inteli
instituto
de tecnologia
e liderança

Controle do Documento

Histórico de revisões

Data	Autor	Versão	Resumo da atividade
<08/08/2022>	<Thainá de Deus Lima>	<1.1>	<Criação do documento; Atualização da seção 2.7.>
<10/08/2022>	<Pedro Rezende>	<1.2>	<Atualização da secção 4.1.1 e 4.1.2.>
<10/08/2022>	<Kathlyn Diwan>	<1.3>	<Atualização da seção 4.2.>
<11/08/2022>	<Thainá de Deus Lima, Kathlyn Diwan, Pedro Rezende>	<1.4>	<Validação de todas as secções.>
<12/08/2022>	<Thainá de Deus Lima>	<1.5>	<Revisão final para publicação no GitHub>
<24/08/2022>	<Thainá de Deus Lima, Luiz Augusto Ferreira, Vinicius Santos, Luiz Ferreira>	<2.1>	<Atualização dos dados da seção 4.2 e início da seção 4.3>
<25/08/2022>	<Kathlyn Diwan, Vinicius Santos, Iago Tavares, Kathlyn Diwan, Pedro Rezende,Isabela Rocha, Luiz Ferreira>	<2.2>	<Atualização dos dados da seção 4.3>
<27/08/2022>	<Thainá de Deus Lima, Pedro Rezende>	<2.3>	<Revisão final>
<28/08/2022>	<Pedro Rezende>	<2.4>	<Implementação do tópico (d) na secção 4.3>
<08/09/2022>	<Kathlyn Diwan>	<2.4>	<Atualização da seção 4.4>
<09/09/2022>	<Thainá de Deus Lima>	<2.5>	<Revisão das legendas referentes às imagens>
<11/09/2022>	<Thainá de Deus Lima>	<2.6>	< Revisão final para publicação no GitHub>
<12/09/2022>	<Pedro Rezende, Isabela Rocha>	<3.0>	< Início da sprint 3 e atualização da seção 4.4>
<16/09/2022>	<Kathlyn Diwan, Iago Tavares>	<3.1>	<Declaração dos parâmetros utilizados, na seção 4.4>
<19/09/2022>	<Kathlyn Diwan, Iago Tavares>	<3.2>	<Análise dos resultados de cada parâmetros testado, na seção 4.5 >

<21/09/2022>	<Kathlyn Diwan, Pedro Rezende>	<4.1>	<Incremento dos hiperparâmetros>
<22/09/2022>	<Kathlyn Diwan, Luiz Ferreira>	<4.2>	<Incremento dos hiperparâmetros>
<23/09/2022>	<Kathlyn Diwan>	<4.3>	<Revisão>
<26/09/2022>	<Iago Tavares, Luiz Ferreira, Thainá de Deus Lima>	<4.4>	<Validação dos artefatos>
<27/09/2022>	<Thainá de Deus Lima, Luiz Ferreira>	<5.1>	<Identificação das seções que precisam ser feitas e revisadas>
<28/09/2022>	<Kathlyn Diwan, Pedro Rezende e Luiz Ferreira>	<5.2>	<Verificação da seção 1>
<29/09/2022>	<Kathlyn Diwan>	<5.3>	<Verificação da seção 2>
<03/10/2022>	<Thainá de Deus Lima, Kathlyn Diwan>	<5.4>	<Verificação de features>
<04/10/2022>	<Thainá de Deus Lima, Isabela Rocha>	<5.5>	<Atualizações do CRISP-DM e PyCaret>
<05/10/2022>	<Thainá de Deus Lima, Iago Tavares>	<5.6>	<Atualizações do deploy e conclusão>
<06/10/2022>	<Thainá de Deus Lima>	<5.7>	<Sumarização das imagens, tabelas e títulos> <Atualização do histórico de revisões>
<06/10/2022>	<Kathlyn Diwan>	<5,7>	Revisão da documentação>
17/03/2023	Escritório de Projetos	5.8	Protegendo dados sensíveis

Sumário

Sumário de imagens

Sumário de tabelas

1. Introdução

A Rappi é uma startup, fundada por Simon, Sebastian e Felipe em 2015, que proporciona dentro de sua plataforma, um comércio de bens e serviços, conectando empreendedores, autônomos e usuários. Além do tradicional delivery de comida, o aplicativo também fornece produtos de farmácia, mercados, lojas de diferentes tipos, ingressos de shows e muito mais. A empresa chega ao Brasil em 2017, e se estabelece em mais de 200 cidades LATAM, 180.000 parceiros ativos, 220.000 RTs, 4,8 M usuários ativos, além de + 5.000 colaboradores.

Com a Rappi, os clientes acessam um ecossistema de comércio, no qual, os empreendedores ampliam seus negócios e as pessoas podem converter o tempo de trabalho na plataforma em uma renda extra. Com a eclosão da pandemia no mundo, o estabelecimento do distanciamento social, a fim de diminuir a contaminação do vírus, serviu como um catalisador para a oferta e procura dos serviços online. Os gastos com delivery em 2021 cresceram 24% a mais que o ano anterior, mostrando que a compra de comida online é um hábito que veio para ficar (GS & MD; Instituto Food Service Brasil, 2022).

Atualmente, observa-se a utilização de modelos preditivos para exercitar a tomada de decisão com maior precisão, por diversas empresas, a partir de testes com a modelagem, que formulam a probabilidade de ações, informações ou números que podem impactar o desempenho da mesma no mercado. Dessa forma, a documentação que registra o Rappi Predict System, contém os insights e anotações de convenção no desenvolvimento de um projeto na área da computação, a fim de auxiliar o cliente no decorrer da utilização do produto.

2. Objetivos e Justificativa

2.1. Objetivos

Com o surgimento de novas plataformas de entrega concorrentes a Rappi, acompanhamos um drástico crescimento na competitividade do segmento de entregas. Diante desse cenário, os

entregadores são chave fundamental para a expansão da plataforma, intensificando a presença da plataforma nas regiões.

O objetivo geral do projeto é entregar um algoritmo preditivo que demonstra os entregadores mais propensos a dar churn na plataforma, baseando-se na base de dados da empresa. A escolha do modelo será feita com base no melhor nível de acurácia, considerando também métricas como Precisão, Recall e F1-score.

2.2. Justificativa

O modelo preditivo baseia-se em dois pilares:

- Melhores métricas estatísticas: Observa-se a utilização de diversos algoritmos, optando pelos que possuem melhores resultados nas métricas aplicadas, se adequando às especificações do cliente.
- Adaptabilidade: Foi criado um algoritmo com alto nível de adaptação, permitindo que o cliente modifique as features caso encontre novos dados que apresentem um alto grau de relevância para o modelo.

A adaptabilidade do modelo foi pensada sabendo que as features que foram entregues podem vir a se tornar menos relevantes, ao passo que novos dados mais relevantes para o algoritmo surgirão.

3. Metodologia

O projeto foi construído com base na metodologia Scrum, que consiste na divisão da construção do produto em 10 semanas, divididas em 5 Sprints incrementais, fator que significa a complementação do projeto a cada 2 semanas.

3.1. CRISP-DM

A metodologia CRISP-DM é um processo ágil que consiste em um mecanismo cíclico, dividido em diversas fases correspondentes ao procedimento de trabalho com os dados. Tal metodologia possibilita retorno às etapas anteriores e trabalhar com melhorias, adaptações e incrementos.

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 foi a interface Google Colab Notebook, que serve para auxiliar no processo de Machine Learning e exploração da Inteligência Artificial. O Google Colab é uma plataforma que trabalha com a linguagem de programação Python permitindo com que os usuários possam rodar e executar os códigos com mais flexibilidade e rapidez. Essa plataforma contém seu próprio sistema de nuvem que armazena as informações.

Outra ferramenta que foi utilizada pelo grupo foi o Pycaret, biblioteca Python que ajuda os programadores desde o tratamento dos dados, até a implementação dos algoritmos e o deploy final do modelo. Esse mecanismo é utilizado na automatização do fluxo de trabalho de aprendizagem da IA.

3.3. Principais técnicas empregadas

Inicialmente, foi feito o merge das tabelas para que os dados possam ser analisados de forma conjunta. Em seguida, para o tratamento de dados, foram removidas informações duplicadas e null.

Por fim, a denominação com label encoding para as categorias dos entregadores. É importante ressaltar que as classificações em churn e não-churn é binário.

Os algoritmos com melhor acurácia foram Random Forest e árvore de decisão, os seus métodos de decisão são respectivamente: uma grande “floresta” de árvores de decisão onde cada árvore parte de um dado diferente das outras árvores, e a árvore de decisão parte de um dado e vai se bifurcando a partir dos dados anteriores.

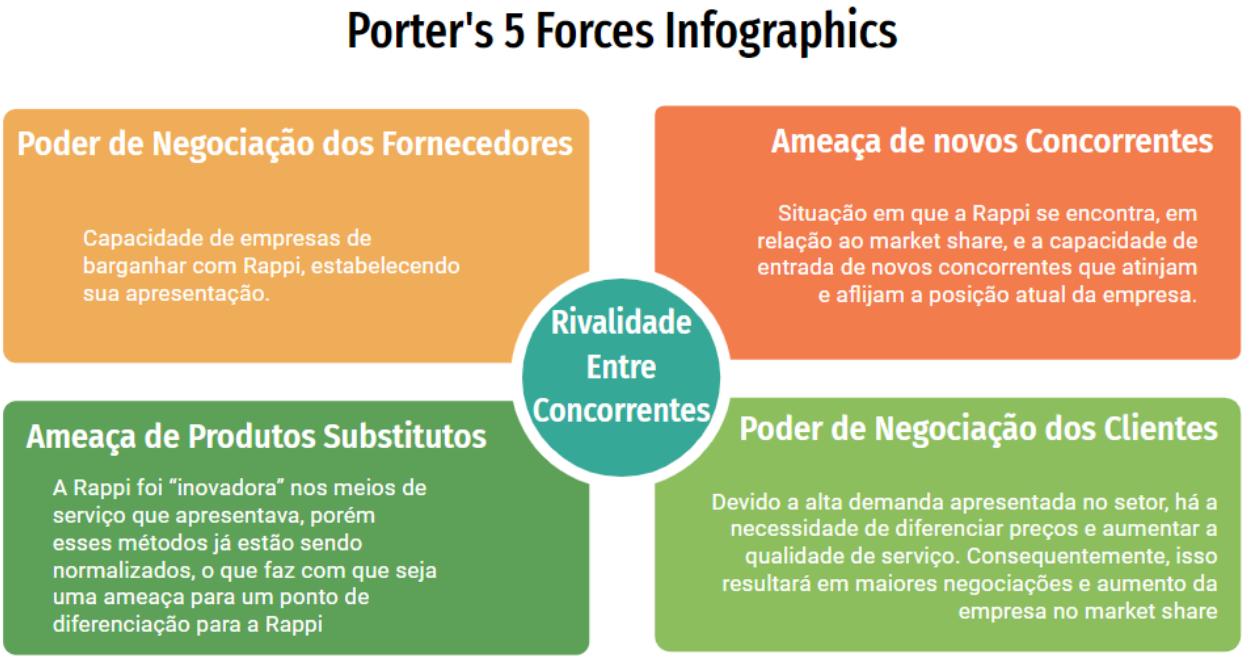
O Random Forest tem a vantagem de partir de vários dados diferentes se tornando menos genérica e cada vez mais específicas sendo geralmente mais precisa. A árvore de decisão parte de um dado e se torna mais precisa baseada nos resultados anteriores de sua análise.

A plataforma Google Colaboraty foi utilizada a fim de estruturar o modelo preditivo de forma que qualquer dataframe por ele selecionado seja funcional para a empresa.

4. Desenvolvimento e Resultados

4.1. Compreensão do Problema

4.1.1. Contexto da indústria



5 Forças de Porter (Imagem 1)

Fontes: Dados dos autores (2022)

A partir da análise de forças, proposta por Michael Porter, se inicia a análise de indústria da Rappi, a fim de entender os principais *players*, modelos de negócios e tendências do mercado.

Modelo de Negócio:

Rappi é uma startup de entrega sob demanda, com sede em grande parte da América Latina. A empresa trabalha no segmento de **delivery** e realiza entregas de Supermercado, Restaurantes, Farmácia, Express, Shopping, Bebidas, Rappi Travel. Isso demonstra como a Rappi, mesmo sendo uma startup, demonstra grande capacidade de crescimento, atuando nas áreas de marketplace (para os restaurantes, principalmente) e como logística.

Rivalidade entre concorrentes: [Ifood, zé delivery, UberEats → principais players]

Devido ao alto nível de competitividade nesse tipo de setor, de delivery, a Rappi como uma empresa relativamente nova ainda possui algumas dificuldades para superar seus concorrentes. Aspectos como: eficiência, tecnologia, suporte aos clientes e entregadores, fazem com que apareçam desafios durante o caminho da Rappi de se tornar a principal empresa de delivery.

Ameaça de novos concorrentes: [mercado livre, 99 food…]

Com um tipo de mercado ativo e que trás bastante retorno financeiro, é claro que haveria diversos concorrentes se expandindo nesse setor. Porém, mesmo com uma grande variedade de concorrentes, o mercado é dominado por grandes empresas como o Ifood, o UberEats e a Rappi, a qual é responsável por uma média de 5% do market share presente no mercado. Devido a estrutura do setor apresentar essa diversidade de concorrentes, a ameaça da aparição de novos players representa algo iminente, fazendo com que a Rappi tenha que constantemente competir arduamente para não perder sua posição.

Ameaça de Produtos substitutos: [aplicativo de delivery dos próprios estabelecimentos]

A ameaça de produtos que podem vir a substituir a Rappi, no setor de delivery, chega a ser média para pequena. A criação de novos apps e o crescimento de novas empresas fazem com que haja um grande potencial e, consequentemente, virar ameaças. Todavia, a Rappi possui uma participação severa no mercado o que torna difícil sua saída e decaimento na sua capacidade, mas não deixa de haver ameaças que venham a diminuir a capacidade da Rappi e tomar seu posto.

Poder de negociação dos clientes:

No que diz respeito ao poder negocial dos clientes, os quais possuem alta palavra de barganha nesse tipo de setor, fica claro os seguintes fatores: a demanda frenética faz com que as empresas tenham que aumentar drasticamente sua eficiência (logística dos apps e atendimento ao cliente) e qualidade no serviço de entrega. A partir desse fator, é visível que a situação da Rappi se encaixa bastante nesse ponto. De acordo com a análise de Porter, entende-se que a força do poder de barganha dos clientes é alta, logo, a Rappi tanto deve aproveitar a situação dos seus concorrentes quando estiverem em eventuais falhas (como caso da UberEats), como também deve se atentar para casos

de próprias falhas e problemas técnicos, que acabam por fazê-los perderem seus clientes e até entregadores.

Poder de negociação dos fornecedores:

O poder de barganha dos fornecedores é forte, no sentido de que as empresas de delivery trabalham com especificamente serviços de entrega constantes, logo, possuem a necessidade de parceiros os quais fornecem o suporte necessário para serem feitas as melhores médias de entregas para o cliente final, mantendo suas posições no mercado ou até crescendo. Porém, um fator é claro, quanto maior o desenvolvimento da empresa e expansão para outros setores de atuação, haverá mais necessidade dos fornecedores e seus produtos.

Tendências de Mercado:

Atualmente, a maior tendência de mercado, nesse tipo de setor, é a capacidade de desenvolvimento tecnológico e inovações a partir de tal. Já que, concorrentes da Rappi, como o Ifood, possuem grande uso de tecnologia e utilização de IA's, as quais antecipam as recomendações de pedidos e etc. Assim, a empresa se mantém atualizada e na linha de competição, no meio técnico.

Conclusão:

Em geral, capacidade de inovar, atualizar e melhorar os meios tecnológicos da empresa, além fazer manutenção do time de suporte técnico, são meios que possam assegurar o crescimento da Rappi no setor. Cativar o cliente a partir dessas melhorias, faria com que assegurasse mais esses clientes somente à Rappi e fazendo com que os fornecedores percebessem o crescimento da empresa.

4.1.2. Análise SWOT

Strengths (Forças)	Weakness (Fraquezas)

- Adaptação à Tecnologia .
- Interface tecnológica ;
- Diversidade de tipos de entregas (além de ser “pioneira nisso”);
  - Mercado;
  - Farmácia;
  - Rap Favor;
  - Pets.
	- Fraca presença no mercado como um todo (na América Latina);
	- Baixa agilidade no desenvolvimento e aprimoramento das ferramentas existentes;
	- Lento desenvolvimento técnico.

Opportunities (Oportunidades)	**Threats (Ameaças)**
- Capacidade de desenvolver aspectos técnicos, assim, cativando clientes; - Possibilidade de expansão na capacidade de prestação dos outros tipos de serviços; - Melhora no cuidado dos entregadores, fazendo com que haja melhora nas entregas; - Aumento crescente de pedidos na plataforma.	- Forte concorrência; - Maior quantidade de trabalho para motoristas da concorrência; - Motoristas com Dupla Jornada de Trabalho (entre mais de uma empresa); - Churn dos Rt 's devido às más qualidades do “emprego”.

Análise SWOT (Tabela 1)

Fontes: Dados dos autores (2022)

4.1.3. Planejamento Geral da Solução

O time de operações da Rappi, percebeu que houve uma crescente no número de churn de entregadores do aplicativo, por conta de inúmeros fatores que estão os levando a ficar insatisfeitos e sair da plataforma. Portanto, a empresa procura um sistema para predição, com inteligência artificial, que rankeie os trabalhadores propensos a dar churn.

O parceiro disponibilizou diversos dados para análise, que serão utilizados para o desenvolvimento da nossa solução. Os dados incluem: a quantidade de Rappitenders, taxa de aceitação dos pedidos, ticket aberto por reclamação, compensação por tickets abertos, incidentes e regras, retorno de produtos, pedidos realizados e pedidos cancelados, tempo (em hora) que o entregador fica conectado no período, tempo de resolução (quanto tempo o RT ficou esperando um retorno do time de Suporte do APP), e informações básicas a respeito dos entregadores (modal, cidade, cadastro, nascimento, ativo ou não, número de ordens, gorjetas, primeiro/último pedido). O objetivo, como equipe, é desenvolver um modelo preditivo que possa ser utilizado de maneira escalar e macro para garantir uma menor quantidade de churns dos RT 's, e otimizar a visualização, em forma de ranking, os rappitender que estão mais propensos a abandonar a empresa. Consequentemente, a solução será utilizada pelos coordenadores de operações para que eles possam entender os principais fatores por meio da análise dos dados que estão influenciando e fazendo com que os entregadores fiquem insatisfeitos resultando na saída da plataforma.

Diante dos dados que foram coletados pela empresa, a tarefa será de caráter classificatório. Desse modo, nossa solução irá apontar a possibilidade de um entregador de dar churn, e abandonar a plataforma. A escala de classificação para realizar essa tarefa ainda não foi definida

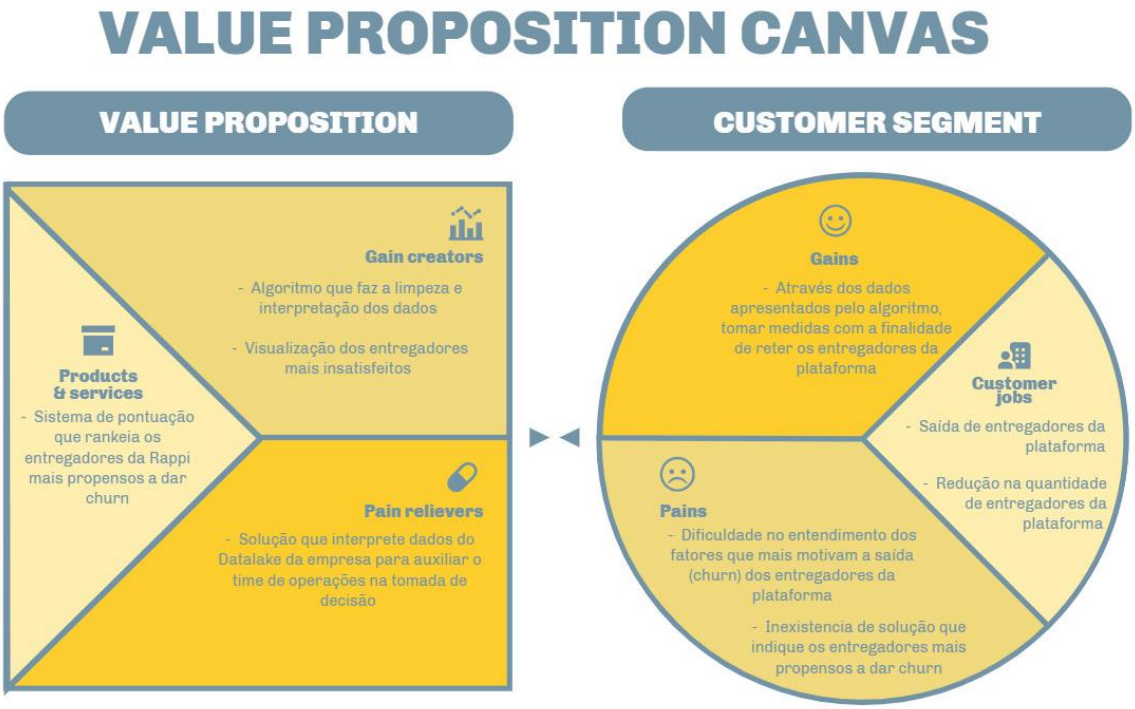
A solução desenvolvida pelo grupo, será utilizada majoritariamente pelos analistas e coordenadores de operações da Rappi que serão responsáveis pelo acompanhamento e andamento dos entregadores. Com a análise dos gráficos e sistema preditivo que será gerado pelo grupo, os próprios gestores da empresa terão a possibilidade de implementar e desenvolver novas estratégias para evitar o churn dos entregadores.

A solução irá facilitar o sistema de operações da Rappi, pois irá apontar quais entregadores estão mais propensos de abandonar a plataforma, consequentemente novas estratégias e maneiras

de garantir a continuidade deles no aplicativo serão geradas. Dessa maneira, observa-se que o sistema poderá ser utilizado para prevenir futuras saídas espontâneas dos entregadores, ou até mesmo para adotar medidas de soluções para melhorar as condições de trabalho e qualidade do serviço.

O critério de sucesso corresponde à assertividade preditiva do modelo. A partir do teste da modelagem, conforme os dados verídicos de entregadores que deram churn ou não, fornecidos pela empresa, observar se o modelo preditivo estruturado consegue predizer o resultado mais próximo da realidade.

4.1.4. Value Proposition Canvas



Value Proposition Canvas (Imagem 2)

Fontes: Dados dos autores (2022)

4.1.5. Matriz de Riscos

Matriz de Riscos - Happi

Matriz de Riscos (Tabela 2)

Fontes: Dados dos autores (2022)

4.1.6. Personas

Com o objetivo de melhor compreender parte dos stakeholders do projeto, foram estruturadas duas personas correspondentes, respectivamente, ao analista e coordenador do Time de Operações e ao entregador da plataforma.

O analista e coordenador do Time de Operações será responsável pela visualização dos entregadores que tem probabilidade de dar churn e os fatores para tal ocorrência, para que eles possam executar medidas pontuais para evitar o churn dos RT 's.

Persona: Fernando Souza (Imagem 3)

Fontes: Dados dos autores (2022)

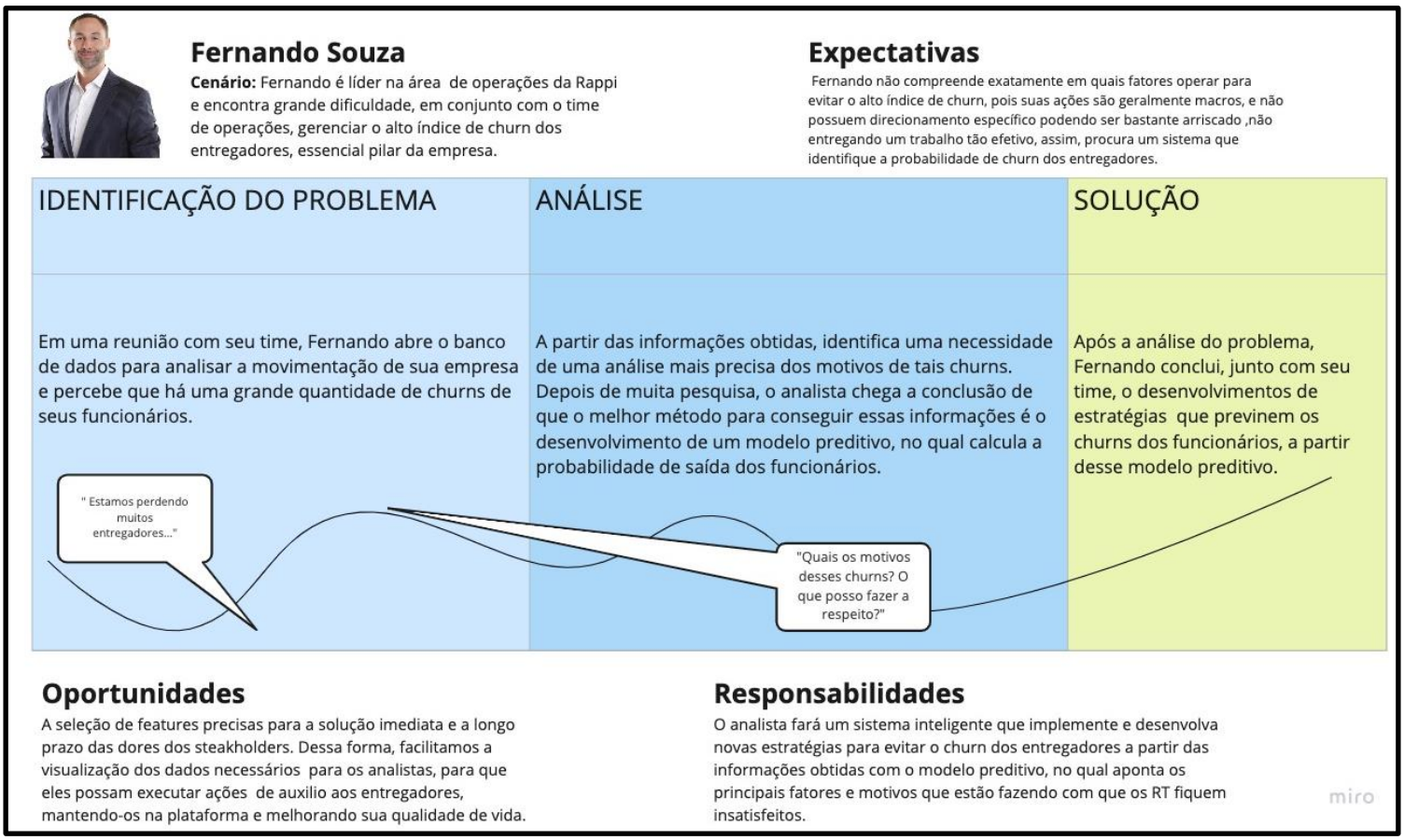
Já relacionado ao entregador, destaca-se a importância de ressaltar as suas dores e motivações que consequentemente resultarão em sua saída da plataforma. A partir do conhecimento e aproximação desses fatores, o Time de Operações pode redirecionar ações que minimizem esses impactos.

Persona: Marcos Aurélio (Imagem 4)

Fontes: Dados dos autores (2022)

4.1.7. Jornadas do Usuário

Mapa Jornada do Usuário - Acesso pelo Miro



Mapa da jornada do usuário (Imagem 5)

Fontes: Dados dos autores (2022)

4.2. Compreensão dos Dados

ABAIXO, SERIA POSSÍVEL VISUALIZAR UMA ETAPA DE EXPLICAÇÃO DA REGRA DE NEGÓCIO, JUNTAMENTE COM OS DADOS FORNECIDOS PELO PARCEIRO PARA A CONSTRUÇÃO DO MODELO PREDITIVO. NO ENTANTO, EM RESPOSTA AO PEDIDO DO PARCEIRO PARA PROTEGER DADOS SENSÍVEIS, OS DADOS FORAM OCULTADOS.

4.3. Preparação dos Dados

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

A especificação “DataFrame Principal”, refere-se à construção da tabela “df_all_store”, que foi criada para ser manipulada e utilizada em outras tabelas, visto que, é possível comparar as colunas relacionadas às hipóteses estruturadas, com a possibilidade de dar churn ou não e o ID de cada entregador.

ABAIXO, SERIA POSSÍVEL VISUALIZAR UMA IMAGEM DA TABELA COM DADOS FORNECIDOS PELO PARCEIRO PARA A CONSTRUÇÃO DO MODELO PREDITIVO. NO ENTANTO, EM RESPOSTA AO PEDIDO DO PARCEIRO PARA PROTEGER DADOS SENSÍVEIS, OS DADOS FORAM OCULTADOS.

DataFrame Principal(Imagem 6)

Fontes: Dados dos autores (2022)

Feature 1 - Quantitativo de punições (Fazer Classificação);

Em relação a Feature 1, explora-se a hipótese de que os entregadores com o maior número de punições têm as maiores chances de dar churn. Para isso, serão utilizadas as planilhas *"df_incidentes"* e, especificamente, as colunas "STOREKEEPER_ID" e "PUNISHMENT_TYPE". Portanto, foram feitas mudanças na planilha "df_warning_punishment", removendo as informações duplicadas, substituindo os valores nulos por 0, e renomeando colunas da forma mais simplificada para a compreensão do usuário.

ABAIXO, SERIA POSSÍVEL VISUALIZAR UMA IMAGEM DA TABELA COM DADOS FORNECIDOS PELO PARCEIRO PARA A CONSTRUÇÃO DO MODELO PREDITIVO. NO ENTANTO, EM RESPOSTA AO PEDIDO DO PARCEIRO PARA PROTEGER DADOS SENSÍVEIS, OS DADOS FORAM OCULTADOS.

Quantitativo de punições (Imagem 7)

Fontes: Dados dos autores (2022)

Feature 2 - Quantitativo de Churns (Fazer classificação); (foi removido do nosso modelo preditivo por conta do viés do algoritmo)

Já na segunda feature, explora-se a hipótese de que o entregador que já deu churn uma vez, tem maiores chances de dar churn novamente. A planilha utilizada para o estudo dessa hipótese foi a *"df_contas_churn"*, a partir das colunas "ID" e "Count". No estudo das features, os valores nulos foram substituídos por 0, localizados na planilha “df_qtd_churns”. Além do mais, foi feito o merge das planilhas “df_all_store” e “count_all_churns”, agrupa-se e soma-se os valores na planilha “count_churns”.

ABAIXO, SERIA POSSÍVEL VISUALIZAR UMA IMAGEM DA TABELA COM DADOS FORNECIDOS PELO PARCEIRO PARA A CONSTRUÇÃO DO MODELO PREDITIVO. NO ENTANTO, EM RESPOSTA AO PEDIDO DO PARCEIRO PARA PROTEGER DADOS SENSÍVEIS, OS DADOS FORAM OCULTADOS.

Quantitativo de churn (Imagem 8)

Fontes: Dados dos autores (2022)

Feature 3 - Modal;

Na feature 3, o merge foi feito nas planilhas “df_infos_gerais” e “df_all_store”, a fim de identificar o modal que possui o maior número de entregadores que deram churn. Além do mais, foram realizados agrupamento e adição da quantidade de entregadores que deram churn em cada uma das modalidades, evidenciado na planilha “df_infos”.

ABAIXO, SERIA POSSÍVEL VISUALIZAR UMA IMAGEM DA TABELA COM DADOS FORNECIDOS PELO PARCEIRO PARA A CONSTRUÇÃO DO MODELO PREDITIVO. NO ENTANTO, EM RESPOSTA AO PEDIDO DO PARCEIRO PARA PROTEGER DADOS SENSÍVEIS, OS DADOS FORAM OCULTADOS.

Modal (Imagem 9)

Fontes: Dados dos autores (2022)

Feature 4 - Quantitativo de defects (média semanal);

Primeiramente, nota-se a fusão das tabelas “df_defects” e "df_all_store", dando origem à tabela "count_defects", composta pelas colunas “ORDERS” (referente aos pedidos com defeitos), “ID”(Identificador do entregador) e “Churn(sim=1/não=0)_x” (target que sinaliza o churn “True ou False” do RT).

Em seguida, foi feito o agrupamento da quantidade de pedidos entregues por cada ID e, a partir dessa perspectiva, o cálculo da média de pedidos com defeitos, informação adquirida conforme a planilha “df_infos”. É importante destacar a substituição dos valores nulos por 0 na planilha “count_defect_na”.

ABAIXO, SERIA POSSÍVEL VISUALIZAR UMA IMAGEM DA TABELA COM DADOS FORNECIDOS PELO PARCEIRO PARA A CONSTRUÇÃO DO MODELO PREDITIVO. NO ENTANTO, EM RESPOSTA AO PEDIDO DO PARCEIRO PARA PROTEGER DADOS SENSÍVEIS, OS DADOS FORAM OCULTADOS.

Quantitativo de defects (média semanal)(Imagem 10)

Fontes: Dados dos autores (2022)

Feature 5 - Taxa de entregas canceladas;

A feature analisada partiu da hipótese de que entregadores com altas taxas de cancelamento dos pedidos, tem maior propensão a darem churn. A planilha utilizada "df_ordens", especificamente, as colunas "ORDERS_CANCEL", "CANCEL_OPS_RT", "STOREKEEPER_ID" e "ORDERS_DONE". O cálculo feito para a construção da taxa de entregas é executado por (ORDERS_CANCEL + CANCELS_OPS_RT)/(ORDERS_DONE + ORDERS_CANCEL + CANCELS_OPS_RT).

ABAIXO, SERIA POSSÍVEL VISUALIZAR UMA IMAGEM DA TABELA COM DADOS FORNECIDOS PELO PARCEIRO PARA A CONSTRUÇÃO DO MODELO PREDITIVO. NO ENTANTO, EM RESPOSTA AO PEDIDO DO PARCEIRO PARA PROTEGER DADOS SENSÍVEIS, OS DADOS FORAM OCULTADOS.

Taxa de entregas canceladas (Imagem 11)

Fontes: Dados dos autores (2022)

Feature 6 - Taxa de aceitação;

Para a delimitação da feature 6, foi feita o merge das tabelas “df_rate” e “df_all_store”, visto que, a hipótese é baseada na afirmação que entregadores com maiores taxas de aceitação possuem menos chances de dar churn. Dessa forma, destaca-se a apresentação das colunas “STOREKEEPER_ID”, “ACCEPTANCE_RATE” E “Churn(sim=1/não =0)_x. Para beneficiar a leitura e maior precisão do algoritmo, foram realizadas a substituição dos valores nulos por 0, na planilha “df_rate_na”.

ABAIXO, SERIA POSSÍVEL VISUALIZAR UMA IMAGEM DA TABELA COM DADOS FORNECIDOS PELO PARCEIRO PARA A CONSTRUÇÃO DO MODELO PREDITIVO. NO ENTANTO, EM RESPOSTA AO PEDIDO DO PARCEIRO PARA PROTEGER DADOS SENSÍVEIS, OS DADOS FORAM OCULTADOS.

Taxa de aceitação (Imagem 12)

Fontes: Dados dos autores (2022)

Feature 7 - Auto-aceite;

A fim de confirmar a hipótese construída com base na afirmação de que entregadores com o auto-aceite(possibilidade dada no aplicativo) desligado, fazem entregas em outras plataformas também e podem ter maiores chances de dar churn. Para a estruturação de tal observação, foi feita a correlação das tabelas “df_auto_aceite” e “df_all_store”.

ABAIXO, SERIA POSSÍVEL VISUALIZAR UMA IMAGEM DA TABELA COM DADOS FORNECIDOS PELO PARCEIRO PARA A CONSTRUÇÃO DO MODELO PREDITIVO. NO ENTANTO, EM RESPOSTA AO PEDIDO DO PARCEIRO PARA PROTEGER DADOS SENSÍVEIS, OS DADOS FORAM OCULTADOS.

Taxa de auto-aceite (Imagem 13)

Fontes: Dados dos autores (2022)

Feature 8 - Earnings por categoria

Para o estudo da Feature 8, realiza-se o merge das tabelas "df_levels" e "df_earnings", e das tabelas “df_all_store” e “df_earnings_categories”, a fim de trabalhar com a hipótese de que os entregadores com ganhos abaixo da média de sua categoria tem uma propensão a dar churn. Para a estruturação da análise, trabalhamos com as colunas "STOREKEEPER_ID", "LEVEL_NAME" e

"EARNINGS". Além do mais, houve também a remoção de IDs duplicados na planilha "df_defects", e de colunas na planilha "df_earnings".

ABAIXO, SERIA POSSÍVEL VISUALIZAR UMA IMAGEM DA TABELA COM DADOS FORNECIDOS PELO PARCEIRO PARA A CONSTRUÇÃO DO MODELO PREDITIVO. NO ENTANTO, EM RESPOSTA AO PEDIDO DO PARCEIRO PARA PROTEGER DADOS SENSÍVEIS, OS DADOS FORAM OCULTADOS.

Colunas utilizadas na feature (Imagem 14)

Fontes: Dados dos autores (2022)

ABAIXO, SERIA POSSÍVEL VISUALIZAR UMA IMAGEM DA TABELA COM DADOS FORNECIDOS PELO PARCEIRO PARA A CONSTRUÇÃO DO MODELO PREDITIVO. NO ENTANTO, EM RESPOSTA AO PEDIDO DO PARCEIRO PARA PROTEGER DADOS SENSÍVEIS, OS DADOS FORAM OCULTADOS.

Earnings por categoria separadamente (Imagem 15)

Fontes: Dados dos autores (2022)

ABAIXO, SERIA POSSÍVEL VISUALIZAR UMA IMAGEM DA TABELA COM DADOS FORNECIDOS PELO PARCEIRO PARA A CONSTRUÇÃO DO MODELO PREDITIVO. NO ENTANTO, EM RESPOSTA AO PEDIDO DO PARCEIRO PARA PROTEGER DADOS SENSÍVEIS, OS DADOS FORAM OCULTADOS.

Earnings por categoria (Imagem 16)

Fontes: Dados dos autores (2022)

b) Se aplicável, como deve ser feita a agregação de registros e/ou derivação de novos atributos.

A agregação de registros e derivação de novos atributos deve ser feita baseada nos conceitos do feature engineering e do modelo crisp, limpando os dados previamente e deixando-os mais "entendíveis" para o algoritmo. A agregação de registros de maneira mais específica, seria o *merge* de dados de outras tabelas para a validação de uma feature, assim como a derivação de novos

atributos seria a criação de novos artifícios para análise, como por exemplo: a junção de duas colunas de uma tabela para a criação de outra com dados baseados nas duas últimas.

c) Se aplicável, como devem ser removidos ou substituídos valores ausentes/em branco.

Para remover ou substituir os valores ausentes ou em branco, é importante se basear no feature engineering, assim dá para fazer uma limpeza adequada desses dedos que estão em branco ou ausente. Desta maneira, sendo esses valores de features que serão levados em consideração para o modelo preditivo, pode ser utilizada a biblioteca Pandas para que facilite a manutenção dessas ocasiões. Esse processo se dá em deletar esses eventos outliers, de forma que os dados que restarem sejam mais precisos e condizentes com a realidade.

Como por exemplo podemos ver na tabela de entregas onde existe uma parcela de entregadores que segundo os dados eles não completaram nem um delivery, nesses caso é importante tirar esse entregador do arquivo para que o algoritmo tenha um arquivo limpo para treinar de forma mais adequada, pois caso contrário a assertividade da predição seria afetada com esse valores outliers.

Para isso primeiramente foi utilizado de um for (estrutura de repetição) com um if/else dentro, para que seja possível saber quais entregadores não tem nem uma entrega concluída, a partir disso dá para criar uma tabela desses entregadores e dar um inner na tabelas que contém todos os RT's, e assim ter a limpeza da tabela adequadamente.

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

Feature 1 - Quantitativo de punições;

→ A feature 1 é baseada na ideia de que os entregadores com maior número de punições têm maior probabilidade de darem churn. Essa feature foi selecionada pela relevância que ela apresenta, no quesito lógico de: +Punições = Entregadores insatisfeitos e estressados, resultando no churn desses mesmos. Além disso, os dados apresentaram confirmações dessa teoria, como também

apresentam Outliers (dados fora da curva, fora do sentido linear dos dados). Dessa forma, a feature está ainda em desenvolvimento e pode ser alterada, já que há entregadores com bastante punições sem darem churn, como entregadores que receberam poucas punições e deram churn.

ABAIXO, SERIA POSSÍVEL VISUALIZAR UMA IMAGEM DA TABELA COM DADOS FORNECIDOS PELO PARCEIRO PARA A CONSTRUÇÃO DO MODELO PREDITIVO. NO ENTANTO, EM RESPOSTA AO PEDIDO DO PARCEIRO PARA PROTEGER DADOS SENSÍVEIS, OS DADOS FORAM OCULTADOS.

Earnings por categoria (Imagem 17)

Fontes: Dados dos autores (2022)

Feature 2 - Quantitativo de Churns

→ A feature explora a hipótese de que o entregador que já deu churn uma vez, tem maiores chances de dar churn novamente. O principal motivo da escolha dessa feature, foi o simples fato e pensamento de que caso um entregador já tenha dado churn alguma vez durante seu tempo no aplicativo (período de 21 dias de inatividade é considerado churn) é possível que ele faça a mesma ação repetidas vezes, de forma que quanto maior o número de churns, maior a tendência a dar churn novamente.

Essa feature foi removida do modelo preditivo, pois estava dando viés ao nosso algoritmo, dado que, a máquina aprendia que o entregar já havia dado churn e automaticamente considerava que iria dar churn, sobressaindo essa feature em detrimento de todas as outras.

ABAIXO, SERIA POSSÍVEL VISUALIZAR UMA IMAGEM DA TABELA COM DADOS FORNECIDOS PELO PARCEIRO PARA A CONSTRUÇÃO DO MODELO PREDITIVO. NO ENTANTO, EM RESPOSTA AO PEDIDO DO PARCEIRO PARA PROTEGER DADOS SENSÍVEIS, OS DADOS FORAM OCULTADOS.

Quantitativo de churn (Imagem 18)

Fontes: Dados dos autores (2022)

Feature 3 - Modal;

→Essa feature partiu da hipótese de que, devido ao uso de certos modais, existe maior probabilidade dos entregadores darem churn, por conta da amplitude de entregas e, consequentemente, de ganhos. O principal motivo de sua escolha foi pelo viés de que há instabilidades nos ganhos dos RT's dependendo do modal utilizado, já que há diferença na taxa cobrada e capacidade de tipos de entregas. Por isso, nós quisemos analisar a discrepância, principalmente dos 3 modais pilares do sistema delivery, para futuramente verificar outros tópicos que envolvam esse tipo de feature.

ABAIXO, SERIA POSSÍVEL VISUALIZAR UMA IMAGEM DA TABELA COM DADOS FORNECIDOS PELO PARCEIRO PARA A CONSTRUÇÃO DO MODELO PREDITIVO. NO ENTANTO, EM RESPOSTA AO PEDIDO DO PARCEIRO PARA PROTEGER DADOS SENSÍVEIS, OS DADOS FORAM OCULTADOS.

ID's por modal e churn (Imagem 19)

Fontes: Dados dos autores (2022)

Feature 4 - Quantitativo de defects (média semanal);

→ A feature abaixo partiu da hipótese de que entregadores que possuem muitas entregas com defeito tem maior propensão a dar churn, por ser um fator prejudicial aos entregadores. Essa feature foi escolhida por apresentar uma possível relação entre: quanto maior o número de defects, maior será a probabilidade dos entregadores darem churn (por causa de estresse, insatisfação e variáveis emocionais).

É importante salientar que, a partir da movimentação da metodologia CRISP-DM, a feature 4 foi excluída em razão da inconsistência dos seus resultados e contaminação do modelo.

ABAIXO, SERIA POSSÍVEL VISUALIZAR UMA IMAGEM DA TABELA COM DADOS FORNECIDOS PELO PARCEIRO PARA A CONSTRUÇÃO DO MODELO PREDITIVO. NO ENTANTO, EM RESPOSTA AO PEDIDO DO PARCEIRO PARA PROTEGER DADOS SENSÍVEIS, OS DADOS FORAM OCULTADOS.

Média semanal de defects (Imagem 20)

Fontes: Dados dos autores (2022)

Feature 5 - Taxa de entregas canceladas;

→ A feature abaixo partiu da hipótese de que entregadores com altas taxas de cancelamento dos pedidos, tem maior propensão a darem churn. O motivo da escolha dessa feature, foi o devido o pensamento lógico de que, quanto maior cancelamento de pedidos, maior será a probabilidade de haver o churn dos entregadores, porém há a necessidade de uma reanálise e reestruturação da ideia dessa feature, pois há uma grande variância entre os dados e o resultado de possíveis churn ou não.

ABAIXO, SERIA POSSÍVEL VISUALIZAR UMA IMAGEM DA TABELA COM DADOS FORNECIDOS PELO PARCEIRO PARA A CONSTRUÇÃO DO MODELO PREDITIVO. NO ENTANTO, EM RESPOSTA AO PEDIDO DO PARCEIRO PARA PROTEGER DADOS SENSÍVEIS, OS DADOS FORAM OCULTADOS.

Entregas canceladas(Imagem 21)

Fontes: Dados dos autores (2022)

Feature 6 - Taxa de aceitação;

→A feature abaixo partiu da hipótese de que entregadores com uma taxa de aceitação baixa possuem uma maior propensão a dar churn. O motivo da escolha de tal feature, foi devido o meio que se apresentam os dados, em relação à: inconsistência nas taxas de índices de aceitação de pedido e números de churn, isso porque há diversos casos Outliers os quais não apresentam taxa de aceitação (possíveis entregadores fantasmas ou contas não utilizadas), ou até mesmo taxa de aceitação alta mas ainda apresenta churn. Mesmo com essas incompatibilidades, ainda é visível o fato de que quanto menor o número de aceitação de pedidos, maior probabilidade de entregadores darem churn.

ABAIXO, SERIA POSSÍVEL VISUALIZAR UMA IMAGEM DA TABELA COM DADOS FORNECIDOS PELO PARCEIRO PARA A CONSTRUÇÃO DO MODELO PREDITIVO. NO ENTANTO, EM RESPOSTA AO PEDIDO DO PARCEIRO PARA PROTEGER DADOS SENSÍVEIS, OS DADOS FORAM OCULTADOS.

ID's por taxa de aceitação, propensão ao churn (Imagem 22)

Fontes: Dados dos autores (2022)

Feature 7 - Auto-aceite;

→ A feature abaixo partiu da hipótese de que entregadores com o auto-aceite desativado também entregam em outras plataformas e possuem maiores chances de dar churn. Essa feature parte da ideia de que há grandes possibilidades de dupla jornada dos entregadores, sendo isso o que a possível grande maioria faz, de forma que continue no app que mais apresenta pedidos.

ABAIXO, SERIA POSSÍVEL VISUALIZAR UMA IMAGEM DA TABELA COM DADOS FORNECIDOS PELO PARCEIRO PARA A CONSTRUÇÃO DO MODELO PREDITIVO. NO ENTANTO, EM RESPOSTA AO PEDIDO DO PARCEIRO PARA PROTEGER DADOS SENSÍVEIS, OS DADOS FORAM OCULTADOS.

Auto-aceite por ID's (Imagem 23)

Fontes: Dados dos autores (2022)

Feature 8 - Earnings por categoria

→ A feature abaixo partiu da hipótese de que entregadores com ganhos abaixo da média de sua categoria tem uma maior propensão a dar churn. Já foi visto que os rookies e bronzes apresentam maior probabilidade de darem churn, porém, com a escolha dessa feature é possível analisar, pelo viés de earnings, que os entregadores em cada categoria se comportam de forma diferente dependendo do quantitativo que ganhavam antes.

ABAIXO, SERIA POSSÍVEL VISUALIZAR UMA IMAGEM DA TABELA COM DADOS FORNECIDOS PELO PARCEIRO PARA A CONSTRUÇÃO DO MODELO PREDITIVO. NO ENTANTO, EM RESPOSTA AO PEDIDO DO PARCEIRO PARA PROTEGER DADOS SENSÍVEIS, OS DADOS FORAM OCULTADOS.

Ganhos considerando a categoria (Imagem 24)

Fontes: Dados dos autores (2022)

4.4. Modelagem

a)Quais os algoritmos que foram escolhidos como adequados ao problema e aos dados e por que?

Os algoritmos escolhidos para a observação das possibilidades no modelo preditivo, primordialmente, foram a árvore de decisão, KNN, Naive Bayes, Random Forest, Regressão Logística, SVM e AdaBoost. Tais algoritmos foram pontuados a partir da recorrente utilização no mercado e classificação do Professor Instrutor Thiago, a fim de favorecer o processo de ensino-aprendizagem.

A árvore de decisão estabelece nós que se relacionam entre si por uma hierarquia, estabelecendo resultados a partir do histórico de dados fornecidos. Existe o nó-raiz, iniciante, e os nós-folha, que são os resultados finais.

```
[ ] dt = DecisionTreeClassifier() # depois tente com max_depth=2 e 3
    # Determinação da classe árvore de decisão.
    dt.fit( x_train, y_train.squeeze() ) # squeeze() -> df para series
    # .squezze remove dados do eixo y que seja de mesmo comprimento da entrada x.

    DecisionTreeClassifier()

[ ] # Teste de Acuracidade (accuracy)
    print('Acuracidade (treino): ', dt.score( x_train, y_train ))
    print('Acuracidade (teste): ', dt.score( x_test, y_test ))
    # Métrica utilizada.

    Acuracidade (treino):  0.9786427126198598
    Acuracidade (teste):  0.8648590530554795
```

Árvore de decisão e acuracidade (Imagem 25)

Fontes: Dados dos autores (2022)

```
[37] # Teste de Acuracidade (accuracy)
     y_pred = dt.predict(x_test)
     print( 'Revocação: ', recall_score( y_test, y_pred, pos_label="CHURN" ))
     print( 'Precisão: ', precision_score( y_test, y_pred, pos_label="CHURN" ))
     print( 'F1_score: ', f1_score( y_test, y_pred, pos_label="CHURN" ))

     Revocação:  0.7964815319347682
     Precisão:  0.9211379186172128
     F1_score:  0.8542862510175542
```

Métodos de avaliação: árvore de decisão (Imagem 26)

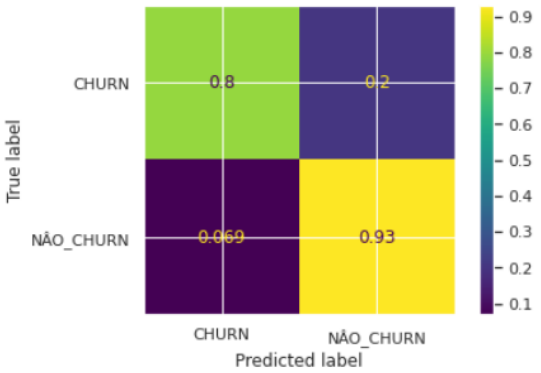
Fontes: Dados dos autores (2022)

A Matriz de confusão (Imagem 28) é utilizada para facilitar a visualização das frequências de classificação para churn ou não-churn do modelo.

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

Explicação da Matriz de Confusão (Imagem 27)

Fontes: Diego Nogare (2020)



Matriz de confusão: árvore de decisão (Imagem 28)

Fontes: Dados dos autores

- O verdadeiro positivo ocorre quando, no conjunto que está sendo analisado, o churn foi previsto corretamente;
- Já o falso positivo é observado quando o churn foi previsto de forma incorreta;
- Falso verdadeiro ocorre quando não-churn foi previsto corretamente;
- Por fim, o falso negativo ocorre quando o não-churn foi previsto incorretamente.

Já o algoritmo KNN, analisa os “vizinhos” mais próximos do dado que está em estudo no momento, classifica-o e conforme a sua classificação, admite-se que o dado de estudo pertence à mesma classe que ele.

```
[ ] # Instaciação do obj Algoritmo
    knn = KNeighborsClassifier(n_neighbors=7)
    # Treino # x = Features, y = Label/Target
    knn.fit( x_train, y_train.squeeze() ) # squeeze() -> df para series
    print('Acuracidade (treino): ', knn.score( x_train, y_train ))
    print('Acuracidade (teste): ', knn.score( x_test, y_test ))

    Acuracidade (treino):  0.790165873302902
    Acuracidade (teste):  0.7227391797317902
```

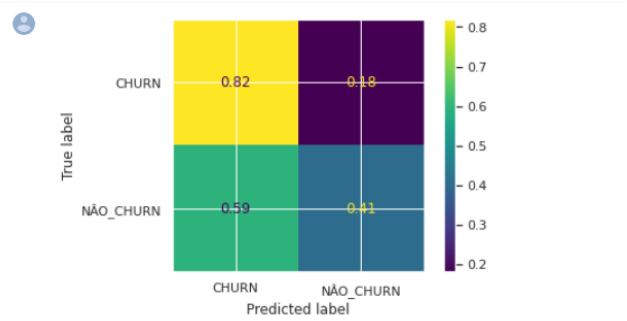
```
[ ] # Teste de Acuracidade (accuracy)
    y_pred = knn.predict(x_test)
    print( 'Revocação: ', recall_score( y_test, y_pred , pos_label="CHURN"))
    print( 'Precisão: ', precision_score( y_test, y_pred, pos_label="CHURN" ))
    print( 'F1_score: ', f1_score( y_test, y_pred, pos_label="CHURN" ))

    Revocação:  0.6641108473124976
    Precisão:  0.7544658648744252
    F1_score:  0.7064108137200108
```

Métodos de avaliação: KNN (Imagem 29)

Fontes: Dados dos autores (2022)

Matriz de confusão: KNN (Imagem 30)



Fontes: Dados dos autores (2022)

Observa-se, então, Naive Bayes como o algoritmo baseado na teoria de Bayes, composto por um estudo de probabilidades, no qual analisa-se os pares de features, considerando que cada uma é independente da outra.

```
[ ] dt = MultinomialNB() # depois tente com max_depth= 2 e 3
    dt.fit( x_train, y_train.squeeze() ) # squeeze() -> df para series
    print('Acuracidade (treino): ', dt.score( x_train, y_train ))
    print('Acuracidade (teste): ', dt.score( x_test, y_test ))

    Acuracidade (treino):  0.6157203319420959
    Acuracidade (teste):  0.6141455213668531

[ ] # Teste de Acuracidade (accuracy)
    y_pred = dt.predict(x_test)
    print( 'Revocação: ', recall_score( y_test, y_pred, pos_label="CHURN" ))
    print( 'Precisão: ', precision_score( y_test, y_pred, pos_label="CHURN" ))
    print( 'F1_score: ', f1_score( y_test, y_pred, pos_label="CHURN" ))

    Revocação:  0.8190946950531273
    Precisão:  0.5823993358239934
    F1_score:  0.6807595264281555
```

Métodos de avaliação: Naive Bayes (Imagem 31)

Fontes: Dados dos autores (2022)

O conceito de Random Forest é entendido como a criação de diversas árvores de decisão e os seus resultados são comparados entre si para verificar a sua constância. Indica-se o Random forest como o algoritmo com os melhores resultados, levando em consideração as features aplicadas na aprendizagem da máquina, visto que, o valor da acurácia (porcentagem de aproximação dos valores de teste do resultado real) foi o maior. Além disso, observa-se na apuração

da revocação, que consiste na identificação de todos os verdadeiros churns possíveis e englobando alguns falsos positivos, o maior resultado entre os outros algoritmos. Essa colocação significa que o modelo está mais próximo de acertar verdadeiros positivos.

```
[ ] modelo = RandomForestClassifier()
    modelo.fit( x_train, y_train.squeeze() )
    print('Acc treino: ', modelo.score(x_train, y_train ))
    print('Acc teste: ', modelo.score(x_test, y_test.squeeze() ))

    Acc treino:  0.978862638920113
    Acc teste:  0.894455956523439

[ ] y_pred = modelo.predict(x_test)
    print( 'Revocação: ', recall_score( y_test, y_pred, pos_label="CHURN" ))
    print( 'Precisão: ', precision_score( y_test, y_pred, pos_label="CHURN" ))
    print( 'F1_score: ', f1_score( y_test, y_pred, pos_label="CHURN" ))

    Revocação:  0.8543572179192777
    Precisão:  0.9298119281599457
    F1_score:  0.8904890367335349
```

Métodos de avaliação: Random Forest(Imagem 32)

Fontes: Dados dos autores (2022)

Quando apontado a regressão logística, nota-se que a saída do modelo é a probabilidade da observação ter classificação 0 ou 1.

```
[ ] modelo = LogisticRegression()
    modelo.fit( x_train, y_train.squeeze() )
    print('Acc treino: ', modelo.score(x_train, y_train ))
    print('Acc teste: ', modelo.score(x_test, y_test.squeeze() ))

    Acc treino:  0.656523991515732
    Acc teste:  0.6542010399968722

[ ] y_pred = modelo.predict(x_test)
    print( 'Revocação: ', recall_score( y_test, y_pred, pos_label="CHURN" ))
    print( 'Precisão: ', precision_score( y_test, y_pred, pos_label="CHURN" ))
    print( 'F1_score: ', f1_score( y_test, y_pred, pos_label="CHURN" ))

    Revocação:  0.7335850231580586
    Precisão:  0.6347837801427995
    F1_score:  0.6806174957118354
```

Métodos de avaliação: Regressão Logística (Imagem 33)

Fontes: Dados dos autores (2022)

SVM(Support Vector Machine) busca uma linha de separação entre as duas classificações aqui analisadas, além dos dois resultados, um de cada classe, mais próximos da outra.

```
[ ] modelo = SVC()
    modelo.fit( x_train, y_train.squeeze() )
    print('Acc treino: ', modelo.score(x_train, y_train ))
    print('Acc teste: ', modelo.score(x_test, y_test.squeeze() ))
```

```
[ ] y_pred = modelo.predict(x_test)
    print( 'Revocação: ', recall_score( y_test, y_pred, pos_label="CHURN" ))
    print( 'Precisão: ', precision_score( y_test, y_pred, pos_label="CHURN" ))
    print( 'F1_score: ', f1_score( y_test, y_pred, pos_label="CHURN" ))
```

‘Métodos de avaliação: SVM (Imagem 34)

Fontes: Dados dos autores (2022)

AdaBoost aplica pesos maiores às classificações errôneas feitas pelo modelo a partir de árvores de decisão, a fim de tratá-las e classificá-las da forma correta.

```
[ ] from sklearn.ensemble import AdaBoostClassifier
    modelo = AdaBoostClassifier()
    modelo.fit( x_train, y_train.squeeze())
    print('Acc treino: ', modelo.score(x_train, y_train ))
    print('Acc teste: ', modelo.score(x_test, y_test.squeeze() ))
```

```
Acc treino:  0.6886723293616273
Acc teste:  0.6862610939515972
```

```
[ ]
    y_pred = modelo.predict(x_test)
    print( 'Revocação: ', recall_score( y_test, y_pred, pos_label="CHURN" ))
    print( 'Precisão: ', precision_score( y_test, y_pred, pos_label="CHURN" ))
    print( 'F1_score: ', f1_score( y_test, y_pred, pos_label="CHURN" ))
```

Métodos de avaliação: AdaBoost (Imagem 35)

Fontes: Dados dos autores (2022)

É importante ressaltar que para a próxima etapa do projeto, que consiste na implementação de hiperparâmetros, selecionam-se os algoritmos de Random Forest, Decision Tree e KNN, a partir do destaque nos resultados obtidos na Revocação.

Atualização -> Hiperparâmetrização

Conforme a metodologia CRISP-DM permite, foram implementados, nos algoritmos, hiperparâmetros com a finalidade de verificar se é possível que os modelos obtenham resultados ainda melhores.

Para a maior agilidade no resultado, foi selecionado o hiperparâmetro Random Search, que em comparação ao Grid Search, o seu processamento é mais rápido, sendo assim, mais apropriado para o cliente.

Dessa forma, foram selecionados os modelos (algoritmos) que obtiveram os maiores valores de assertividade, assumindo que a Revocação é a principal métrica de avaliação, para que possa-se explorar suas tendências (melhorias, pioras ou conservação) quando aplicados os hiperparâmetros.

Atualização -> PyCaret e Deploy

A biblioteca PyCaret é uma ferramenta de low-code, que facilita o procedimento de análise, exploração, processamento e treinamento de dados. Após a escolha do Random Forest, como o algoritmo que adquiriu melhores resultados, o PyCaret foi utilizado para a validação do modelo.

O Pycaret apontou os excelentes resultados que o Random Forest alcançou, corroborando, assim, com as modelagens feitas manualmente pelos desenvolvedores.

Por fim, salva-se o modelo dado pelo PyCaret, além dos parâmetros correspondentes aos melhores para serem aplicados manualmente no Random Forest.

Atualização -> Final

A entrega é feita em um Collab final, no qual, salva-se o modelo do deploy (resultante do PyCaret), e aplica-se o SHAP (SHapley Additive exPlanations), que permite a classificação de 1 (menor probabilidade de churn) a 5 (maior probabilidade de churn), para cada amostra selecionada (ID do entregador).

Por fim, visando a facilidade do manuseio da ferramenta pelo Time de Operações da Rappi, os resultados classificatórios podem ser exportados em Excel, mecanismo amplamente utilizado na área.

b) Qual foi a estratégia de avaliação escolhida e por que?

Inicialmente, as estratégias de avaliação escolhidas e aplicadas durante o processo de formulação dos algoritmos, são as métricas de precisão, revocação e F1-score, que como anteriormente dito, vão exibir a diferentes verdadeiros positivos de todo o modelo, dos resultados de teste e a média desses valores.

Para melhor noção do que essas métricas abrangem, disserta-se a respeito de cada especificidade:

- Acurácia: Antes de qualquer resultado de avaliação dos algoritmos, temos o cálculo da sua acurácia. Ao ser aplicado, tanto o cálculo de treino quanto o cálculo de teste, o fator acurácia indica uma performance geral do modelo. “Dentre todas as classificações, quantas o modelo classificou corretamente”;
- Precisão: A métrica de precisão calcula, dentre todas as classificações de classe Positivo que o modelo fez, quantas estão corretas;
- Revocação: A métrica de revocação/recall calcula, dentre todas as situações de classe Positivo como valor esperado, quantas estão corretas;
- F1-Score: A métrica de F1-Score faz um cálculo da média harmônica entre precisão e revocação.

Assim, a estratégia de avaliação escolhida pelo grupo tange a observação do resultado da Revocação de todos os algoritmos testados. A partir desse resultado, foram definidos os três melhores algoritmos para a aplicação dos hiperparâmetros.

Atualização -> Hiperparâmetrização

Hiperparâmetros são variáveis que controlam o próprio processo de treinamento. O ajuste de hiperparâmetro realiza vários testes em um único treino.

Para o projeto, especificamente, utilizaremos o hiperparâmetro Random Search em detrimento ao Grid Search, visto que, o mesmo é processado de forma mais rápida. Além do mais,

nota-se, a partir da análise de requisitos feita com o stakeholder, como fundamentais premissas para o modelo preditivo, flexibilidade e rapidez.

Dessa forma, para a implementação dos hiperparâmetros, foram selecionados os algoritmos com melhor desempenho (Random Forest, Decision Tree e KNN), pautado na métrica de avaliação Revocação, que identifica todos os verdadeiros positivos podendo também englobar falsos positivos.

Para cada algoritmo selecionado, a documentação "scikit-learn"(SKlearn) indica determinados parâmetros que possuem especificações de melhoria.

```
parametros = {'criterion': ['gini', 'entropy','log_loss'],
              'max_depth': [4,90,5,422],
              'min_samples_split': [2, 4, 8]
```

Parâmetros da árvore de decisão (Imagem 36)

Fonte: Dados dos Autores (2022)

```
parametros = {'n_estimators': [101, 178],
              'criterion': ['gini', 'entropy','log_loss'],
             'max_depth': [4,90,5,29,78],
             'min_samples_split': [7, 15, 30] ]
```

Parâmetros do Random Forest (Imagem 37)

Fonte: Dados dos Autores (2022)

```
parametros = {'n_neighbors': [3, 5, 7, 9, 13, 17, 21, 29],
              'weights': ['uniform', 'distance'],
              'algorithm': ['auto',
                            'ball_tree',
                            'kd_tree',
                            'brute'],
              'leaf_size': [15, 30, 45, 60]}
```

Parâmetros do KNN (Imagem 38)

Fonte: Dados dos Autores (2022)

Justifica-se, então, a aplicação dos hiperparâmetros a fim de melhorar os resultados obtidos pelos algoritmos.

É de indubitável importância destacar que existem três possibilidades para os novos resultados das métricas avaliativas, após a aplicação dos hiperparâmetros. Os indicadores podem diminuir, aumentar ou estabelecer constância. Caso haja redução, é necessária a re-avaliação do algoritmo, e a não-utilização do hiperparâmetro. Se as métricas aumentarem, revela-se a relevância da aplicação do hiperparâmetro. Por fim, a invariabilidade indica que o algoritmo possui os melhores resultados possíveis e não é necessário o processo de melhoria.

Atualização -> Pycaret e Deploy

Aplica-se a ferramenta Pycaret, pois ela retorna os melhores resultados de cada modelo, apresentando, então, os parâmetros preferíveis para serem aplicados no algoritmo.

Após a utilização do Pycaret, foram coletados os parâmetros do modelo com os melhores resultados (Random Forest). A partir desses parâmetros apontados, testa-se manualmente o algoritmo, sem o uso da ferramenta, para assim prosseguir com o deploy final .

Para deploy do modelo, emprega-se o método dump, pertencente à biblioteca joblib. A biblioteca joblib é utilizada para facilitar a exportação de soluções preditivas de forma otimizada. Dessa forma, o cliente poderá importar o algoritmo de previsão, obedecendo o nome das features, que precisam ser semelhantes ao modelo de treino.

Dessa forma, o cliente poderá importar o algoritmo de previsão, obedecendo o nome das features, que precisam ser semelhantes ao modelo de treino.

c) Resultados preliminares obtidos

Quando realiza-se a análise da Acuracidade, Precisão, Revocação e F1-score foram dos algoritmos, encontra-se os seguintes resultados tabelados:

	KNN	Arvore de Decisão	Naive Bayes	Random Forest	Adaboost
Acuracidade	0.72	0.87	0.62	**0.89**	0.68
Precisão	0.75	0.92	0.58	**0.93**	0.69
Revocação	0.66	0.80	0.82	**0.85**	0.69
F1_score	0.70	0.85	0.68	**0.86**	0.68

Resultado da acuracidade e métricas em cada algoritmo (Tabela 3)

Fonte: Dados dos autores (2022)

Com base nesses valores, concluímos que o Random Forest apresenta os melhores resultados para as métricas que mais valorizamos no modelo preditivo. Em seguida, obtém-se Decision Tree e KNN.

O indicador SVM não está presente na documentação, visto que, a partir da tentativa de resolução no Google Colaboratory não foi possível ter os resultados de forma minimamente satisfatória para a solução desejada.

Já o Adaboost, consiste na técnica estatística de classificação e combinação de modelos de aprendizagem fracos e transformá-los em modelos fortes. Ao analisar o nível de acurácia e performance geral do modelo obtivemos os seguintes resultados: Acurácia (treino): 0.69, Acurácia (teste): 0.68, Revocação: 0.68; Precisão: 0.68; e F1-score: 0.68. Percebe-se, então, que o grau de acerto é baixo em relação aos outros algoritmos utilizados. Além disso, evidencia-se que esse algoritmo segue um padrão similar aos resultados encontrados na Regressão Logística e Naive Bayes.

A fim de dar continuidade ao procedimento de implementação, destaca-se os algoritmos com melhores resultados(Random Forest, Decision Tree e KNN), para que a partir da aplicação do hiperparâmetro Random Search, analisar o comportamento dos resultados obtidos pelos três melhores modelos.

Atualização -> Hiperparâmetrização

Primordialmente, evidencia-se que cada algoritmo exibiu uma possibilidade de resultado após a aplicação dos hiperparâmetros, dado que, os resultados podem melhorar, piorar ou continuar da mesma forma.

Salienta-se, então, que o Random Forest obteve uma diminuição média de 2%. Logo, nota–se que o algoritmo não precisa de melhorias, já que seus resultados iniciais são satisfatórios.

Já no modelo da árvore de decisão, observa-se a permanência da maioria dos resultados. Esse fator expressa que o treinamento e teste do algoritmo foi feito da forma correta, portanto, não é necessário fazer melhorias.

Por fim, ressalta-se o aumento expressivo de 9.2% no KNN. É possível constatar a possibilidade de melhoria do algoritmo, feita pelo hiperparâmetro.

Modelo Preditivo		**Acurácia (treino)**	**Acurácia (teste)**	**Precisão**	**Revocação**	**F1-score**	**Best Score**
Random Forest	pré hiperparâmetros	98%	89%	85%	93%	89%	
	hiperparâmetros	97%	87%	83%	90%	87%	86%
Árvore de Decisão	pré hiperparâmetros	98%	87%	80%	92%	86%	
	hiperparâmetros	98%	86%	86%	80%	92%	85%

KNN	pré hiperparâmetros	79%	72%	67%	76%	71%	
	hiperparâmetros	97%	82%	70%	92%	80%	82%

Tabela comparativa dos resultados obtidos nos algoritmos (Tabela 4)

Fonte: Dados dos autores (2022)

Atualização -> Pycaret e Deploy

Após a execução da ferramenta Pycaret, que tem como finalidade maximizar os resultados, obtivemos os seguintes parâmetros do modelo Random forest:

"RandomForestClassifier(bootstrap=True, ccp_alpha=0.0, class_weight=None,

criterion='gini', max_depth=None, max_features='auto',

max_leaf_nodes=None, max_samples=None,

min_impurity_decrease=0.0, min_impurity_split=None,

min_samples_leaf=1, min_samples_split=2,

min_weight_fraction_leaf=0.0, n_estimators=100,

n_jobs=-1, oob_score=False, random_state=8178, verbose=0,

warm_start=False)"

Os parâmetros do Random Forest foram coletados e aplicados manualmente, isso foi necessário porque algumas modificações que o Pycaret fez acabou desbalanceando o resultado.

Para a importação e utilização do modelo que foi criado, novamente será necessário utilizar a biblioteca joblib, através do método load, que carrega o algoritmo de previsão.

Após carregar o método, deve-se aplicar o método predict_proba(), que exibe a probabilidade de cada categoria ocorrer (churn ou não churn). Com essa informação, foi

desenvolvido um algoritmo que classifica a probabilidade do entregador dar churn (de 1 a 5), de acordo com a solicitação do cliente. Segue código abaixo:

```
# a estrutura abaixo tem como finalidade classificar as probabilidades de churn em categorias de 1 a 5 (do menos a mais propenso a dar churn)

prob_predict = []

for i in range(len(previsionRF)):
  # entregadores com probabilidade de churn maior que 0.8 são classificados com 5
  if previsionRF[i][1] >= 0.8:
    prob_predict.append(5)

  # entregadores com probabilidade de churn entre 0.8 e 0.65 são classificados com 4
  elif previsionRF[i][1] < 0.8 and previsionRF[i][1] >= 0.65:
    prob_predict.append(4)

  # entregadores com probabilidade de churn entre 0.65 e 0.35 são classificados com 3
  elif previsionRF[i][1] < 0.65 and previsionRF[i][1] >= 0.35:
    prob_predict.append(3)

  # entregadores com probabilidade de churn entre 0.35 e 0.15 são classificados com 2
  elif previsionRF[i][1] < 0.35 and previsionRF[i][1] >= 0.15:
    prob_predict.append(2)

  # entregadores com probabilidade de churn abaixo de 0.15 são classificados com 1
  elif previsionRF[i][1] < 0.15:
    prob_predict.append(1)
```

Classificação do entregador (Imagem 39)

Fonte: Dados dos Autores (2022)

Finalmente, após realizarmos a predição, se faz necessário salvar a nova coluna de previsão, para que assim o cliente possa visualizar os entregadores mais propensos a dar churn da plataforma.

Para salvar a nova coluna, primeiramente é necessário converter o data frame em excel (xlsx) para depois salvar o arquivo e realizar o download.

```
# criando um arquivo excel
datatoexcel = pd.ExcelWriter('Predict_Churn.xlsx')

# transformando o dataframe em excel
df_final.to_excel(datatoexcel)

# salvando o excel
datatoexcel.save()
print('DataFrame is written to Excel File successfully.')

# baixando o documento
from google.colab import files
files.download('Predict_Churn.xlsx')
```

```
DataFrame is written to Excel File successfully.
```

Salvando os resultados em Excel (Imagem 40)

Fonte: Dados dos Autores (2022)

4.5. Avaliação

a)Análise dos resultados (Qual a qualidade dos resultados? Qual a taxa de erro? Os algoritmos escolhidos foram adequados? Forneça exemplos e justifique suas respostas) (seção 4.5)

É significativo ressaltar, que para a construção da solução, utiliza-se a metodologia CRISP-DM. Dessa forma, a análise de resultados, primordialmente, será realizada a partir dos primeiros testes em algoritmos de predição, e no decorrer das passagens cíclicas pelas fases da metodologia, observações e mudanças serão atualizadas na documentação.

Como destacado nas seções anteriores, foram utilizados 7 (sete) algoritmos, sendo eles: árvore de decisão, KNN, Naive Bayes, Random Forest, Regressão Logística, SVM e AdaBoost. Para que melhores resultados fossem encontrados, evidencia-se que fatores de avaliação precisam ser determinados, dessa forma, é possível discorrer a respeito do seu grau de precisão. Então, após a análise dos operadores e métodos utilizados, evidencia-se o detalhamento da Acuracidade, Precisão, Revocação e F1-score. Esses fatores determinam a proximidade dos resultados obtidos pelo modelo próximo ao que é expresso pela realidade de predição, próxima de 100% (cem por cento), ressaltando que dados com total acerto podem indicar viés ou impureza nos dados.

Ao verificar-se todos os valores obtidos pelos modelos preditivos (seção 4.4), é possível visualizar o excelente resultado do algoritmo Random Forest em comparação aos demais, visto que, os resultados dos métodos de avaliação foram os maiores. Além do mais, com o intuito de estabelecer as análises seguintes (hiperparâmetrização) da forma mais abrangente possível e adequada para a solução, seleciona-se Decision Tree e KNN (algoritmos que tiveram resultados mais próximos do Random Forest). Esses valores podem ser verificados na tabela 4.

A fim de garantir a qualidade dos dados, é importante evidenciar que anteriormente ao treinamento dos modelos, a base de dados foi re-balanceada (Imagem 35) e houve novamente a limpeza dos mesmos (Imagem 36). Além disso, para melhor otimizar o treinamento, os dados organizados para essa função foram selecionados de forma randomizada(Imagem 35).

O erro relativo percentual pode ser calculado a partir da diferença entre o valor da acuracidade de treino e de teste dividido pela acuracidade de teste. Ressalta-se os valores de acuracidade encontrados no modelo Random Forest, Acuracidade (treino): 0.97 e Acuracidade (teste): 0.88 (Imagem 31). O resultado obtido em erro relativo no modelo utilizado é de 0.9%.

Em seguida, observa-se os resultados encontrados no algoritmo Decision Tree, no qual, identifica-se a Acuracidade (treino): 0.98 e Acuracidade (teste): 0.87. O erro relativo obtido é de 0.13%.

Por fim, o erro relativo do KNN é calculado a partir da Acuracidade (treino): 0.80 e Acuracidade (teste): 0.73. Obtém-se como resolução 0.1%.

A partir da resultância dos erros relativos, expressos por cada algoritmo, é possível perceber o grau de assertividade dos modelos e a pequena margem de erro, adequada para o modelo preditivo requerido.

```
# Balanceamento das features

oversample = RandomOverSampler(sampling_strategy='minority')
x_over, y_over = oversample.fit_resample(x, y)


# Dividindo dados para treino e dados para teste
x_train, x_test, y_train, y_test = train_test_split(x_over, y_over,
                                                    test_size = 0.2,
                                                    random_state = 1) # qualquer valor como semente do pseudo-random
```

Balanceamento das features (Imagem 41)

Fontes: Dados dos autores (2022)

```
all_storekeepers = all_storekeepers.drop(columns = ["Churn(sim=1/não=0)_x"])
auto_accept = auto_accept.drop(columns = ["Churn(sim=1/não=0)_x"])
churns_quantitative = churns_quantitative.drop(columns = ["Churn(sim=1/não=0)_x"])
defects_mean = defects_mean.drop(columns = ["Churn(sim=1/não=0)_x"])
media_modal = media_modal.drop(columns = ["Churn(sim=1/não=0)_x"])
punishment_quantitative = punishment_quantitative.drop(columns = ["Churn(sim=1/não=0)_x"])
storekeeper_cancel_tax = storekeeper_cancel_tax.drop(columns = ["Churn(sim=1/não=0)_x"])
storekeeper_category = storekeeper_category.drop(columns = ["Churn(sim=1/não=0)_x"])
```

Limpeza dos dados duplicados (Imagem 42)

Fontes: Dados dos autores (2022)

Atualização -> Hiperparametrização

Inicialmente, ressalta-se que com a implementação da condição hiperparametrizada, os resultados correspondentes ao Random Forest sofreram uma queda.

Para que seja analisado o erro relativo (métrica de comparação entre os erros absolutos e de teste) dentro de tal modelo, utiliza-se a Acuracidade (treino): 0.98 e Acuracidade (teste): 0.88. É possível obter o erro relativo de 0.11.

O Random Forest demonstra que o algoritmo aplicado isoladamente, isto é, sem a aplicação dos hiperparâmetros, foi treinado e testado da melhor forma e atingirá resoluções preferíveis.

Como anteriormente abordado, com a aplicação dos hiperparâmetros na Decision Tree, nota-se a constância entre as métricas de avaliação. Esse fator evidencia a qualidade do treinamento e testagem do modelo.

Para a validação da assertividade de tal, com a implementação da hiperparametrização, calcula-se o erro relativo, condizente à Acuracidade (treino): 0.98 e Acuracidade (teste): 0.87. Portanto, a resolução adquirida é de 0,13%.

O algoritmo demonstra excelente desempenho de treinamento e teste, sem precisar de melhorias e parametrizações específicas, entretanto, não possui os melhores resultados nas métricas de avaliação quando comparado ao Random Forest, por exemplo.

Já quando aborda-se a respeito do KNN, evidencia-se que a implementação dos hiperparâmetros possibilitaram a melhoria relevante do modelo. A partir desse fato, constata-se a aplicabilidade da hiperparametrização em aprimorar os resultados do algoritmo.

A fim de garantir a maior exatidão do modelo, verifica-se o erro relativo que cabe ao algoritmo, tendo como base a Acuracidade (treino): 0.98 e Acuracidade (teste): 0.82. O erro relativo resultante é de 0.19.

Por fim, o KNN apresenta excelentes resultados quando os hiperparâmetros são aplicados, todavia, ainda são baixos quando em comparação com o Random Forest. **Atualização -> Pycaret e Deploy**

É observável que a utilização do PyCaret para validar o Random Forest, garante que tal algoritmo é a melhor opção para ser utilizada no modelo. Além do mais, é possível obter os parâmetros que, manualmente, alcançam os melhores valores.

Evidencia-se, então, a comparação do Random Forest sem hiperparâmetros (Imagem 32), aplicado hiperparâmetros em aleatoriedade (Tabela 4) e recomendados pelo PyCaret (Imagem 43). É notável, a melhoria nos resultados.

```
[ ] # puxando as metricas para analisar se o modelo está apresentando um bom resultado
    y_pred = rf.predict(x_test)
    print('Acc treino: ', rf.score(x_train, y_train ))
    print('Acc teste: ', rf.score(x_test, y_test.squeeze() ))
    print( 'Revocação: ', recall_score( y_test, y_pred))
    print( 'Precisão: ', precision_score( y_test, y_pred ))
    print( 'F1_score: ', f1_score( y_test, y_pred ))

    Acc treino:  0.9786036146109259
    Acc teste:  0.8904484497790984
    Revocação:  0.8475849453158448
    Precisão:  0.9280630726614106
    F1_score:  0.8860002441108263
```

Resultados do Random Forest após a aplicação dos parÂmetros indicados pelo PyCaret (Imagem 43)

Fonte: Dados dos Autores (2022)

4.6 Comparação de Modelos

O trabalho de comparação dos modelos consiste, primeiramente, na busca do hiperparâmetro que mais se adequam ao modelo, gerando uma melhoria nas métricas (Precisão,

Acurácia, Revocação, F1-score), e posteriormente, realiza-se a análise comparativa entre os algoritmos pré-selecionados, para que seja elegido aquele que apresenta as melhores métricas para o modelo de predição.

Portanto, conforme expresso nas seções anteriores, foram aplicados hiperparâmetros nos modelos que apresentaram as melhores resultados nas métricas de avaliação (Random Forest, Decision Tree, KNN).

Ressalta-se ainda, que para satisfazer o requisito do cliente e levando em consideração as nuances do negócio, o hiperparâmetro escolhido para o modelo de predição configura-se no Random Search, dado que, a sua agilidade é expressivamente superior ao Grid Search e, logo, um facilitador para a equipe de operações que precisa observar os fatores que têm maior impacto no churn dos entregadores.

O modelo preditivo com melhores resultados escolhido foi o *Random Forest,* que apontou um nível de assertividade de treino de 0.98, e Acurácia de teste de 0.89. Esse modelo trouxe tendências positivas, com valores de Revocação, Precisão e F1-score altos.

Ao configurar os parâmetros do Random Forest, chegamos no Best Score, com um valor de 0.86. E com base nos resultados obtidos percebemos que nossos valores de acurácia, revocação, precisão e f1-score diminuíram, quando comparado aos resultados obtidos pré configuração dos hiperparâmetros. Entretanto, concluímos que a diferença de valores não foi alta, mas mesmo assim como podemos ver nas informações acima, vimos que o Random Forest foi o modelo preditivo que nos apresentou os melhores resultados.

5. Conclusões e Recomendações

O projeto possui resultados excelentes, a partir das features aplicadas, o que garante a maior quantidade de verdadeiros positivos e precisão do modelo preditivo. Ademais, a equipe elaborou

documentações detalhadas a cada etapa efetuada na construção da modelagem, tanto no presente documento quanto no Google Collaboratory.

O modelo predito foi projetado para que o cliente possa inserir as features de sua preferência, visto que, tais informações são variáveis ou incrementais com o passar do tempo. Desse modo, o Time de Operações precisa, apenas, selecionar o ID do entregador que gostaria de averiguar a propensão de churn. É importante ressaltar que, caso seja a escolha do time, as porcentagens escolhidas para classificar o entregador de 1 a 5 podem ser substituídas.

Em síntese, a fim de incluir o modelo preditivo próximo às ferramentas de trabalho utilizadas pelo Time de Operações da Rappi, é possível exportar os resultados em Excel.

O impacto causado pelo mecanismo é latente, dado que, caso o RT esteja próximo de dar churn, intervenções serão feitas. Portanto, durante o tratamento de dados que será feito caso o stakeholder mude a base de dados, é de extrema importância a realização da limpeza e balanceamento dos dados. Dessa maneira, mitiga-se a possibilidade de viés nos dados aplicados aos modelo de predição.

6. Referências

SCIKIT LEARN. Scikit Learn. *In*: Scikit Learn. [*S. l.*], [20]. Disponível em: https://scikit-learn.org/stable/. Acesso em: 8 ago. 2022.

MINISTÉRIO DA SAÚDE. Resolução nº 466, de 12 de dezembro de 2012. Pesquisa envolvendo seres humanos. RESOLUÇÃO Nº 466, DE 12 DE DEZEMBRO DE 2012, [*S. l.*], 12 dez. 2012. Disponível em: https://conselho.saude.gov.br/resolucoes/2012/Reso466.pdf. Acesso em: 8 set. 2022.

MINISTÉRIO DA SAÚDE. Resolução nº 510, de 7 de abril de 2016. Pesquisa envolvendo seres humanos. RESOLUÇÃO Nº 510, DE 7 DE ABRIL DE 2016, [*S. l.*], 7 abri. 2016. Disponível em: https://bvsms.saude.gov.br/bvs/saudelegis/cns/2016/res0510_07_04_2016.html. Acesso em: 8 set. 2022.

GS & MD; INSTITUTO FOOD SERVICE BRASIL. Panorama do mercado de Food Service no Brasil. *In*: Panorama do mercado de Food Service no Brasil. [*S. l.*], 17 abr. 2022. Disponível em: https://mercadoeconsumo.com.br/17/04/2015/noticias/panorama-mercado-de-food-service-brasil/. Acesso em: 18 ago. 2022.