PONTIFÍCIA UNIVERSIDADE CATÓLICA DE MINAS GERAIS
NÚCLEO DE EDUCAÇÃO A DISTÂNCIA
Pós-graduação *Lato Sensu* em Ciência de Dados e Big Data

ROBERTO TEIXEIRA DE OLIVEIRA

PREVISÃO DE RISCO DE INADIMPLÊNCIA COM AUXÍLIO DE DADOS SOCIOECONÔMICOS

Belo Horizonte
2022

ROBERTO TEIXEIRA DE OLIVEIRA

PREVISÃO DE RISCO DE INADIMPLÊNCIA COM AUXÍLIO DE DADOS SOCIOECONÔMICOS

Trabalho de Conclusão de Curso apresentado ao Curso de Especialização em Ciência de Dados e Big Data como requisito parcial à obtenção do título de especialista.

Belo Horizonte
2022

SUMÁRIO

1. Introdução

1.1. Contextualização

"Em dezembro de 2021, o número de famílias que informaram estar endividadas atingiu uma marca de 76,3%, segundo dados da CNC (Confederação Nacional do Comércio de Bens, Serviços e Turismo). É o maior valor desde o início da pesquisa, em 2010."

Esta é apenas mais uma notícia publicada, desta vez, pelo Jornal Nexo em sua página no dia primeiro de fevereiro de 2022, que evidencia uma característica que o brasileiro adquiriu nos últimos anos, o endividamento. Pesquisadores explicam, que a alta da inflação e por consequência o aumento dos preços dos itens básicos, fizeram com que as famílias optassem por direcionar a renda para os itens essenciais, e para as demais despesas buscassem formas atrativas de pagamento da dívida, como renegociação e contratação de prazos mais longos, possibilitando às famílias condições de ampliar o endividamento e o consumo, porém, mantendo a sua capacidade de pagamento.

No entanto, boa parte destas famílias não conseguem estes "benefícios", e para estas, restam realizar empréstimos pessoais, mesmo com juros elevados, buscando o controle de suas finanças.

Sabendo destas condições, as empresas que fazem concessões de créditos, buscam cada vez mais identificar os melhores clientes na visão financeira, visando o aumento de seus lucros.

É neste ponto, que apresenta-se como uma das soluções mais utilizadas para este problema, a análise de risco de crédito, sendo esta, a probabilidade de um cliente faltar com a quitação de qualquer tipo de dívida. Neste trabalho, iremos apresentar na prática um problema de análise de crédito e como solucionar através de técnicas de *Machine Learning.*

1.2. O problema proposto

A análise de risco de crédito é a maneira como a empresa prevê o descumprimento de acordos, se revelando como o potencial de perda financeira decorrente do não pagamento do principal e dos juros gerados nas condições do empréstimo.

O juros, é uma forma de compensar o credor pelo risco de crédito. Logo, a análise de crédito é fundamental na definição do percentual de juros que será cobrado na operação.

Mas além dos empréstimos concedidos, a análise de crédito ganha cada vez mais espaço no mercado financeiro na solicitação de novos cartões de crédito. Além dos dados concedidos pelo cliente no momento da solicitação, hoje, facilmente é possível adquirir novos dados do solicitante através de empresas que vendem informações exatamente com este propósito, servindo de insumo para predições que levarão a tomada de decisão do companhia de aprovar ou não o início da linha de crédito solicitada, além do limite estabelecido.

Neste trabalho, apresentaremos a resolução de um desafio proposto pela Nubank em 2018, chamado *Data Challenge*. O problema traz como ideia proporcionar aos participantes uma experiência das tarefas do cotidianas de um cientista de dados que trabalha em uma grande companhia financeira.

1.2.1 Sobre o Nubank

O Nubank é uma *fintech*, uma empresa de tecnologia que cria soluções para o mercado financeiro. Não se encaixando como um banco tradicional, o Nubank nasceu para diminuir a burocracia, sendo todos os seus problemas resolvidos via aplicativo.

1.2.2 Sobre o *Data Challeng*

A problemática do desafio proposto pelo Nubank, consistia em construir 3 modelos de *Machine Learning*, sendo eles:

- Risco de Crédito: Previsão do risco de inadimplência de um cliente utilizando seus dados de aquisição;
- Propensão a Gasto: Distinguir os clientes de acordo com a sua propensão a gasto;

- Fraude: Identificação de possíveis fraudadores.

Nosso estudo, será baseado na resolução do modelo de Risco de crédito, em que, deve-se prever a probabilidade de um cliente Nubank deixar de pagar a sua fatura de crédito e se tornar inadimplente.

1.2.3 Sobre a base disponibilizada

O Nubank disponibilizou para o desafio diversos arquivos que atendem aos três modelos propostos, para o atual trabalho, o arquivo de interesse é chamado de "acquisition_train.csv", este, contêm as informações do solicitante coletadas no momento da requisição, assim como variáveis extras coletadas pelo Nubank através de empresas terceiras com informações financeiras.

Os dados concedidos seguem o formato dos dados reais que utiliza-se no Nubank. Portanto, boa parte das variáveis possuem significado e valores iguais às que lida-se diariamente. Porém, como trata-se de dados sobre informações pessoais, nenhum dos mesmos são referentes a pessoas reais.

O enriquecimento dos dados é a adição de novas informações na base através de outras fontes, visando identificar novas variáveis que somando as informações iniciais, acrescentem na análise.

Na atualidade, é possível encontrar diversas fontes que são facilmente utilizadas como enriquecimento. Nesta ocasião, serão utilizados dados econômicos do Índice de Desenvolvimento Humano (IDH), junto a dados do Censo dos municípios brasileiros de 2010. Estes foram coletados na comunidade Kaggle, famosa por ser utilizada para divulgação de desafios da área de Ciência de Dados.

1.3. Objetivos

Este trabalho terá como objetivo principal construir um modelo de *Machine Learning* que seja capaz de apresentar a probabilidade de um cliente se tornar inadimplente, a partir dos dados coletados de sua aquisição, junto a dados socioeconômicos como fonte de enriquecimento.

Para chegar neste objetivo, assim como toda análise de *Machine Learning*, é possível dividir a mesma em etapas, trataremos cada uma como um objetivo específico.

1. Seleção de variáveis: Identificar quais variáveis foram escolhidas para o modelo final, apresentando os critérios utilizados.
2. Ajuste do Modelo: Dentre os algoritmos testados, apresentar qual o melhor e o porquê este foi escolhido.
3. Variáveis importantes: Apresentar quais variáveis apresentaram uma maior importância para o modelo escolhido.
4. Validação: Apresentar quais métricas foram utilizadas para validação.

Acima de tudo, a finalidade deste trabalho é apresentar um passo a passo da resolução de um problema de ciência de dados, passando por todas as etapas necessárias para o processo.

2. Coleta de Dados

O presente estudo trata-se de um *Data Challenge* promovido pelo Nubank em 2018, seus dados foram concedidos na época e diversos participantes aproveitaram a oportunidade para disponibilizar seus resultados em portfólios pessoais. Para este trabalho, utilizou-se os dados concedidos do desafio no github (Plataforma de hospedagem de códigos e arquivos de controle de versão usando o Git) por um usuário, este armazenou toda a documentação fornecida pela companhia no período.

É possível encontrar o mesmo no link https://github.com/luanprates/nubank-challenge/blob/master/questions.md.

2.1 Dados de aquisição

Para ajustar o modelo de Risco de crédito, utilizou-se dados de duas fontes de informações, a principal será os dados de aquisição fornecidos pela companhia no desafio, esta conta com as seguintes colunas.

Coluna	Tipo	Descrição
ids	String	identificador único de um solicitante
email	String	Provedor de e-mail do solicitante
tags	String	Tags descritivas dadas pelo provedor de dados
score_1	String	Score de crédito 1, categorias
score_2	String	Score de crédito 2, categorias
score_3	Float	Score de crédito 3
score_4	Float	Score de crédito 4
score_5	Float	Score de crédito 5
score_6	Float	Score de crédito 6
risk_rate	Float	Risco associado ao solicitante
last_amount_borrowed	Float	Valor do último empréstimo que o solicitante tomou
last_borrowed_in_months	Int	Duração do último empréstimo que o solicitante tomou
reason	String	Razão pela qual foi feita uma consulta naquele cpf
income	Float	Renda estimada pelo provedor dos dados para o solicitante
facebook_profile	Bool	Se o solicitante possui perfil no Facebook
state	String	Estado de residência do solicitante
zip	String	Código postal do solicitante
shipping_zip_code	Int	Código do endereço de entrega
shipping_state	String	Estado do endereço de entrega
channel	String	Canal pelo qual o solicitante aplicou

job_name	String	Profissão do solicitante
real_state	String	Informação sobre habitação do solicitante
ok_since	Float	Quantidade de dias que
n_bankruptcies	Float	Quantidade de bancarrotas que o solicitante já experimentou
n_defaulted_loads	Float	Quantidade de empréstimos não pagos no passado
n_accounts	Float	Número de contas que o solicitante possui
n_issues	Float	Número de reclamações de terceiros feitas em alguma das contas do solicitante
user_agent	String	Informação sobre dispositivo usado para a aplicação
reported_income	Int	Renda informada pelo próprio aplicante
profile_phone_number	String	Número de telefone, ex: 210-2813414
marketing_channel	String	Canal de marketing pelo qual o solicitante chegou na página de pedido de crédito
lat_lon	Object	Latitude e longitude da localização
external_data_provider_fraud_score	Int	Score de fraude
external_data_provider_first_name	String	Primeiro nome do solicitante
external_data_provider_email_seen_before	String	Se o e-mail já foi consultado junto ao provedor de dados
external_data_provider_credit_checks_last_year	Int	Quantidade de consultas de crédito na janela de um ano
external_data_provider_credit_checks_last_month	Int	Quantidade de consultas de crédito na janela de um mês
external_data_provider_credit_checks_last_2_year	Int	Quantidade de consultas de crédito na janela de dois anos
application_time_in_funnel	Int	Tempo gasto pelo solicitante durante o processo de aplicação
application_time_applied	Date	Horário de aplicação
profile_tags	String	Tags
target_fraud	String	Variável default para o modelo de Fraude
target_default	String	Variável default para o modelo de Inadimplência

Tabela 1: Variáveis de aquisição.

2.2 Enriquecimento com dados socioeconômicos

Para o enriquecimento da base de aquisição, utilizou-se dados econômicos dos municípios brasileiros, disponibilizados através na comunidade Kaggle. A base é contida por diversas informações de diferentes áreas, porém, filtrou-se apenas as variáveis relacionadas

a economia de cada município. O link para acessar o conjunto de dados está disponível em https://www.kaggle.com/datasets/pauloeduneves/hdi-brazil-idh-brasil?select=desc.csv.

Coluna	Tipo	Descrição
corte1	Float	Renda per capita máxima do 1º quinto mais pobre
corte2	Float	Renda per capita máxima do 2º quinto mais pobre
corte3	Float	Renda per capita máxima do 3º quinto mais pobre
corte4	Float	Renda per capita máxima do 4º quinto mais pobre
corte9	Float	Renda per capita mínima do décimo mais rico
gini	Float	Índice de Gini
pind	Float	% de extremamente pobres
pindcri	Float	% de crianças extremamente pobres
pmpob	Float	% de pobres
pmpobcri	Float	% de crianças pobres
ppob	Float	% de vulneráveis à pobreza
ppobcri	Float	% de crianças vulneráveis à pobreza
pren10ricos	Float	Percentual da renda apropriada pelos 10% mais ricos
pren20	Float	Percentual da renda apropriada pelos 20% mais pobres
pren20ricos	Float	Percentual da renda apropriada pelos 20% mais ricos
pren40	Float	Percentual da renda apropriada pelos 40% mais pobres
pren60	Float	Percentual da renda apropriada pelos 60% mais pobres
pren80	Float	Percentual da renda apropriada pelos 80% mais pobres
prentrab	Float	% da renda proveniente de rendimentos do trabalho
r1040	Float	Razão 10% mais ricos / 40% mais pobres
r2040	Float	Razão 20% mais ricos / 40% mais pobres
rdpc	Float	Renda per capita
rdpc1	Float	Renda per capita média do 1º quinto mais pobre
rdpc10	Float	Renda per capita média do décimo mais rico
rdpc2	Float	Renda per capita média do 2º quinto mais pobre
rdpc3	Float	Renda per capita média do 3º quinto mais pobre
rdpc4	Float	Renda per capita média do 4º quinto mais pobre
rdpc5	Float	Renda per capita média do quinto mais rico
rdpct	Float	Renda per capita , exceto renda nula
rind	Float	Renda per capita média dos extremamente pobres
rmpob	Float	Renda per capita média dos pobres
rpob	Float	Renda per capita média dos vulneráveis à pobreza
theil	Float	Índice de Theil - L - Mede a Desigualdade segundo a renda domiciliar percapita
cpr	Float	% de trabalhadores por conta própria - 18 anos ou mais
emp	Float	% de empregadores - 18 anos ou mais
p_agro	Float	% dos ocupados no setor agropecuário - 18 anos ou mais
p_com	Float	% dos ocupados no setor comércio - 18 anos ou mais
p_constr	Float	% dos ocupados no setor de construção - 18 anos ou mais
p_extr	Float	% dos ocupados no setor extrativo mineral - 18 anos ou mais
p_formal	Float	Grau de formalização dos ocupados - 18 anos ou mais

p_fund	Float	% dos ocupados com fundamental completo - 18 anos ou mais
p_med	Float	% dos ocupados com médio completo - 18 anos ou mais
p_siup	Float	% dos ocupados no SIUP - 18 anos ou mais
p_serv	Float	% dos ocupados no setor serviços - 18 anos ou mais
p_super	Float	% dos ocupados com superior completo - 18 anos ou mais
p_transf	Float	% dos ocupados na indústria de transformação - 18 anos ou mais
ren0	Float	% dos ocupados sem rendimento - 18 anos ou mais
ren1	Float	% dos ocupados com rendimento de até 1 s.m. - 18 anos ou mais
ren2	Float	% dos ocupados com rendimento de até 2 s.m. - 18 anos ou mais
ren3	Float	% dos ocupados com rendimento de até 3 s.m. - 18 anos ou mais
ren5	Float	% dos ocupados com rendimento de até 5 s.m. - 18 anos ou mais
renocup	Float	Rendimento médio dos ocupados - 18 anos ou mais
t_ativ	Float	Taxa de atividade - 10 anos ou mais
t_ativ1014	Float	Taxa de atividade - 10 a 14 anos
t_ativ1517	Float	Taxa de atividade - 15 a 17 anos
t_ativ1824	Float	Taxa de atividade - 18 a 24 anos
t_ativ18m	Float	Taxa de atividade - 18 anos ou mais
t_ativ2529	Float	Taxa de atividade - 25 a 29 anos
t_des	Float	Taxa de desocupação - 10 anos ou mais
t_des1014	Float	Taxa de desocupação - 10 a 14 anos
t_des1517	Float	Taxa de desocupação - 15 a 17 anos
t_des1824	Float	Taxa de desocupação - 18 a 24 anos
t_des18m	Float	Taxa de desocupação - 18 anos ou mais
t_des2529	Float	Taxa de desocupação - 25 a 29 anos
theiltrab	Float	Índice de Theil-L dos rendimentos do trabalho - 18 anos ou mais
trabcc	Float	% de empregados com carteira - 18 anos ou mais
trabpub	Float	% de trabalhadores do setor público - 18 anos ou mais
trabsc	Float	% de empregados sem carteira - 18 anos ou mais
t_luz	Float	% da população em domicílios com energia elétrica
parede	Float	% de pessoas em domicílios com paredes inadequadas
t_nestuda_ntrab_mmeio	Float	% de pessoas de 15 a 24 anos que não estudam nem trabalham
t_ocupdesloc_1	Float	% de pessoas vulneráveis à pobreza e que gastam mais de uma hora até o trabalho
t_rmaxidoso	Float	% de pessoas em domicílios vulneráveis à pobreza e dependentes de idosos
t_sluz	Float	% de pessoas em domicílios sem energia elétrica

Tabela 2: Variáveis de enriquecimento – Dados socioeconômicos.

3. Tratamento de Dados

Nesta seção, apresentaremos o tratamento realizado nas bases relacionadas ao estudo. Para isto, trabalhou-se com a linguagem R versão 4.1.3, sendo esta uma das principais linguagens utilizadas entre os Cientistas de Dados.

Para início do estudo, os arquivos “acquisition_train.csv” e “atlas.csv"” (enriquecimento) foram carregadas através da função *read.csv.* Abaixo, encontra-se as dimensões iniciais de cada base.

Base	Linhas	Colunas
Aquisição	45.000	43
Atlas 2010 – Variáveis Econômicas	5.565	74

Tabela 3: Dimensões iniciais.

3.1 Enriquecimento

Bases carregadas, o interesse passa a ser unificar as bases. Para isto, utilizou-se a variável *lat_lon*, variável do tipo *string* que apresenta as informações de Latitude e Longitude do solicitante no momento da aplicação. Dessa forma, tratou-se a variável utilizando pacote *Stringr*, separando-a em duas novas colunas, Latitude e Longitude. Agora o desafio se torna através dessas informações de localização, identificar o município do solicitante. Através de pesquisas, encontrou-se o pacote *tidygeocoder*, este possibilita através da função *reverse_geocode*, obter diversas informações relacionadas a localização a partir dos campos Latitude e Longitude.

Após isto, novos tratamentos precisaram ser realizados para então relacionar as informações das duas bases através dos nomes dos municípios. Como chave, utilizou-se o padrão nomeMunicipio-UF, uma vez que existem no país municípios com o mesmo nome.

O próximo passo é a identificação dos endereços através da função *reverse_geocode*, nesta, foi possível identificar 43.193 municípios. Para finalizar, fez-se a mescla da base de aquisição com a base de enriquecimento atlas através da chave nomeMunicipio-UF, encerrando esta etapa com 37.857 registros enriquecidos, equivalente a 84% da base total.

3.2 Ajustes de variáveis

Começando o processo de limpeza dos dados, analisamos a variável *target_default* para verificar a existência de nulos e identificou-se 3.259 registros, como esta variável é a principal do estudo, pois é a que identifica o evento de inadimplência do cliente, removeu-se estes registros do *dataset*, restando um total de 41.741.

Ainda sobre a limpeza dos dados, os processos listados abaixo foram executados.

- Troca de campos “” para NA, isto ocorreu em 52.384 valores.
- Correções de valores no campo *email*.
- Substituição de valores Inf para NA no campo *reported_income*.
- No campo *external_data_provider_email_seen_before*, substituição de valores -999 para NA.

3.3 Criação de novas *features*

O processo de criação de novas variáveis é fundamental em qualquer análise de *Machine Learning*, em nosso estudo criamos quatro novas variáveis, na tabela abaixo apresentamos as variáveis criadas.

Variável	Tipo	Descrição
hora_aplicacao	Integer	Hora extraída do campo *application_time_applied*
Região	String	Região baseado na variável de Latitude e Longitude
Turno	String	Turno em relação a variável *hora_aplicacao*
sins_state	String	Grupos em relação a variável codificada state
Group_lat_lon	String	Grupos calculados pelas variáveis Latitude e Longitude pelo algoritmo Kmeans

Tabela 4: *Features* criadas.

Antes do ajuste final do modelo, novas *features* foram criadas a partir de transformações de variáveis já existentes, seguindo metodologias conhecidas da área, porém, está etapa é apresentada apenas no capítulo 5.

3.4 Valores Ausentes

Após efetuar os passos listados acima, analisaremos os variáveis que as apresentaram valores nulos. A Figura 1 apresenta um gráfico de barras com o percentual de valores nulos encontrados em 21 variáveis. A variável *target_fraud*, apresentou 96,75% dos registros nulos, porém, esta variável é referente ao default de outro modelo proposto pelo Nubank no desafio, logo, ela já seria removida na etapa de seleção de variáveis.

Para a geração da figura a seguir, apresentamos somente as variáveis da base de aquisição, já que todas as variáveis enriquecidas pela base do IDH, terão a mesma quantidade de nulo, 6.644 registros e todas serão imputadas em etapas posteriores.

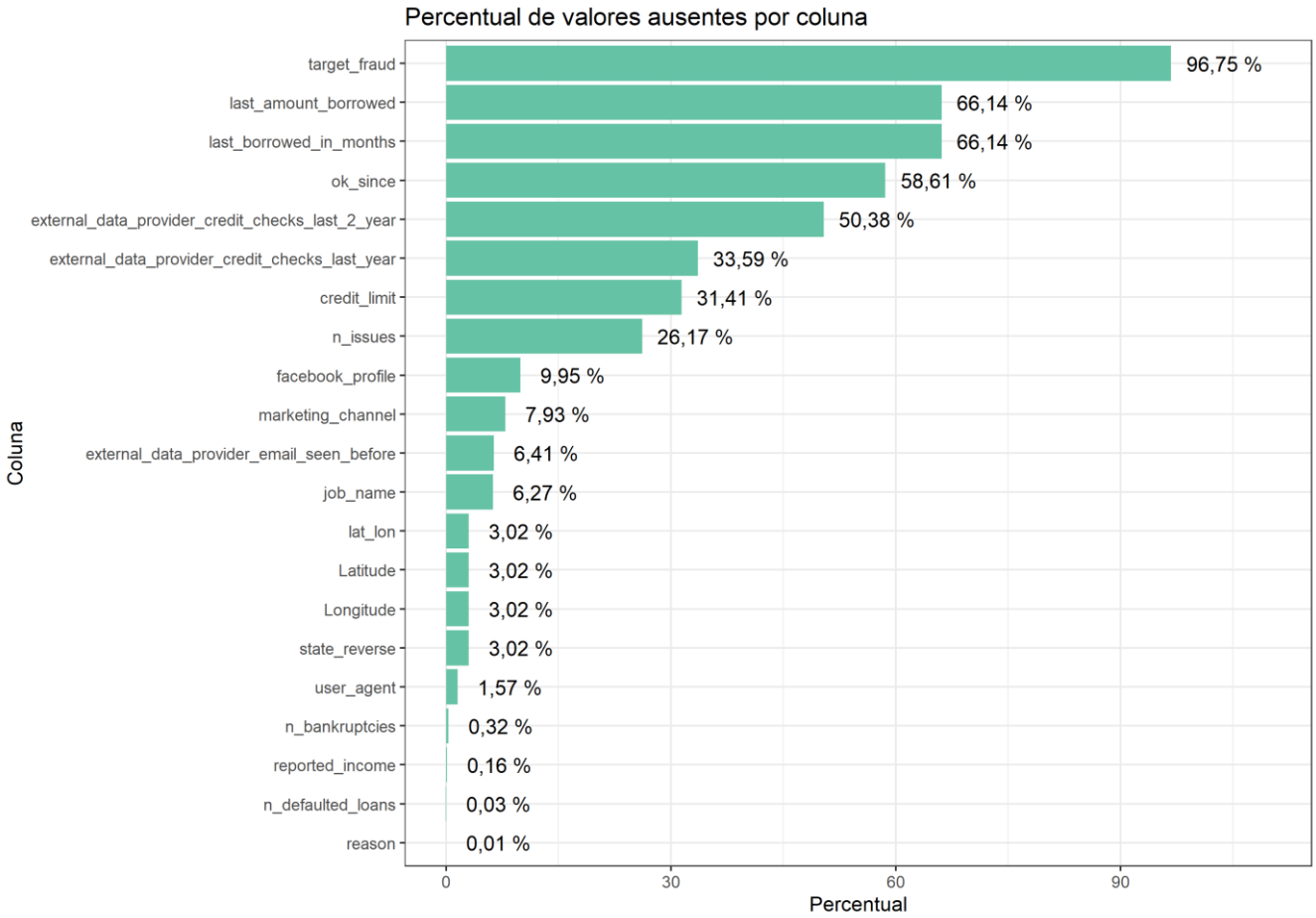


Figura 1: Percentual de valores nulos por variável.

Para analisar a necessidade de imputação dos dados ou remoção a da variável, examinou-se caso a caso e tomou-se as devidas decisões a respeito das variáveis com nulos apresentadas na Figura 1.

1. As variáveis *external_data_provider_credit_checks_last_2_year* será removida por apresentar apenas um único valor.

2. As variáveis *job_name*, *user_agent* e *reason*, serão removidas por se tratar de variáveis codificadas ou categóricas que possuem um elevado número de classes.

3. As variáveis *marketing_channel*, *n_bankruptcies*, *n_defaulted_loans* e *external_data_provider_credit_checks_last_year*, são variáveis categóricas e serão imputadas pela classe com maior frequência a variável *facebook_profile* será imputada com uma nova classe chamada ausente, está decisão foi tomada devido na análise descritiva esta variável se apresentar promissora.

4. As variáveis *las_amount_borrwed*, *last_borrewd_in_months*, *ok_since*, *n_issues*, *credit_limit*, *external_data_provider_email_seen_before* e *reported_income*, são variáveis numéricas e receberão valores imputados a partir da mediana. Esta etapa será realizada no pré-processamento da base, apresentada no capítulo 5.

4. Análise e Exploração dos Dados

Nessa seção será apresentado uma análise descritiva em relação a base criada até o momento. Trabalharemos com foco na principal variável do modelo *target_default*, buscaremos entender como algumas variáveis se comportam em relação aos níveis desta variável.

Após toda a etapa de enriquecimento e tratamento dos dados, neste momento, a base do estudo encontra-se com 41.741 linhas e 135 colunas, dessa forma, optamos por apresentar o comportamento somente de algumas variáveis.

4.1 Variável Target

A principal variável de nosso estudo se chama *target_default*, ela apresenta, se o evento inadimplência ocorreu ou não após a entrada do cliente na companhia. A Figura 1, apresenta a distribuição das classes desta variável.

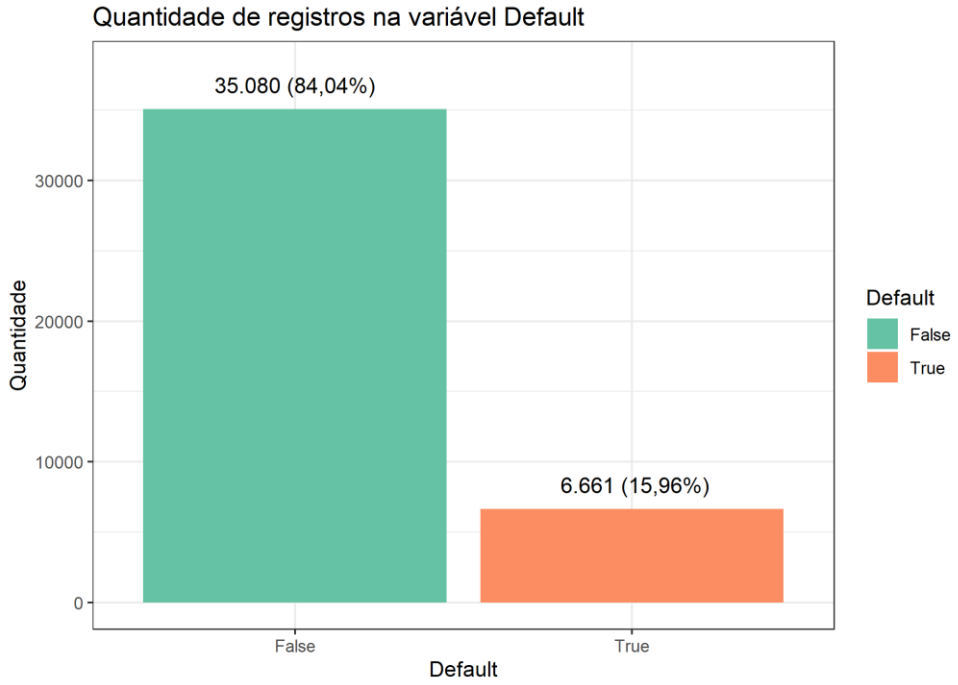


Figura 2: Distribuição da variável *target_default* (*Default*).

Nota-se que se deparamos com uma situação de desbalanceamento, em que a classe *True* possui somente 15,96% registros de toda a base. Trataremos melhor sobre este tema no capítulo 5, pois será feito um processo de balanceamento das classes antes do ajuste final dos modelos.

4.2 Variável *facebook_profile*

Iniciando a busca de insights, analisamos a variável *facebook_profile* que apresenta a informação se o solicitante possuía conta na rede social facebook no momento da aquisição.

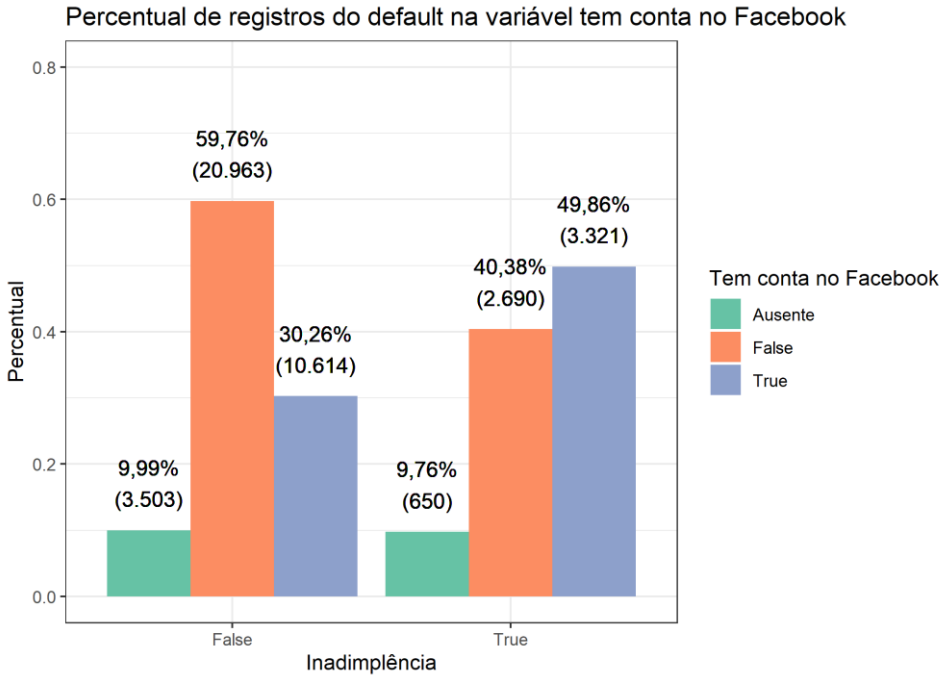


Figura 3: Distribuição da variável *facebook_profile* em relação ao *target* (inadimplência).

Nota-se através da Figura 3, que dentro do público que apresentaram inadimplência, aproximadamente 50% tinham conta no facebook, já no público que não apresentaram inadimplência 30,26% possuía conta na rede social, levando a uma hipótese de que a informação do solicitante possuir conta na plataforma no momento da solicitação, leva a uma probabilidade maior deste se tornar inadimplente.

4.3 Variável *email*

A variável *email* também foi analisada, esta informa o provedor do *e-mail* do solicitante. Novamente calculamos a distribuição das classes de forma separada para cada categoria da variável default e o resultado pode ser visto na Figura 1.

Analisando o gráfico, percebe-se que a distribuição dos provedores é muito próxima quando segregada pela variável *target*, isto nos diz que o evento se tornar inadimplente, independe do provedor de e-mail do solicitante.

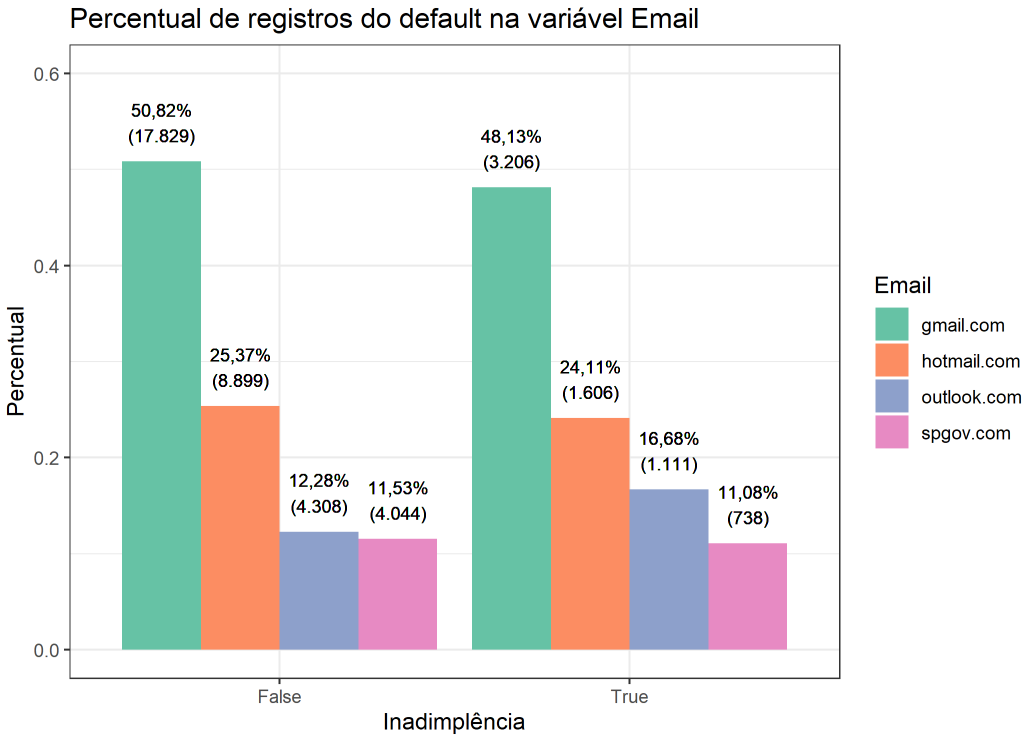


Figura 4 Distribuição da variável *e-mail* em relação ao *target* (inadimplência).

4.4 Analisando o percentual de inadimplência dos Estados

A variável *lat_lon* que trouxe a informação da localização do solicitante no momento da requisição do crédito, foi responsável por nos apresentar o endereço exato de cada cliente, dessa forma, analisamos o estado coletado e calculamos o percentual de inadimplência de cada.

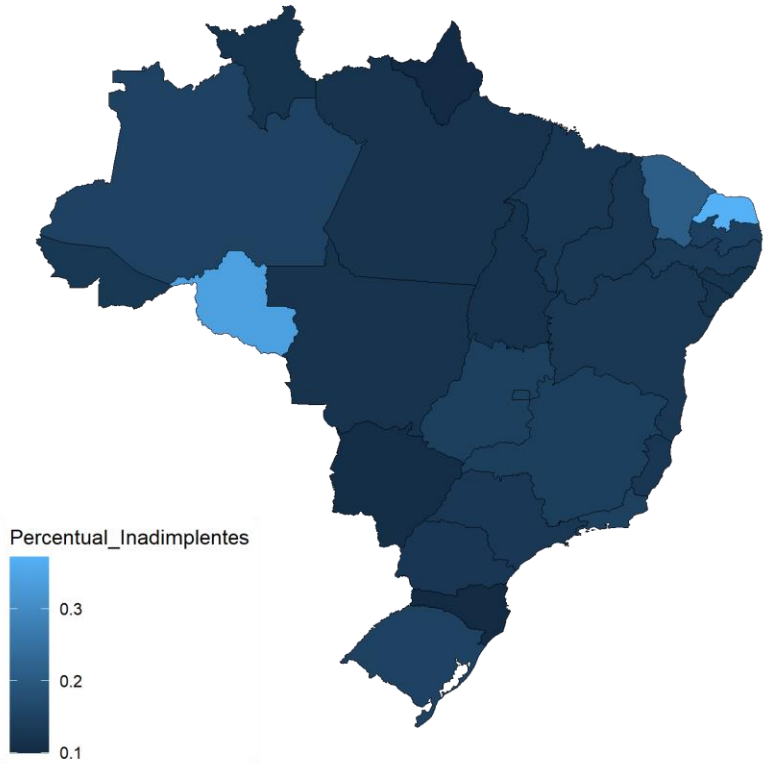


Figura 5: Mapa dos estados brasileiros pelo percentual de inadimplentes

Analisando a Figura 5, nota-se que 2 estados se destacaram dos demais, são eles, Rondônia e Rio Grande do Norte, na Tabela 5 apresenta o percentual de inadimplência de cada um. Para entender a que esse número representa, basta compara-los com o percentual de inadimplentes da base, que é de 15,96%, logo, estes dois estados possuem em pontos percentuais, mais que o dobro do comportamento comum.

Estado	Quantidade Registros	Quantidade Inadimplentes	Percentual Inadimplentes
Rondônia	1286	439	34,14%
Rio Grande do Norte	1771	658	37,15%

Tabela 5: Estados que se destacaram na inadimplência.

4.5 Variáveis de Score

A base de aquisição nos trouxe seis variáveis de scores de crédito de provedores terceiros. As variáveis *score_1* e *score_2*, estão codificadas, logo não iremos analisá-las.

Na busca da identificação de insights entre uma variável categoria (*target*) e outra contínua (variáveis de scores), é comum analisar gráficos, um dos mais utilizados para isto se chama *boxplot*.

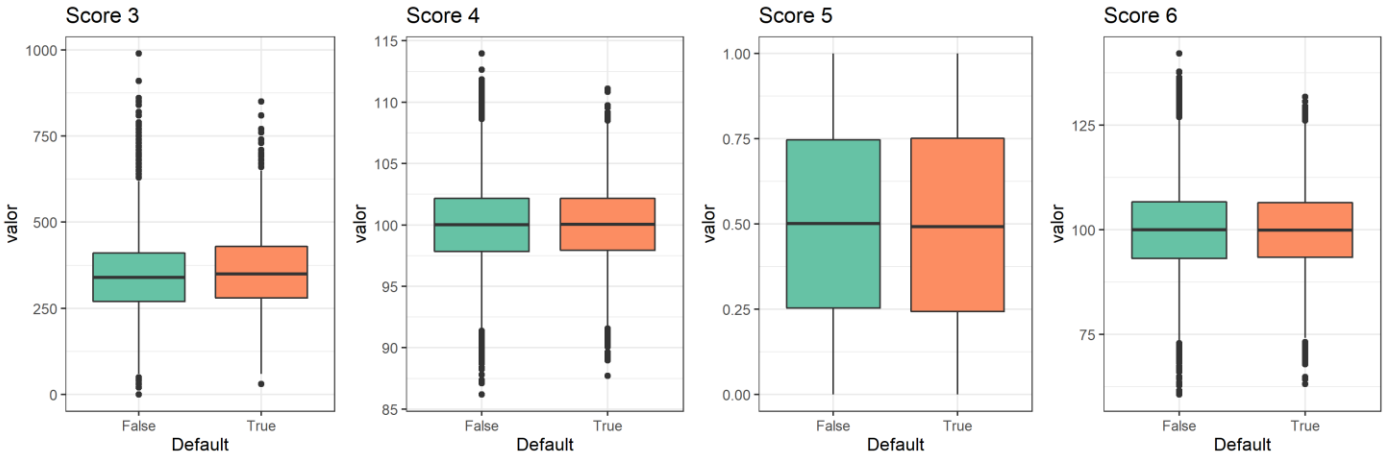


Figura 6: *Boxplot* das variáveis score_3, socre_4, score_5 e score_6 segregado pelo *target*.

A Figura 6 apresenta os gráficos de *boxplot* para as variáveis de scores que não estavam codificadas. A caixa destacada no gráfico representa que 50% dados estão localizados naquela região. Comparando as caixas em todos os gráficos, nota-se que em nenhum score o comportamento da caixa se altera quando comparado ao evento d*efault*, ou seja, nenhum dos scores aparenta influenciar na probabilidade do evento inadimplência.

4.6 Variáveis de Enriquecimento

Novamente utilizou-se do gráfico de *boxplot* para buscar insights, agora em relação a três variáveis que foram enriquecidas.

- Renda Per capita
- Rendimento médio dos ocupados – 18 anos ou mais
- Percentual de extremamente pobres

Novamente, nota-se que estas variáveis aparenta não influenciar no evento se tornar inadimplente.

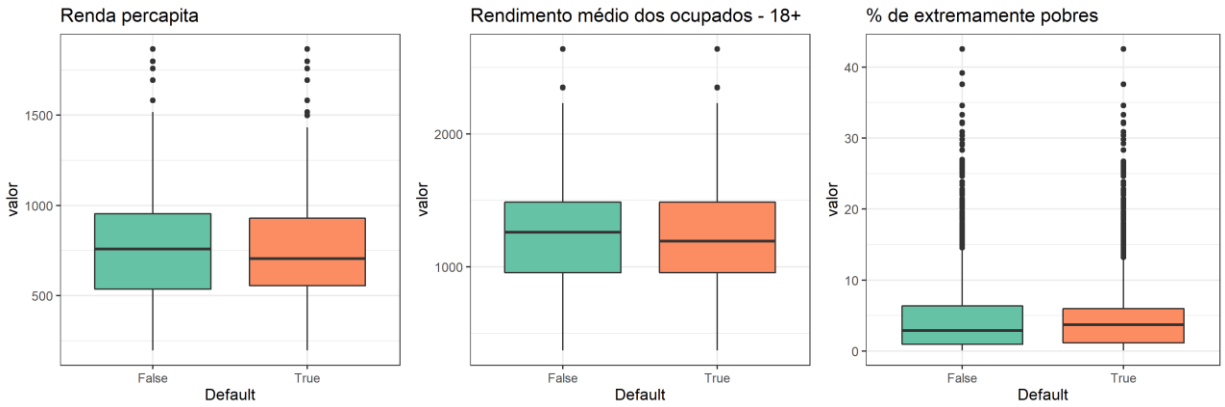


Figura 7: *Boxplot* de variáveis socioeconômicas.

5. Criação de Modelos de *Machine Learning*

Considera-se esta seção a principal deste trabalho, uma vez que nela será apresentada todo o passo a passo para a construção do nosso modelo de *Machine Learning*. Iniciaremos apresentando o processo de seleção de variáveis, logo após, ajuste dos Modelos e por fim o comparativos entre eles. Serão exibidos os principais códigos utilizados.

Assim como no capítulo tratamento de dados, continuaremos utilizando a linguagem R versão 4.1.3. Todas as bibliotecas necessárias serão informadas no decorrer do capítulo.

5.1 Remoção de variáveis

Inicia-se etapa com um refinamento nas variáveis, uma análise detalhada foi realizada e diversas variáveis foram removidas cada uma com seu motivo.

Variáveis	Motivo da remoção
name_state, state.y, cidade_atlas, uf.y, uf.x, cidade, codmun6, codmun7, município, Municipio	Variáveis duplicadas que surgiram através das mesclas de bases
channel e external_data_provider_credit_checks_last_2_year	Por apresentar um único valor
ids, score_1, score_2, reason, state_codificado, zip, job_name, profile_tags, user_agent, shipping_state,state_reverse	Variáveis que estavam codificadas ou possuíam um elevado número de classes
external_data_provider_first_name, profile_phone_number, shipping_zip_code	Dados sem interpretação
lat_lon, Latitude, Longitude	Sem informação para modelagem
target_fraud	Default para outro modelo

Tabela 6: Variáveis removidas.

Após a remoção das variáveis listadas na Tabela 6, passamos a ter uma base com dimensão de 41.741 linhas por 105 colunas.

5.2 Pré-processamento

Neste momento trabalharemos no pré-processamento dos dados, última etapa antes do ajuste do modelo. Aqui iremos criar novas variáveis, imputar dados nulos, realizar transformações e fazer novas reduções de variáveis.

5.2.1 *One_hot_encoder* ou *dummy*

Para iniciar, uma estratégia bastante conhecida e utilizada no pré-processamento é o *one hot encoder*, este processo é aplicado as variáveis categóricas explicativas, criando uma coluna para cada categoria com os valores de 0 ou 1.

Em nossa base as variáveis *real_state, email, marketing_channel, estado_loc, região, bins_state, turno* e *group.lat.lon*, passaram por este processo.

Abaixo temos o código utilizado nesse processo. O pacote *dataPreparation* foi utilizado na operação.

```
b_oneHotEncoder = dataPreparation::one_hot_encoder(base_preparada,drop=T)
```

Após este passo, o conjunto de dados passou a ter 174 colunas.

5.2.3 População de treino e teste

A partir deste momento, utilizaremos em específico o pacote *tidymodels* junto as suas dependências. Este pacote possibilita realizar o processo completo de modelagem.

Para iniciar, utiliza-se o função *inicial_split* do pacote *rsample* para separarmos nossa base em treino e teste.

```
dt_split = rsample::initial_split(data = b_oneHotEncoder , prop = 0.75, strata = target_default)
```

O parâmetro prop, recebe o percentual da base designado a base de treino, nota-se que utilizamos uma separação de 75% para treino e 25% para teste.

5.2.3 *Recipes*

Dando prosseguimento a construção do modelo, iremos agora trabalhar com o pacote *recipes*, com este é possível criamos um objeto que armazenará passos a se realizar nas bases de treino e teste.

Para criamos um objeto *recipe*, precisamos repassar uma base inicial e em seguida elencar os *steps* que serão processados. Abaixo temos a criação do objeto *recipe* utilizado no estudo e posteriormente a explicação de cada passo.

```
data_recipe = rsample::training(x = dt_split) %>%
                    recipes::recipe(target_default ~ .) %>%
themis::step_downsample(target_default, under_ratio = 1) %>%
  recipes:: step_impute_median (recipes::all_numeric_predictors(), neighbors = 3) %>%
recipes::step_novel(recipes::all_nominal_predictors(), -recipes::all_outcomes()) %>%
  recipes::step_center(recipes::all_numeric_predictors()) %>%
  recipes::step_scale(recipes::all_numeric_predictors()) %>%
  recipes::step_corr(recipes::all_numeric_predictors(), threshold = 0.8, method = "pearson")
%>%
  recipes::step_zv(recipes::all_numeric_predictors(), -recipes::all_outcomes())
```

Balanceamento

O balanceamento da nossa base de treino é feito a partir do *step_downsample*, neste passo estamos realizando um *undersamplig* em nossa base, ou seja, estamos igualando os valores da classe majoritária para classe minoritária através de amostras aleatórias.

Target	Quantidade Base Original	Base Teste (25%)	Base Treino (75%)	Base Treino com *Undersampling*
1	6.661	1.666	4.995	4.995
0	35.080	8.770	26.310	4.995

Tabela 7: Balanceamento e *Split* das bases de treino e teste.

Imputação de dados nulos

No capítulo 3 apresentamos os valores nulos, algumas variáveis numéricas ficaram de ser imputadas e este processo é realizado através da função *step_impute_median*, utilizou-se a mediana por dois motivos, esforço computacional reduzido, já que temos muitos registros a imputar e devido a mediana ser uma métrica que não é sensível a dados discrepantes.

Nominais em Fator

Para transformar as variáveis nominais em fator é utilizado a função *step_novel*.

Variáveis correlacionadas

Através da função *step_corr*, removemos as variáveis com alta correlação entre elas, efeito conhecido como multicolinearidade. Neste passo utilizamos um valor limite de 0,8.

Após este passo, 48 colunas foram removidas, indo para o modelo final 126 variáveis.

Normalizando

Finalizando o pré-processamento aplicamos a normalização das variáveis numéricas, fazendo com que elas passem a ter uma média zero e desvio padrão um. Estes passos são realizados pelas funções *step_center* e *step_scale*.

5.3 Ajuste do Modelo

Nesta sessão apresentaremos o passa a passo dos modelos analisados. Continuamos utilizando as extensões do pacote *tidymodels* no R para isto.

5.3.1 Cross-Validation

Para iniciarmos o ajuste do modelo, ajustamos as configurações de *cross-validation*, no *tidymodels*, utilizamos a função *vfold* do pacote *rsample* para esta etapa.

```
cv = rsample::vfold_cv(data= rsample::training(x = dt_split), v = 6, repeats = 2, strata = target_default)
```

A função *rsample* recebe os parâmetros data que recebe o conjunto de dados de treino, v que recebe o número de dobras, *repeats* que recebe o número de vezes que será repetido os particionamentos das dobras e *strata* em que passamos a variável default.

Em nosso estudo, trabalhamos com 6 *folds* e 2 repetições para cada conjunto de parâmetros.

5.3.2 Treinamento

Para ajustar os modelos, utilizaremos os pacotes *workflow*, *parsnip* e *tune*, também fazendo parte do *tidymodels*.

Todos os modelos ajustados seguirão uma mesma estrutura.

- Especificação do modelo utilizado: Nestas etapa informamos qual modelo queremos treinar, junto a quais hiperparâmetros iremos ajustar.
- Criação de *workflow*: Criação um objeto que agrupa as solicitações de pré-processamento, modelagem e pós-processamento.
- Criação da grade de parâmetros: Dataframe com os hiperparâmetros para ser treinados e avaliados.
- Ajuste do modelo: Ajuste efetivamente calcula um conjunto de métricas de desempenho. Para isto, recebe todos os objetos anteriores criados.
- Análise de desempenho: Após isto, avalia-se no treino e principalmente no teste as métricas importantes para o estudo.

Randon Forest

O primeiro modelo estudado será o *randon forest*, em português florestas aleatórias. Este modelo opera construindo infinitas árvores aleatórias em tempo de treinamento. O algoritmo cria uma estrutura simular a um fluxograma, com nós no de uma condição é verificada, e se atendida o fluxo segue por um ramo, caso contrário, por outro, sempre levando ao próximo nó, até a finalização da árvore.

Abaixo segue os códigos utilizados para o treinamento do modelo. Os resultados serão apresentados logo mais.

```
# ESPECIFICANDO O MODELO
rf_spec = parsnip::rand_forest() %>%
  set_args(mtry = tune()) %>%
  parsnip::set_engine("ranger", importance = "impurity") %>%
  parsnip::set_mode("classification")
# CRIANDO WORKFLOW
rf_workflow = workflows::workflow() %>%
  workflows::add_recipe(data_recipe) %>%
  workflows::add_model(rf_spec)
# GRID DE PARAMETROS
rf_grid = expand.grid(mtry = c(5,10,30,50,60,70))
# AJUSTE DO MODELO
rf_fit_tune = rf_workflow %>%
  tune::tune_grid(resamples = cv, # CV object
                  grid = rf_grid, # grid of values to try
                  metrics = yardstick::metric_set(recall,f_meas,accuracy,kap,roc_auc,sens),
control = tune::control_resamples(save_pred = TRUE,verbose=T)
  )
```

SVM - *Support Vector Machine*

O segundo modelo estudo foi o SVM, este é um algoritmo plota-se cada item de dados como um ponto no espaço n-dimensional (n é o número de recursos), com o valor de cada recurso sendo o valor de uma determinada coordenada. Então, nós executamos a classificação encontrando o hiperplano que melhor diferencia as classes.

Para o ajuste do modelo SVM no *tidymodels* utilizamos o código abaixo.

```
# ESPECIFICANDO O MODELO
SVM_spec = parsnip::svm_rbf(
    cost = tune(),
    rbf_sigma = tune()) %>%
  parsnip::set_engine("kernlab", importance = "impurity") %>%
  parsnip::set_mode("classification")
# CRIANDO WORKFLOW
SVM_workflow = workflows::workflow() %>%
  workflows::add_recipe(data_recipe) %>%
  workflows::add_model(SVM_spec)
# GRID DE PARAMETROS
SVM_grid = expand.grid(cost = c(0.1,1),rbf_sigma=c(0.01,0.001))
# AJUSTE DO MODELO
SVM_fit_tune = SVM_workflow %>%
  tune::tune_grid(resamples = cv, # CV object
                  grid = SVM_grid, # grid of values to try
```

```
            metrics = yardstick::metric_set(recall,f_meas,accuracy,kap,roc_auc,sens),
control = tune::control_resamples(save_pred = TRUE,verbose=T)
  )
```

XGBoost

Por último, o terceiro modelo analisado é o XGBoost. Este algoritmo é baseado em árvore de decisão e utiliza uma estrutura de *Gradiente boosting*.

Abaixo segue os códigos utilizados para ajustar este modelo.

```
# ESPECIFICANDO O MODELO
XGB_spec <- parsnip::boost_tree(
  trees = 1000,
  tree_depth = tune(),
  min_n = tune(),
  loss_reduction = tune(),
  sample_size = tune(),
  mtry = tune(),
  learn_rate = tune()
) %>%
  parsnip::set_engine("xgboost") %>%
  parsnip::set_mode("classification")
# GRADE DE PARAMETROS
XGB_grid <- grid_latin_hypercube(
  tree_depth(),
  min_n(),
  loss_reduction(),
  sample_size = sample_prop(),
  finalize(mtry(),  training(dt_split)),
  learn_rate(),
  size = 5
)
# CRIANDO WORKFLOW
XGB_workflow = workflows::workflow() %>%
  workflows::add_recipe(data_recipe) %>%
  workflows::add_model(XGB_spec)
# AJUSTE DO MODELO
XGB_fit_tune = XGB_workflow %>%
  tune::tune_grid(resamples = cv, # CV object
            grid = XGB_grid, # grid of values to try
            metrics = yardstick::metric_set(recall,f_meas,accuracy,kap,roc_auc,sens),
            control = tune::control_resamples(save_pred = TRUE,verbose=T)
)
```

5.2 Avalição de métricas

O objetivo de nosso modelo é predizer com o máximo de exatidão se o cliente será inadimplente, logo, necessitamos minimizar os falsos negativos e assim maximizando a identificação de cliente inadimplentes, para isto acontecer necessitamos trabalhar com a métrica sensibilidade.

Nas análises a seguir inicialmente apresentamos para cada modelo os resultados olhando somente para acurácia e posteriormente fazendo uma otimização de *threshold* para maximizar a sensibilidade, limitando a especificidade em 0,5, ou seja, a probabilidade de um cliente adimplente ser classificado como inadimplente ser 50%.

É válido ressaltar que todos as métricas aqui apresentadas foram calculadas na base final de teste.

5.2.1 Avaliação Randon Forest

O modelo de *randon forest* foi treinado avaliando diferentes valores do parâmetro *mtry*, que se ajustou melhor com o valor 45. A Figura 8 apresenta a curva ROC para as duas repetições no treinamento, este tipo de gráfico ilustra o desempenho de um sistema classificador binário.

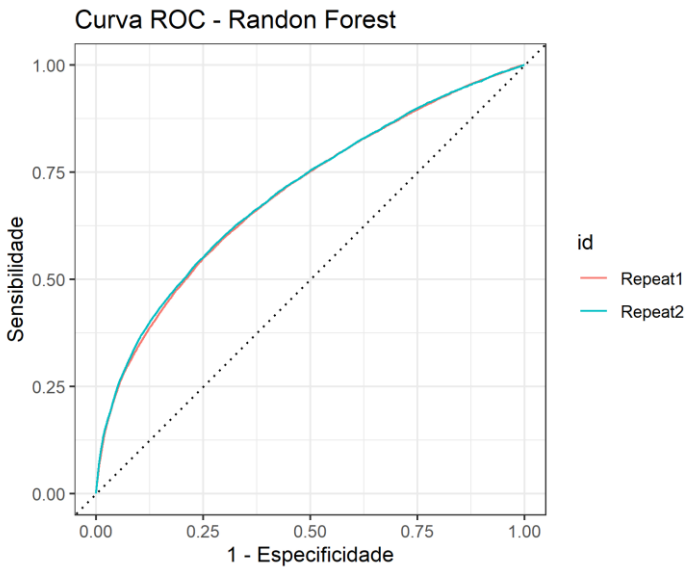
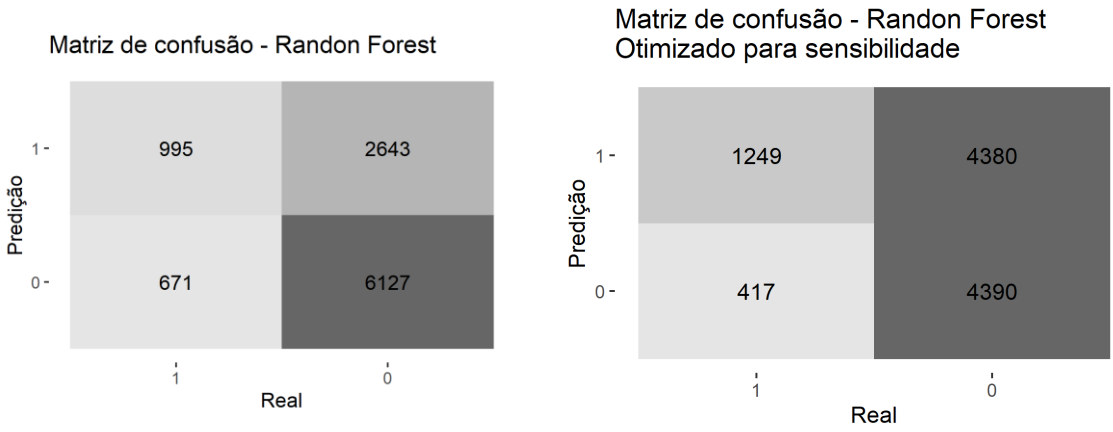


Figura 8: Balanceamento e *Split* das bases de treino e teste.

O melhor modelo encontra apresentou uma acurácia de 68,2% e sensibilidade de 59,7%. O próximo passo é encontrar o melhor *threshold* que irá maximizar a sensibilidade fazendo com que a especificidade não fique abaixo de 50%.

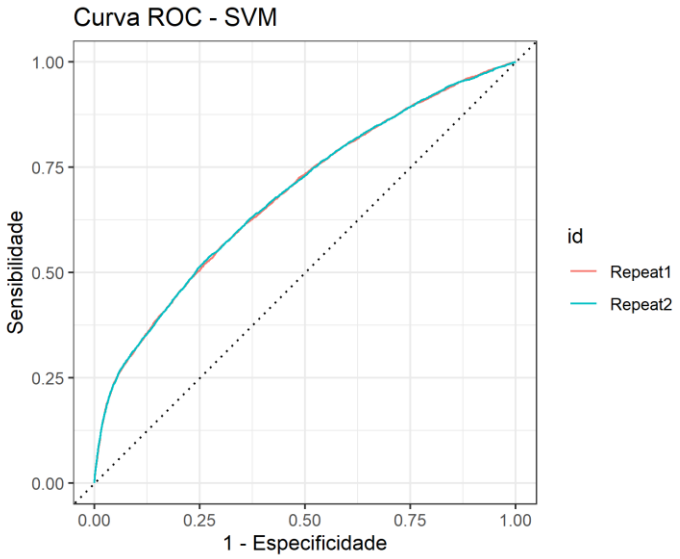
Após aplicado da etapa de otimização, encontrou-se um *threshold* do valor de 0,416 que apresenta uma sensibilidade de 75,%. Abaixo visualiza-se as matrizes de confusão antes e depois do novo *threshold* ser aplicado.



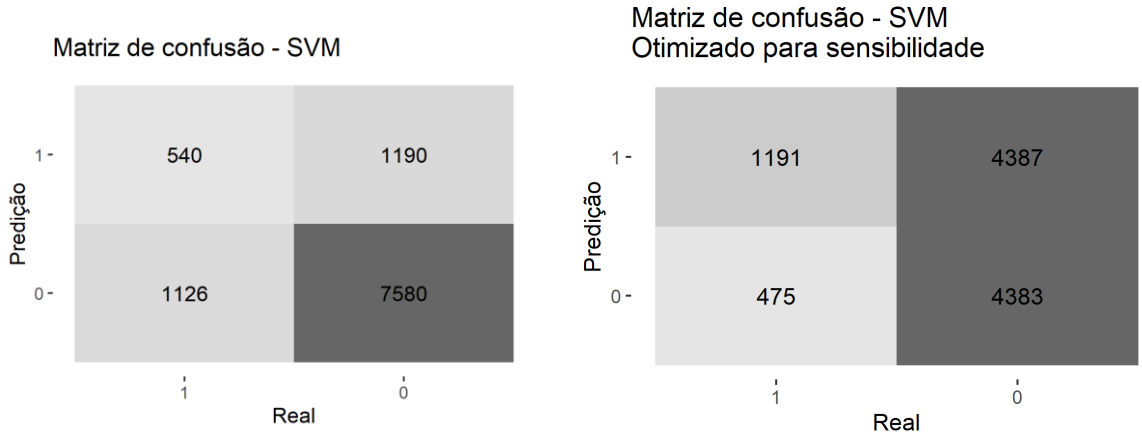
Figuras 9 e 10: Matrizes de confusão para o modelo Randon Forest com threshold 0,5 e otimizado para benefício da sensibilidade

5.2.1 Avaliação SVM

O modelo SVM foi ajustado variando valores dos hiperparâmetros *cost* e *rbf_sigma*, os melhores resultados apresentados foram respectivamente 0,1 e 0,01. A Figura 11 apresenta a curva ROC para o melhor modelo encontrado com este classificador.



Figuras 11: Curva ROC modelo SVM



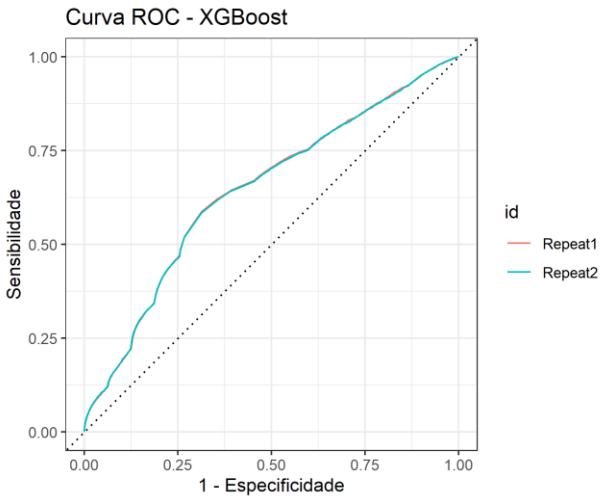
Figuras 12 e 13: Matrizes de confusão para o modelo SVM com *threshold* 0,5 e otimizado para benefício da sensibilidade

Para o modelo SVM o melhor modelo ajustou obteve-se uma acurácia de 77,8%, sendo o melhor resultado avaliando essa métricas entre todos os modelos, porém, ao avaliarmos a sensibilidade, métrica escolhida como principal neste estudo, o resultado foi de apenas 32,4%.

Ao otimizar o *threshold*, encontrou-se um valor de 0,413 que apresentava uma sensibilidade de 71,5%.

5.2.1 Avaliação XGBoost

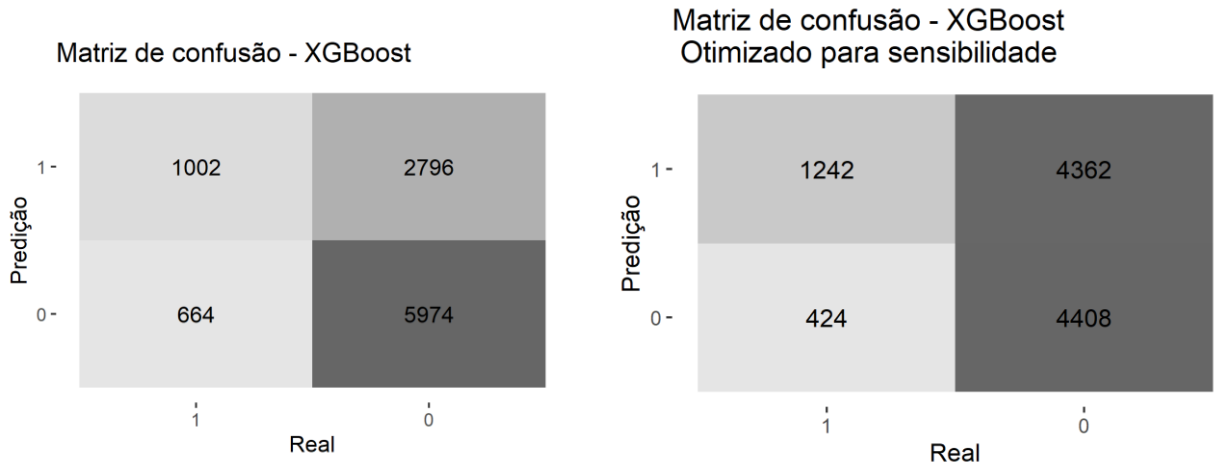
Terceiro modelo avaliado, o XGBoost apresentou seu melhor modelo com os seguintes parâmetros e resultados, *mtry* com 124, *min_n* com 24 , *tree_depth* com 15, *learn_rate* com 0,000000160, *loss_reduction* com 0,00000257 e *sample_size* com 0,886.



Figuras 14: Curva ROC modelo XGBoost

A Figura 14 apresenta a curva ROC do melhor modelo ajustado do algoritmo XGBoost.

Para finalizar a avaliação das métricas, otimizou-se o *threshold* para maximizar a sensibilidade. Encontrou-se um valor de 0,4999863 que apresentava no conjunto de treino uma sensibilidade de 74,5%. A seguir apresentamos a matriz de confusão obtida no conjunto de teste para o modelo com *threshold* de 0,5 e otimizado.



Figuras 15 e 16: Matrizes de confusão para o modelo XGBoost com *threshold* 0,5 e otimizado para benefício da sensibilidade

5.2.1 Comparativo entre os modelos

Calculado todas métricas, avaliamos agora o desempenho de forma conjunta para escolhermos o melhor modelo.

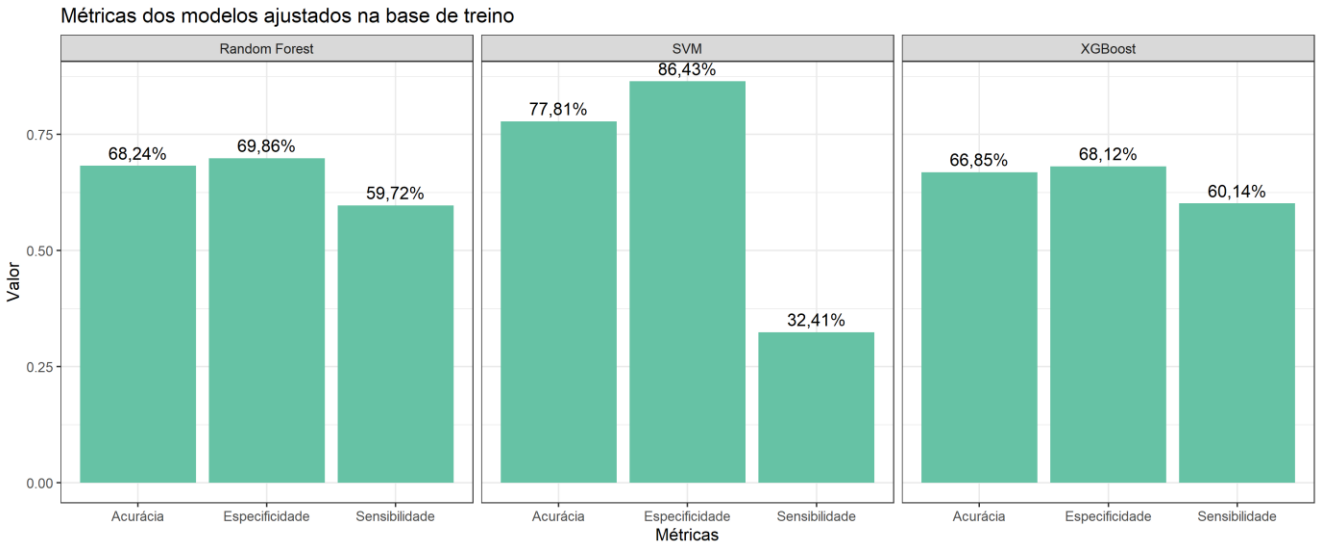


Figura 17: Comparação de métricas para os modelos ajustados utilizando threshold de 0,5

A Figura 17 nos apresenta as métricas acurácia, sensibilidade e especificidade para cada modelo quando utilizado o *threshold* padrão de 0,5. Nota-se que nesta visão, se utilizássemos a acurácia como métrica principal deste estudo, o modelo escolhido seria o SVM chegando a um valor de 77,81%.

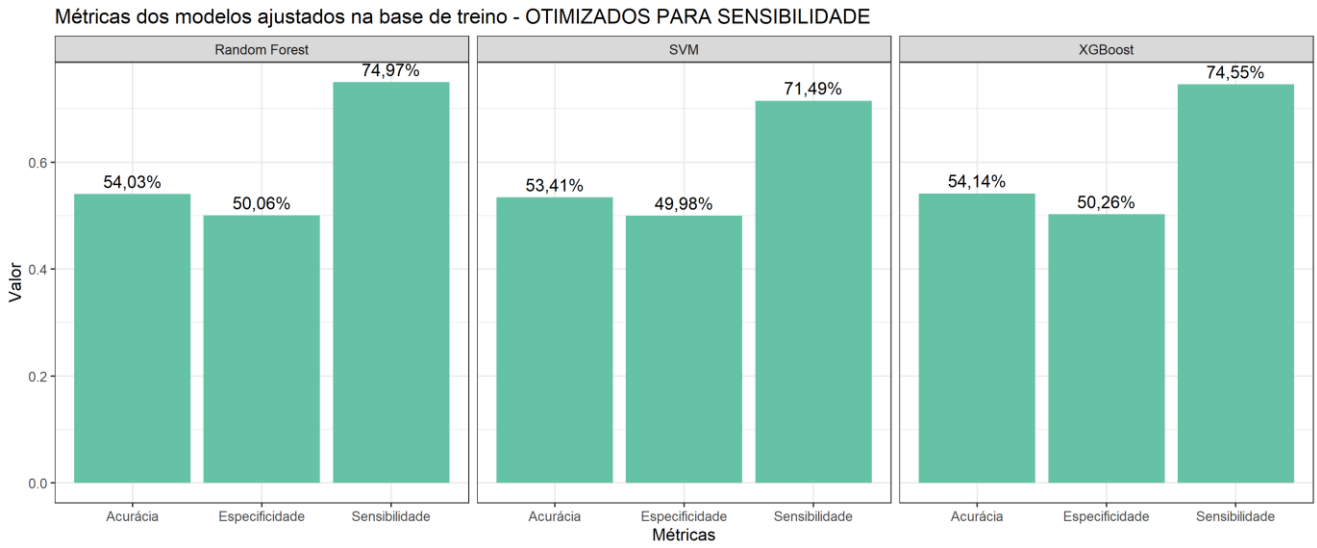


Figura 18: Comparação de métricas para os modelos ajustados utilizando *threshold* otimizado

Porém, como explicado no início desta sessão, a melhor métrica a se analisar no contexto estudado é a sensibilidade. A Figura 18 nos apresenta um comparativo de como ficou todas as métricas quando otimizado a sensibilidade, limitando a especificidade a 50%.

Através desta nova visão, optou-se por escolher o modelo *randon forest* como o melhor entre os três avaliados no estudo. Este, após ter *threshold* otimizado, apresentou resultados próximos ao modelo XGBoost, mas por décimos ganhou na sensibilidade.

5.2.1 Importância das variáveis

Escolhido o melhor modelo, podemos agora analisar quais variáveis apresentaram a maior importância no ajuste. Utilizou-se o pacote vip do R para construir a gráfico apresentado na Figura 19.

Nota-se que as variáveis *income, facebook_profile* e *risk_rate* foram as que contribuíram mais para o modelo *Randon Forest* ajustado.

Observa-se que entre as 10 variáveis com maior importância, não presenciamos nenhuma variável da base de enriquecimento do estudo.

Outra informação interessante é a variável *facebook_profile* se apresentar como a segunda mais importante. Esta variável foi notada como uma possível variável que influenciaria nos resultados na análise descritiva deste estudo.

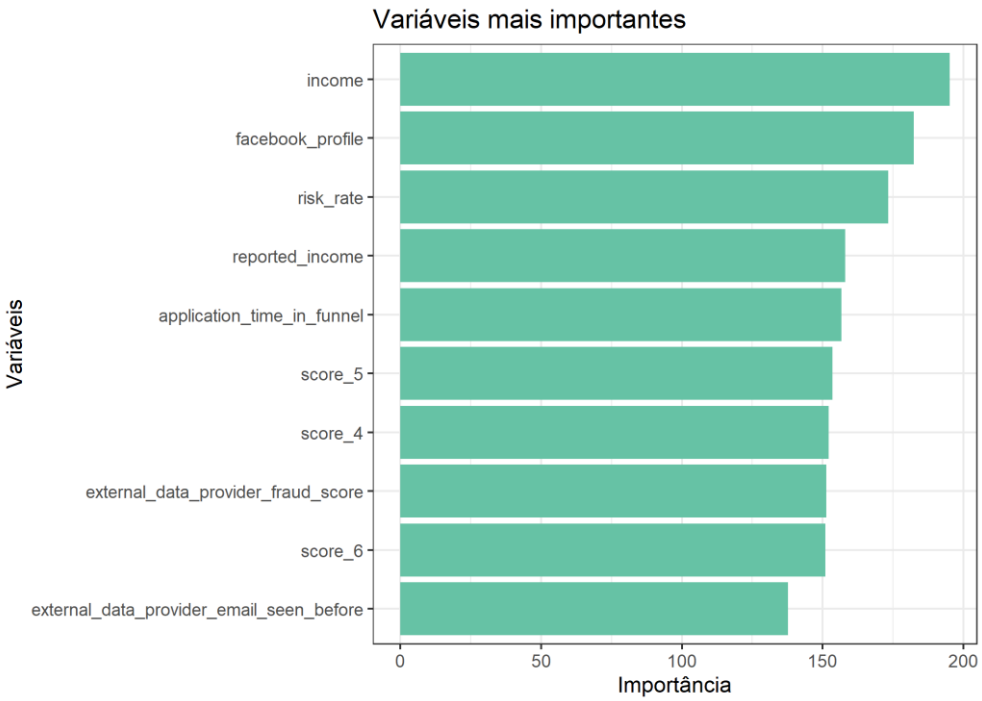


Figura 19: Importância das variáveis ajustadas no modelo *Randon Forest*

6. Interpretação dos Resultados

Este trabalho apresentou um passo a passo da resolução de uma solução de *machine learning* utilizando a linguagem R. Nosso estudo tratava-se de uma análise de crédito e tinha como objetivo prever se cliente seria inadimplente, levando a tomada de decisão de concessão de crédito ou não.

Na análise descritiva, observou-se poucas variáveis correlacionadas com a variável *target_default*, evento de nosso estudo. As que se destacaram foram as variáveis *facebook_profile* e estados. Dentre estas a variável *facebook_profile* se apresentou com a segunda mais importante no ajuste, confirmando a hipótese apresentada da variável no capítulo 3.

No ajuste do modelo, analisou-se os resultados dos algoritmos Randon Forest, SVM e XGBoost. Avaliamos inicialmente as métricas utilizando o *threshold* a 0,5 padrão dos algoritmos e posteriormente otimizados para beneficiar a sensibilidade. Esta que foi escolhida como métrica principal no estudo por prevalecer a intensão de acertar o máximo de inadimplentes possível, limitando a especificidade em 50%.

A Tabela 8 apresenta um resultado geral dos modelos ajustados. Escolheu-se o modelo Randon Forest como o melhor devido a este apresentar a maior sensibilidade.

Tipo de *threshold*	Modelo	Valor *threshold*	Acurácia	Sensibilidade	Especificidade
NORMAL	*Randon Forest*	0,5	68,2%	59,7%	69,9%
	SVM	0,5	77,8%	32,4%	86,4%
	XGBoost	0,5	66,8%	60,1%	68,1%
OTIMIZADO PARA SENSIBILIDADE	*Randon Forest*	0,416	54,0%	75,0%	50,1%
	SVM	0,413	53,4%	71,5%	50,0%
	XGBoost	0,4999863	54,1%	74,5%	50,3%

Tabela 8: Tabela geral com métricas dos modelos ajustados

7. Resumo do Estudo

PREVISÃO DE RISCO DE INADIMPLÊNCIA COM AUXÍLIO DE DADOS SOCIOECONÔMICOS

1. Problema a se resolver

O objetivo do modelo é predizer se um cliente se tornará inadimplente a partir de dados coletados em sua aquisição e valores socioeconômicos do município em que este estava localizado no momento da solicitação.

2. Variáveis Utilizadas na solução

Variáveis explicativas:
Dados na aquisição:

- Escore de provedores de terceiros, Possui Facebook,
- Renda estimada por provedores
- Entre outras.

Dados Socioeconômicos (Município):

- Renda percepita
- Percentual de probreza
- Percentual de empregados
- Entre outras.

Variável resposta: Inadimplente (1) , Não Inadimplente (0)

3. Fonte dos dados

Um vez que o presente estudo é referente a um Data Challeng de 2018, encontrou-se os arquivos em repositório público no GitHub.

Outras fontes utilizadas agora como enriquecimento foram de dados socioeconômicos do IGBE disponibilizados na comunidade Kaggle.

4. Modelos utilizados

Optou-se por utilizar três modelos algoritimos de machine learning, foram eles:

- Random Forest
- Support Vector Machine - SVM
- XGBoost

5. Avaliação dos modelos e resultados

A principal métrica utilizada neste estudo foi a sensibilidade, pois prevaleceu a intenção de acertar o máximo de clientes inadimplentes possíveis.

Utilizou-se técnicas de otimização de threshold para maximizar os valores de sensibilidade.

O melhor modelo apresentado obteve uma sensibilidade de 75%, sendo este o Randon Forest.

6. Preparação dos dados utilizados nos modelos

Para preparar a base de dados ajusta na modelagem as etapas abaixo em ordem foram executadas.

1. Coleta das informações (Dados de Aquisição)
2. Enriquecimemnto com dados Socioeconômicos
3. Exclusão de registros com target nulos
4. Criação de novas features
5. Aplicação de OneHotEncoder
6. Imputação utilizando valor mais frequente e mediana
7. Exclusão de variáveis correlacionadas
8. Divisão dos dados em 75% para treino e 25% para teste
9. Balanceamento utilizando técnica Undersampling na base de treino

8. Links

Abaixo encontra-se os links para o vídeo da apresentação resumida de 5 minutos e do repositório do github com todos os dados utilizados no trabalho.

Link para o vídeo: https://www.youtube.com/watch?v=WKgnaXtKt0k

Link para o repositório: https://github.com/robertooliveira94/POS-nubank-challenge-2018

APÊNDICE

Scripts R utilizados no trabalho

```
# DATA CHALLENGE 2018 NUBANK TCC
# LIMPANDO O AMBIENTE ----------------------------------------------------------------
rm(list = ls())
library("janitor")
library(dataPreparation)
# PACOTES ----------------------------------------------------------------
pacotes = c("dplyr", "stringr","tidyverse","tidymodels","ranger","tune","probably",
        "xgboost" ,"tictoc", "ggplot2",  "abjutils",

"ggthemes","dataPreparation","tidygeocoder","Cluster","ClusterR","themis","janitor")
novos.pacotes = pacotes[!(pacotes %in% installed.packages()[, "Package"])]
if(length(novos.pacotes)) install.packages(novos.pacotes, repos = 'https://cran.us.r-
project.org')
options(warn = -1)
unlist(lapply(pacotes, require, character.only = TRUE))

# CARREGANDO AS BASES INICIAIS ----------------------------------------------------------------
acquisition_train = read.csv("codigo-fonte/data/acquisition_train.csv")

# DICIONARIO DA BASE DE AQUISICAO
dicionario = read.table("codigo-fonte/data/dicionario.txt",sep = "-",header = T,encoding =
"UTF-8")

# ENRRIQUECIMENTO DOS DADOS ----------------------------------------------------------------

# CORRIGINDO COLUNAS DE ESTADO E SEPARANDO COLUNAS DE LATITUDE E LONGITUDE
acquisition_train_ = acquisition_train %>%
  dplyr::mutate(shipping_state = stringr::str_sub(string = shipping_state,start = 4,end = 5),
            lat_lon = str_replace(lat_lon, "\\(",""),
            lat_lon = str_replace(lat_lon, "\\)",""))

# NESTA ETAPAZ FOI REALIZADO UMA BUSCA DE ENDERECOS ATRAVES DAS VARIAVEIS DE
LATITUDE E LONGITUDE COM A AJUDA DO PACOTE tidygeocoder
# SPLIT COLUNA lat_lon
# base_split = str_split_fixed(acquisition_train_$lat_lon, ",", 2)
# acquisition_train_$Latitude = as.numeric(base_split[,1])
# acquisition_train_$Longitude = as.numeric(base_split[,2])
# reverse <- acquisition_train_ %>%
#   reverse_geocode(lat = Latitude, long = Longitude, method = 'osm',
#                address = address_found, full_results = TRUE)
# save(reverse,file = "codigo-fonte/data/reverse.Rdata")
```

```
# CARREGANDO OBJETO BAIXADO
load(file = "codigo-fonte/data/reverse.Rdata",verbose = T)

# REMOVENDO VARIAVEIS EXTRAS ORIUNDAS DO REVERSE_GEOCODE
base_reverse = reverse %>%
  dplyr::mutate(municipio = gsub("^.*? de ","",municipality)) %>%
  dplyr::select(c(ids:target_fraud,Latitude,Longitude,municipio,state...60)) %>%
  dplyr::rename(state = state...16,
                state_reverse = state...60)

# ENRRIQUECIMENTO DOS DADOS ----------------------------------------------------------
# BASES DO IBGE IDH
atlas = read.table("codigo-fonte/data/atlas.csv",encoding = 'UTF-8',sep = ',',header = T)
desc = read.csv("codigo-fonte/data/desc.csv",encoding = 'UTF-8',sep = ',')
estados = read.table("codigo-fonte/data/estados.txt",encoding = 'UTF-8',sep = ';',header = T)

# BASE DE 2010 COM COLUNAS RELACIONADAS A ECONOMIA DO MUNICIPIO
atlas_2010 = atlas %>%
  dplyr::filter(ano=='2010') %>%
  dplyr::select(codmun6,uf,codmun7,município,desc$SIGLA[desc$cat=='econ']) %>%
  dplyr::left_join(estados,by=c("uf"="uf")) %>%
  dplyr::mutate(cidade = paste0(município,"-",state)) %>%
  dplyr::mutate(cidade = str_to_upper(abjutils::rm_accent(cidade)),
                cidade_atlas = str_to_upper(abjutils::rm_accent(cidade)))

# JOIN DAS BASES PELO NOME DO MINICIPIO
base_mesclada = base_reverse %>%
  dplyr::mutate(municipio = stringr::str_to_upper(abjutils::rm_accent(municipio))) %>%
  dplyr::mutate(shipping_state  =  stringr::str_to_upper(abjutils::rm_accent(shipping_state)))
%>%
  dplyr::mutate(state_reverse   =   stringr::str_to_upper(abjutils::rm_accent(state_reverse)))
%>%
  dplyr::rename(state_codificado = state) %>%
  dplyr::left_join(estados %>%
                     dplyr::mutate(name_state                                                =
stringr::str_to_upper(abjutils::rm_accent(name_state))),
                   by = c("state_reverse"="name_state")) %>%
  dplyr::mutate(cidade = ifelse(is.na(municipio),NA,paste0(municipio,"-",state))) %>%
  dplyr::left_join(atlas_2010,by = c("cidade"="cidade"))

# LIMPANDO AS VARIAVEIS  -------------------------------------------------------------------

# REMOVENDO LINHAS EM QUE A VARIAVEL TARGET_DEFAULT E NULA
base_mesclada_1 = base_mesclada %>%
  dplyr::mutate(target_default = ifelse(target_default=="",NA,target_default)) %>%
  dplyr::filter(!is.na(target_default))
```

```
# QUANTIDADE DE REGISTROS ""
sum(as.data.frame(base_mesclada_1) == "",na.rm = T)

# SUBSTITUINDO "" POR NA
base_mesclada_2 = as.data.frame(base_mesclada_1)
base_mesclada_2[base_mesclada_2 == ""] <- NA

# TRATANDO ALGUMAS VARIAVEIS
# APLICATION_TIME_APLIED: RETIRAR HORA DA APLICACAO
# EXTERNAL_DATA_PROVIDER_EMAIL_SEEN_BEFORE = -999 POR NA
# email.hotmail.com e email.hotmaill.com
# variavel "reported_income" nos valores Inf tranforma-se em NA
# Criando variavel Regiao

base_mesclada_3 = base_mesclada_2 %>%
 dplyr::mutate(hora_aplicacao = as.numeric(substr(application_time_applied,start = 1,stop =
2)),
        external_data_provider_email_seen_before =
ifelse(external_data_provider_email_seen_before==-
999,NA,external_data_provider_email_seen_before),
        email = ifelse(email=="hotmaill.com","hotmail.com",email),
        email = ifelse(email=="gmaill.com","gmail.com",email),
        reported_income = ifelse(reported_income==Inf,NA,reported_income),
        regiao = dplyr::case_when(state.y %in% c("AM","RR","AP","PA","TO","RO","AC") ~
"Região Norte",
                  state.y %in% c("MA","PI","CE","RN","PE","PB","SE","AL","BA") ~
"Região Nordeste",
                  state.y %in% c("MT","MS","GO") ~ "Região Centro-Oeste",
                  state.y %in% c("SP","RJ","ES","MG") ~ "Região Sudeste",
                  state.y %in% c("PR","RS","SC") ~ "Região Sul",TRUE ~ "")) %>%
 dplyr::select(-c(application_time_applied))

# TIPO
DADOS/QTDE_VALORES_AUSENTES/PERC_VALORES_AUSENTES/QTDE_VALORES_UNICOS
df_colunas = function(base_analise){
 colunas = data.frame(row.names =
c("Coluna","Tipo","qtde_Valores_ausentes","Perc_Valores_ausentes"))
 for(i in 1:dim(base_analise)[2]){
  colunas_i = data.frame(coluna=colnames(base_analise)[i],
            Tipo = class(base_analise[,i]),
            qtde_Valores_ausentes = sum(is.na(base_analise[,i])),
            Perc_Valores_ausentes =
round(sum(is.na(base_analise[,i]))*100/nrow(base_analise),2),
            qtde_valores_unicos = length(unique(base_analise[,i])))
  colunas = rbind(colunas,colunas_i) %>% dplyr::arrange(desc(Perc_Valores_ausentes))
 }
 return(colunas)
```

```
}
colunas = df_colunas(as.data.frame(base_mesclada_3))

# TRATANDO COLUNAS COM NULOS
colunas_nulos = colunas %>%
  dplyr::filter(qtde_Valores_ausentes>0) %>%
  dplyr::arrange(desc(qtde_Valores_ausentes)) %>%
  dplyr::mutate(perc_label = paste0(as.character(format(Perc_Valores_ausentes,
decimal.mark = ',')),"%"))
colunas_nulos$coluna <- factor(colunas_nulos$coluna,levels =
rev(colunas_nulos$coluna),ordered = T)

# REMOVER COLUNAS DA BASE DE ENRRIQUECIMENTO
library(Hmisc)
colunas_nulos2=colunas_nulos %>%
  dplyr::filter(qtde_Valores_ausentes!=6644) %>%
  dplyr::filter(coluna %nin% c("municipio", "cidade", "uf.x", "uf.y","state.x"))

# GRAFICO DE COLUNAS COM NULOS
graf_nulos = ggplot(colunas_nulos2, aes(x=coluna))+
  labs(title = "Percentual de valores ausentes por coluna",x="Coluna",y="Percentual")+
  geom_bar(aes(weight = Perc_Valores_ausentes,fill = "#66C2A5"))+
  ylim(0,110)+
  geom_text(aes(label = perc_label, y = Perc_Valores_ausentes),hjust = -0.2,size = 4) +
  coord_flip()+
  scale_fill_brewer(palette = "Set2") +
  theme_bw()+theme(legend.position = "none"); graf_nulos
# SALVANDO GRAFICO
ggsave(plot = graf_nulos,filename = "codigo-fonte/imagens/graf_colunas_nulas.png",width =
10,height = 7)

# VARIAVEIS COM NULOS QUE RECEBERAO A CLASSE MAIS FREQUENTE
Mode <- function(x){ ux <- sort(unique(x));ux[which.max(tabulate(match(x, ux)))]}
base_mesclada_3.1 = base_mesclada_3 %>%
  dplyr::mutate(facebook_profile = ifelse(is.na(facebook_profile),'Ausente',facebook_profile),
          marketing_channel =
ifelse(is.na(marketing_channel),Mode(base_mesclada_3$marketing_channel),marketing_ch
annel),
          n_bankruptcies =
ifelse(is.na(n_bankruptcies),Mode(base_mesclada_3$n_bankruptcies),n_bankruptcies),
          n_defaulted_loans =
ifelse(is.na(n_defaulted_loans),Mode(base_mesclada_3$n_defaulted_loans),n_defaulted_lo
ans),
          external_data_provider_credit_checks_last_year =
ifelse(is.na(external_data_provider_credit_checks_last_year),

Mode(base_mesclada_3$external_data_provider_credit_checks_last_year),
```

```
                                        external_data_provider_credit_checks_last_year))

# CRIANDO VARIAVEIS
# BINS State
# Turno através da variavel hora_aplicação
df_state_bins = data_frame(state = unique(acquisition_train$state), bins_state =
paste0("bins_",rep(seq(1,length(unique(acquisition_train$state)),5),5)[1:51]))
base_mesclada_4 = base_mesclada_3.1 %>%
dplyr::left_join(df_state_bins,by=c("state_codificado"="state"))%>%
 dplyr::mutate(turno = dplyr::case_when(hora_aplicacao>=0 & hora_aplicacao< 6 ~
"Madrugada",
                       hora_aplicacao>=6 & hora_aplicacao< 12 ~ "Manha",
                       hora_aplicacao>=12 & hora_aplicacao< 18 ~ "Tarde",
                       hora_aplicacao>=18 ~ "Noite")) %>%
 dplyr::mutate(Latitude = ifelse(is.na(Latitude),0,Latitude)) %>%
 dplyr::mutate(Longitude = ifelse(is.na(Longitude),0,Longitude)) %>%
 dplyr::mutate(last_amount_borrowed =
ifelse(is.na(last_amount_borrowed),0,last_amount_borrowed),
         last_borrowed_in_months =
ifelse(is.na(last_borrowed_in_months),0,last_borrowed_in_months),
         ok_since = ifelse(is.na(ok_since),0,ok_since))

# Criando Cluster para a variável longitude e latitude
df = base_mesclada_4 %>% dplyr::select(Latitude,Longitude)
set.seed(240) # Setting seed
kmeans.re <- kmeans(df, centers = 10, nstart = 20)
base_mesclada_4$Group_lat_lon = paste0("cluster_",kmeans.re$cluster)

save(x=base_mesclada_4,file="codigo-fonte/data/base_mesclada_4.Rdata")

# PRE PROCESSAMENTO  ----------------------------------------------------------------

# SELECAO VARIAVEIS

# Removendo variáveis duplicadas que surgiram nos joins
base_mesclada_5 = base_mesclada_4 %>%
 dplyr::rename(estado_loc = state.x) %>%
 dplyr::select(-
c(name_state,state.y,cidade_atlas,uf.y,uf.x,cidade,codmun6,codmun7,municipio,município))

# Removendo variáveis que tiveram mais de 50% de valores Nulos
base_mesclada_6 = base_mesclada_5 %>%
 dplyr::select(-c(channel,external_data_provider_credit_checks_last_2_year))

# Variaveis que serão removidas por conta da codificacao ou ter classe demais
base_mesclada_7 = base_mesclada_6 %>%
```

```
 dplyr::select(-c(ids,score_1,score_2,reason,state_codificado,zip,job_name, profile_tags,
user_agent,shipping_state,state_reverse))

# Outras removidas
base_mesclada_8 = base_mesclada_7 %>%
 dplyr::select(-c(external_data_provider_first_name,lat_lon, shipping_zip_code,
target_fraud,
          Latitude,Longitude,profile_phone_number))

# TRANSFORMANDO COLUNA TARGET EM 0 OU 1, E depois em fator
base_mesclada_9 = base_mesclada_8 %>%
 dplyr::mutate(target_default = ifelse(target_default=="True",1,0)) %>%
 dplyr::mutate(target_default = factor(target_default,levels = c(1,0),ordered = T))

# GARANTINDO QUE OS NOMES ESTEJAM EM MINUSCULO
base_preparada <- base_mesclada_9 %>% janitor::clean_names()
dim(base_preparada)

# DEFININDO SEMENTE
set.seed(123)
base_preparada$facebook_profile = factor(base_preparada$facebook_profile,ordered = T)
base_preparada$facebook_profile = ifelse(base_preparada$facebook_profile=="True",1,0)
b_oneHotEncoder = dataPreparation::one_hot_encoder(base_preparada,drop=T)
b_oneHotEncoder$target_default = factor(b_oneHotEncoder$target_default,levels =
c(1,0),ordered = T)
dim(b_oneHotEncoder)

# DIVIDINDO A POPULACAO EM TREINO E TESTE
dt_split = rsample::initial_split(data = b_oneHotEncoder , prop = 0.75, strata =
target_default)
# save(dt_split, file = "codigo-fonte/data/dt_split.Rdata")

# DATA PREPARATION
data_recipe = rsample::training(x = dt_split) %>%
 recipes::recipe(target_default ~ .) %>% # INFORMANDO QUAL O DEFAULT
 themis::step_downsample(target_default, under_ratio = 1) %>%
 recipes::step_impute_median(recipes::all_numeric_predictors()) %>%
 recipes::step_novel(recipes::all_nominal_predictors(), -recipes::all_outcomes()) %>%
#TRANSFORMANDO AS VARIAVEIS NOMINAIS EM FATOR
 recipes::step_center(recipes::all_numeric_predictors()) %>% # NORMALIZANDO OS DADOS
NUMERICOS PARA TER UMA MEDIA DE ZERO
 recipes::step_scale(recipes::all_numeric_predictors()) %>% # NORMALIZANDO OS DADOS
NUMERICOS PARA TER UM DESVIO PADRAO DE UM
 recipes::step_corr(recipes::all_numeric_predictors(), threshold = 0.85, method = "pearson")
%>% #REMOVENDO VARIAVEIS NUMERICAS ALTAMENTE CORRELACIONADAS
 recipes::step_zv(recipes::all_numeric_predictors(), -recipes::all_outcomes()) #REMOVENDO
VARIAVEIS COM VARIABILIDADE PROXIMA DE ZERO
```

```
# AJUSTE DOS MODELOS ------------------------------------------------------------

# CROSS VALIDATION
cv = rsample::vfold_cv(rsample::training(x = dt_split), v = 6, repeats = 2, strata = target_default)

# RANDOM FOREST ========================
# ESPECIFICANDO O MODELO
rf_spec = parsnip::rand_forest() %>%
  set_args(mtry = tune()) %>%
  parsnip::set_engine("ranger", importance = "impurity") %>%
  parsnip::set_mode("classification")
# CRIANDO WORKFLOW
rf_workflow = workflows::workflow() %>%
  workflows::add_recipe(data_recipe) %>%
  workflows::add_model(rf_spec)
# GRID DE PARAMETROS
rf_grid = expand.grid(mtry = c(45,50,55))
# AJUSTE DO MODELO
rf_fit_tune = rf_workflow %>%
  tune::tune_grid(resamples = cv, # CV object
            grid = rf_grid, # grid of values to try
            metrics = yardstick::metric_set(recall,f_meas,accuracy,kap,roc_auc,sens), # metrics we care about
            control = tune::control_resamples(save_pred = TRUE,verbose=T)
  )
# SALVANDO O MODELO EM UM ARQUIVO .RDATA
# save(rf_fit_tune, file = "codigo-fonte/data/rf_fit_tune_predicao_1.Rdata")
# save(rf_workflow, file = "codigo-fonte/data/rf_workflow.Rdata")
# CARREGANDO O MODELO
# load(file = "codigo-fonte/data/rf_fit_tune_predicao_1.Rdata", verbose = T)
# load(file = "codigo-fonte/data/rf_workflow.Rdata", verbose = T)

# METRICAS NO TREINO
rf_fit_tune %>% collect_metrics()

# MOSTRANDO O MELHOR MODELO NO TREINO
rf_fit_tune %>% tune::show_best(metric = "recall")
rf_fit_tune %>% tune::show_best(metric = "accuracy")

# AVALIE O MODELO NO CONJUNTO DE TESTE ------------------------------------

# FINALIZANDO O FLUXO DE TRABALHO
param_final <- rf_fit_tune %>% tune::select_best(metric = "accuracy")
rf_workflow_best <- rf_workflow %>% finalize_workflow(param_final)
```

```
# Ajustar no conjunto de treinamento e avaliar no conjunto de teste
rf_fit <- rf_workflow_best %>% tune::last_fit(dt_split)

# COLENTANDO METRICAS MELHOR MODELO AVALIADO no TESTE
rf_fit %>% collect_metrics()

# GERANDO PREDICOES DO DATASET DE TREINO
train_predictions <- rf_fit %>% collect_predictions(); train_predictions

# CONFUSION MATRIX
graf_MC_rf = rf_fit %>%
  collect_predictions() %>%
  conf_mat(truth = target_default, estimate = .pred_class, dnn = c("Predição", "Real")) %>%
  autoplot(type = "heatmap") +
  labs(title = "Matriz de confusão - Randon Forest");graf_MC_rf
ggsave(plot = graf_MC_rf,filename = "codigo-fonte/imagens/graf_MC_rf.png",width =
4,height =3)

# METRICAS
rf_metrics <- metric_set(accuracy, sens, spec)
train_predictions %>% rf_metrics(truth = target_default, estimate = .pred_class)

# CURVA ROC
rf_curva_ROC = rf_fit_tune %>%
  collect_predictions() %>%
  group_by(id) %>% # id contains our folds
  roc_curve(target_default, .pred_1) %>%
  autoplot() +
  labs(title="Curva ROC - Randon Forest",
       x = "1 - Especificidade",
       y = "Sensibilidade",
       fill="Repetição")+
  scale_fill_brewer(palette = "Set2") +
  theme_bw(); rf_curva_ROC
ggsave(rf_curva_ROC,filename = "codigo-fonte/imagens/rf_curva_ROC.png",width =
6,height = 4.2)

# OTIMIZANDO O CUTOFF RF ---------------------------------------

# OTIMIZANDO O PONTO DE CORTE
rf_fit %>%
  tune::collect_predictions() %>%
  dplyr::group_by(id) %>%
  yardstick::roc_curve(target_default, .pred_1) %>%
  dplyr::filter(specificity >= 0.5) %>%
  dplyr::arrange(desc(sensitivity)) %>%
  dplyr::filter(dplyr::row_number() == 1)
```

```
# OTIMIZANDO O PONTO DE CORTE
set.seed(123)
rf_fit_preds = rf_fit %>% collect_predictions()
rf_fit_preds_new <-
  rf_fit_preds %>%
  mutate(.pred_class = make_two_class_pred(.pred_1, levels(target_default), threshold =
0.416))

# OTIMIZANDO O PONTO DE CORTE
rf_metrics <- metric_set(accuracy, sens, spec)
rf_fit_preds_new %>% rf_metrics(truth = target_default, estimate = .pred_class)

# CONFUSION MATRIX nova
graf_MC_rf_otim = rf_fit_preds_new %>%
  conf_mat(truth = target_default, estimate = .pred_class, dnn = c("Predição", "Real")) %>%
  autoplot(type = "heatmap") +
  labs(title = "Matriz de confusão - Randon Forest \nOtimizado para sensibilidade");
graf_MC_rf_otim
ggsave(plot = graf_MC_rf_otim,filename = "codigo-
fonte/imagens/graf_MC_rf_otim.png",width = 4,height =3)

# GRAFICO ANALISANDO A PREDICAO DO TREINO AJUSTADO
graf_dens_rf_train = rf_fit_preds_new %>%
  ggplot() +
  geom_density(aes(x = .pred_1, fill = target_default), alpha = 0.5) +
  labs(x = "Predição",y="Densidade",
       title = " Distribuição das predições para as classes 1 e 0 - Modelo Randon
Forest",fill="Default") +
  scale_fill_viridis_d() +
  theme_bw(); graf_dens_rf_train

# SVM ========================
# ESPECIFICANDO O MODELO
SVM_spec = parsnip::svm_rbf(
    cost = tune(),
    rbf_sigma = tune()) %>%
  parsnip::set_engine("kernlab", importance = "impurity") %>%
  parsnip::set_mode("classification")
# CRIANDO WORKFLOW
SVM_workflow = workflows::workflow() %>%
  workflows::add_recipe(data_recipe) %>%
  workflows::add_model(SVM_spec)
# GRID DE PARAMETROS
SVM_grid = expand.grid(cost = c(0.1),rbf_sigma=c(0.01))
# AJUSTE DO MODELO
SVM_fit_tune = SVM_workflow %>%
```

```
 tune::tune_grid(resamples = cv, # CV object
            grid = SVM_grid, # grid of values to try
            metrics = yardstick::metric_set(recall,f_meas,accuracy,kap,roc_auc,sens), # 
metrics we care about
            control = tune::control_resamples(save_pred = TRUE,verbose=T)
 )

# SALVANDO O MODELO EM UM ARQUIVO .RDATA
# save(SVM_fit_tune, file = "codigo-fonte/data/SVM_fit_tune_predicao_1.Rdata")
# save(SVM_workflow, file = "codigo-fonte/data/SVM_workflow.Rdata")

# CARREGANDO O MODELO
# load(file = "codigo-fonte/data/SVM_fit_tune_predicao_1.Rdata", verbose = T)
# load(file = "codigo-fonte/data/SVM_workflow.Rdata", verbose = T)

# METRICAS NO TREINO
SVM_fit_tune %>% collect_metrics()

# MOSTRANDO O MELHOR MODELO NO TREINO
SVM_fit_tune %>% tune::show_best(metric = "recall")
SVM_fit_tune %>% tune::show_best(metric = "accuracy")

# AVALIE O MODELO NO CONJUNTO DE TESTE ------------------------------------

# FINALIZANDO O FLUXO DE TRABALHO
param_final <- SVM_fit_tune %>% tune::select_best(metric = "accuracy")
SVM_workflow_best <- SVM_workflow %>% finalize_workflow(param_final)

# Ajustar no conjunto de treinamento e avaliar no conjunto de teste
SVM_fit <- SVM_workflow_best %>% tune::last_fit(dt_split)

# COLENTANDO METRICAS MELHOR MODELO AVALIADO no TESTE
SVM_fit %>% collect_metrics()

# GERANDO PREDICOES DO DATASET DE TREINO
train_predictions <- SVM_fit %>% collect_predictions(); train_predictions

# CONFUSION MATRIX
graf_MC_svm = SVM_fit %>%
  collect_predictions() %>%
  conf_mat(truth = target_default, estimate = .pred_class, dnn = c("Predição", "Real")) %>%
  autoplot(type = "heatmap") +
  labs(title = "Matriz de confusão - SVM "); graf_MC_svm
ggsave(plot = graf_MC_svm, filename = "codigo-fonte/imagens/graf_MC_svm.png", width = 
4, height =3)

# METRICAS
```

```
SVM_metrics <- metric_set(accuracy, sens, spec)
train_predictions %>% SVM_metrics(truth = target_default, estimate = .pred_class)

# CURVA ROC
SVM_curva_ROC = SVM_fit_tune %>%
  collect_predictions() %>%
  group_by(id) %>% # id contains our folds
  roc_curve(target_default, .pred_1) %>%
  autoplot() +
  labs(title="Curva ROC - SVM",
       x = "1 - Especificidade",
       y = "Sensibilidade")+
  scale_fill_brewer(palette = "Set2") +
  theme_bw();SVM_curva_ROC
ggsave(SVM_curva_ROC,filename = "codigo-fonte/imagens/SVM_curva_ROC.png",width = 6,height = 4.2)


# OTIMIZANDO O CUTOFF SVM ----------------------------------------

# OTIMIZANDO O PONTO DE CORTE PARA SENSIBILIDADE
SVM_fit %>%
  tune::collect_predictions() %>%
  dplyr::group_by(id) %>%
  yardstick::roc_curve(target_default, .pred_1) %>%
  dplyr::filter(specificity >= 0.5) %>%
  dplyr::arrange(desc(sensitivity)) %>%
  dplyr::filter(dplyr::row_number() == 1)
# 0.413

# OTIMIZANDO O PONTO DE CORTE
set.seed(123)
SVM_fit_preds = SVM_fit %>% collect_predictions()
SVM_fit_preds_new <-
  SVM_fit_preds %>%
  mutate(.pred_class = make_two_class_pred(.pred_1, levels(target_default), threshold = 0.413))

# OTIMIZANDO O PONTO DE CORTE
SVM_metrics <- metric_set(accuracy, sens, spec)
SVM_fit_preds_new %>% SVM_metrics(truth = target_default, estimate = .pred_class)

# CONFUSION MATRIX nova
graf_MC_svm_otim = SVM_fit_preds_new %>%
  conf_mat(truth = target_default, estimate = .pred_class, dnn = c("Predição", "Real")) %>%
  autoplot(type = "heatmap") +
```

```
 labs(title = "Matriz de confusão - SVM \nOtimizado para sensibilidade"); graf_MC_svm_otim
ggsave(plot = graf_MC_svm_otim,filename = "codigo-fonte/imagens/graf_MC_svm_otim.png",width = 4,height =3)

# GRAFICO ANALISANDO A PREDICAO DO TREINO AJUSTADO
graf_dens_SVM_train = SVM_fit_preds_new %>%
  ggplot() +
  geom_density(aes(x = .pred_1, fill = target_default), alpha = 0.5) +
  labs(x = "Predição",y="Densidade",
       title = " Distribuição das predições para as classes 1 e 0 - Modelo SVM",fill="Default") +
  scale_fill_viridis_d() +
  theme_bw(); graf_dens_SVM_train

# Xgboost ========================
# ESPECIFICANDO O MODELO
xgb_spec <- parsnip::boost_tree(
  trees = 1000,
  tree_depth = tune(),
  min_n = tune(),
  loss_reduction = tune(),
  sample_size = tune(),
  mtry = tune(),
  learn_rate = tune()
) %>%
  parsnip::set_engine("xgboost") %>%
  parsnip::set_mode("classification")

# GRADE DE PARAMETROS
xgb_grid <- grid_latin_hypercube(
  tree_depth(),
  min_n(),
  loss_reduction(),
  sample_size = sample_prop(),
  finalize(mtry(),  training(dt_split)),
  learn_rate(),
  size = 5
)

# CRIANDO WORKFLOW
xgb_workflow = workflows::workflow() %>%
  workflows::add_recipe(data_recipe) %>%
  workflows::add_model(XGB_spec)
# AJUSTE DO MODELO
xgb_fit_tune = xgb_workflow %>%
  tune::tune_grid(resamples = cv, # CV object
                  grid = XGB_grid, # grid of values to try
```

```
            metrics = yardstick::metric_set(recall,f_meas,accuracy,kap,roc_auc,sens),
            control = tune::control_resamples(save_pred = TRUE,verbose=T)
 )

# SALVANDO O MODELO EM UM ARQUIVO .RDATA
# save(xgb_fit_tune, file = "codigo-fonte/data/xgb_fit_tune_predicao_1.Rdata")
# save(xgb_workflow, file = "codigo-fonte/data/xgb_workflow.Rdata")
# CARREGANDO O MODELO
# load(file = "codigo-fonte/data/xgb_fit_tune_predicao_1.Rdata", verbose = T)
# load(file = "codigo-fonte/data/xgb_workflow.Rdata", verbose = T)
# METRICAS NO TREINO
xgb_fit_tune  %>% collect_metrics()

# MOSTRANDO O MELHOR MODELO NO TREINO
xgb_fit_tune %>% tune::show_best(metric = "accuracy")

# CURVA ROC
xgb_curva_ROC = xgb_fit_tune %>%
  collect_predictions() %>%
  group_by(id) %>% # id contains our folds
  roc_curve(target_default, .pred_1) %>%
  autoplot() +
  labs(title="Curva ROC - XGBoost",
      x = "1 - Especificidade",
      y = "Sensibilidade")+
  scale_fill_brewer(palette = "Set2") +
  theme_bw(); xgb_curva_ROC
ggsave(plot = xgb_curva_ROC,filename = "codigo-fonte/imagens/xgb_curva_ROC.png",width
= 6,height = 4.2)

# AVALIE O MODELO NO CONJUNTO DE TESTE ---------------------------------------

# FINALIZANDO O FLUXO DE TRABALHO
param_final <- xgb_fit_tune %>% tune::select_best(metric = "accuracy")
xgb_workflow_best <- xgb_workflow %>% finalize_workflow(param_final)

# Ajustar no conjunto de treinamento e avaliar no conjunto de teste
xgb_fit <- xgb_workflow_best %>% tune::last_fit(dt_split)

# COLENTANDO METRICAS MELHOR MODELO AVALIADO no TESTE
xgb_fit %>% collect_metrics()

# GERANDO PREDICOES DO DATASET DE TREINO
train_predictions <- xgb_fit %>% collect_predictions(); train_predictions

# CONFUSION MATRIX
graf_MC_xgb = xgb_fit %>%
```

```
  collect_predictions() %>%
  conf_mat(truth = target_default, estimate = .pred_class, dnn = c("Predição", "Real")) %>%
  autoplot(type = "heatmap") +
  labs(title = "Matriz de confusão - XGBoost"); graf_MC_xgb
ggsave(plot = graf_MC_xgb,filename = "codigo-fonte/imagens/graf_MC_xgb.png",width = 4,height =3)

# METRICAS
xgb_metrics <- metric_set(accuracy, sens, spec)
train_predictions %>% xgb_metrics(truth = target_default, estimate = .pred_class)
ggsave(plot = xgb_curva_ROC,filename = "codigo-fonte/imagens/xgb_curva_ROC.png",width = 6,height = 4.2)

# OTIMIZANDO O CUTOFF XGB ---------------------------------------

# OTIMIZANDO O PONTO DE CORTE
a = xgb_fit %>%
  tune::collect_predictions() %>%
  dplyr::group_by(id) %>%
  yardstick::roc_curve(target_default, .pred_1) %>%
  dplyr::filter(specificity >= 0.5) %>%
  dplyr::arrange(desc(sensitivity)) %>%
  dplyr::filter(dplyr::row_number() == 1)

# OTIMIZANDO O PONTO DE CORTE
set.seed(123)
xgb_fit_preds = xgb_fit %>% collect_predictions()
xgb_fit_preds_new <-
  xgb_fit_preds %>%
  mutate(.pred_class = make_two_class_pred(.pred_1, levels(target_default), threshold = 0.4999863))

# OTIMIZANDO O PONTO DE CORTE
xgb_metrics <- metric_set(accuracy, sens, spec)
xgb_fit_preds_new %>% xgb_metrics(truth = target_default, estimate = .pred_class)

# CONFUSION MATRIX nova
graf_MC_xgb_otim = xgb_fit_preds_new %>%
  conf_mat(truth = target_default, estimate = .pred_class, dnn = c("Predição", "Real")) %>%
  autoplot(type = "heatmap")+
  labs(title = "Matriz de confusão - XGBoost\n Otimizado para sensibilidade");graf_MC_xgb_otim
ggsave(plot = graf_MC_xgb_otim,filename = "codigo-fonte/imagens/graf_MC_xgb_otim.png",width = 4,height =3)

confusion_matrix_new = xgb_fit_preds_new %>%
  yardstick::conf_mat(truth = target_default, estimate = .pred_class);confusion_matrix_new
```

```
True_Positive = confusion_matrix_new$table[1] # Eram inadimplentes e eu disse que era inadimplente ()
False_Positive = confusion_matrix_new$table[3] # Eram adimplentes e eu disse que era inadimplentes ()
False_Negativo = confusion_matrix_new$table[2] # Eram inadimplentes e eu disse que era adimplentes (Pior)
True_Negative = confusion_matrix_new$table[4] # Eram adimplentes e eu disse que era adimplentes ()


# GRAFICO ANALISANDO A PREDICAO DO TREINO AJUSTADO
graf_dens_xgb_train = xgb_fit_preds_new %>%
  ggplot() +
  geom_density(aes(x = .pred_1, fill = target_default), alpha = 0.5) +
  labs(x = "Predição",y="Densidade",
       title = " Distribuição das predições para as classes 1 e 0 - Modelo Xgboost",fill="Default")
+
  scale_fill_viridis_d() +
  theme_bw(); graf_dens_xgb_train

# COMPARANDO MODELOS COM METRICAS DE TESTE ----------------------------------------------------
table_metrics <- metric_set(accuracy, sens, spec)

# MODEL METRICS TREINO
rf_metrics = rf_fit %>%
  collect_predictions() %>%
  table_metrics(truth = target_default, estimate = .pred_class) %>%
  dplyr::mutate(model = "Random Forest")

svm_metrics = SVM_fit %>%
  collect_predictions() %>%
  table_metrics(truth = target_default, estimate = .pred_class) %>%
  dplyr::mutate(model = "SVM")

xgb_metrics = xgb_fit %>%
  collect_predictions() %>%
  table_metrics(truth = target_default, estimate = .pred_class) %>%
  dplyr::mutate(model = "XGBoost")

model_compare = dplyr::bind_rows(
  rf_metrics,
  svm_metrics,
  xgb_metrics
) %>% dplyr::mutate(.metric = dplyr::case_when(.metric=="accuracy" ~ "Acurácia",
                                  .metric=="sens" ~ "Sensibilidade",
```

```
                                .metric=="spec" ~ "Especificidade")) %>%
  dplyr::rename(Metricas =.metric,
                Modelos = model) %>%
  dplyr::mutate(label_name                                                    =
paste0(format(round(.estimate*100,2),decimal.mark=","),"%"))

graf_comp_modelos = ggplot(data=model_compare, aes(x=Metricas, y=.estimate,
group=Metricas)) +
  geom_bar(stat="identity",fill="#66C2A5")+
  geom_text(aes(label=label_name), vjust=-0.5, size=4)+
  scale_fill_brewer(palette = "Set2") +
  labs(title = "Métricas dos modelos ajustados na base de treino",
       y= "Valor",
       x = "Métricas")+
  theme_bw()+
  facet_grid(. ~ Modelos); graf_comp_modelos
ggsave(plot              =              graf_comp_modelos,filename              =              "codigo-
fonte/imagens/graf_comp_modelos.png",width = 12,height =5)


# MODEL METRICS TREINO OTIMIZADO
rf_metrics_otm = rf_fit %>%
  collect_predictions() %>%
  mutate(.pred_class  =  make_two_class_pred(.pred_1,  levels(target_default),  threshold  =
0.416)) %>%
  table_metrics(truth = target_default, estimate = .pred_class) %>%
  dplyr::mutate(model = "Random Forest")

svm_metrics_otm = SVM_fit %>%
  collect_predictions() %>%
  mutate(.pred_class  =  make_two_class_pred(.pred_1,  levels(target_default),  threshold  =
0.413)) %>%
  table_metrics(truth = target_default, estimate = .pred_class) %>%
  dplyr::mutate(model = "SVM")

xgb_metrics_otm = xgb_fit %>%
  collect_predictions() %>%
  mutate(.pred_class  =  make_two_class_pred(.pred_1,  levels(target_default),  threshold  =
0.4999863)) %>%
  table_metrics(truth = target_default, estimate = .pred_class) %>%
  dplyr::mutate(model = "XGBoost")

model_compare_otm = dplyr::bind_rows(
  rf_metrics_otm,
  svm_metrics_otm,
  xgb_metrics_otm
) %>% dplyr::mutate(.metric = dplyr::case_when(.metric=="accuracy" ~ "Acurácia",
```

```
                                .metric=="sens" ~ "Sensibilidade",
                                .metric=="spec" ~ "Especificidade")) %>%
  dplyr::rename(Metricas =.metric,
                Modelos = model) %>%
  dplyr::mutate(label_name =
paste0(format(round(.estimate*100,2),decimal.mark=","),"%"))

graf_comp_modelos_otim = ggplot(data=model_compare_otm, aes(x=Metricas,
y=.estimate, group=Metricas)) +
  geom_bar(stat="identity",fill="#66C2A5")+
  geom_text(aes(label=label_name), vjust=-0.5, size=4)+
  scale_fill_brewer(palette = "Set2") +
  labs(title = "Métricas dos modelos ajustados na base de treino - OTIMIZADOS PARA
SENSIBILIDADE",
       y= "Valor",
       x = "Métricas")+
  theme_bw()+
  facet_grid(. ~ Modelos);graf_comp_modelos_otim
ggsave(plot = graf_comp_modelos_otim,filename = "codigo-
fonte/imagens/graf_comp_modelos_otim.png",width = 12,height =5)

# ======= Variáveis mais importantes
library(vip)
# Finalizando o modelo
wf_rf_final <- rf_workflow %>% finalize_workflow(select_best(rf_fit_tune, "accuracy"))
# treinando o modelo com a base de treino
modelo_final <- fit(wf_rf_final, rsample::training(x = dt_split))
graf_var_importantes = vip(modelo_final$fit$fit) +
  aes(fill = "#66C2A5") +
  labs(title = "Variáveis mais importantes",
       x= "Variáveis",
       y = "Importância")+
  scale_fill_brewer(palette = "Set2") +
  theme_bw() +
  theme(legend.position = "none");graf_var_importantes
ggsave(plot = graf_var_importantes,filename = "codigo-
fonte/imagens/graf_var_importantes.png",width = 7,height =5)
```