, que levam a dívidas no valor dos produtos e custos com devoluções (gasolina, tempo de deslocamento).*
  - Subconjunto relevante: arquivo "comp defects", com as devoluções pendentes. Verificar se existe uma relação entre devoluções e "churn".

- *Hipótese 3: Os entregadores deixam a plataforma por baixa remuneração ou poucos pedidos.*
  - Subconjunto relevante: arquivos "earnings", "Ordens done e cancel" e, especialmente, "Infos gerais". Verificar se existe uma relação entre rendimentos, número de ordens realizadas e "churn".

Restrições de segurança*:* foi mencionado que são dados sensíveis e que devem ser geridos com cuidado. Portanto, não vamos publicar essa base de dados, nem repassar essas informações para terceiros. O uso dos dados será exclusivamente para desenvolver um modelo preditivo para a possibilidade de "churn" de entregadores na plataforma Rappi.

4.2.2. Estatística Descritiva dos Dados

Link para o Google Colab com o código das análises:
https://colab.research.google.com/drive/1s7qv2MbyDceq2NBiZt31mBhXniZ3x8j5?usp=sharing

A partir dos dados fornecidos, podemos estabelecer algumas análises pautadas em estatística descritiva e elaborar gráficos para testar as hipóteses suscitadas inicialmente.

- *Hipótese A: Os entregadores deixam a plataforma por baixa remuneração ou poucos pedidos*

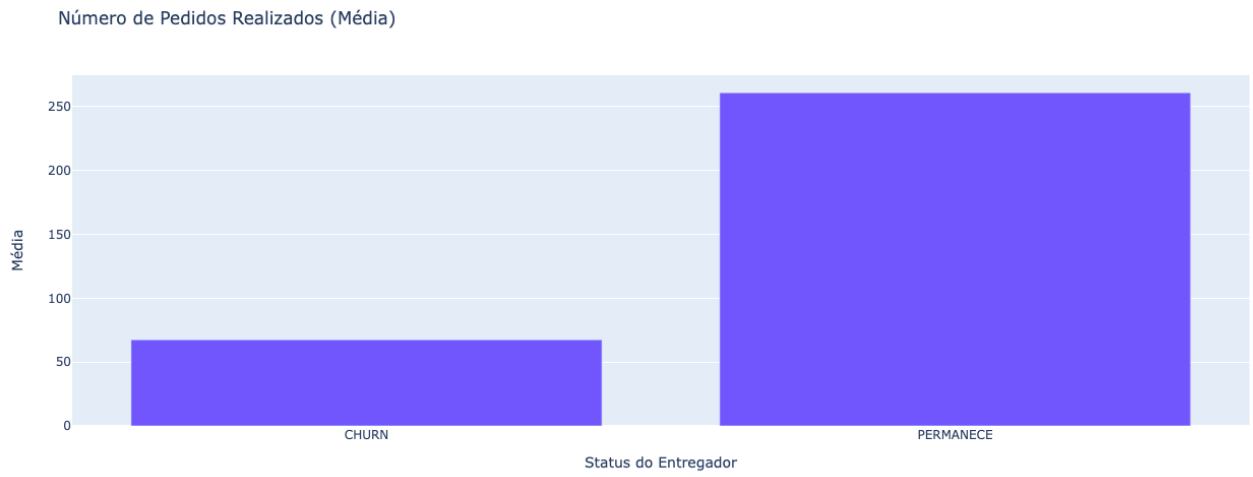
Para fins de comparação para essa hipótese, foi necessário trabalhar com dois grupos – os entregadores que saíram da plataforma e aqueles que permanecem na mesma. Para tanto, dividimos o arquivo "Infos Gerais" em dois grupos, os que entregaram o último pedido há mais de 21 dias ("CHURN")- critério informado em entrevista com a Rappi para classificação de Churn - e os que entregaram pedidos após essa data ("PERMANECE"):

```
df_infos = pd.read_csv('/content/drive/MyDrive/data/infos gerais.csv')
df_infos.sort_values('ULTIMO_PEDIDO', ascending=False)
# Churns são entregadores com o último pedido há mais de 21 dias
```

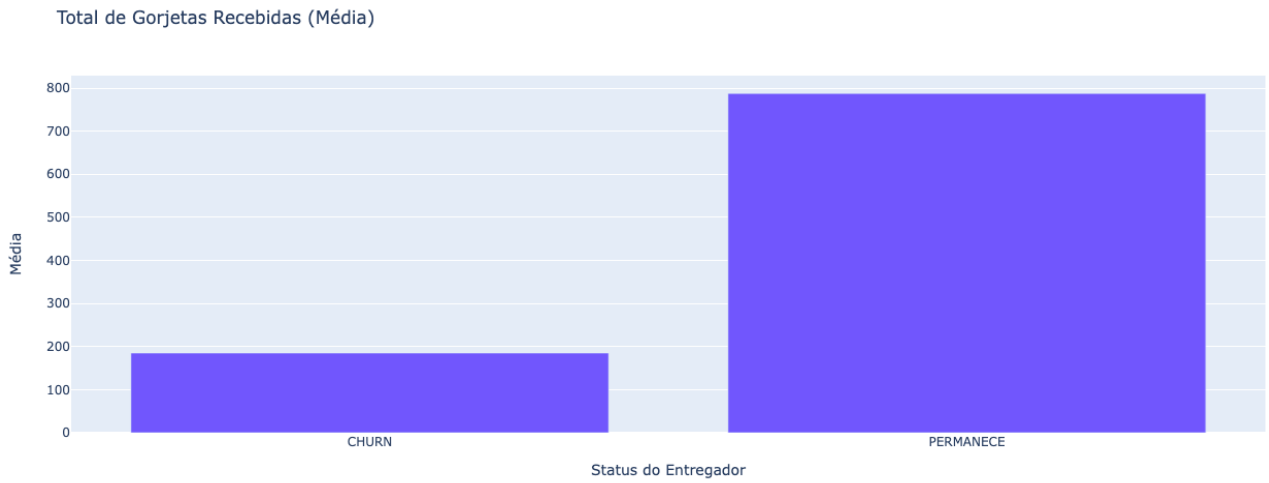
```
# Os dados vão até 01/08/2022, logo a data de corte é 11/07/2022
df_not_churned = df_infos[(df_infos['ULTIMO_PEDIDO'] > '2022-07-11')]
df_churned = df_infos[(df_infos['ULTIMO_PEDIDO'] < '2022-07-11')]
```

Realizando uma comparação entre os dois grupos, obtivemos resultados interessantes.

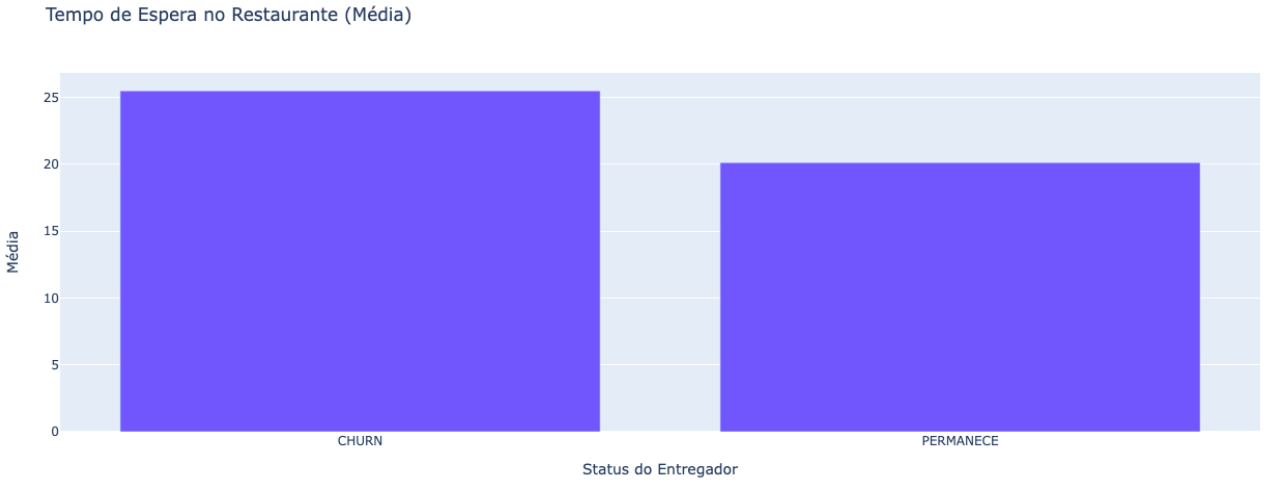
- Os entregadores que sofreram "churn" realizaram menos entregas do que aqueles que permaneceram na plataforma, sendo o número de pedidos cerca de 4 vezes maior. Isso pode ser um indicador de que quanto mais corridas o entregador recebe, menor sua propensão a sair da plataforma:



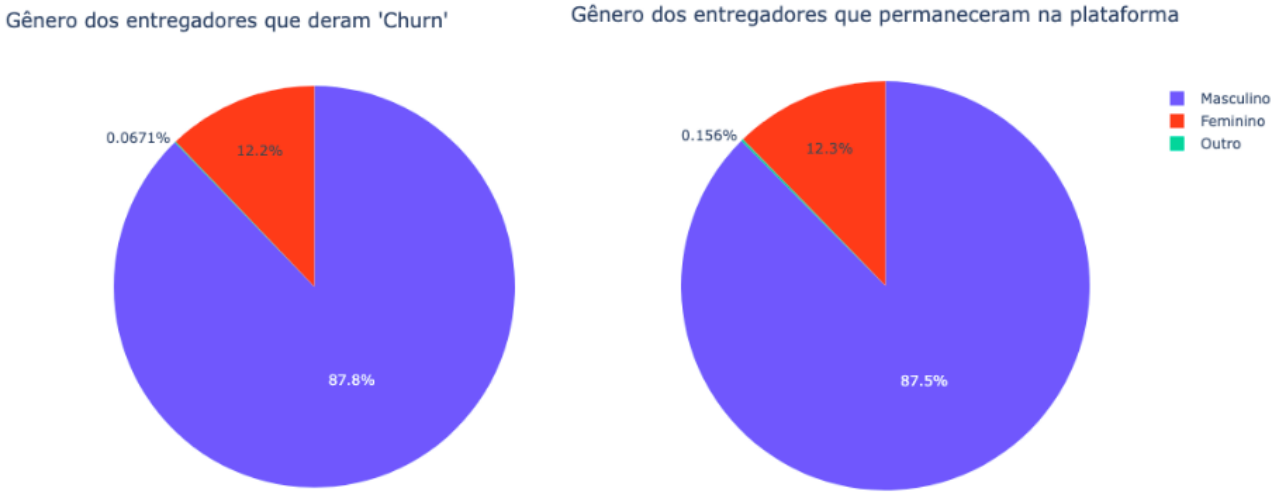
- Os entregadores que permaneceram na plataforma ganharam quase o quádruplo de gorjetas daqueles que deram "churn". Isso pode indicar que as gorjetas (ou ausência delas) são um fator de permanência na plataforma:



- Entregadores que permaneceram na plataforma em geral esperam menos tempo para o pedido ficar pronto no restaurante. Não conseguimos entender qual a causa desse fator:

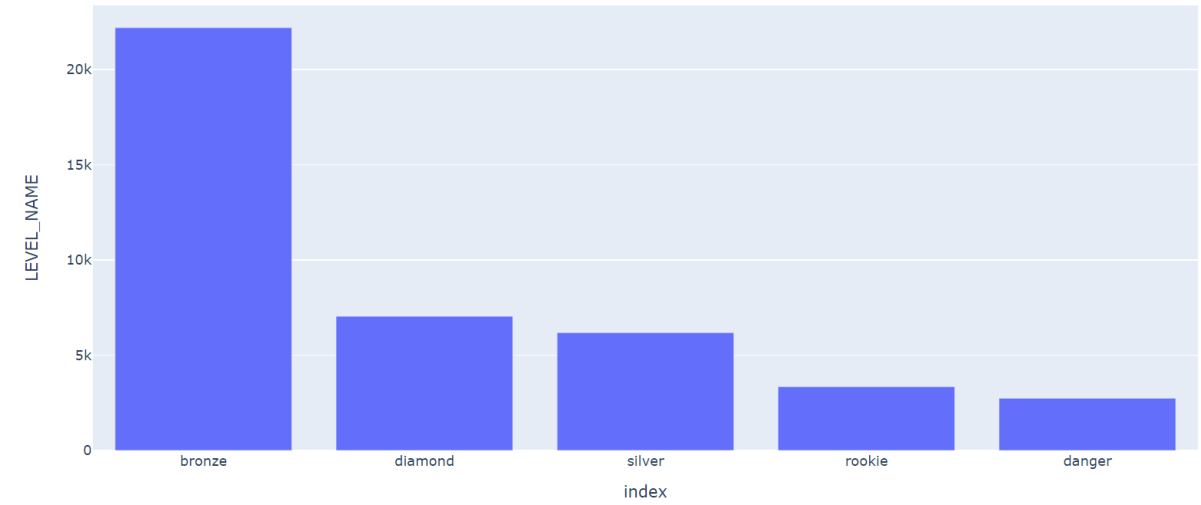


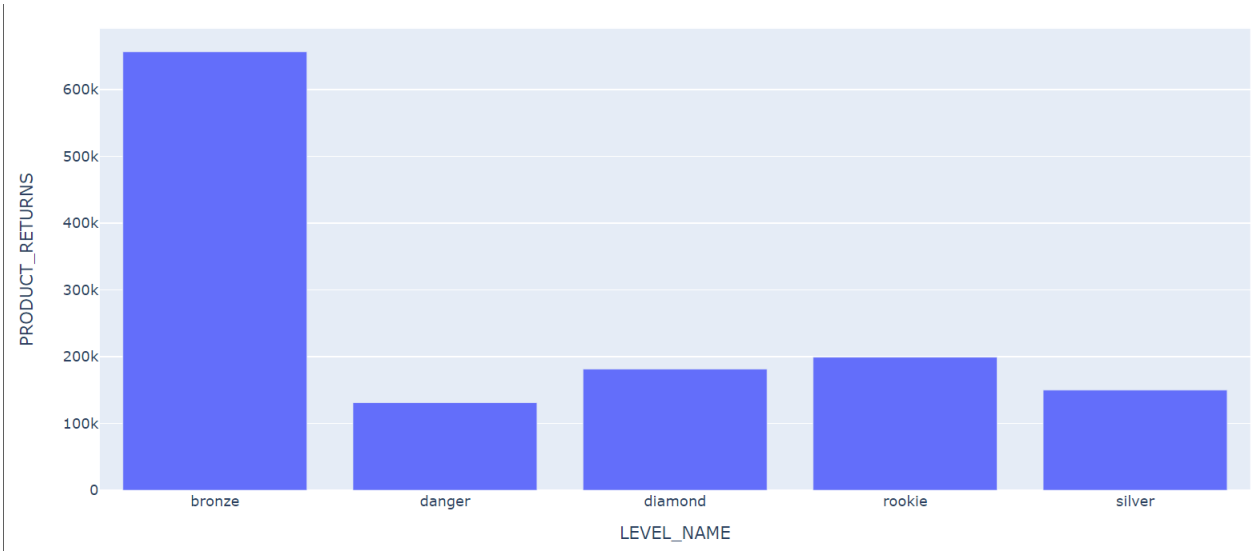
- Aspectos demográficos, como gênero do entregador, aparentam não impactar de forma expressiva o "churn". A composição dos dois grupos é praticamente idêntica:



- *Hipótese B: Os entregadores deixam a plataforma por devoluções excessivas de pedidos*

    - Para trabalhar com essa hipótese, verificamos se algum grupo está representado de forma desproporcional no valor de pedidos a devolver pendentes. Assim, comparamos a quantidade de entregadores em cada nível e a quantidade de devoluções pendentes:





Nesse contexto, verificamos que há uma quantidade expressiva de devolução no nível bronze (porém proporcional), e uma quantidade muito acima do esperado no nível "rookie" e "danger". O que mais se destaca nessa análise é que o nível "bronze" pode representar um grupo descontente com a plataforma, dado o alto volume de devoluções pendentes.

4.2.3. Descrição da Predição Desejada ("Target")

A predição desejada ("target") consiste em classificar se um entregador ("RT") irá sair da plataforma Rappi (evento de "churn") ou se irá permanecer na plataforma.

Diferentemente de um modelo de regressão contínua, em que se deseja encontrar um valor numérico específico por interpolação (por exemplo, o valor de uma casa em determinado mercado imobiliário), no modelo de classificação o mais importante é prever em qual categoria se deve encaixar uma situação (por exemplo, um paciente tem uma patologia ou não tem).

Nosso caso, portanto, consiste em um modelo de **classificação binária**, em que o "target" pode ser encaixado em uma de duas categorias: a) o entregador saiu; *ou* b) permaneceu na plataforma durante determinado período.

4.3. Preparação dos Dados

- Link para o notebook de "*feature engineering*":

https://colab.research.google.com/drive/1gcgucA_kH5xj9dncwh1sUfaLRROkbm3G?usp=sharing

Após cuidadosa análise dos dados disponibilizados pelo cliente, iniciamos a preparação e também a seleção das características, propriedades e atributos ("features") que serão utilizadas em nosso modelo preditivo. Nesta fase não nos limitamos em apenas buscar features prontas, mas também criamos novas features com base nos dados fornecidos para que pudéssemos explicitar informações valiosas, mas que não estavam visíveis nos dados.

4.3.1. Manipulação

Importamos as bases de dados em formato ".CSV" para nosso ambiente de desenvolvimento("Google Colaboratory") e realizamos algumas otimizações para conseguir lidar com problemas gerados pelo tamanho das bases. As principais mudanças que realizamos foram:

- mudança de tipo de algumas variáveis, como por exemplo de int64 para int16 em algumas variáveis que não armazenam grandes valores que exigissem, portanto, grande quantidade de memória volátil ("RAM");
- mudança das variáveis categóricas de 'object' para 'category', também para economizar memória RAM;
- encoding das variáveis categóricas usando Label-Encoding (se ordinais) e One-Hot Encoding (se nominais);
- mudança de 'object' para 'datetime' nas colunas que representam datas;
- transformação dos 'datetimes' em 'ordinais', para introdução no modelo;
- renomeação de algumas colunas para nomes mais intuitivos.

4.3.2. Agregação

Para juntar as bases de dados realizamos algumas operações de merge através da biblioteca "pandas" da linguagem Python. Utilizamos a operação de Inner Join - que retorna um dataframe com apenas as linhas que estão presente em ambos os dataframes, concatenados através de um valor de referência (no caso, o ID dos entregadores). Utilizamos também a operação de Left Join - que retorna um dataframe com todas as linhas dos primeiro dataframe que está sendo concatenado.

Além disso, aplicamos a função "*groupby*" para conseguirmos:

- agrupar os ganhos e as gorjetas de cada entregador;
- somar os minutos de punição de cada entregador, de forma que conseguíssemos visualizar o tempo total de punição de cada ID (entregador) da base;
- realizar uma contagem, por entregador, de pedidos em aberto para o suporte, a soma de devoluções que constam como pendentes na base e, por fim, uma contagem de pedidos com problema por entregador.

4.3.3. Remoção e Substituição

Retirar entregadores sem earnings, Inner Join em 'earnings', 'ordens realizadas e canceladas', 'média de km rodados', 'taxa de aceitação dos pedidos'.

Fizemos também algumas operações de remoção e substituição nas bases. Através de operações de Inner join - comentadas previamente no tópico 4.3.2 - juntamos os dataframes que continham as informações de earnings, ordens realizadas e canceladas, média de km rodados e taxa de aceitação dos pedidos – removendo, assim, qualquer entregador que não possuísse qualquer um desses atributos.

Na tabela de 'taxa de aceitação dos pedidos', alguns entregadores tinham valores em branco como porcentagem de aceitação. Realizamos uma operação 'dropna' para eliminar esses entregadores.

Em outras tabelas, realizamos operações de Left Join, realizando as substituições adequadas para os valores ausentes na tabela à direita na operação de merge com o método "*fillna*" do Pandas:

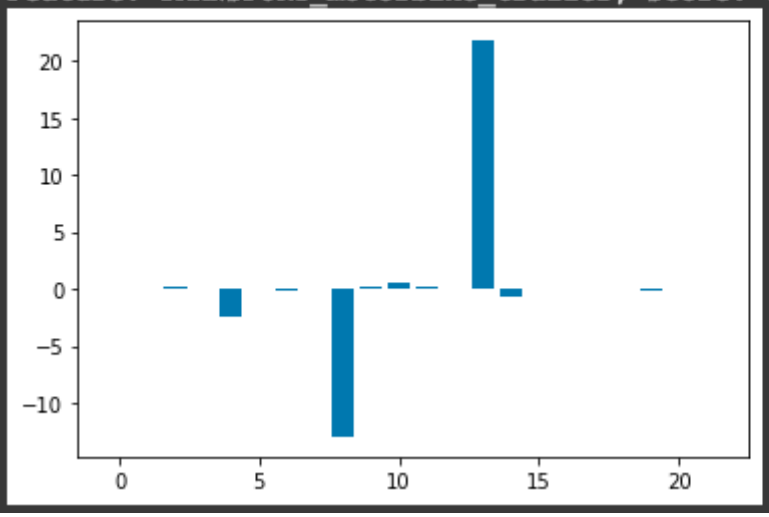
- "Incidentes Regras RT" (Medidas Disciplinares contra Entregadores) – se o entregador não tinha qualquer registro na tabela de medidas disciplinares, substituímos o valor vazio de "PUNISHMENT_MINUTES" por "0";
- "Quantidade de pedidos ao suporte da Rappi"- se o entregador não tinha qualquer registro na tabela de pedidos de suporte, substituímos o valor vazio de "SUPPORT_TICKET_COUNT" contagem por "0";
- "Devoluções Pendentes" – se o entregador não tinha qualquer devolução pendente na tabela, substituímos o valor "PRODUCT_RETURNS" por "0";
- "Pedidos com Problema" - se o entregador não tinha qualquer pedido com problema em seu ID, substituímos o valor "DEFECTS_COUNT" por "0".

4.3.4. Seleção das features

ABAIXO, SERIA POSSÍVEL VISUALIZAR A TABELA DE ARQUIVO CONTENDO TODOS OS DADOS FORNECIDOS PELO PARCEIRO PARA A CONSTRUÇÃO DO MODELO PREDITIVO. NO ENTANTO, EM RESPOSTA AO PEDIDO DO PARCEIRO PARA PROTEGER DADOS SENSÍVEIS, A TABELA FOI OCULTADA.

Ao invés de realizarmos uma análise gráfica exploratória para cada uma das features acima, optamos por construir um modelo preliminar usando regressão logística (disponível em https://colab.research.google.com/drive/1kowwKCfPTls6mnsk5RbLtMWLaypdAG3D#scrollTo=OJyQcGeaRkWg) para demonstrar como podemos apresentar as hipóteses mais pertinentes através de um modelo. Os resultados obtidos foram os seguintes:

```
Feature: LEVEL_NAME, Score: 0.05326
Feature: AUTO_ACCEPT, Score: 0.06191
Feature: ACCEPTANCE_RATE, Score: 0.12545
Feature: ORDERS_DONE, Score: -0.06278
Feature: ORDERS_LAST_30D_COUNT, Score: -2.50861
Feature: ORDERS_CANCEL, Score: 0.07066
Feature: ORDERS_CANCELED_LAST_30D_COUNT, Score: -0.13663
Feature: FIRST_ORDER_DATE, Score: -0.02853
Feature: LAST_ORDER_DATE, Score: -12.97138
Feature: EARNINGS, Score: 0.22665
Feature: TIPS, Score: 0.48481
Feature: PUNISHMENT_MINUTES, Score: 0.25026
Feature: PRODUCT_RETURNS, Score: 0.04352
Feature: SUPPORT_TICKET_COUNT, Score: 21.75265
Feature: DEFECTS_COUNT, Score: -0.72360
Feature: AVG_DISTANCE_TO_USER, Score: 0.02992
Feature: TRANSPORT_bicycle, Score: 0.01427
Feature: TRANSPORT_car, Score: -0.02671
Feature: TRANSPORT_motorbike, Score: 0.00896
Feature: TRANSPORT_neither, Score: -0.11032
Feature: TRANSPORT_cargo_van, Score: 0.00927
Feature: TRANSPORT_motorbike_trailer, Score: 0.00000
```



No gráfico acima, quanto *mais negativo* o valor da feature em questão, *menos* propenso um entregador é de dar "churn" se tiver um valor alto naquela feature. Inversamente, quanto *mais positivo* o valor da feature no gráfico acima, *mais propenso* um entregador é de dar "churn" se tiver um valor alto naquela feature.

No caso, se observa um valor muito negativo para "LAST_ORDER_DATE", ou seja, quanto mais tempo um entregador fica sem entregar um pedido, mais cresce sua chance de dar "churn". Isso pode representar um viés, porém, pois uma das definições de "churn" que aplicamos é a de um entregador que está há mais de 21 dias sem realizar entregas.

Outro ponto de destaque é o valor muito positivo para "SUPPORT_TICKET_COUNT". Isso representa o número absoluto de chamados abertos por um entregador para o suporte da Rappi. Todavia, entendemos que esse dado pode representar um ruído, pois em uma análise manual notamos que os entregadores que não deram "churn" estão sem *nenhum* ticket aberto para o suporte, o que é um forte indício de que se trata de um dado enviesado:

```
df_infos.groupby('IS_CHURN')['SUPPORT_TICKET_COUNT'].describe()
```

IS_CHURN	count	mean	std	min	25%	50%	75%	max
False	31133.0	0.000000	0.000000	0.0	0.0	0.0	0.0	0.0
True	132160.0	12.544597	31.301608	0.0	0.0	2.0	12.0	1184.0

4.4. Modelagem

Para o teste de todo modelo preditivo, os dados foram divididos em duas parcelas: dados de treino, e dados de teste, em uma proporção de 70% e 30%, respectivamente.

Realizamos um “undersampling” durante o processo de modelagem com a biblioteca "imbalanced-learn", pois o conjunto de dados de entregadores que deram “churn” era consideravelmente superior ao dos que permaneceram na plataforma. Esse tratamento essencialmente consiste em eliminar parte do conjunto com dados em excesso para equilibrá-lo com o de dados sub representados. Caso esse tratamento do desbalanceamento identificado não fosse realizado, isso poderia enviesar as métricas apresentadas a seguir.

Excluímos as features 'ORDERS_CANCELED_LAST_30D_COUNT', 'ORDERS_LAST_30D_COUNT' e 'LAST_ORDER_DATE', pois apresentam alta correlação com o target e poderiam enviesar o modelo ao fornecer dicas muito fortes de um entregador prestes a dar churn. Um entregador que realizou poucas ordens nos últimos 30 dias ou ficou muitos dias sem realizar uma ordem seria classificado como churn extremamente provável e traria um viés excessivo para o modelo. Outra feature removida foi 'SUPPORT_TICKET_COUNT', pois por algum motivo os entregadores que não deram churn estavam com essa métrica zerada, o que também traz um viés excessivo para o modelo.

Passada essa análise preliminar, utilizamos, majoritariamente, quatro métricas de avaliação, cada uma tendo sua peculiaridade. Vale ressaltar que esses indicadores se baseiam na “matriz de confusão”, uma matriz que compara as predições falsas e positivas, com o dado real do target.

		Detectada	
		Sim	**Não**
Real	**Sim**	Verdadeiro Positivo (VP)	Falso Negativo (FN)
	Não	Falso Positivo (FP)	Verdadeiro Negativo (VN)

Matriz de confusão

Acurácia: Indica a performance geral do modelo, apresentando uma porcentagem de quantas classificações o modelo classificou corretamente.

Precisão: Indica quantos dos positivos feitos pelo modelo foram realmente assertivos.

Revocação/Recall: Indica quantos dos positivos dados pelo modelo foram verdadeiros ou falsos.

F1 Score: É a média harmônica entre precisão e recall (2 * precisão * recall) / (precisão + recall).

Para fins de aprofundamento, as fórmulas para cálculo de cada uma das métricas já foram mencionadas na seção 4.1.3.

4.4.1. Avaliação e Seleção de Modelos (Algoritmos)

Dentre as quatro principais métricas de avaliação de modelos, demos prioridade a duas, acurácia e recall. Basicamente, buscamos um modelo com uma alta taxa de acertos, por isso priorizamos a acurácia. Também desejamos o menor número possível de falso negativos, por valorizamos a revocação ("recall") sobre a precisão, dado que é mais importante que o modelo identifique todos os entregadores propensos a dar churn ("recall") do que prevenir a identificação incorreta como churn um entregador que na realidade não é churn ("precisão").

Para termos um sentido inicial dos modelos que poderiam apresentar as melhores métricas, usamos a biblioteca Pycaret. Essa ferramenta possibilita rodar uma série de modelos de classificação e observar as métricas de cada um, para que possamos escolher o modelo mais adequado para implementar manualmente.

Atualmente, após o teste com vários algoritmos, elegemos os quatro melhores em termos de acurácia, sendo eles:

1. *Light Gradient Boosting Machine (LightGBM);*
2. *Random Forest Classifier;*
3. *Gradient Boosting Classifier; e*
4. *Ada Boost Classifier*.

No caso específico do modelo Extra Trees Classifier, optamos por excluí-lo pelo fato de ser muito semelhante ao Random Forest. Abaixo estão as métricas de cada modelo, com a Acurácia, Revocação, Precisão e F1 Score.

A tabela abaixo contém os modelos de machine learning aplicados ao nosso "dataset" pela ferramenta Pycaret, em ordem decrescente de acurácia:

Modelo	Acurácia	Revocação	Precisão	F1 Score
Light Gradient Boosting Machine	0.9040	0.9398	0.9434	0.9415
Random Forest Classifier	0.9016	0.9371	0.9429	0.9400
Extra Trees Classifier	0.9014	0.9379	0.9420	0.9399
Gradient Boosting Classifier	0.8750	0.8905	0.9544	0.9213

Ada Boost Classifier	0.8660	0.8800	0.9534	0.9152
Decision Tree Classifier	0.8589	0.8958	0.9300	0.9126
K Neighbors Classifier	0.8163	0.8261	0.9435	0.8809
SVM - Linear Kernel	0.8106	0.8154	0.9470	0.8761
Logistic Regression	0.8100	0.8135	0.9481	0.8757
Ridge Classifier	0.8074	0.8140	0.9441	0.8743
Linear Discriminant Analysis	0.8074	0.8140	0.9441	0.8743
Naive Bayes	0.2102	0.0409	0.9706	0.0785
Quadratic Discriminant Analysis	0.1776	0.0000	0.0000	0.0000
Dummy Classifier	0.1776	0.0000	0.0000	0.0000

O notebook que usamos para obter a tabela acima pode ser encontrado no link a seguir:

https://colab.research.google.com/drive/1_l8X6shyH90NFIJso10Mw7Lq_NReoq6m#scrollTo=58ZtXVvXFGas

É importante observar que as métricas apresentadas no teste automatizado são ligeiramente diferentes da seção 4.5 com os modelos feitos manualmente, pois os "datasets" de treino e teste são sorteados aleatoriamente e, portanto, irão divergir. Além disso, a biblioteca Pycaret realiza alguns ajustes que optamos por fazer de forma diferente quando fizemos a construção dos modelos de forma manual.

Apesar disso, essa é uma excelente forma de avaliar quais modelos apresentam os resultados mais promissores e serviu para embasar a seleção dos modelos para avaliação manual.

4.4.2. Configuração de Hiperparâmetros

LightGBM

LightGBM é uma estrutura de aumento de gradiente que usa uma série de árvores de decisão para embasar suas escolhas ("ensemble"). Ele pode ser usado para tarefas de classificação ou outras tarefas de machine learning.

Selecionamos os seguintes hiperparâmetros, baseados na documentação disponível em Parameters Tuning — LightGBM 3.3.2.99 documentation:

'learning_rate', 'n_estimators', 'num_leaves', 'max_depth', 'min_child_samples', 'subsample', 'colsample_bytree', 'reg_alpha', 'reg_lambda', 'min_split_gain', 'min_child_weight', 'subsample_freq', 'max_bin', 'cat_smooth', 'cat_l2', 'max_cat_to_onehot', 'cat_l2', 'cat_smooth', 'max_cat_threshold', 'metric', 'n_jobs', 'random_state'.

Random Forest Classifier

Random Forest é um algoritmo de aprendizado de máquina "ensemble", que usa o resultado de diversas árvores de decisão para embasar suas decisões.

Selecionamos os seguintes hiperparâmetros, baseados no seguinte post: Random Forest Hyperparameter Tuning in Python | Machine learning

'n_estimators', 'max_features', 'criterion', 'max_depth', 'min_samples_split', 'min_samples_leaf'

Gradient Boosting Classifier

Gradient Boosting, assim como LightBGM, é um algoritmo de aumento de gradiente baseado em árvores de decisão. Contudo, o LightBGM é comparativamente mais eficiente.

Selecionamos os seguintes hiperparâmetros, baseados no seguinte post: Gradient Boosting | Hyperparameter Tuning Python

'loss', 'learning_rate', 'n_estimators', 'subsample', 'min_samples_split', 'min_samples_leaf', 'min_weight_fraction_leaf', 'max_depth', 'min_impurity_decrease', 'init', 'max_features', 'random_state'

AdaBoost

AdaBoost é uma sigla para "Adaptive Boosting". É um algoritmo de aprendizado de máquina muito usado para tarefas de classificação binária. Ele usa uma série de algoritmos de aprendizado mais "fracos" e consolida seus resultados para obter uma decisão mais robusta.

Selecionamos os seguintes hiperparâmetros, baseados no seguinte post: Tuning of Adaboost with Computational Complexity | by Srijani Chaudhury | Medium

'n_estimators', 'learning_rate', 'algorithm', 'random_state'

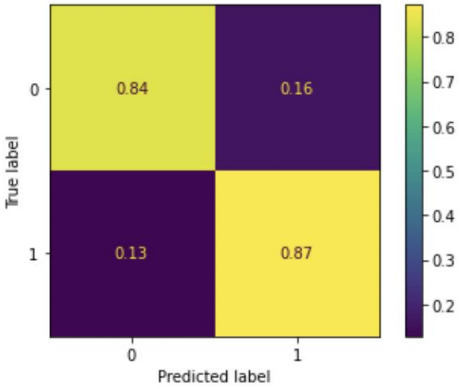
4.4.3. Resultados Obtidos

Após selecionarmos os hiperparâmetros para cada modelo como citado na seção anterior, utilizamos o RandomSeach para realizar o tuning dos modelos, achando quais seriam os pontos ótimos de cada

hiperparâmetro escolhido. Os resultados obtidos rodando os modelos com as melhores configurações dos hiperparâmetros foram os seguintes:

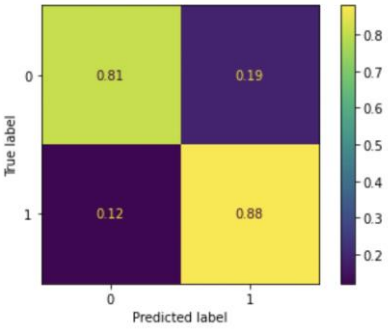
LightGBM

Acc treino: 91,26%
Acc teste: 86,51%
Revocação: 87,12%
Precisão: 96,13%
F1_score: 91,40%



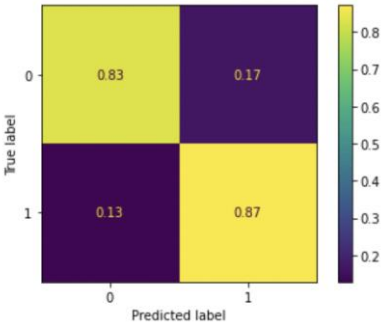
Random Forest Classifier

Acc treino: 85,57%
Acc teste: 86,71%
Revocação: 87,95%
Precisão: 95,56%
F1_score: 91,59%



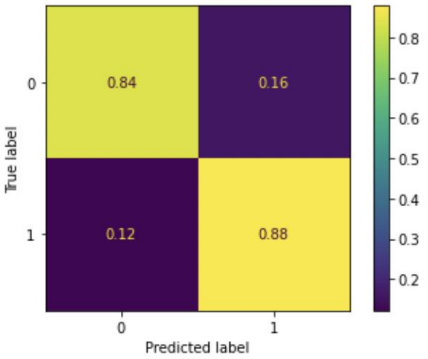
Gradient Boosting Classifier

Acc treino: 87,49%
Acc teste: 86,47%
Revocação: 87,16%
Precisão: 96,04%
F1_score: 91,38%



AdaBoost

Acc treino: 91,26%
Acc teste: 86,51%
Revocação: 87,80%
Precisão: 96,17%
F1_score: 91,80%



4.5. Avaliação

A partir dos resultados obtidos na seção 4.4 com a modelagem e testes preliminares, pudemos selecionar os modelos mais promissores para avaliação e implementação manual, no notebook linkado a seguir:

https://colab.research.google.com/drive/1kowwKCfPTls6mnsk5RbLtMWLaypdAG3D#scrollTo=OJyQcGeaRkWg

Treinamos cada um dos modelos manualmente e verificamos para cada uma:

1. acurácia de treino e de teste;
2. revocação;
3. precisão;
4. F1 score;
5. matriz de confusão; e
6. importância de cada feature para o modelo.

Dessa forma, podemos verificar a qualidade dos resultados obtidos e a taxa de erro, constatando se os resultados preliminares do Pycaret estão de acordo com os modelos do scikit-learn.

Observa-se que os algoritmos escolhidos na seção 4.4. foram adequados pois são todos destinados para tarefas de *classificação* – que é o problema de negócio enfrentado pelo cliente, como apontado na seção 4.1.3. A própria biblioteca que usamos para testes preliminares (Pycaret) apenas usa modelos relacionados com essa tarefa, como pode ser visto na linha de importação do módulo relevante (from pycaret.*classification* import *).

Sendo assim, vamos passar a avaliar os modelos e seus resultados.

4.5.1. Análise dos Resultados

LightGBM

Esse modelo retornou em geral métricas muito boas e deu bastante importância para as features de ordens canceladas e defeitos nas ordens.

```
Acc treino:  0.8752060336646521
Acc teste:  0.8705572619569987
```

```
Revocação:  0.8778389626208541
Precisão:  0.9616111636181687
F1_score:  0.9178174871652007
```

Matriz de confusão:

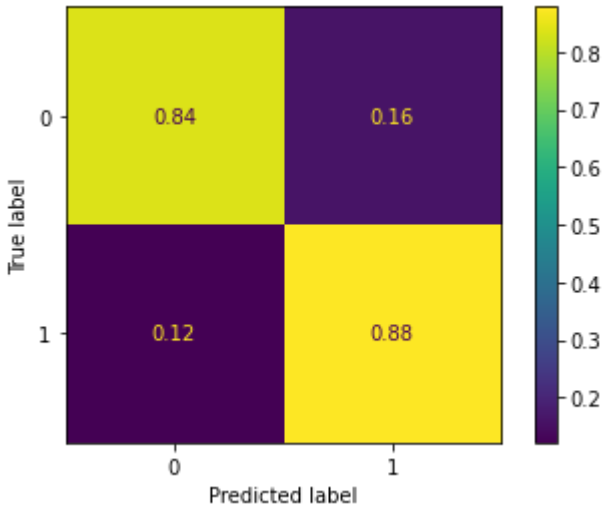
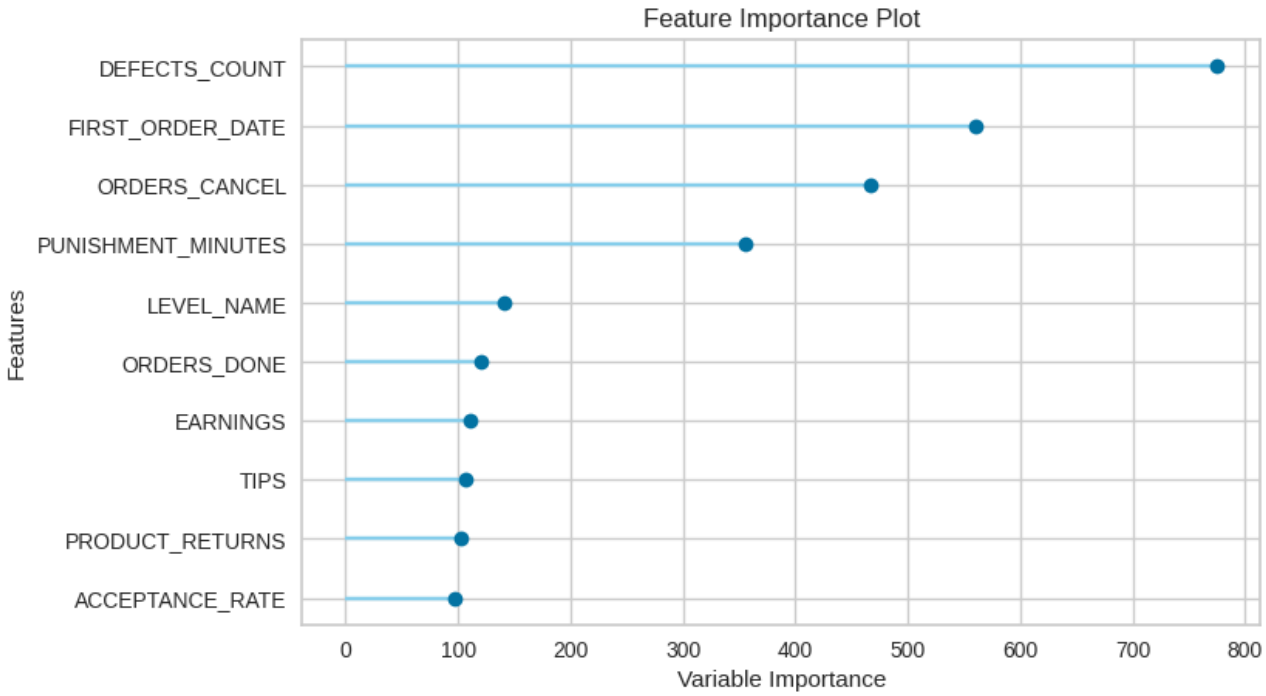
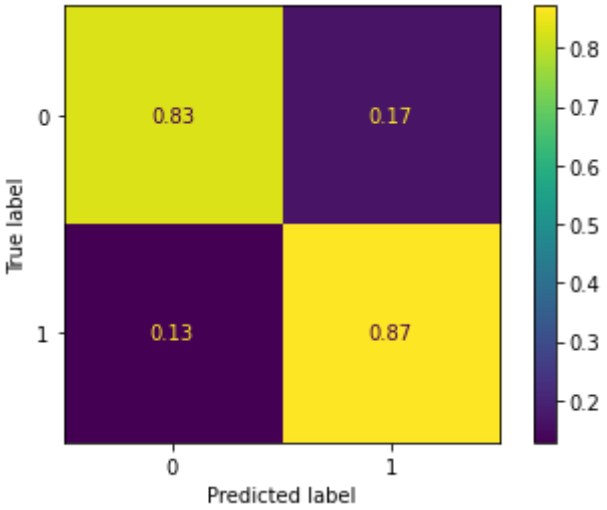


Gráfico de importância das features:



Após o ajuste dos hiperparâmetros, tivemos os seguintes resultados:

```
Acc treino:  0.9126666999650367
Acc teste:  0.8651455316659353
Revocação:  0.8712665262516812
Precisão:  0.9613327733445332
F1_score:  0.914086419424403
```



Random Forest Classifier

O modelo apresentou ótima acurácia, especialmente no dataset de teste. Também apresentou a melhor revocação. Porém, a precisão e acurácia no dataset de treino foram inferiores ao do LightGBM.

```
Acc treino:  1.0
Acc teste:  0.8644769008963831
```

```
Revocação:  0.8715710406780521
Precisão:  0.9601632607419418
F1_score:  0.9137247599031632
```

Matriz de confusão:

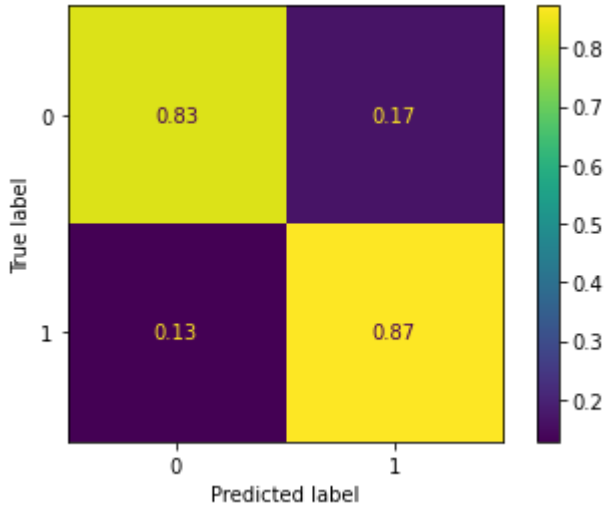
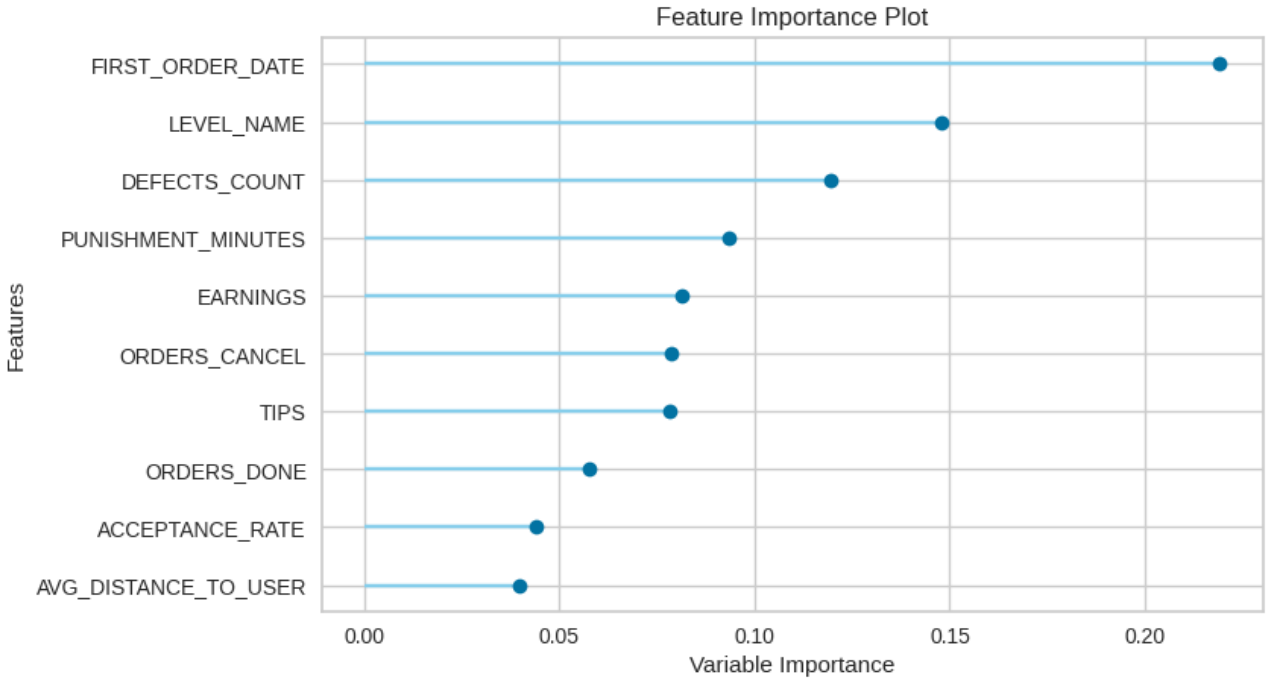
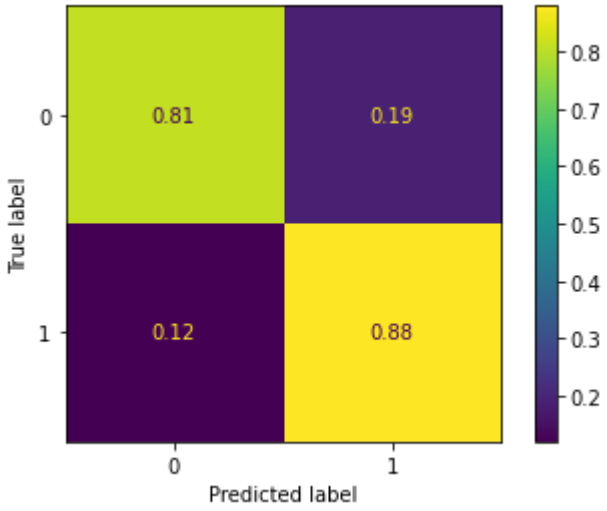


Gráfico de Importância das Features:



Após o ajuste dos hiperparâmetros, tivemos os seguintes resultados:

```
Acc treino:  0.8557514609659858
Acc teste:  0.8671723186861405
Revocação:  0.879539168168092
Precisão:  0.9556106975461814
F1_score:  0.9159982557448102
```



Gradient Boosting Classifier

Os resultados obtidos entre o LightBGM e Gradient Boosting Classifier foram muito semelhantes,, com exceção da acurácia de treino, que foi significantemente superior no LightBGM.

```
Acc treino 0.8538035063183658
Acc teste 0.861133747048622
```

```
Revocação 0.8671048290912782
Precisão 0.9603979875769415
F1 Score 0.9113701224228522
```

Matriz de confusão:

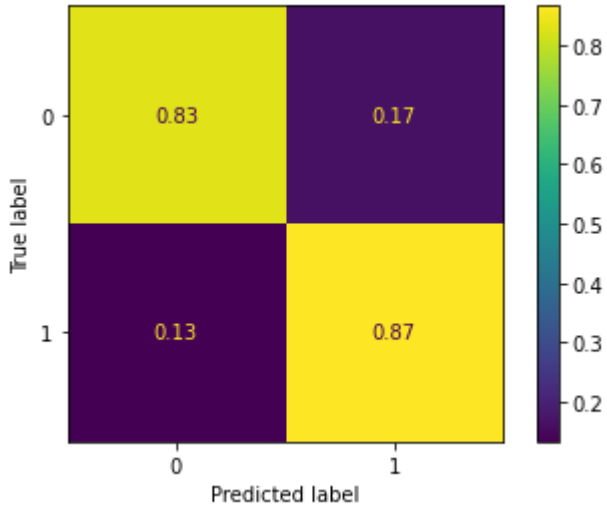
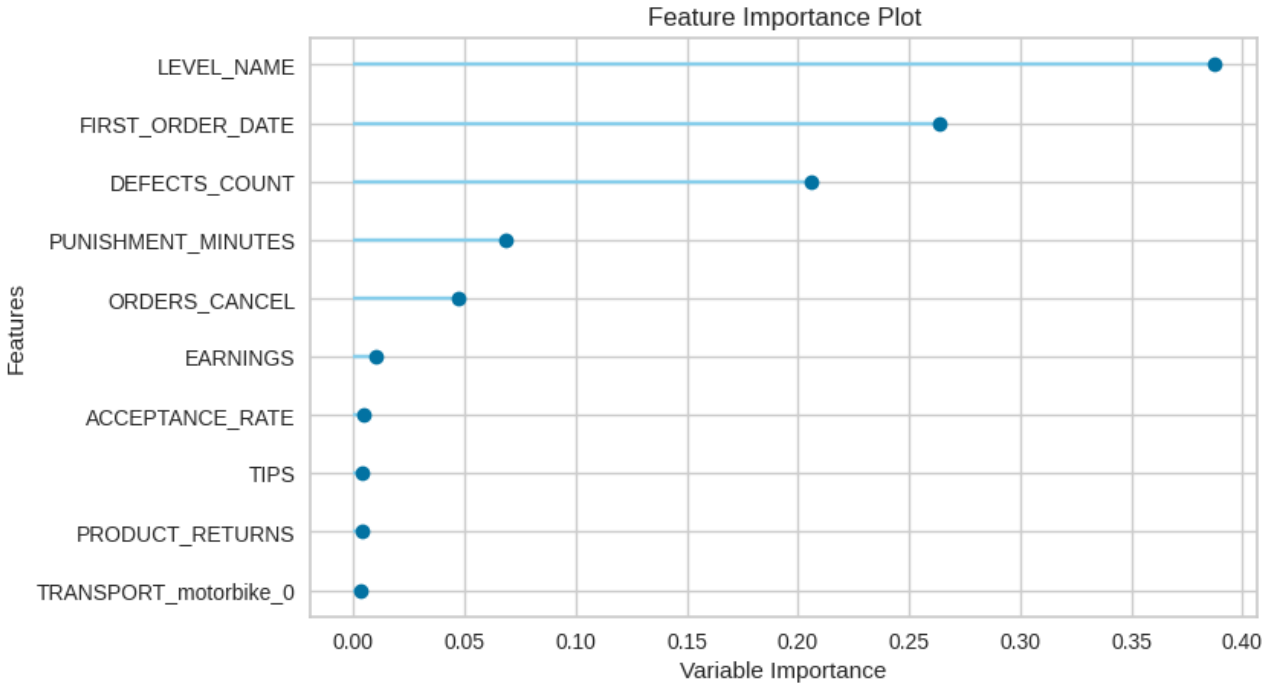
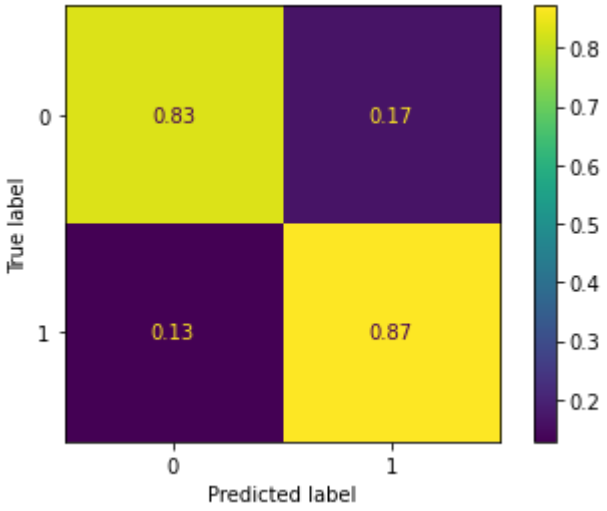


Gráfico de importância das features:



Após o ajuste dos hiperparâmetros, tivemos os seguintes resultados:

```
Acc treino:  0.8749313221117826
Acc teste:  0.8647067427234167
Revocação:  0.8716217930824472
Precisão:  0.9604071132982888
F1_score:  0.9138630588923919
```



AdaBoost

Esse algoritmo apresentou boas métricas, porém elas foram inicialmente inferiores às dos demais algoritmos testados. Após o ajuste dos hiperparâmetros, as métricas melhoraram consideravelmente.

```
Acc treino:  0.8449378152939414
Acc teste:  0.8522952840636036
```

```
Revocação:  0.8563453193595046
Precisão:  0.9599476588723901
F1_score:  0.905191722214026
```

Matriz de confusão:

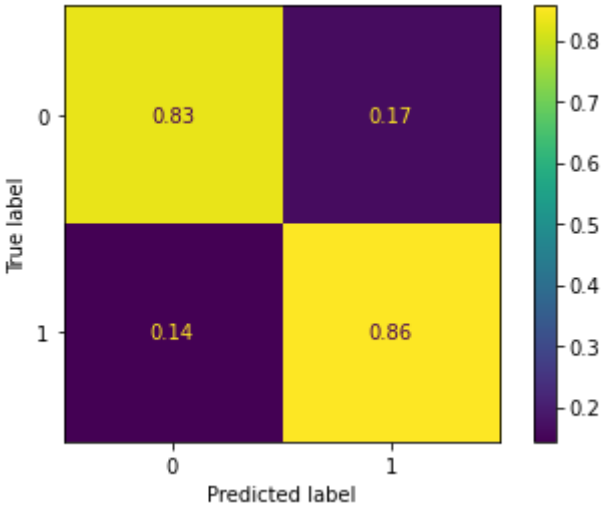
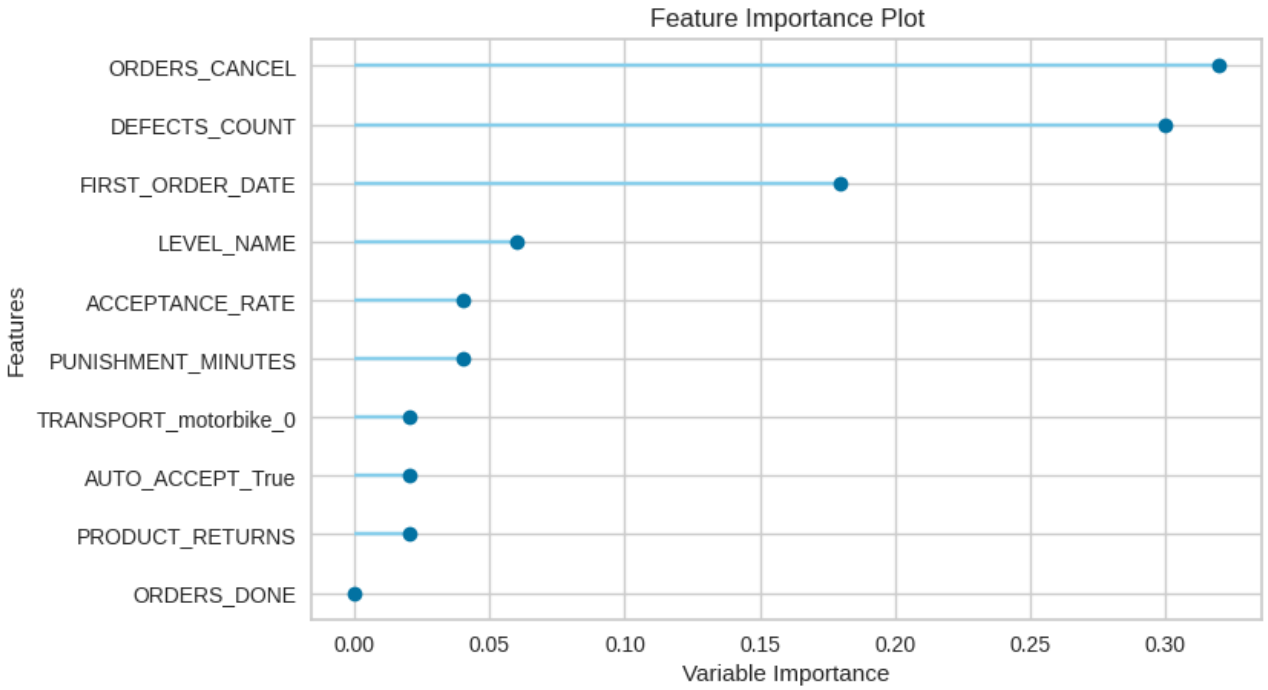
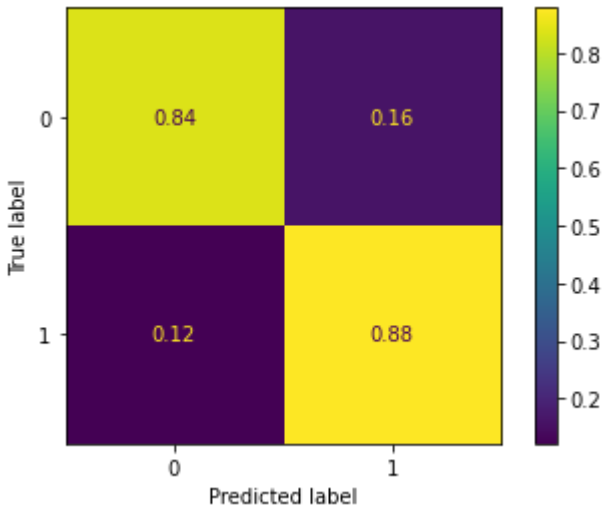


Gráfico de importância das features:



Após o ajuste dos hiperparâmetros, tivemos os seguintes resultados:

```
Acc treino:  0.9126666999650367
Acc teste:  0.8651455316659353
Revocação:  0.8780673484406324
Precisão:  0.9617811379492454
F1_score:  0.9180197389366445
```



4.5.2. Análise Comparativa

Todos os algoritmos escolhidos para avaliação foram adequados e confirmaram os resultados obtidos nos testes preliminares. Todos apresentaram acurácia acima de 85% e taxas de revocação, precisão e score F1 acima de 90%. As matrizes de confusão também apontam que os dados não estão viciados, devido a correção feita com o "undersampling".

Após o ajuste dos hiperparâmetros, obtivemos os seguintes resultados:

Modelo	Acurácia de Treino	Acurácia de Teste	Revocação	Precisão	F1 Score
Light Gradient Boosting Machine	91,27%	86,51%	87,12%	96,13%	91,40%
Random Forest Classifier	85,57%	86,71%	87,95%	95,56%	91,59%
Gradient Boosting Classifier	87,49%	86,47%	87,16%	96,04%	91,38%
Ada Boost Classifier	91,26%	86,51%	87,80%	96,17%	91,80%

Notamos que o modelo Random Forest Classifier obteve a melhor acurácia quando testado e também obteve a melhor revocação. Conforme mencionado anteriormente na seção 4.4.1, esses foram os indicadores que priorizamos, portanto selecionamos o modelo **Random Forest** como o algoritmo mais adequado para solucionar o problema de predição de "churn" da Rappi.

4.6. Modelo Escolhido

Como citado anteriormente, no tópico 4.5.2, decidimos seguir trabalhando com o modelo Random Forest Classifier, pois ele não somente atingiu um nível alto de acurácia quando comparado com outros modelos, como também atingiu um alto nível de revocação. Achamos prudente atribuir uma alta importância para a revocação, pois uma das exigências do cliente era de que conseguíssemos mapear todos os casos de churn, mesmo que isso abrisse brecha para falsos positivos. Dessa forma, o indicador que mede esse efeito é justamente a revocação - Logo, ficamos com o modelo Random Forest, modelo com maior acurácia e revocação.

A configuração ótima de hiperparâmetros, para o modelo de Random Forest, que foi encontrada através do RandomSearch foi a seguinte:

{'n_estimators': 6,

'min_samples_split': 2,

'min_samples_leaf': 8,

'max_features': 'sqrt',

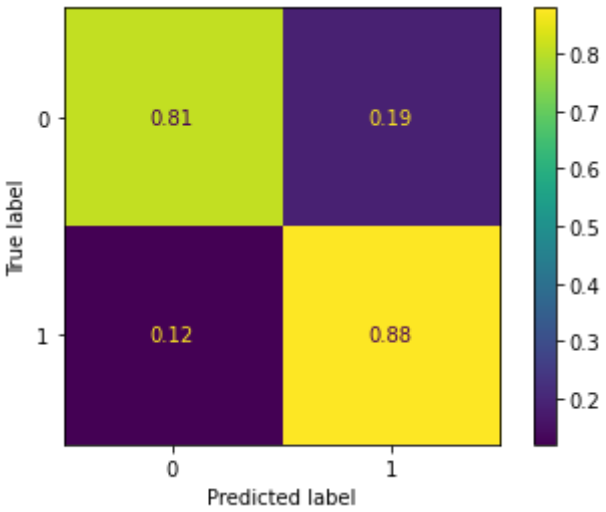
'max_depth': 10,

'criterion': 'gini'}

Além disso, o desempenho das métricas desse modelo foram os seguintes:

```
Acc treino:  0.8557514609659858
Acc teste:  0.8671723186861405
Revocação:  0.879539168168092
Precisão:  0.9556106975461814
F1_score:  0.9159982557448102
```

E a matriz de confusão:



4.6.1. Possíveis falhas

Possível falha	Solução possível
Tentar rodar o modelo com um dataset com colunas diferentes das que foram usadas na criação do modelo. Ou seja, as colunas do dataset não batem com as features selecionadas.	Remover as colunas que não fazem parte das features selecionadas, de tal forma que as colunas do dataset sejam iguais as informadas na parte de feature engineering.
Utilizar um dataset com diversos valores NaNs e/ou nulos.	Tratar o dataset que se deseja utilizar para predição, de forma a remover valores nulos e não numéricos.

5. Conclusões e Recomendações

Os resultados se mostraram bastante satisfatórios. O modelo escolhido apresenta métricas elevadas, e pode ser realmente utilizado para tratar a questão levantada. É importante que para a utilização do modelo, os dados sigam o mesmo padrão estabelecido previamente na Feature Engineering.

É recomendado que a porcentagem de confiança do modelo seja verificada, a fim de que a pessoa que irá utilizar as previsões feitas pelo modelo, tenha maior confiabilidade nas ações que irá tomar.

É importante também se atentar ao fato que, embora o modelo tenha pontuações altas, ele não é perfeito, portanto, não é coerente acreditar em todas as previsões feitas por ele. O recomendado é que o resultado gerado pelo modelo não seja dado como verdade absoluta, e que seja tratado apenas como uma ferramenta para analisar um perfil de churn, portanto, como qualquer outra aplicação, erros podem acontecer.

Além disso, vale ressaltar que ao se analisar os gráficos de feature importance que foram mostrados na seção 4.5.2, é possível notar que em todos os modelos testados, 2 features se mostraram particularmente relevantes para explicar os casos de churn, sendo elas: DEFECTS_COUNT e FIRST_ORDER_DATE.

Nesse sentido, achamos prudente fazer 2 recomendações ao parceiro: Tendo em vista que a quantidade de defeitos em um pedido é um fator que influencia de maneira significativa o churn, recomendamos melhorar o atendimento aos RT's, de forma a mitigar esse impacto. Ademais, a data do primeiro pedido foi uma feature que mostrou ter grande relevância também para explicar os churns, dessa forma achamos razoável a sugestão de criar um processo de onboarding para os novos RT's, de forma a tentar fidelizar alguns desses entregadores e evitar o churn.

6. Referências

ANSELMO, Fernando. **Machine Learning na Prática**. São Paulo, 2020.

AZANK, Felipe; GURGEL, Gustavo Korzune. **Dados Desbalanceados — O que são e como lidar com eles**. Disponível em: https://medium.com/turing-talks/dados-desbalanceados-o-que-são-e-como-evitá-los-43df4f49732b. Acesso em: 20 ago. 2022.

CORRÊA, Danielle Modelli. **Feature Engineering: preparando dados para aprendizado de máquina**. Disponível em: https://ateliware.com/blog/feature-engineering. Acesso em: 15 ago. 2022.

FERREIRA, André Carlos Ponce de Leon. **Inteligência Artificial - Uma Abordagem De Aprendizado De Máquina**. 2. ed. São Paulo: Ltc, 2021.

SILVA, Leandro Augusto; PERES, Sarajane Marques; BOSCARIOLI, Clodis. **Introdução à Mineração de Dados - Com Aplicações em R**. São Paulo: Gen Ltc, 2016.

SOUZA, Emanuel G de. **Entendendo o que é Matriz de Confusão com Python**. Disponível em: https://medium.com/data-hackers/entendendo-o-que-é-matriz-de-confusão-com-python-114e683ec509. Acesso em: 15 set. 2022.

VAZ, Arthur Lamblet. **Otimizando os hiperparâmetros**. Disponível em: https://medium.com/data-hackers/otimizando-os-hiperparâmetros-621de5e9be37. Acesso em: 20 set. 2022.