Rappi Hour
Rappi

inteli
instituto
de tecnologia
e liderança

Controle do Documento

Histórico de revisões

Data	Autor	Versão	Resumo da atividade
02/08/2022	“Rappi Hour”	1.0	Criação do Documento
09/08/2022	Caio Martins	1.1	Atualização da Matriz de Riscos; Atualização da Análise de Indústria.
10/08/2022	Caio Martins; Sarah Ribeiro; Mateus Almeida	1.2	Atualização da documentação: Seções 2; 4.1.3; 4.1.4; 4.2
14/08/2022	Sarah Ribeiro	1.3	Atualização da seção 4.2
17/08/2022	Raissa Sabino	1.4	Adicionando as Personas (seção 4.1.6)
20/08/2022	Sarah Ribeiro	1.5	Adicionando o mapa da Jornada do Usuário. (seção 4.1.7)
07/09/2022	Caio Martins; Mateus Almeida	1.6	Preenchimento das secções 4.4 (Modelagem) e 4.5 (Avaliação)
20/09/2022	Marcos Silva	2.0	Seção 4.5
22/09/2022	Sarah Ribeiro	2.1	Seção 4.4
04/10/2022	Caio Martins, Mateus Almeida, Sarah Ribeiro e Raissa Sabino	2.2	Revisão da documentação

Sumário

1. Introdução

A Rappi é uma empresa multinacional que atua em nove países da América Latina e em 250 cidades, oferecendo serviços, como entregas turbo, as quais levam até 10 minutos para serem entregues ao cliente, e um banco digital que pode ofertar cartões de crédito em minutos.

A empresa é diversificada em suas áreas de atuação e, portanto, é difícil incluí-la em um único setor. Dentro do mercado brasileiro de delivery, a Rappi domina 5% do market share, porcentagem considerada alta, dado que sua maior concorrente domina 83% desse setor.

A Rappi enfrenta um obstáculo relacionado à alta rotatividade dos entregadores na plataforma. Nesse sentido, o parceiro expôs o conceito de churn, que se refere à inatividade de um entregador, denominado “Rappitendero” ou RT, na plataforma por, pelo menos, 21 dias. A título de expressividade desse problema, foi verificado em uma análise da base de dados do cliente que, no último ano, ocorreram cerca de 32 milhões de casos de churn. Essa é a dor que levou o cliente a requisitar a solução de machine learning.

Dentro da Rappi, a proposta de projeto está englobada na área de atuação do time de operações. A solução desenvolvida tem como objetivo fundamentar iniciativas da empresa para com os entregadores, as quais visam evitar a saída de couriers da plataforma, e, assim, minimizar prejuízos relacionados com a falta de mão de obra.

2. Objetivos e Justificativa

2.1. Objetivos

A solução proposta pelo grupo tem objetivo de solucionar a dor do cliente mediante acertada definição de variáveis para a construção do modelo e uma análise detalhada dos dados fornecidos, bem como sua estruturação em tabelas com dados correlatos. A benesse do projeto se encontra na predição de grupos de RT's que estão propensos a ficarem inativos da plataforma e indicar possíveis soluções para estes entregadores dependendo da granularidade temporal suficiente.

2.2. Justificativa

O desenvolvimento desta solução se justifica pela facilitação do processo de tomada de decisão por parte da Rappi em relação a seus colaboradores, neste caso, os RT's. É possível inferir sua necessidade através da dificuldade da empresa em avaliar sua base de entregadores e entender como atender os problemas enfrentados por estes fazem com que eles deem churn na plataforma.

3. Metodologia

3.1. CRISP-DM

O CRISP-DM é uma metodologia ágil que consiste num processo cíclico devido ao processo de concepção do aprendizado de máquina (machine learning), além do alto nível de informações. A metodologia propicia um modelo. O grupo optou por utilizar as metodologias ágeis como o CRISP-DM, metodologia realizada na matemática e estatística no cruzamento de dados, para garantir uma melhoria no relacionamento com o cliente, transformando o volume do conjunto de dados em informações úteis para o gerenciamento e tomada de decisões de uma equipe. Essa metodologia foi bastante utilizada durante nosso projeto devido os incrementos e mudanças contínuas que ocorrem a cada Sprint e validação com nosso parceiro,

3.2. Ferramentas

A principal ferramenta utilizada pelos grupos no estudo dos dados foi a interface Google Colab Notebook, que serve para auxiliar no processo de Machine Learning e exploração da base de dados.

Além disso, utilizamos a biblioteca Pandas da linguagem Python e o scikitlearn para fazer o "plot" dos gráficos e matrizes de confusão.

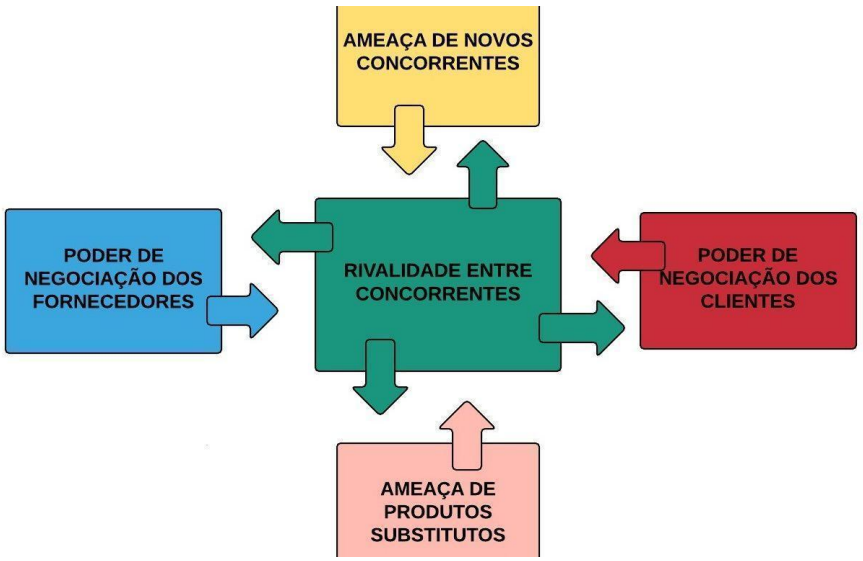
3.3. Principais técnicas empregadas

Empregamos os modelos de Random e Oversampling para fazer o balanceamento da amostra contida na base de dados, utilizamos os modelos de aprendizado de máquina LightGBM, ADABoost e RandomForest, sendo que estes modelos utilizam métodos de ensemble sendo o primeiro aquele que cria diversas florestas de decisão, ou seja, cria modelos de RandomForest mais fracos para obter uma métrica geral melhorada, já o segundo utiliza métodos de regressão mais fracos para realizar uma predição mais acurada, enquanto aquele último cria diversas árvores de decisão para gerar um output mais acurado.

4. Desenvolvimento e Resultados

4.1. Compreensão do Problema

4.1.1. Contexto da indústria



Cinco forças de Poter (imagem 1)

A partir da análise de forças, proposta por Michael Porter, se inicia a análise do setor em que está inserida a Rappi, a fim de entender os principais players, modelos de negócios e tendências do mercado.

Modelo de negócio: Rappi é um aplicativo de entregas sob demanda que atua em dois ramos principais, como plataforma de logística e um marketplace onde restaurantes podem se cadastrar e operar com a venda dos seus produtos. A empresa é considerada um

exemplo de “unicórnio” na América Latina e tem diversificado sua atuação, partindo para áreas como e-commerce e viagens [5].

Ameaça de novos concorrentes: Com um mercado que contabilizou em torno de R$40,5 bilhões é esperado que haja diversos players procurando entrar neste setor tão importante como o delivery. Contudo, indo em contramão a esta suposição, o mercado é dominado por grandes corporações tais como o Ifood, o finado Ubereats e a Rappi sendo esta responsável por um market share de 5%, em média, do setor de delivery[1]. Pela composição do setor se dar por grandes players, a ameaça de novos concorrentes representa força irrisória para empresa que compete num setor de mar vermelho.

Poder de negociação dos clientes: Mediante ao crescimento de 24% no setor de delivery no ano de 2021 [2], os clientes têm conquistado cada vez mais poder de barganha neste mercado. A alta demanda força as empresas no setor a praticar preços mais competitivos ao passo de que aumentam eficiência e qualidade no serviço de entrega, como é o caso da Rappi. Como define Porter, o poder de barganha dos clientes é uma peça fundamental para uma empresa analisar o cenário em que está inserida, o que, como no caso da Rappi, se demonstra como altíssimo, uma vez que as poucas concorrentes aptas do setor tendem a aumentar seu market share em detrimento de qualquer eventual falha da Rappi em atender seus clientes ou atender as necessidades dos entregadores ou parceiros do app.

Ameaça de produtos substitutos: A ameaça por produtos substitutos para o setor de delivery é mediana, a criação de novos aplicativos individualizados para cada estabelecimento pode ser uma ameaça potencial, uma vez que, ainda que existam empresas bem estabelecidas, como a Rappi, cujas estão expandindo suas áreas de atuação para captar cada vez mais clientes[3], o desenvolvimento destes app’s já ocorre e pode se tornar um problema futuro.

Poder de negociação dos fornecedores: O poder de barganha dos fornecedores é alto, atualmente, uma vez que a empresa trabalha quase que exclusivamente com serviço de entrega e, portanto, necessita de parceiros para fornecer a melhor entrega para o cliente final e manter sua posição no mercado. Ainda como citado anteriormente, por estar expandindo para outros setores de atuação, a empresa se torna ainda mais dependente de seus fornecedores.

Rivalidade entre concorrentes: O nível de competitividade é alto neste setor em decorrência do foco dos concorrentes em eficiência, tecnologia e suporte ao cliente. As empresas no setor têm superado os desafios de sua área de atuação através da manutenção de imagem, inovação tecnológica, qualidade dos produtos, orientação da empresa em torno do cliente e suporte ao entregador, peça fundamental para o crescimento no setor, o que torna a competição acirrada pela fidelização da clientela.

Tendências: A inovação tecnológica, como supracitado anteriormente, é essencial para o desenvolvimento de um negócio sustentável, uma vez que concorrentes fazem seu uso para antecipar futuros movimentos de mercado, fazer recomendações de pedidos e etc[4] . Desta forma, manter-se atualizado é essencial para continuar sendo competitivo na indústria e seguir as tendências da era informacional tem poder de potencializar o negócio exponencialmente.

Conclusão: Em termos gerais, os gastos em delivery subiram 24% no último ano. A capacidade de inovação, manutenção de qualidade e orientação de desenvolvimento em torno do cliente, tem assegurado o crescimento do setor, e da empresa, como um exemplo de confiabilidade e inovação, além da preocupação com os colaboradores, entregadores e fornecedores,o que torna a empresa cada vez mais propícia a captar novos clientes.

Principais Concorrentes: Um dos principais concorrentes da Rappi são empresas de entrega de delivery de forma geral, tais como o ifood que é um gigante do mercado, abarcando em média 87% do mercado, a rede Pão de Açúcar que já conta com delivery próprio para entrega dos produtos da rede e redes de entrega de produtos de mercado tal como a Shopper.

4.1.2. Análise SWOT

Strengths (Forças) - Plataforma multilateral, prestadora de diversos serviços de entrega, não só de alimentos; - Marca forte em países da américa latina.	**Weakness (Fraquezas)** - Marketing deficitário dos demais serviços da empresa - Baixa participação no mercado
Opportunities (Oportunidades) - Área em ascensão - Aumento de pedidos de comida por aplicativo. - Crescimento tecnológico. - Poucos competidores.	**Threats (Ameaças)** - Regulamentação da obrigação trabalhista da Rappi para com os RT's; - Mercado acirrado, seu competidor domina o mercado. - Retração da demanda com o fim da pandemia. - Aumento da inflação, a diferença dos preços é ressarcida pela Rappi.

Análise SWOT (tabela 1)

4.1.3. Planejamento Geral da Solução

A solução é um modelo preditivo que capaz de analisar dados e identificar quais RTs cadastrados estão mais propensos a dar churn, ou seja,ficarem inativos por mais de 21 dias. Essa proposta poderá ser utilizada pelo cliente com a finalidade de observar o nível de satisfação dos entregadores com a plataforma e, assim, entender quais atitudes podem ser tomadas para que essa desistência não ocorra.

A inteligência artificial desenvolvida prediz uma classificação, ou seja, categoriza os entregadores em uma determinada classe. Essa classificação retorna três possíveis valores: 0 (churn definitivo), 1 (não churn)e 2 (churn, mas inconsistente). O modelo rotula como 0 aqueles entregadores que estão previstos para não dar churn, como 1 aqueles que abandonarão a plataforma - deixam de ser ativos na base de dados - e como 2 aqueles que darão churn, mas não deixarão a Rappi de vez - continuando ativos na plataforma.

Após a categorização, mediante as métricas de avaliação, a solução também indica, em uma escala de 1 a 5, o nível de certeza da predição realizada pelo algoritmo, permitindo que o parceiro também tenha esse fator como uma métrica de priorização. Nesse processo, o método usado é o de probabilidade da previsão, então o algoritmo divide as predições (0, 1 e 2) nos tipos "certeza de churn" (probabilidade de ser 0 maior que 70%), "certeza de não churn" (probabilidade de ser 1 maior que 70%), "RT inconsistente" (probabilidade de ser 2 maior que 70%), "provável churn" (probabilidade de ser 2 menor que 70%, mas maior que a de ser 1) e "provável não churn" (probabilidade de ser 1 menor que 70%, mas maior que a de ser 2).

O modelo deve ser utilizado periodicamente pelos gestores, os quais irão acompanhar os entregadores que estão com uma alta probabilidade de dar churn. A ferramenta apresenta, em escala, a tendência de decisão do trabalhador, mostrando de forma gráfica os dados analisados, assim pretende-se facilitar o trabalho do gestor de operações, indicando quais são as métricas que mais afetam o RT classificado como “churn” ou “possível churn”.

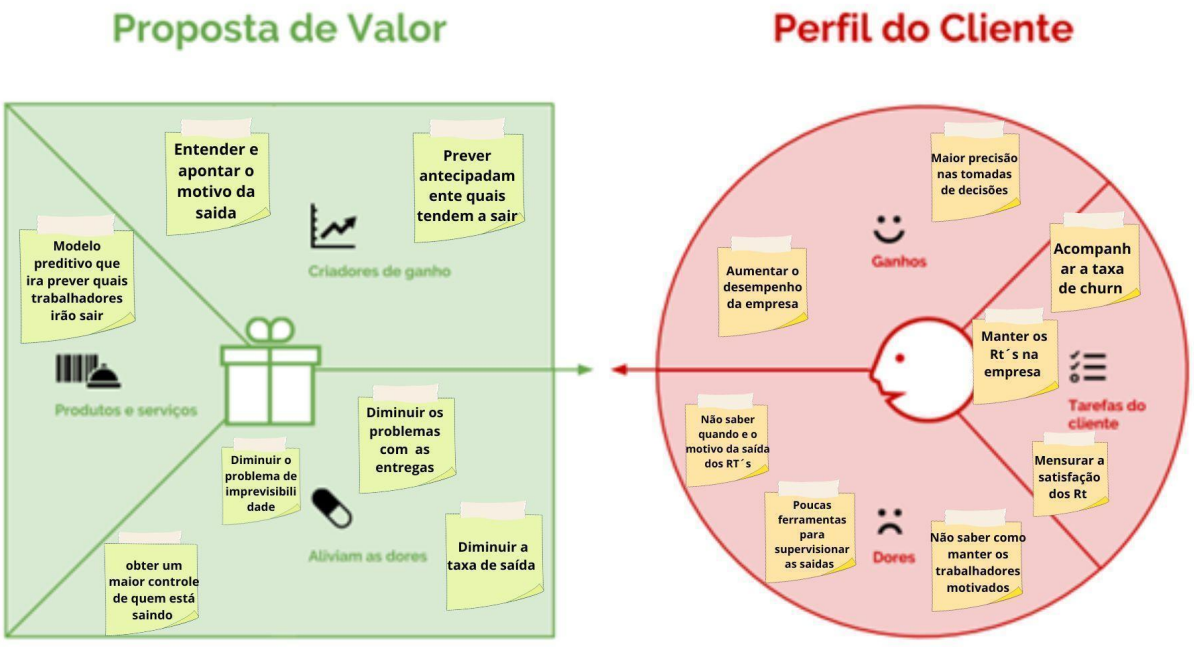
O critério de avaliação para os outputs, resultados obtidos pelo modelo preditivo, será obtido em porcentagem e através disso será feita a classificação. As respostas do modelo foram categorizadas em positivo ou negativo para cada classificação e em verdadeiro ou falso, quando comparadas ao valor real de cada entregador. O modelo em questão leva em consideração a revocação (“recall”), priorizando a quantidade de verdadeiros positivos em detrimento de falsos negativos. Assim, as chances do modelo classificar um entregador incorretamente como negativo são minimizadas, respeitando a regra de negócio apresentada pelo cliente de que é melhor identificar o maior número de churns possível, ainda que alguns entregadores com pouca propensão de dar churn sejam incluídos nesse grupo.

Segue a fórmula de revocação:

$$Recall = \frac{True\ Positive(TP)}{True\ Positive(TP) + False\ Negative(FN)}$$

Fórmula recall (imagem 2)

4.1.4. Value Proposition Canvas



4.1.5. Matriz de Riscos

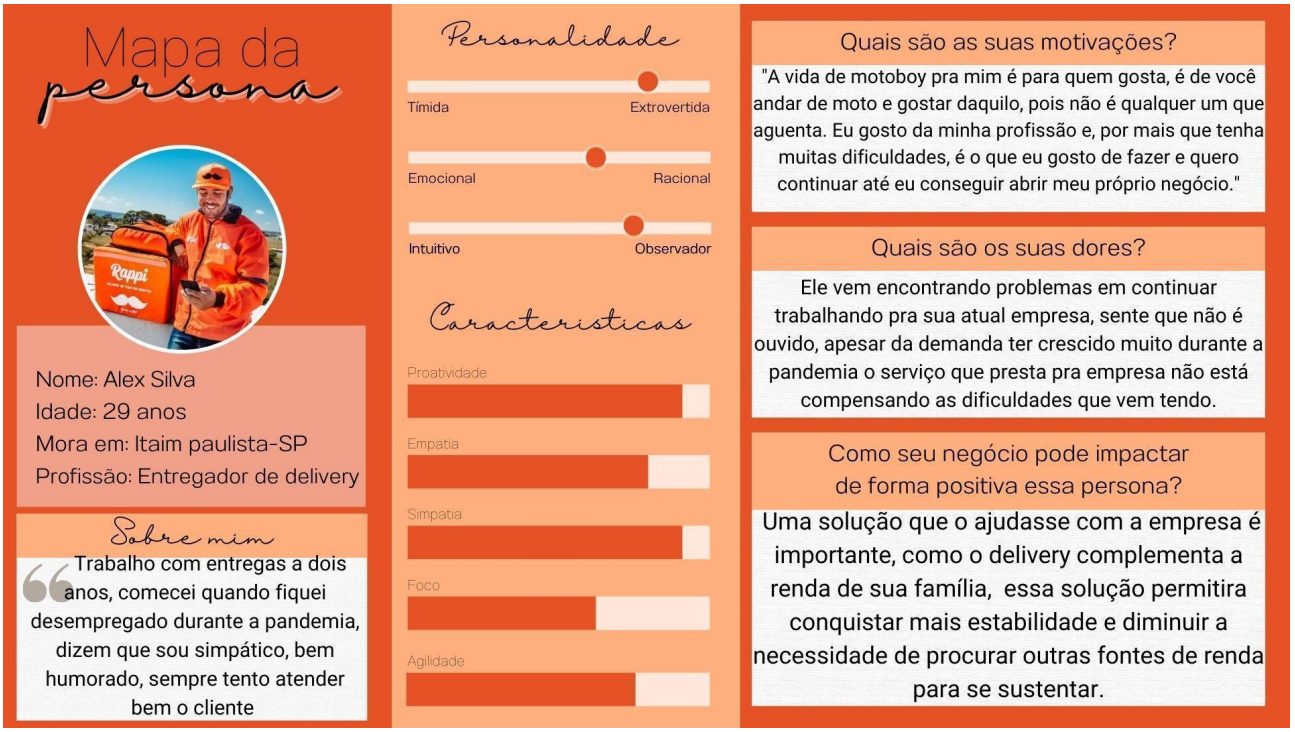
Probabilidade	Ameaças					Oportunidades				
90%										
70%		Não dar devida atenção à documentação		Falta de contato com os RT´s		Fazer a definição de variáveis cabíveis para o modelo	Boa difusão de tarefas			
50%			Individualização das tarefas	Baixa taxa de assertividade do modelo	Errar na análise primária dos dados	Terminar o treino do modelo antes do prazo estipulado	Ter contato direto e próximo com o parceiro			
30%				Escassez de informação importantes	Erros significativos na base de dados					
10%				Priorizar a quantidade de acertos em detrimento do número de situações contempladas						
	Muito Baixo	Baixo	Moderado	Alto	Muito Alto	Muito Alto	Alto	Moderado	Baixo	Muito Baixo
	Impacto									

Matriz de riscos (tabela 2)

Para tratar antecipadamente os riscos descritos na matriz acima, o grupo pretende tomar as seguintes medidas de contingência: dedicar maior tempo na revisão das entregas para que a documentação sempre esteja alinhada, sempre nomear um revisor da documentação, fazer sempre alinhamentos em grupo do que será feito para evitar a individualização das atividades, alinhar com orientador e com o cliente os possíveis problemas nas informações disponíveis para mitigar os possíveis erros na base de dados, rodar o crisp-dm quantas vezes for necessário para ter uma alta assertividade e refazer a análise dos dados caso seja necessário para não cometer erros.

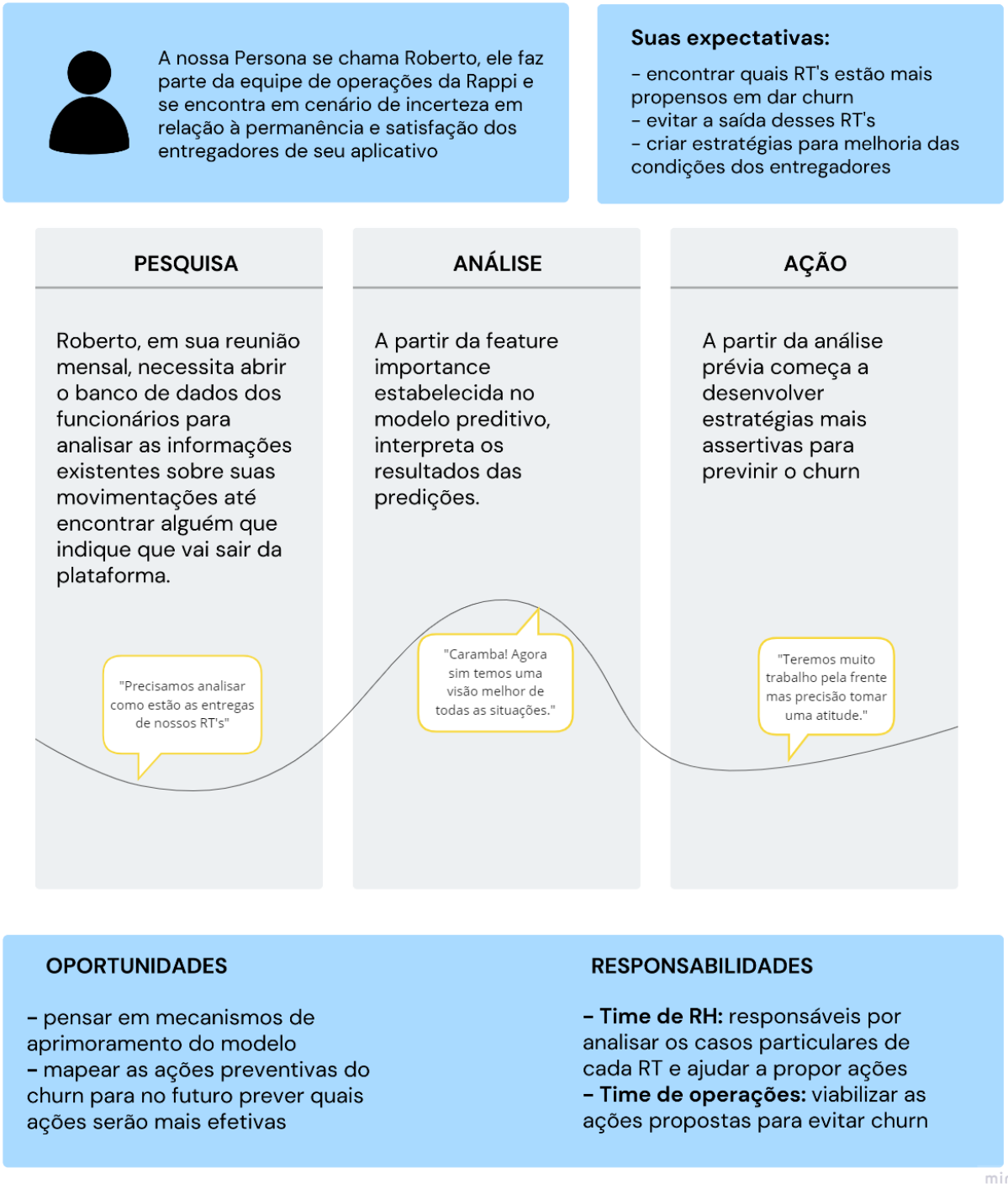
4.1.6. Personas

Primera persona (imagem 4)



Segunda persona (imagem 5)

4.1.7. Jornadas do Usuário



Jornada do usuário (imagem 6)

4.2. Compreensão dos Dados

Os dados que iremos utilizar para formulação do nosso modelo preditivo foram disponibilizados pelo cliente no formato **CSV,** todas as planilhas estão organizadas em linhas e colunas, com o intuito de gerar uma maior compreensão do mesmo. Ao total, foram apresentadas 10 bases contendo diversas informações:

- Taxa de pedidos aceitos pelo entregador (identificado pelo seu ID);
  - presente na base attendance_rate.csv (7MB) - 566.099 linhas e 2 colunas
- A receita de cada entregador (nela consta os ganhos e gorjetas);
  - presente na base earnings.csv (22 MB) - 566.099 linhas e 4 colunas
- Os pedidos que foram concluídos e cancelados ;
  - presente na base Ordens Done e Cancel.csv (9,1 MB) - 653.166 linhas e 4 colunas
- O tempo em horas que o RT fica/ficou conectado em um certo período;
  - presente na base supply.csv (10,1 MB) - 124.526 linhas e 10 colunas
- As informações gerais de cada entregador (nome, cidade, data de nascimento, etc);
  - presente na base infos gerais.csv (29,9 MB) - 180.178 linhas e 25 colunas
- O retorno de um produto uma vez que a ordem foi cancelada;
  - presente na base Product return.csv (4,6 MB) - 41.535 linhas e 11 colunas
- Quanto tempo o entregador fica esperando o suporte;
  - presente na base Tempo de Resolução e Modal.csv (2,63 GB) - 32.000.000 linhas e 9 colunas
- Regras para incidentes;
  - presente na base Incidentes_Regras RT.csv (234,2 MB) - 2.405.601 linhas e 9 colunas
- Todos os casos em que o usuário abre uma reclamação sobre algum pedido;
  - presente na base comp defects.csv (388,9 MB) - 6.783.958 linhas e 10 colunas
- Todos os entregadores que já deram churn;
  - presente na base criacao contas churn.csv (2,63 GB) - 32.000.000 linhas e 9 colunas

Colunas de cada CSV:

- attendance_rate.csv
  - storekeper_id: ID do entregador.
  - ACCEPTANCE_RATE: Porcentagem de pedidos aceitos pelo entregador em relação ao total de pedidos recebidos.

- earnings.csv
  - STOREKEEPER_ID: ID do entregador.
  - MONTH: Primeiro dia do mês, em formato ISO-8601.
  - EARNINGS: Receita do entregador.

        - TIPS: Gorjetas do entregador.

    - Ordens Done e Cancel.csv
        - STOREKEEPER_ID: ID do entregador.
        - ORDERS_DONE: Número de pedidos realizado.
    - ORDERS_CANCEL: Ordens totais canceladas pelo entregador.
    - CANCEL_OPS_RT: Ordens canceladas manualmente pelo time de operação.

- supply.csv
    - STOREKEEPER_ID: ID do entregador.
    - CITY:Cidade de atuação.
    - DATE:Dia da semana (YYYY-MM-DD).
    - WEEK:Segunda-feira da semana em referência (YYYY-MM-DD).
    - CREATED_CARD: Data de Criação do Cartão.
    - LEVEL_NAME_2: Nível do entregador.
    - HAVE_CARD: cartão.
    - TRANSPORT_MEDIA_TYPE: Modal de transporte.
    - NUM_ORDERS: Número de pedidos atendidos pelo entregador naquele dia.
    - SUPPLY_HOURS: Horas em que o entregador ficou conectado no período.

- infos gerais.csv
    - ID: ID do entregador.
    - NOME: Nome do entregador.
    - SOBRENOME: Sobrenome do entregador.
    - GENERO: Genero do entregador.
    - DATA_NASCIMENTO: Data de nascimento do entregador.
    - CIDADE: Cidade de atuação.
    - IS_ACTIVE: Se está ativo na plataforma.
    - TRANSPORTE: Modal de transporte.
    - AUTO ACEITA: Se aceita os pedidos automaticamente.
    - COUNT_ORDERS_LAST_7D: Ordens realizadas nos últimos 7 dias.
    - COUNT_ORDERS_LAST_30D: Ordens realizadas nos últimos 30 dias.
    - COUNT_ORDERS_CANCELED_LAST_7D: Ordens canceladas nos últimos 7 dias.
    - COUNT_ORDERS_CANCELED_LAST_30D: Ordens canceladas nos últimos 30 dias.
    - GORJETA: Gorjetas recebidas pelo entregador.
    - PRIMEIRO_PEDIDO: Data do primeiro pedido entregue.
    - ULTIMO_PEDIDO: Data do último pedido entregue.
    - COUNT_ORDERS_RESTAURANTES: Ordens de restaurantes.
    - COUNT_ORDERS_MERCADO: Ordens de mercado.
    - COUNT_ORDERS_FARMACIA: Ordens de farmacia.
    - COUNT_ORDERS_EXPRESS: Ordens de turbo.
    - COUNT_ORDERS_ECOMERCE: Ordens de e-commerce.
    - COUNT_ORDERS_ANTOJO: Ordens de lojas que não estão no aplicativo.
    - FRETE_MEDIO: Média do tempo gasto nos pedidos.
    - COOKING_TIME_MEDIO: Média de tempo esperando o pedido ficar pronto.
    - ITENS_MEDIO: Média de itens nas entregas.

- Product return.csv
    - ID_ENTREGADOR: ID do entregador.
    - LEVEL_NAME: Nível do entregador
    - MODAL :Meio de locomoção
    - CITY:Cidade de atuação
    - CREATED_AT:Data de cadastro
    - ORDER_ID:ID do pedido
    - PRODUCT_RETURNS: Valor da devolução (negativo)
    - VERTICAL_SUB_GROUP: Categoria do pedido (farmácia, mercado)
    - COUNT_TO_GMV:Abater do GMV

        - GMV:Gross Merchandise Value(valor total pago no pedido pela Rappi)
        - STORE_ID:ID da loja

    - Tempo de Resolução e Modal.csv
  - STOREKEEPER_ID: ID do entregador

- Incidentes_Regras RT.csv
  - STOREKEEPER_ID: ID do entregador.
  - DATE: Data do Incidente (formato YY-MM-DD)
  - NAME: Nome do Incidente
  - INCIDENT_ID: ID do incidente
  - PUNISHMENT_MINUTES :Minutos de suspensão
  - PUNISHMENT_TYPE: Tipo de punição (permanent_block, temporary_block, warning)
  - DISCIPLINE_RULE_BUCKET: É uma derivada de um indicador interno da Rappi
  - CATEGORY_RULE: Gênero da regra aplicável
  - ORDER_ID: ID do pedido.

- comp defects.csv
  - STOREKEEPER_ID: ID do entregador.
  - WEEK:Segunda-feira da semana em referência (YYYY-MM-DD).
  - CITY:Cidade de atuação
  - LEVEL_NAME: Nome do nível do entregador.
  - ORDERS: Número de pedidos que aconteceu algum problema.
  - GMV_TOTAL:Total da transação (GMV = custo total pago pela Rappi para a loja).
  - COMPENSATIONS:Valor pago (devolvido) para o usuário
  - DEFECT_COMPENSATIONS: ID do pedido
  - DEFECT_ORDER: ID do pedido, a confirmar

- criacao contas churn.csv

  - ID: ID do entregador.
  - FIRST_NOME: Nome do entregador.
  - GENDER: Gênero do entregador.
  - CITY: Cidade de atuação.
  - SK.CREATED_AT::DATE: Data da criação da conta.
  - TRANSPORT_MEDIA_TYPE: Modal de transporte.
  - CARTAO: Tem cartão pré-pago.
  - LEVEL_NAME: Nível do entregador.
  - FECHA_ULT: Última interação no aplicativo.

Uma alternativa que encontramos para a mesclagem de todos esses dados que nos foram disponíveis foi a união de informações que possuem o mesmo identificador. Por exemplo, conseguimos mapear todas as informações disponíveis em relação a um entregador específico quando puxamos seu ID de todas as tabelas. Dessa forma, montaremos nossas pesquisas e análises para que possamos ter uma boa base para formação de nossas hipóteses.

Quando analisada a qualidade das bases de dados disponibilizadas pelo nosso cliente nos deparamos com muitas inconsistências, informações desorganizadas e sem tipificação. Como também muitos dados que fogem totalmente do padrão o que causa uma certa dificuldade na análise e estudo destes.

No quesito cobertura/diversidade dos dados, pudemos notar a grande quantidade de informações presentes em diversas tabelas. Essa gama de informações serão analisadas e interpretadas com muita cautela para que não haja conflito no momento da mesclagem.

Especificamente na questão de acesso, a Rappi se prontificou em proporcionar o máximo de informações possíveis, porém, em decorrência de regras internas da empresa, dados que poderiam contribuir para a formulação de hipóteses e desenvolvimento do modelo não serão fornecidos.

Quanto às restrições de segurança tomadas em relação ao projeto, tivemos a cautela de não publicar ou compartilhar os dados sensíveis e pessoais referentes à Rappi, priorizamos por uma conduta ética que esteja de acordo com a LGPD.

Toda a nossa descrição estatística dos dados foi feita na plataforma Google Colab, onde nós importamos todas as bases e começamos a analisar e identificar quais seriam os atributos que mais serviram para modelagem de nosso sistema.

Nesta análise inicial, feita por nosso grupo, esmiuçamos cada tabela para trazer todos os dados quantitativos que possam ser úteis para nosso sistema. Também trouxemos representações gráficas que embasam nossas primeiras hipóteses, são elas:

1. A visualização da distribuição de rendimentos dos couriers e ex-outliers para nos contextualizarmos sobre a situação de ganhos médios mensais dos entregadores que nos traz uma contextualização da renda dos entregadores. (disponível na seção "Data Frame Earnings" do Notebook.)
2. Visualização da distribuição de rendimentos no sweet spot que nos disponibiliza uma visualização mais real do que seriam os verdadeiros ganhos mensais de um entregador, ou seja, foram retirados os pontos fora da curva (outliers). Essa visualização pode ser encontrada também na seção "Data Frame Earnings".
3. Uma visão da correlação alta entre o nível de renda e ganhos com gorjetas para que possamos entender se uma influencia a outra, e, se isso é um causador de "churn" do entregador. Também disponível na seção "Data Frame Earnings" do Notebook.

Definição do target:

O target definido é uma classe. Durante a análise de dados, percebemos a existência de três ocorrências:O entregador que sai definitivamente da plataforma; O entregador que é inconsistente, ou seja, sai e retorna à plataforma com frequência; O entregador que não saiu da plataforma no período analisado, sendo que este representa a base de entregadores com a qual a Rappi conta atualmente para continuar operando. Portanto, para que o nosso modelo se encaixasse melhor ao contexto analisado, separamos as 3 possibilidades para o target, assim, seguem as três categorias de RT's definidas:

True: entregadores que nunca deram churn;

Quasi: inconsistentes;

False: entregadores que deram churn definitivo.

4.3. Preparação dos Dados

Feature Tempo de Resolução Modal:

Manipulações:

Para a construção da feature Tempo de Resolução Modal, denotada pelo dataframe "df_resTime" no Google Collaboratory (Collab) do grupo, foi utilizada a planilha tempo-resolução-modal-def. A manipulação utilizada para iniciar a tratativa dos dados foi a exclusão de das colunas pouco relevantes para o desenvolvimento da feature, tais como: Cldade(CITY), Hora de envio (SENT_HOUR), Horário de resposta (RESPONSE_AT), Meio de transporte (TRANSPORT_MEDIA_TYPE), Tempo de resposta (RESPONSE_TIME) e Categorização do tempo de resposta (RESOLUTION_TIME_BUCKET).

```
df_resTime = df_resTime.drop(columns=[ 'SENT_DATA', 'SENT_HOUR', 'RESPONSE_AT',
                                       'CITY' ,'TRANSPORT_MEDIA_TYPE',
                                       'RESPONSE_TIME', 'RESOLUTION_TIME_BUCKET'])
```

O nome da coluna "STOREKEEPER_ID" foi alterada para "ID", de modo a facilitar a futura junção desta feature com a tabela consolidada (consol_feature) que será utilizada para iniciar as predições.

```
df_resTime.rename(columns={
    "STOREKEEPER_ID": "ID"
    }, inplace=True)
```

Para facilitar a análise futura, os valores da coluna Tempo de resolução (Resolution_time) foram alterados de sua forma padrão em segundos para horas, e após convertidos esses valores são formatados para apresentarem apenas duas casas decimais.

```
df_resTime['RESOLUTION_TIME'] = df_resTime['RESOLUTION_TIME'].div(3600)
pd.options.display.float_format = '{:.2f}'.format
```

Para remoção dos itens duplicados na coluna TICKET_ID, foi utilizado um método de sort nos valores da coluna Resolution_time que elencou todos os valores em ordem decrescente e selecionou, para um mesmo ticket, apenas o maior valor cabível, pois este representava o tempo total que o ticket ficou aberto na plataforma.

```
df_resTime = df_resTime.sort_values('RESOLUTION_TIME', ascending=False).drop_duplicates(
    'TICKET_ID').sort_index()
```

Derivação de novos atributos:

Para quantificar o tempo total que o entregador(RT) "perdeu" com tickets aberto na plataforma da Rappi, foi criada uma nova tabela denominada "temp_sum", sendo que esta utiliza o método sum do python para fazer o somatório de todos os itens e método "groupBy" da biblioteca Pandas que agrupa os ID's iguais e facilita o somatório. Além disto, o nome da coluna desta nova tabela, a qual recebe o nome da coluna utilizada anteriormente "RESOLUTION_TIME" é alterado para "RES_TIME_TOTAL" de modo categorizar a informação que está feature trará.

```
temp_sum = df_resTime.groupby('ID').sum()
temp_sum.rename(columns={'RESOLUTION_TIME': 'RES_TIME_TOTAL'}, inplace = True)
```

Para quantificar o tempo médio dos tickets do entregador(RT) abertos na plataforma da Rappi, foi criada uma nova tabela denominada "temp_mean", sendo que esta utiliza o método mean do python para fazer a média de todos os itens e método "groupBy" da biblioteca Pandas que agrupa os ID's iguais e facilita o processo de cálculo da média para cada ID. Além disto, o nome da coluna desta nova tabela, a qual recebe o nome da coluna utilizada anteriormente "RESOLUTION_TIME" é alterado para "RES_TIME_MEAN" de modo categorizar a informação que essa feature trará.

```
temp_mean = df_resTime.groupby('ID').mean()
temp_mean.rename(columns={'RESOLUTION_TIME': 'RES_TIME_MEAN'}, inplace = True)
```

Para quantificar o número de tickets abertos na plataforma da Rappi, foi criada uma nova coluna na feature "resTime" denominada "TOTAL_TICKETS", sendo que esta utiliza o método map do python para fazer a iteração entre os elementos da coluna "ID" de todos os entregadores e calcular sua frequência de aparecimento na mesma coluna e o método

"value_counts()" que faz a contagem de cada ID. Além disso, é utilizado um método "drop_duplicates()" da biblioteca pandas para remover os tickets duplicados, na coluna "TICKET_ID", e assim indexar a frequência de aparecimento de cada ID na nova coluna "TOTAL_TICKETS" .

```
df_resTime['TOTAL_TICKETS'] = df_resTime['ID'].map(df_resTime['ID'].value_counts())
df_resTime = df_resTime.drop(columns=['TICKET_ID', 'RESOLUTION_TIME']).drop_duplicates()
```

Após estas três manipulações, é feita uma mescla entre a tabela feature resTime, a tabela temp_sum e a tabela temp_mean através do método "pd.merge()" da biblioteca pandas através da verificação do ID, ou seja, as colunas "RES_TIME_MEAN" e "RES_TIME_TOTAL" só serão adicionadas à tabela da da feature resTime se houver o mesmo ID em ambas as tabelas. A seção dita ao final pode ser verificada no parâmetro "on = 'ID' ", no final do comando.

```
df_resTime_ = pd.merge(df_resTime, temp_sum, on= 'ID')
df_resTime_ = pd.merge(df_resTime_, temp_mean, on = 'ID')
```

Remoção de dados nulos:

As linhas com valores nulos nesta tabela são um dado irrisório que representa apenas 0,5% do total, desta forma sua exclusão é feita utilizando o método “dropna()” da biblioteca pandas. Esta manipulação ainda será analisada para verificar se não o modelo não se torna enviesado, contudo será mantido desta forma até a análise dos outputs resultantes da predição feita pelo algoritmo selecionado pelo grupo.

```
df_resTime = df_resolution_time.dropna()
```

Identificação da Feature selecionada:

Ao final, obtemos a seguinte tabela abaixo.

```
[31] df_resTime
```

	ID	LEVEL_NAME	TOTAL_TICKETS	RES_TIME_TOTAL	RES_TIME_MEAN
97824	1499730	silver	1	125.89	125.89
24138	695719	rookie	1	96.84	96.84
77090	1345703	rookie	1	48.03	48.03
77092	1405979	rookie	1	19.44	19.44
77095	617040	rookie	1	11.60	11.60
...	...	...	...	...	...
4820	1359488	diamond	230	6103.86	26.54
6000	1243322	bronze	232	10349.56	44.61
985	342502	bronze	267	42996.66	161.04
492	773351	bronze	275	64370.34	234.07
4497	1221039	bronze	328	143.63	0.44

97825 rows × 5 columns

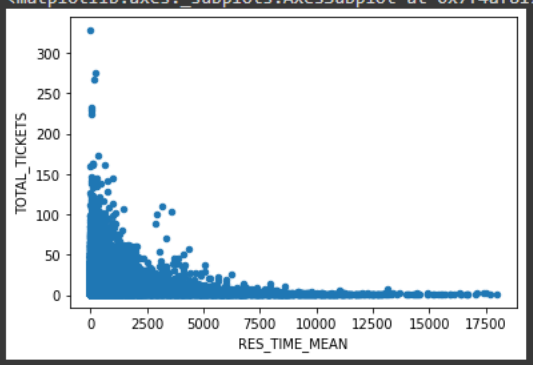
A feature têm relevância para o modelo preditivo, ao passo que relaciona a quantidade de tempo total que o entregador despende na plataforma quando necessita da resolução de algum problema, qualquer que seja, o número de vezes que este abriu chamados na plataforma no período de 1 (um) ano, o tempo médio de resposta para cada ticket e o nível do entregador.

Desta forma, pretendemos correlacionar a quantidade de horas perdidas com tickets abertos e probabilidade do entregador dar churn da plataforma.

No gráfico abaixo podemos verificar o número de tickets abertos em função da quantidade de hora média para resolução destes na plataforma, no formato de uma dispersão.

```
[34] df_fe_resolution_time.plot(x ='RES_TIME_MEAN', y='TOTAL_TICKETS', kind = 'scatter')
```

```
<matplotlib.axes._subplots.AxesSubplot at 0x7f4af8178390>
```



Features Incidentes e Regras:

Manipulações:

Para a construção das features relacionadas aos incidentes e regras dos RTs no notebook da equipe, a base “incidentes-regras-rt-def” foi utilizada. Primeiramente, dados pouco relevantes para o modelo final foram excluídos, como ID do pedido ('ORDER_ID'), ID do incidente ('INCIDENT_ID') e data ('DATE'). As colunas 'DISCIPLINE_RULE_BUCKET' e 'NAME' não foram adotadas também, pois não possuem um padrão claro e não fariam muita diferença na análise final.

```
[ ] df_incidentes = df_incidentes_o.drop(columns=['ORDER_ID', 'INCIDENT_ID', 'DATE', 'DISCIPLINE_RULE_BUCKET', 'NAME'])
```

Depois disso, os dados de tipo de punição (“PUNISHMENT_TYPE”) e de categoria de regra infringida nos incidentes (“CATEGORY_RULE”) foram transformados em dados “dummy”. Assim, essas informações passaram a ser tratadas como 1 ou 0 (verdadeiro ou falso), para que o computador entenda como foi a recorrência desses dados.

```
[ ] df_incidentes = pd.get_dummies(df_incidentes, columns=['PUNISHMENT_TYPE'])
```

```
[ ] df_incidentes = pd.get_dummies(df_incidentes, columns=['CATEGORY_RULE'])
```

Com essas operações a coluna “PUNISHMENT_TYPE” se transformou em três novas colunas (possíveis valores: warning, temporary block e permanent block) e a coluna “CATEGORY_RULE”

se transformou em sete novas colunas (possíveis valores: covid, other, discipline, fraud, manual, performance e warning).

Analisando a quantidade de valores em cada item criado, as categorias "COVID" e "OTHER" foram excluídas da análise, pois apresentam números desprezíveis quando comparados ao total de outras categorias e ao tamanho da base consolidada.

```
[ ]  df_incidentes[['CATEGORY_RULE_Covid', 'CATEGORY_RULE_Other',
                    'CATEGORY_RULE_Discipline', 'CATEGORY_RULE_Fraud',
                    'CATEGORY_RULE_Manual', 'CATEGORY_RULE_Performance',
                    'CATEGORY_RULE_Warning']].sum()

    CATEGORY_RULE_Covid              476
    CATEGORY_RULE_Other                4
    CATEGORY_RULE_Discipline     2217495
    CATEGORY_RULE_Fraud            71962
    CATEGORY_RULE_Manual           27200
    CATEGORY_RULE_Performance       2902
    CATEGORY_RULE_Warning          85562
    dtype: int64

[ ]  df_incidentes = df_incidentes.drop(columns=['CATEGORY_RULE_Covid', 'CATEGORY_RULE_Other'])
```

A variável "STOREKEEPER_ID" teve seu nome alterado para "ID", por motivos de compatibilidade na hora de juntar as bases com a base consolidada. As outras colunas também tiveram seus nomes alterados, para que a leitura desses dados fique mais fácil.

```
[ ]  df_incidentes = df_incidentes.rename(columns={'STOREKEEPER_ID': 'ID',
                                                   'PUNISHMENT_TYPE_permanent_block': 'PERMANENT_BLOCK',
                                                   'PUNISHMENT_TYPE_temporary_block': 'TEMPORARY_BLOCKS',
                                                   'PUNISHMENT_TYPE_warning': 'WARNINGS',
                                                   'CATEGORY_RULE_Discipline' : 'DISCIPLINE_INCIDENTS',
                                                   'CATEGORY_RULE_Fraud' : 'FRAUD_INCIDENTS',
                                                   'CATEGORY_RULE_Manual' : 'MANUAL_INCIDENTS',
                                                   'CATEGORY_RULE_Performance' : 'PERFORMANCE_INCIDENTS',
                                                   'CATEGORY_RULE_Warning' : 'WARNING_INCIDENTS'})
```

Derivação de novos atributos:

Para ter a quantidade de cada uma das variáveis acima por entregador, foram realizadas as operações de agrupar por ID e somar os valores de cada coluna. Assim, pode-se visualizar o número de minutos de punição, bloqueios (temporários e permanentes), advertências e regras envolvidas em cada incidente registrado de cada RT.

```
[ ]  df_incidentes = df_incidentes.groupby(by=['ID']).sum().reset_index()
```

Identificação da Feature selecionada:

Ao final, a tabela das features relacionadas aos incidentes e às regras dos RT's ficou da seguinte forma:

```
[ ] df_incidentes
```

	ID	PUNISHMENT_MINUTES	PERMANENT_BLOCK	TEMPORARY_BLOCKS	WARNINGS	DISCIPLINE_INCIDENTS	FRAUD_INCIDENTS	MANUAL_INCIDENTS	PERFORMANCE_INCIDENTS	WARNING_INCIDENTS
0	32799	7200	0.0	5.0	0.0	0.0	5.0	0.0	0.0	0.0
1	32825	0	0.0	0.0	1.0	0.0	0.0	0.0	0.0	1.0
2	32924	0	0.0	0.0	1.0	0.0	0.0	0.0	0.0	1.0
3	32932	360	0.0	1.0	0.0	1.0	0.0	0.0	0.0	0.0
4	33027	21600000	1.0	0.0	0.0	1.0	0.0	0.0	0.0	0.0
...	...	...	...	...	...	...	...	...	...	...
185915	1558561	15	0.0	1.0	0.0	1.0	0.0	0.0	0.0	0.0
185916	1558691	15	0.0	1.0	0.0	1.0	0.0	0.0	0.0	0.0
185917	1558770	15	0.0	1.0	0.0	1.0	0.0	0.0	0.0	0.0
185918	1558815	15	0.0	1.0	0.0	1.0	0.0	0.0	0.0	0.0
185919	1558936	15	0.0	1.0	0.0	1.0	0.0	0.0	0.0	0.0

185920 rows × 10 columns

Essa feature é de extrema relevância para o modelo, visto que, uma das principais reclamações feitas pelos entregadores é de que as punições são feitas de maneira “indevidas”, gerando penalizações no aplicativo ou até mesmo no impedimento do trabalho dos couriers.

Feature Supply:

Manipulações:

A base “SUPPLY-DEF” traz informações sobre as entregas, mais especificamente quanto pedidos entregues (“NUM_ORDERS”) e o número de horas ativas no aplicativo de um determinado RT (“SUPPLY_HOURS”). Esses dados foram coletados ao longo de cinco semanas, e cada linha relaciona os dados previamente apresentados com sua respectiva semana e o ID do entregador.

Primeiramente, dados desnecessários para a criação da feature foram retirados do csv. Essas informações já estão em outras bases. As informações de data e semana não são relevantes para a predição de churn.

```
[ ] df_supply = df_supply_o.drop(columns=['CITY', 'DATE', 'WEEK', 'CREATED_CARD',
                                          'LEVEL_NAME_2', 'HAVE_CARD', 'TRANSPORT_MEDIA_TYPE'])
```

Em seguida, os dados foram agrupados por entregador e as informações de pedidos entregues e o número de horas ativas no aplicativo foram somadas de acordo com o “STOREKEEPER_ID”. Assim, conseguimos ter os números totais dessas informações, independente da semana na qual os dados foram registrados.

```
[ ] df_supply = df_supply.groupby(['STOREKEEPER_ID']).sum().reset_index()
```

Derivação de novos atributos:

A feature consiste na média de pedidos entregues por hora ativa no aplicativo. Nesse sentido, pode-se ter ideia da produtividade média de cada entregador. Para isso, uma nova

coluna (“ORDERS_PER_SUPPLY_HOURS”) foi criada, contendo os valores da coluna “NUM_ORDERS” divididos pelos valores na coluna “SUPPLY_HOURS”.

```
[ ]  df_supply['ORDERS_PER_SUPPLY_HOURS'] = df_supply['NUM_ORDERS'] / df_supply['SUPPLY_HOURS']
```

Depois disso, os dados brutos de número de pedidos e de horas no app foram excluídos, já que a feature já está desenvolvida e, então, eles se tornaram desnecessários para a implantação na base consolidada final.

```
[ ]  df_supply = df_supply.drop(columns=['NUM_ORDERS', 'SUPPLY_HOURS'])
```

Por fim, a variável “STOREKEEPER_ID” foi renomeada para “ID”, buscando uma maior compatibilidade de chaves de merge com a base consolidada.

```
[ ]  df_supply = df_supply.rename(columns={'STOREKEEPER_ID': 'ID'})
```

Identificação da Feature selecionada:

Ao final, a tabela da feature de supply dos RTs ficou da seguinte maneira:

```
[ ]  df_supply
```

	ID	ORDERS_PER_SUPPLY_HOURS
0	33203	0.614341
1	33355	0.684830
2	33564	1.284929
3	33737	1.574782
4	34420	1.678784
...	...	...
6555	1538876	0.482228
6556	1539013	0.000000
6557	1539017	0.729122
6558	1539162	0.386199
6559	1539478	0.337753

6560 rows × 2 columns

Essa feature é muito importante para o modelo, visto que ela evidencia a produtividade do entregador. Está relacionada com um dos principais problemas da Rappi: a baixa demanda de pedidos, fazendo com que os entregadores fiquem sem trabalhar, ainda que estejam disponíveis no aplicativo.

Como evidenciado na análise exploratória, esses dados têm grande impacto nas taxas de earnings (ganhos) e de churn no aplicativo. Abaixo, estão exemplos de dois gráficos gerados que evidenciam essa relação. O primeiro mostra que existem entregadores que passam muitas horas ativos no app, mas que não entregam de maneira compatível. Já o segundo mostra como isso é refletido nos ganhos mensais desses entregadores.

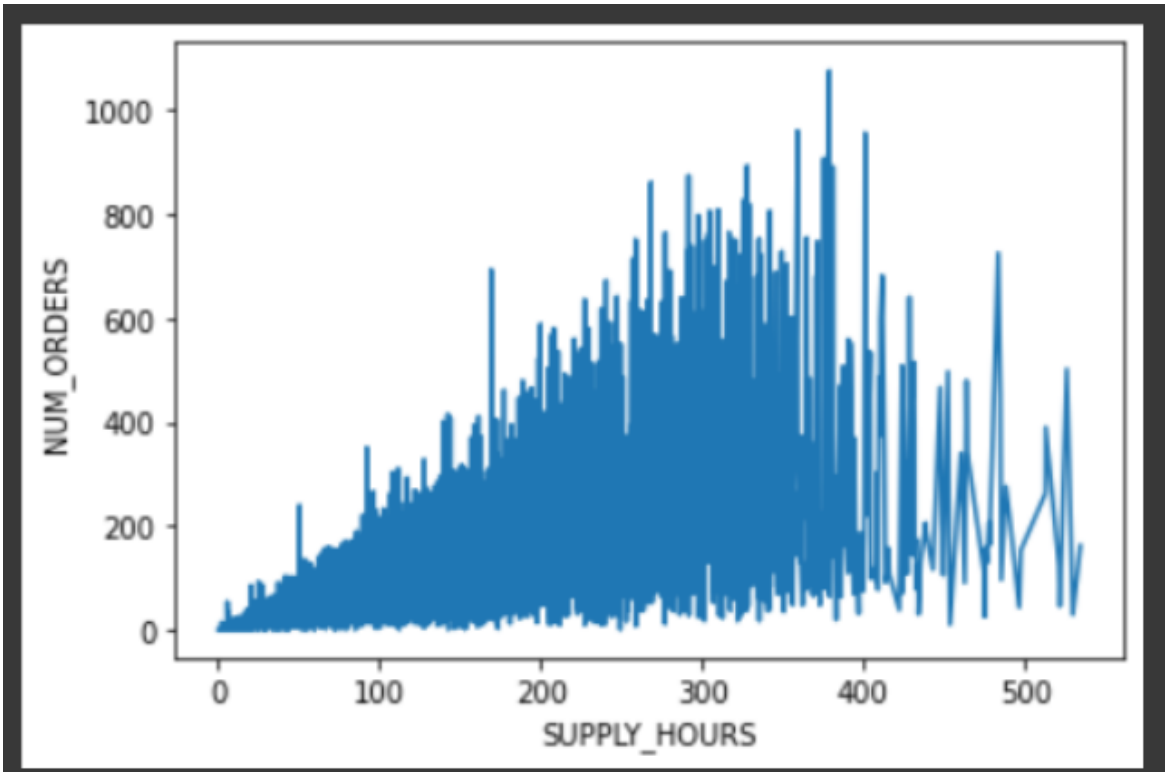


Gráfico quantidade de pedidos x quantidade de horas ativas no app.

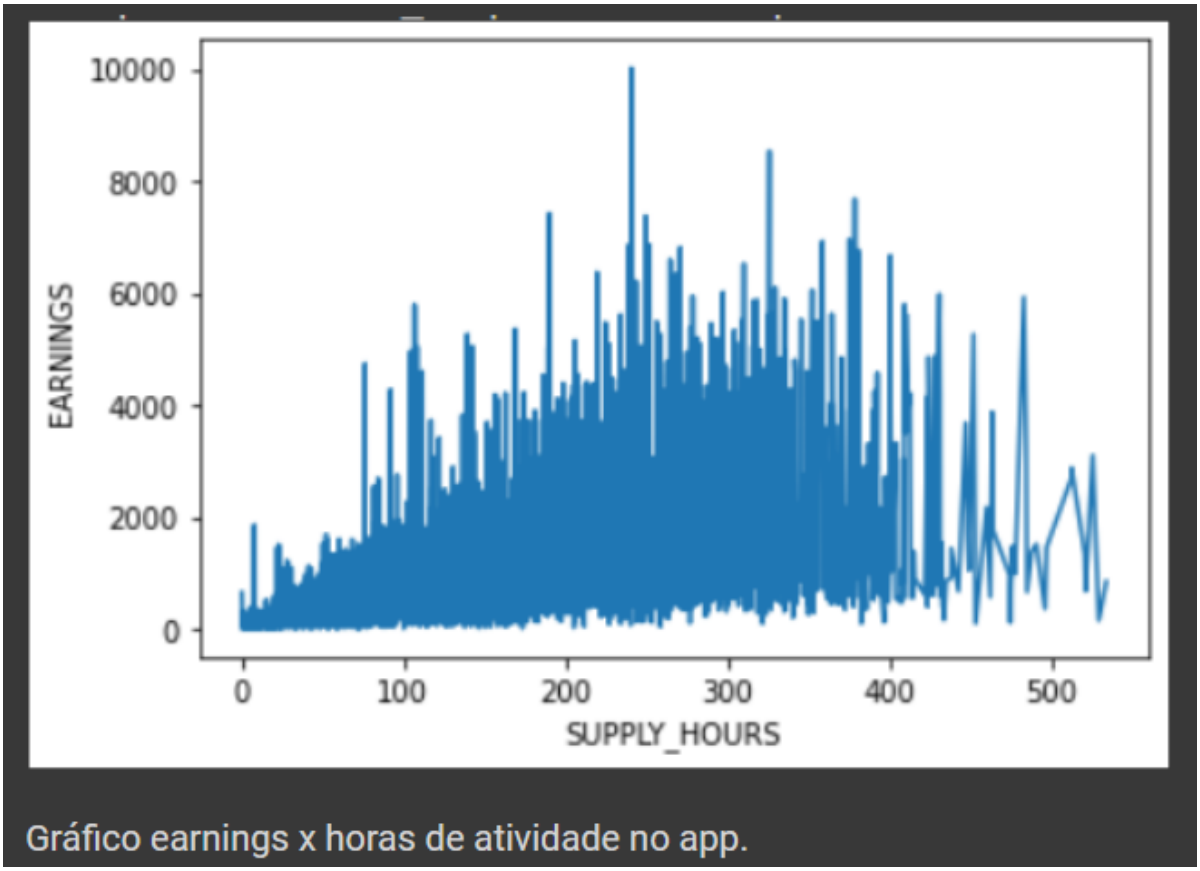


Gráfico earnings x horas de atividade no app.

Um problema dessa feature é que os dados de supply dos entregadores fornecidos pela Rappi, apesar de muito importantes para o modelo, contempla apenas 6560 couriers, um número muito pequeno se comparado com a quantidade de IDs em tabelas como "INFOS-GERAIS".

Feature Products Return

Manipulações:

Para estruturação da feature Products Return utilizamos a base "product-return.csv", disponibilizada pelo nosso parceiro. Com o fim de deixá-la mais funcional e simplificada, começamos nossa feature retirando todas as colunas que não seriam muito úteis para o nosso modelo preditivo, são elas: Modo de locomoção do RT ("MODAL'), Cidade ("CITY"), ID da loja ("STORE_ID"), Contagem do GMV("COUNT_TO_GMV"), Valor total do GMV("GMV"), Data de criação ("CREATED_AT"), a categoria do RT ("LEVEL_NAME") e a categoria da entrega ("VERTICAL_SUB_GROUP").

```
# ----------------------------------------
# dropa as colunas que não serão utilizadas
# renomeia a coluna de ID para possibilitar o merge com df_consol
# calcula media e soma de pedido de retorno de produto por entregador
# ----------------------------------------
df_productReturn.drop([
    'MODAL',
    'CITY',
    'STORE_ID',
    'COUNT_TO_GMV',
    'GMV',
    'CREATED_AT',
    'LEVEL_NAME',
    'VERTICAL_SUB_GROUP' ], axis=1, inplace=True)
```

Depois disso, renomeamos a coluna "ID_ENTREGADOR" para "ID" com o objetivo de ser possível a junção desta tabela com a tabela final que nomeamos de "consol_feature".

```
df_productReturn.rename(columns={"ID_ENTREGADOR": "ID"}, inplace = True)
```

Derivação de novos atributos:

Pelo fato desta base que estamos trabalhando estar separada apenas por pedidos e não por entregadores, tivemos que fazer um trabalho de agrupamento de todos os pedidos de cada entregador. Juntamente com isso, fizemos uma média do valor total de todos os pedidos retornados de cada entregador, por exemplo: se um RT x teve 3 pedidos retornados, nós

somamos o valor total desses pedidos e dividimos por três para obter uma média para esse courier. Assim, podemos ter uma visão mais ampla de cada entregador. Para isso, nós utilizamos um comando chamado ".groupby" que foi extraído da biblioteca "pandas".

```
df_temp_mean = df_productReturn.groupby(['ID']).mean().drop(['ORDER_ID'], axis= 1)
df_temp_count = df_productReturn.groupby(['ID']).count().drop(['PRODUCT_RETURNS'], axis=1)
```

Como, com a mudança que fizemos com o valores, a coluna de "ORDER_ID" já não fazia mais menção a cada pedido e sim a soma de todos eles, renomeamos essa coluna para "Nº de pedidos". Após isso juntamos todas as informações em uma única tabela final que se chama "df_productReturn".

```
df_productReturn_ = pd.merge(df_temp_mean, df_temp_count, on = 'ID').rename(columns={"ORDER_ID": "Nº_PEDIDOS"})
```

Identificação da Feature selecionada:

Ao final, obtemos a seguinte tabela abaixo.

ID	Nº_PEDIDOS	PRODUCT_RETURNS
33051	1	-0.025
33161	1	-72.444
33189	1	-2.145
33213	1	-24.718
33271	1	-16.337
...	...	...
1539018	1	-41.750
1539433	1	-89.424
1539461	1	-100.436
1540050	1	-20.625
1541264	1	-26.630

25719 rows × 2 columns

O objetivo dessa feature é entender o quanto os produtos retornados podem interferir na decisão do entregador de dar chun ou não a partir da comparação do valor ou da quantidade de pedidos de cada entregador. O problema que encontramos ao analisar essas informações foi que só encontramos 25.719 entregadores que passaram por essa situação e esse número é muito baixo comparado com a quantidade de IDs que estamos tratando na tabela consolidada final “Iconsol-feature-eng” que são, no total, 159.576.

Feature Attendance Rate

Manipulações:

Revisar a descrição dessa feature

Analisando as bases nós recolhemos as informações da tabela “attendance-rate-def.csv que denota a taxa de aceitação pelo id de cada entregador, o objetivo dessa feature é obter a porcentagem da taxa de aceitação dos pedidos e relacionar com a saída dos entregadores. Como essa tabela possui apenas duas colunas, nós iniciamos as manipulações tratando os dados, os ID 's que pareciam vazios foram zerados e a taxa foi convertida em porcentagem.

Começamos a converter os dados armazenando a coluna “ACCEPTANCE_RATE” dentro de um novo "data frame", com a função “round” diminuímos as casas decimais, logo após multiplicamos a taxa por 100 para obter uma porcentagem.

```
df_porncentagem_attendance_rate = round((df_attendance_rate ['ACCEPTANCE_RATE'] )*100)
```

Para realocar a coluna no data frame original nós inserimos o data frame criado agora como coluna, com o nome de “PERCENTAGEM_ACCEPTANCE_RATE” .

```
df_porcentagem = df_porncentagem_attendance_rate .fillna(0)
df_attendance_rate["PERCENTAGEM_ACCEPTANCE_RATE "] =  df_porcentagem
df_attendance_rate
```

Com a função drop excluímos a coluna da antiga taxa:

```
df_attendance_rate.drop(columns=['ACCEPTANCE_RATE'])
```

Por fim, renomeamos a coluna “STOREKEEPER_ID” para “ID” para juntar com a tabela final.

```
df_attendance_rate.rename(columns ={'STOREKEEPER_ID': 'ID'} )
```

Substituição de dados nulos:

Na tabela aparecem alguns dados nulos ou vazios, como isso pode indicar possíveis saídas nós preenchemos os dados com zero. A função "fillna" substitui os valores NaN que estão no dataframe por algum valor determinado, nesse caso o zero.

```
df_porcentagem = df_porncentagem_attendance_rate .fillna(0)
df_porcentagem
```

Identificação da Feature selecionada:

Após todas as alterações teremos a seguinte feature:

	ID	PERCENTAGEM_ACCEPTANCE_RATE
0	907442.0	67.0
1	1393441.0	56.0
2	1061798.0	78.0
3	103460.0	16.0
4	1259998.0	86.0
...	...	...
653162	57801.0	0.0
653163	911388.0	0.0
653164	537609.0	0.0
653165	491467.0	0.0
653166	706668.0	0.0

653167 rows × 2 columns

Essa Feature indica a porcentagem de aceitação que se refere a quantos pedidos eles estão aceitando e fazendo, ela é de grande importância pois pode indicar quando os RT's estão deixando de fazer entregas e assim deixando a Rappi, no nosso modelo preditivo essa informação pode ajudar a classificar quais tendem a sair, assim ela se torna decisiva e essencial.

Feature Earnings

Manipulações:

Para a construção das features relacionadas aos ganhos dos RTs no notebook da equipe foi utilizada a base "`earnings-def.csv`".

A base foi armazenada na variável df_earnings_D usando a biblioteca pandas no python.

```
df_earnings_D = pd.read_csv("/content/drive/Shareddrives/grupo4-rappi-hour/bases-rappi/earnings-def.csv")
```

	MONTH	STOREKEEPER_ID	EARNINGS	TIPS
0	2021-07-01T00:00:00Z	33161	1047.82	316.0
1	2021-07-01T00:00:00Z	33189	916.58	304.0
2	2021-07-01T00:00:00Z	33194	3076.38	552.0
3	2021-07-01T00:00:00Z	33207	160.42	44.0
4	2021-07-01T00:00:00Z	33213	268.63	59.0

Com auxílio de um laço de repetição a coluna “MONTH” foi formatada de texto para um valor numérico somente com os meses no intuito de possibilitar a relação churn com ganhos por período.

```
MONTH = df_earnings_D["MONTH"]
mes = []
for m in MONTH:
  a = m.split("-")
  mes.append(int(a[1]))


df_earnings_D["MONTH"] = mes

df_earnings_D["MONTH"].value_counts()
df_earnings_D.head()
```

Após a formatação da base as features escolhidas foram a **`EARNINGS`** e **`TIPS`**, pensando nas demais foi concluído que os meses não é um dado importante para se importar para o modelo, visto que não há uma maneira de distribuir-los de maneira não aleatória para os demais Rts’s que não estavam presentes na base “`earnings-def.csv`” e que constam no modelo final.

Por fim, a coluna "**STOREKEEPER_ID**" foi renomeada para "**ID**" no intuito de cumprir com o combinado da equipe em relação a nomenclatura utilizadas. Em seguida houve o cálculo da média de todos os ID's para não haver recorrência de ID.

```
df_earnings.rename(columns={
    "STOREKEEPER_ID": "ID"
    }, inplace=True)

df_earnings = df_earnings.groupby("ID").mean()
```

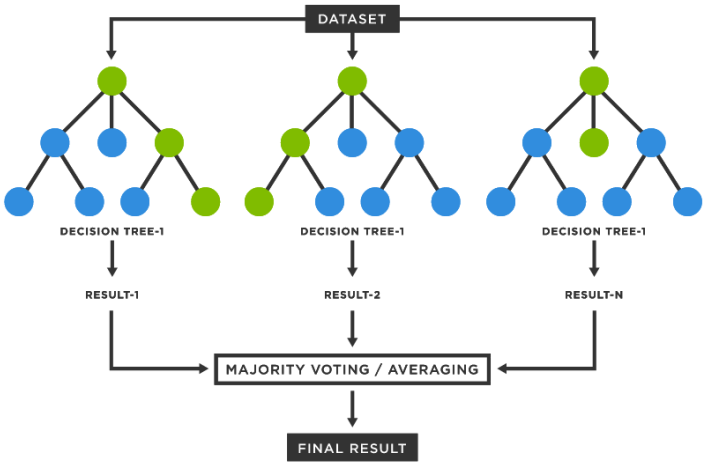
Essa feature é de extrema relevância para o modelo, visto que, a aderência dos entregadores no app é baseada em seus ganhos, outro ponto é que notamos ao cruzar as tabelas "Supply" com "Earnings" que a taxa de horas de trabalho dos couriers não são condizentes com seus ganhos. Assim acredito que os Rt's que não foram compensados de maneira simétrica com as suas horas de atividade têm mais chances de dar churn.

4.4 Modelagem

Os seguintes modelos foram escolhidos para fazer o aprendizado de máquina mediante o tratamento de dados realizados anteriormente:

Random Forest

O Random Forest ou "Floresta de decisão" é um método de aprendizagem "ensemble", usado para classificações, regressões e outras tarefas, que opera construindo uma infinidade de árvores de decisão (decision tree) cada uma com os mesmos nós, mas usando dados diferentes que levam a folhas diferentes. Ele mescla as decisões de várias árvores de decisão para encontrar uma resposta, que representa a média de todas essas árvores de decisão.



Modelo Random forest (imagem 7)

A razão pela qual esse modelo funciona tão bem é que as árvores se protegem de seus erros individuais (desde que não errem constantemente na mesma direção). Enquanto algumas árvores podem estar erradas, muitas outras estarão certas, então, como um grupo, as árvores podem se mover na direção correta.

Como forma de aplicação desse modelo, o importamos da biblioteca “sklearn” do pyhton e, então, determinamos o nosso x e y de treino assim como representado na imagem abaixo:

```
from sklearn.ensemble import RandomForestClassifier

# Instaciação do obj Algoritmo
ranfor = RandomForestClassifier()
# Treino # x = Features, y = Label/Target
ranfor.fit( x_train, y_train ) # squeeze() -> df para series

RandomForestClassifier()
```

Além disso, utilizamos hiperparâmetros para o refinamento dos resultados do nosso modelo. Esses hiperparâmetros servem como moldes que determinam como e até que ponto os dados devem ser analisados. Recorremos a dois tipos de ferramentas que automatizam o processo de ajuste dos parâmetros: GridSearch e RandomSearch.

GridSearch:

De maneira sistemática, faz diversas combinações dos parâmetros e depois de avaliá-los os armazena num único objeto.

```
[ ] ranfor_grid_search = GridSearchCV(estimator = RandomForestClassifier(), param_grid=parameters_gs)
    ranfor_grid_search.fit(x_train, y_train.squeeze())
```

```
GridSearchCV(estimator=RandomForestClassifier(),
             param_grid={'class_weight': ['balanced', 'balanced_subsample'],
                         'criterion': ['gini', 'entropy', 'log_loss'],
                         'max_depth': [6, 8], 'min_samples_leaf': [2, 4]})
```

```
[ ] print(ranfor_grid_search.best_score_)
    print(ranfor_grid_search.best_params_)
```

```
0.8208910275175267
{'class_weight': 'balanced_subsample', 'criterion': 'gini', 'max_depth': 8, 'min_samples_leaf': 2}
```

RandomSearch:

É um método no qual combinações aleatórias de hiperparâmetros são selecionadas e usadas para treinar um modelo. Uma distribuição de amostragem é definida para cada hiperparâmetro para fazer um Random Search.

```
[ ] ranfor_random_search = RandomizedSearchCV(estimator = RandomForestClassifier(), param_distributions=parameters_rs)
    ranfor_random_search.fit(x_train, y_train.squeeze())
```

```
RandomizedSearchCV(estimator=RandomForestClassifier(),
                   param_distributions={'class_weight': ['balanced',
                                                         'balanced_subsample'],
                                        'criterion': ['gini', 'entropy',
                                                      'log_loss'],
                                        'max_depth': [4, 5, 6, 7, 8, 9],
                                        'min_samples_leaf': [1, 2, 3, 4, 5, 6]})
```

```
[ ] print(ranfor_random_search.best_score_)
    print(ranfor_random_search.best_params_)
```

```
0.8209998866881916
{'min_samples_leaf': 2, 'max_depth': 8, 'criterion': 'entropy', 'class_weight': 'balanced_subsample'}
```

Como complemento, utilizamos uma técnica chamada “Cross Validation” (avaliação cruzada) que protege o nosso modelo de Overfitting. Na validação cruzada, você cria um número fixo de dobras (ou partições) dos dados, executa a análise em cada dobra e, em seguida, calcula a média da estimativa de erro geral. A ideia é que, uma vez que tenhamos identificado nossa melhor combinação de parâmetros, testamos o desempenho desse conjunto de parâmetros em um contexto diferente. Sua aplicação ocorre da seguinte forma:

```
[ ]  from sklearn.ensemble import RandomForestClassifier

     ranfor = RandomForestClassifier()

[ ]  ranfor_scores = cross_validate(ranfor, x, y, cv=5, error_score='raise', scoring=scoring)

[ ]  ranfor_scores

     {'fit_time': array([25.48084664, 27.60660076, 27.29132056, 26.0064764 , 26.10364151]),
      'score_time': array([0.90615225, 0.93442965, 1.00150537, 0.96426654, 0.94860005]),
      'test_precision_macro': array([0.86148803, 0.89675985, 0.87479634, 0.8784946 , 0.87770068]),
      'test_recall_macro': array([0.79310455, 0.83411478, 0.8493921 , 0.83415747, 0.82754613])}
```

LightGBM

O LightGBM cria árvores de decisão que crescem em folha, o que significa que, dada uma condição, apenas uma única folha é dividida, dependendo do ganho. Às vezes, as árvores foliares podem se ajustar em excesso, especialmente com conjuntos de dados menores. Limitar a profundidade da árvore pode ajudar a evitar o overfitting.

Para a aplicação desse modelo, o importamos da biblioteca “lightbgm” do python e, depois determinamos o nosso x e y de treino para as primeiras análises.

```
import lightgbm as lgb

x_train, x_test, y_train, y_test = train_test_split(x, y, test_size = 0.3, random_state=42)
# Criando o modelo

lgb = lgb.LGBMClassifier()

lgb.fit(x_train, y_train)

LGBMClassifier()
```

Assim como no exemplo de modelo acima, a ferramenta de GridSearch foi usada para o ajuste dos hiperparâmetros.

GridSearch:

```
from sklearn.model_selection import GridSearchCV

model = lgb

# Cria o GridSearchCV
parameters = {
    'boosting_type':['gbdt','dart','goss','rf'],
    'num_leaves':[42,31],
    'max_depth':range(-3,0),
    'n_estimators':[80,100,200,120]
    }
modelGS = GridSearchCV(model, parameters)

# Treina os modelos e guarda na variável modelGS o melhor modelo
modelGS.fit(x_train, y_train)
modelGS.best_params_
```

```
{'boosting_type': 'gbdt',
 'max_depth': -3,
 'n_estimators': 200,
 'num_leaves': 42}
```

Cross Validation:

```
[ ]  import lightgbm as lgb

     lgb = lgb.LGBMClassifier()
```

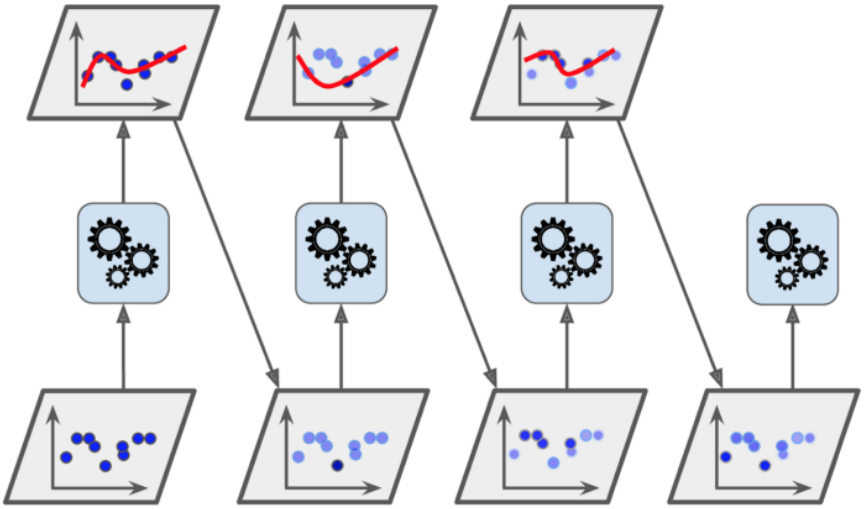
```
[ ]  scores_lgb = cross_validate(lgb, x, y, cv=5, error_score='raise', scoring=scoring)
```

```
[ ]  scores_lgb
```

```
{'fit_time': array([5.92588234, 4.63918281, 7.32697725, 4.75694203, 4.71540022]),
 'score_time': array([0.35538578, 0.40030575, 0.38600755, 0.38497496, 0.37556386]),
 'test_precision_macro': array([0.86681858, 0.90165403, 0.87691125, 0.88097714, 0.87797645]),
 'test_recall_macro': array([0.8279951 , 0.84859355, 0.85780917, 0.84447514, 0.83471705])}
```

ADABoost

AdaBoost é um dos métodos mais populares de Boosting, conceito que se refere a combinação de diversos modelos de aprendizagem fracos para criar um modelo de aprendizagem forte. A idéia por trás consiste em treinar preditores sequencialmente, cada um tentado corrigir seu predecessor



Modelo ADAboost (imagem 8)

Grid Search:

```
[15] adaboost_grid_search = GridSearchCV(estimator = AdaBoostClassifier(), param_grid=parameters_gs)
     adaboost_grid_search.fit(x_train, y_train.squeeze())

     GridSearchCV(estimator=AdaBoostClassifier(),
                  param_grid={'algorithm': ['SAMME', 'SAMME.R'],
                              'learning_rate': [0.5, 1.0, 1.5],
                              'n_estimators': [35, 50, 65]})
```

Através do Grid Search encontramos os melhores hiperparâmetros para o modelo.

4.5. Avaliação

Usando os resultados que foram obtidos anteriormente com o teste de diversos modelos e a realização de testes preliminares, foi possível selecionar os modelos mais acurados para utilização final, conforme dados disponíveis no notebook do projeto.

Os modelos foram treinados e as métricas de avaliação utilizadas foram as típicas de algoritmos de regressão, como acurácia de treino e de teste, revocação, precisão, F1 score, matriz de confusão e importância de cada feature para o modelo.

Observa-se que os algoritmos escolhidos na seção anterior são adequados para classificação conforme as regras de negócio estabelecidas pelo cliente, como apontado na seção específica desse documento.

Vale destacar também que todos os modelos foram testados utilizando os dados iniciais conforme disponibilizados pelo cliente e também passaram por um tratamento de *resampling* com o objetivo de melhorar os resultados obtidos.

Abaixo segue avaliação dos modelos utilizados e como linguagens de programação trabalham com número ao invés de letras ou palavras, foi feita a "tradução" dos termos do target para:

- **Churn Definitivo: 0**
- **Entregador Constante (Não sai da plataforma): 1**
- **Entregador inconsistente: 2**

AdaBoost

AdaBoost é o algoritmo utilizado para tarefas de classificação binária e parte do princípio de Adaptive Boosting. Ele utiliza uma série de algoritmos considerados "fracos" e, posteriormente, consolida os resultados obtidos em uma decisão final.

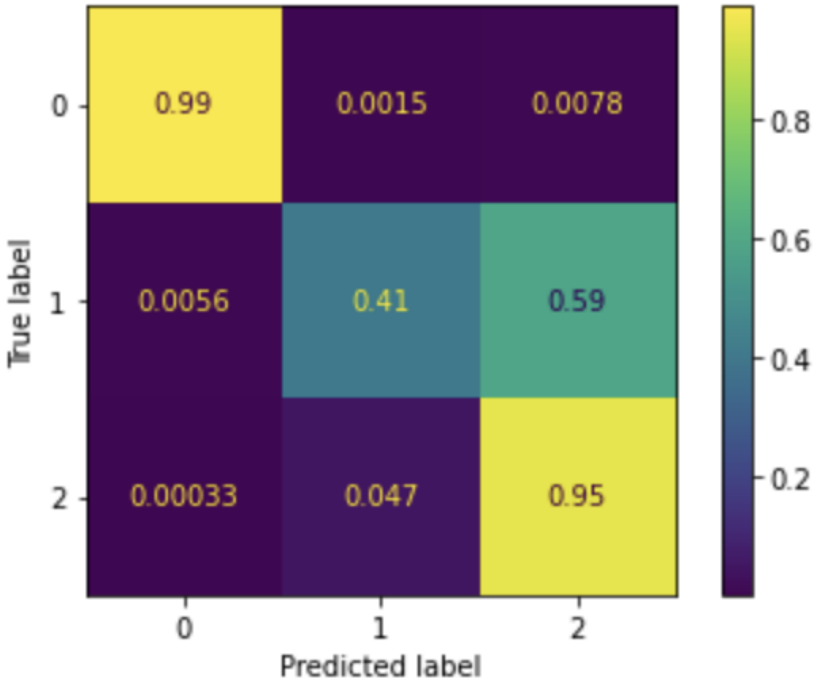
Acuracidade utilizando dados iniciais:

```
Acuracidade (treino):  0.8273140462128841
Acuracidade (teste):  0.8268029889291082
```

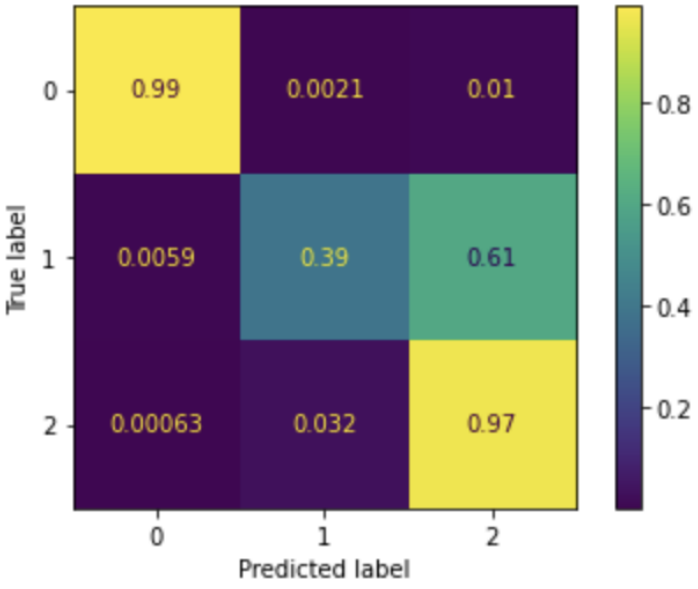
Acuracidade utilizando dados *resampled*:

```
Acuracidade (treino):  0.8296455560696369
Acuracidade (teste):  0.8306767426599776
```

Matriz de confusão utilizando dados iniciais:



Matriz de confusão utilizando dados resampled:



Random Forest Classifier

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

O modelo apresentou excelente acurácia, tanto no dataset de treino quanto no de teste, embora nesse caso tenha se aproximado de um overfit.

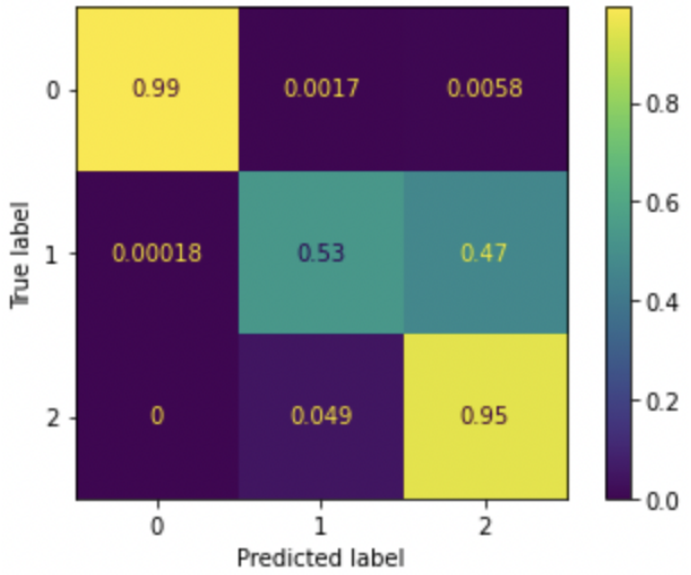
Acuracidade utilizando dados iniciais:

```
Acuracidade (treino):  0.9999818559544221
Acuracidade (teste):  0.8557185495649965
```

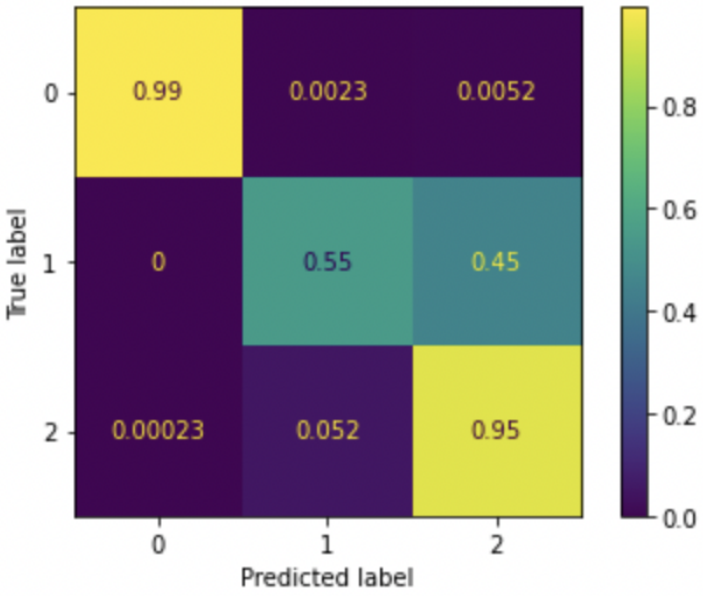
Acuracidade utilizando dados *resampled*:

```
Acuracidade (treino):  0.9999727839316332
Acuracidade (teste):  0.8569674647022714
```

Matriz de confusão utilizando dados iniciais:



Matriz de confusão utilizando dados resampled:



LightGBM

Foi possível obter métricas aceitáveis a partir desse modelo, que utiliza uma estrutura de aumento de gradiente baseado em 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.

Especificamente, o modelo atingiu métricas padrão acima de 90% utilizando a base de dados *resampled*, porém não apresentou performance adequada utilizando os dados iniciais.

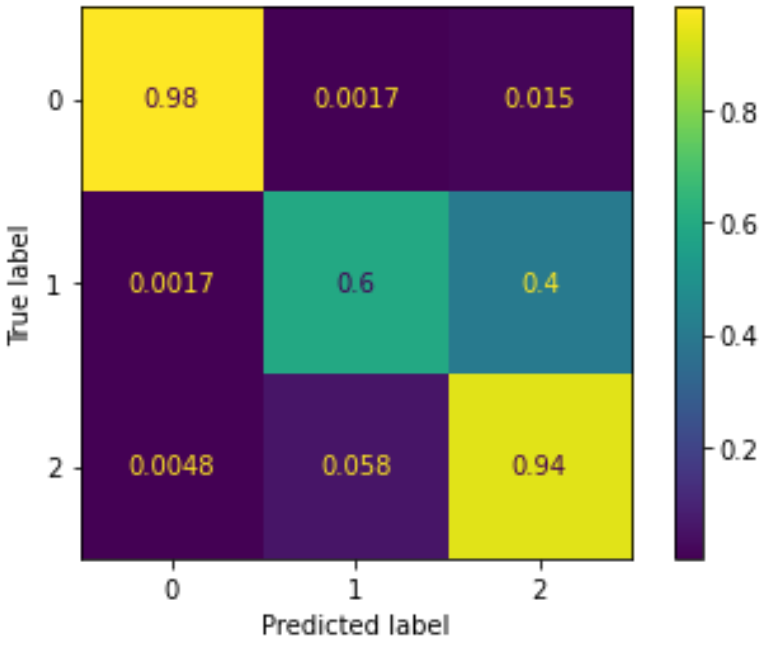
Acuracidade utilizando dados iniciais:

```
Acuracidade (treino):  0.8724201435194006
Acuracidade (teste):  0.8654981901314536
```

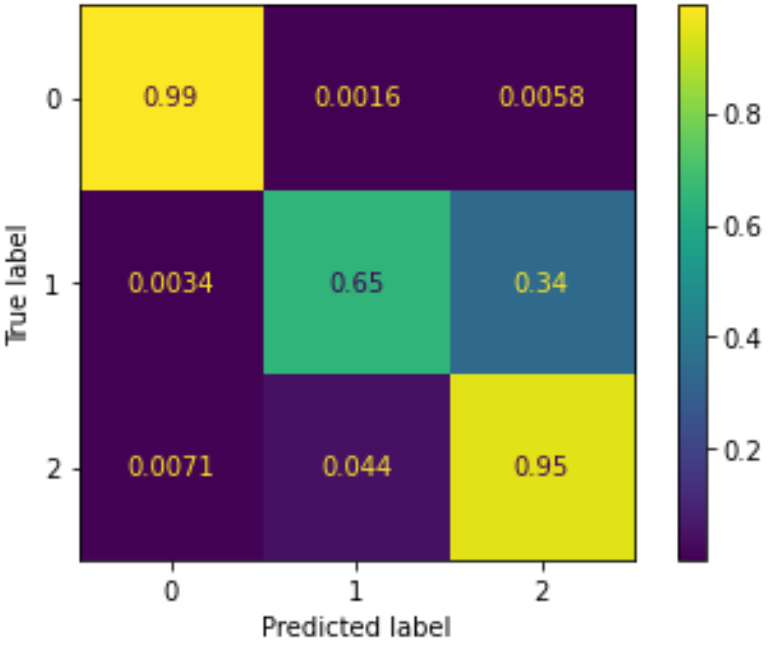
Acuracidade utilizando dados *resampled*:

```
Acuracidade (treino):  0.8867448675031072
Acuracidade (teste):  0.8694142799686713
```

Matriz de confusão utilizando dados iniciais:



Matriz de confusão utilizando dados resampled:



Conclusão:.

Para critérios de comparação, vemos que os modelos têm uma revocação alta para as categorias “Churn Definitivo” e “Entregador inconsistente", sendo que estes são os indivíduos almejados pelo time de operação da Rappi, as políticas da empresa serão desenvolvidas em cima destes entregadores. Já para o caso dos entregadores que não saem da plataforma, foram utilizadas diversas métricas para que sua revocação fosse aumentada, desde hiper parametrização, Under e Over sampling e reavaliação das features, mas o resultado não foi satisfatório. Ainda sim, é fundamental salientar que os modelos de classificação, principalmente aqueles que utilizam métodos de ensemble, neste caso, o Random Forest, o ADABoost e o LighGradientBoostModel, são os que melhores conseguem classificar o target de acordo com a análise feita.

4.6 Comparação de Modelos

Ademais, utilizando das métricas demonstradas na seção anterior, é possível ver que o LighGradientBoostModel indica o melhor resultado frente aos outros modelos, pois este apresenta resultados satisfatórios mediante a entrada de novos dados, fator que pode ser analisado pela baixa diferença entre a acuracidade de treino e de teste. Infere-se a partir do que foi obtido como resultado que o modelo após passado em um conjunto de treino, definido por um subconjunto aleatório do ***dataframe*** original, apresenta um resultado quase que equivalente ao modelo quando aplicado ao conjunto de dados de teste, amostra restante após a subtração do subconjunto de treino do ***dataframe*** original.

Como linguagens de programação trabalham com número ao invés de letras, foi feita a "tradução" dos termos do target para:

- **Churn Definitivo: 0**
- **Entregador Constante (Não sai da plataforma): 1**
- **Entregador inconsistente: 2**

Acuracidade LGBM utilizando dados iniciais:

```
Acuracidade (treino):  0.8724201435194006
Acuracidade (teste):  0.8654981901314536
```

Acuracidade LGBM utilizando dados *resampled*:

```
Acuracidade (treino):  0.8867448675031072
Acuracidade (teste):  0.8694142799686713
```

Como vemos nos dois resultados obtidos para o modelo LGBM ele apresenta uma diferença média de ~1% para a aplicação do modelo nos conjuntos de treino e teste, respectivamente.

Fator que não ocorre no caso do Random Forest Classifier, pois este apresenta uma diferença média de ~14%, mesmo após o resampling dos dados. Este fator é caracterizado no processo de Machine Learning como Overfitting, ou seja, o modelo fica viciado ao analisar os dados de treino e não consegue fazer uma análise acurada quando recebe dados novos.

Seguem os resultados que demonstram tal diferença:

Acuracidade RandomForestClassifier utilizando dados iniciais:

```
Acuracidade (treino):  0.9999818559544221
Acuracidade (teste):  0.8557185495649965
```

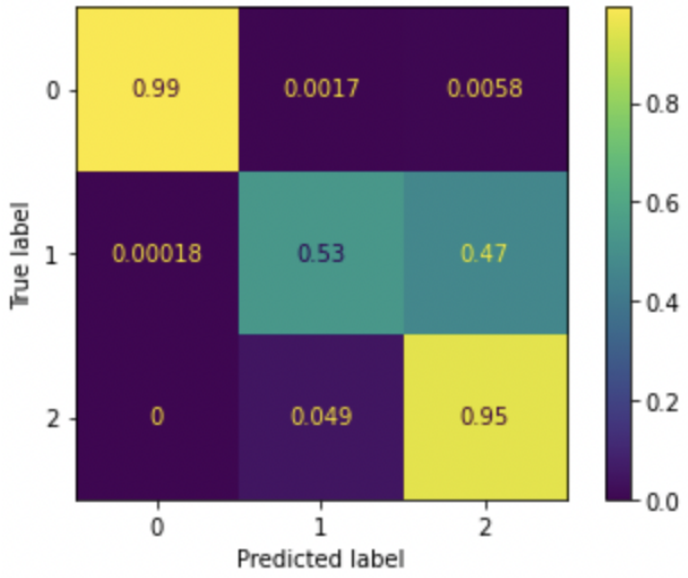
Acuracidade RandomForestClassifier utilizando dados *resampled*:

```
Acuracidade (treino):  0.9999727839316332
Acuracidade (teste):  0.8569674647022714
```

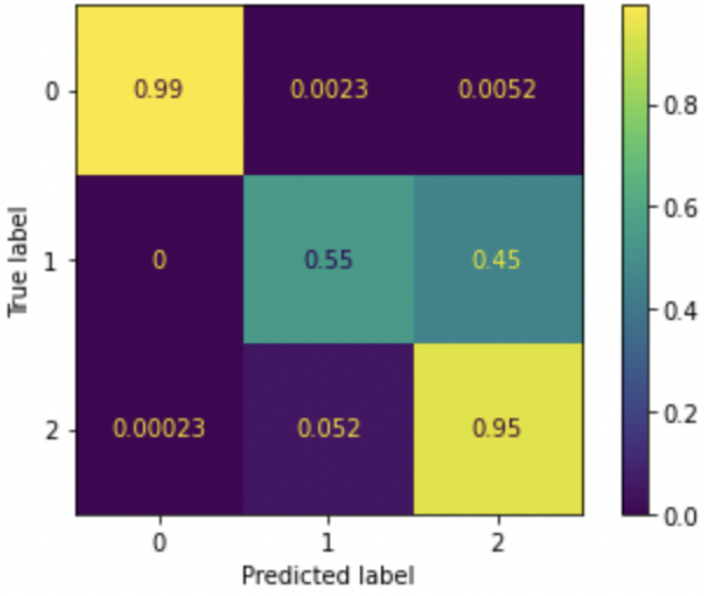
Já para o caso do modelo ADABoost, foi feita uma escolha arbitrária para sua não utilização, este não faz uma classificação adequada do entregador que não saiu da plataforma no período analisado, "Não Churn" ou "False", como consta na documentação ao final da seção 4.2.

Podemos observar esta dificuldade em categorizar este target através das matrizes de confusão seguintes:

Matriz de confusão utilizando dados iniciais:



Matriz de confusão utilizando dados resampled:



Em contramão vemos que este modelo consegue trabalhar bem com novos dados, mas sua classificação deficitária no target "Não Churn" foi definitiva para sua não utilização. Neste caso, o target referido é o valor no centro da matriz que apresenta 53% e 55% de revocação, respectivamente.

5. Conclusões e Recomendações

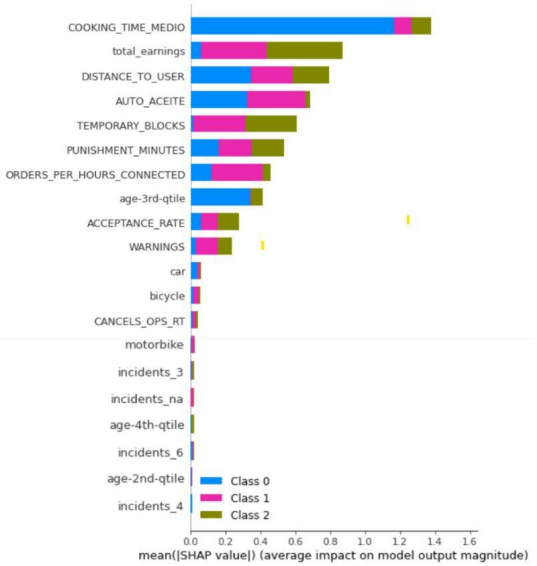
Este trabalho teve como intuito utilizar algoritmos de inteligência artificial e técnicas de machine learning para auxiliar a Rappi a identificar os entregadores que irão sair da empresa, assim colaborando com o planejamento de estratégias que irão fidelizar o trabalhador e aumentar a produtividade.

Alcançamos o objetivo do trabalho utilizando um algoritmo de classificação para dividir em três classes: saída (churn), inconsistência(churn constante) e permanência (não churn), elas mostram a tendência de cada trabalhador deixar a empresa. Para melhorar a visualização utilizamos uma escala que vai de 1 a 5 onde: 1 é rotulado como “Certeza de não churn” que indica quais trabalhadores têm uma porcentagem igual ou superior a 90% na classe que não irá sair, 2 é "provável não churn” que representa uma probabilidade inferior a 90% na classe que não irá sair, 3 foi denominado como "RT_ Inconsistente" que indica quais têm uma porcentagem igual ou superior a 90% na classe de inconsistência, 4 é "Provável churn” que ilustra uma probabilidade inferior a 90% na classe que irá sair ,5 “Churn definitivo” que indica quais possuem uma porcentagem igual ou superior a 90% na classe que irá sair.

5	Certeza de Churn Definitivo
4	Provável Churn Definitivo
3	Inconsistentes
2	Provável Não Churn
1	Certeza de Não Churn

Foram testados diversos tipos de algoritmos e de todos eles o que apresentou um melhor desempenho foi o LGBM que obteve uma acurácia de 88% no treino e 86% no teste, esse tipo de algoritmo funciona criando árvores de decisão e aumentando os ramos no sentido que os resultados foram melhores, assim ele analisa uma série de características dos entregadores e os classifica em umas das classes. Com isso, vemos que o conjunto de dados é de grande importância para as predições.

Quando analisamos a explicabilidade do modelo para alocar o trabalhador em cada uma das classes, chegamos a conclusão que para a classe de “churn” o que mais influencia o modelo é o tempo médio que o entregador espera o pedido ficar pronto e a distância que o entregador percorre até o usuário, para a classificação “não churn” o algoritmo considera mais o total de ganhos, o auto aceite, os bloqueios temporários e as ordens por hora conectado, já o que influencia a classificação de “churn constante” é o total de ganhos e os bloqueios temporários. Assim, entregadores que permanecem ou saem e voltam, fazem isso pelo total de ganhos e pelos bloqueios temporários, o que condiz com as hipóteses levantadas na fase de elaboração de features.



Ao final desse projeto entendemos que este modelo de machine learning pode melhorar a relação da empresa com os trabalhadores, mostrar os déficits que causam a evasão, aumentar a eficiência da empresa, dar uma grande vantagem competitiva e gerar um diferencial com relação aos demais players no setor. Com relação aos colaboradores da Rappi que são afetados indiretamente pela solução elaborada, nós inferimos que todos os dados devem ser tratados de forma ética e que a partir dos resultados apresentados a empresa elabore incentivos e manobras para mitigar as irregularidades que os levam ao abandono da plataforma.

6. Referências

[1]: https://economia.uol.com.br/noticias/redacao/2022/04/02/ifood-ficou-tao-grande-que-prejudica-os-brasileiros.htm

[2]:

https://www.cnnbrasil.com.br/business/gasto-com-delivery-sobe-24-em-2021-veja-tendencias-de-consumo-do-pos-pandemia/

[3]:
https://sbvc.com.br/silenciosamente-o-rappi-comeca-a-virar-um-grande-varejista/

[4]:
https://news.ifood.com.br/onde-e-como-o-ifood-utiliza-inteligencia-artificial/

[5]:
https://sacra.com/research/rappi-the-meituan-of-latin-america/#:~:text=Rappi%20is%20a%20Latin%20American%20on%2Ddemand%20delivery%20app%20with,as%20e%2Dcommerce%20and%20travel.

Tabela 2:

Matriz de Risco - Figma

Anexos

Manual do usuário