Projeto de Feature Engineering

Esta apresentação explora o processo detalhado de feature engineering aplicado ao dataset, destacando as decisões técnicas que sustentam a melhoria da qualidade final. Abordaremos os principais métodos adotados, seus impactos intermediários e como esses refletiram nas métricas do modelo.

Turma: 10 DTS

Professor: Felipe Teodoro.

RM 357240 – José Carlos Basílio.

RM 358285 – Rafael Henrique Gallo.

RM 358024 – Lucas Nascimento dos Santos.

RM 358921 – Társis Fortes Tavares.

Problema de negócio

A QuantumFinance possui um modelo de risco de crédito desatualizado, que tem concedido cartões para um número muito alto de mau pagadores e gerando problemas financeiros para a companhia.

QUANTUM FINANCE

Objetivo:

Desenvolver um modelo de crédito mais preciso para identificar maus pagadores.

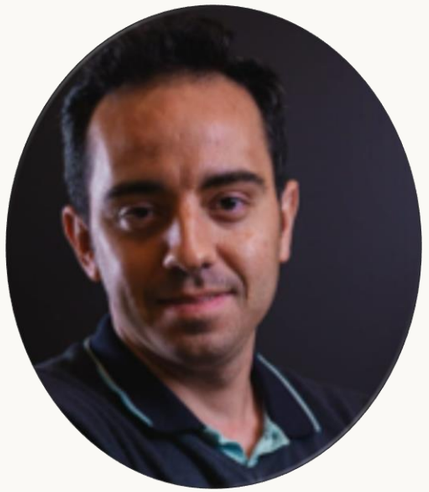
Equipe de trabalho:

Neste contexto, o HEAD de Ciência de Dados chamou os melhores talentos de sua equipe para ajuda-lo na construção de um novo modelo que será utilizado antes da concessão de cartão de crédito aos requerentes.

José Basílio

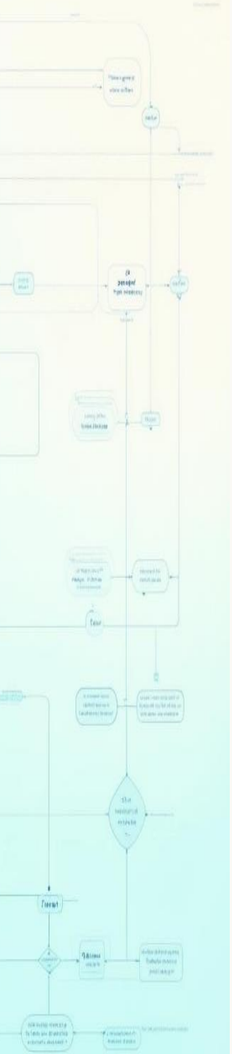
Lucas Nascimento dos Santos

Rafael Gallo



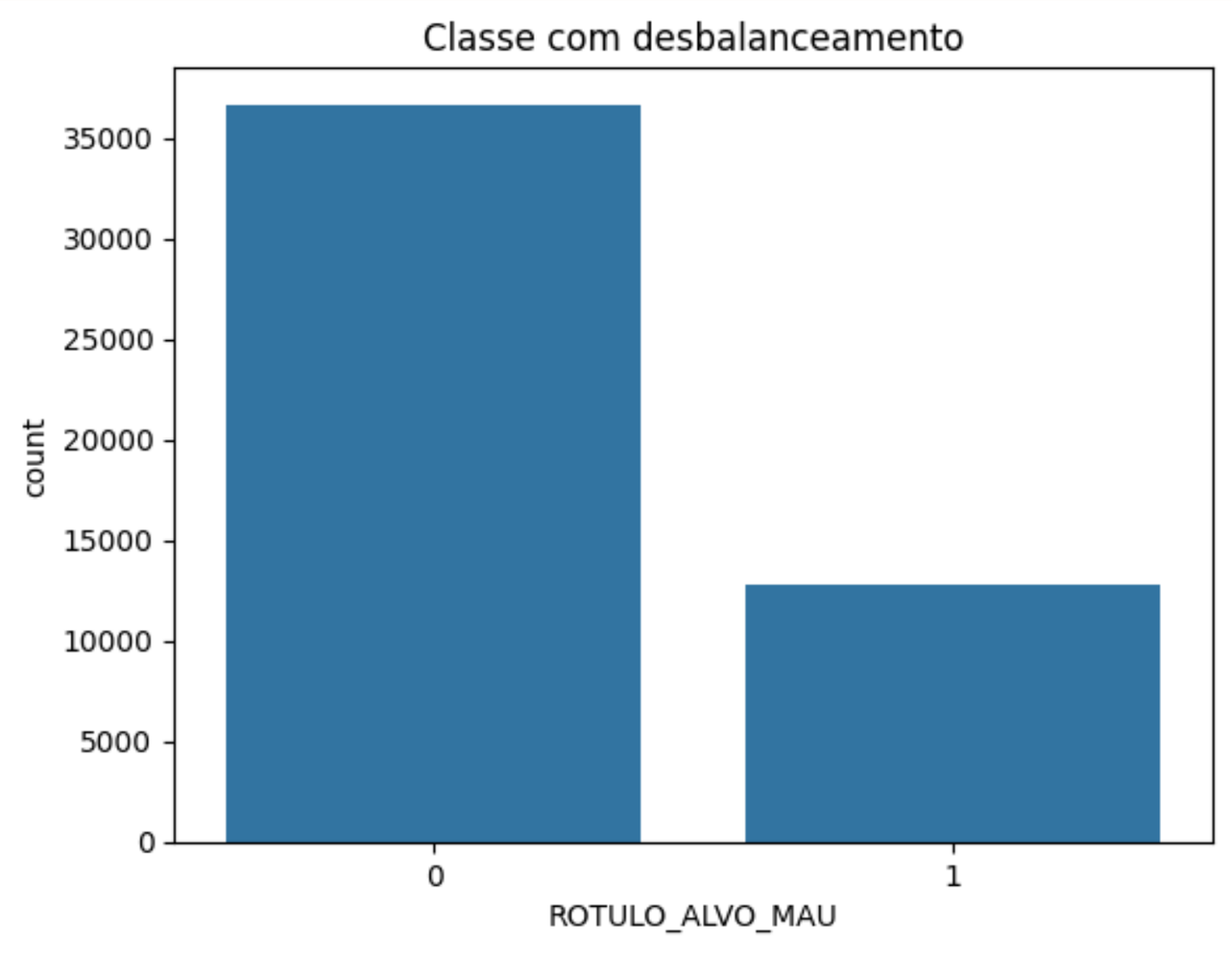
Társis Tavares

Estrutura do credit scoring



O código com todos os detalhes está descrito no notebook. A estrutura dos passos principais se encontram abaixo:

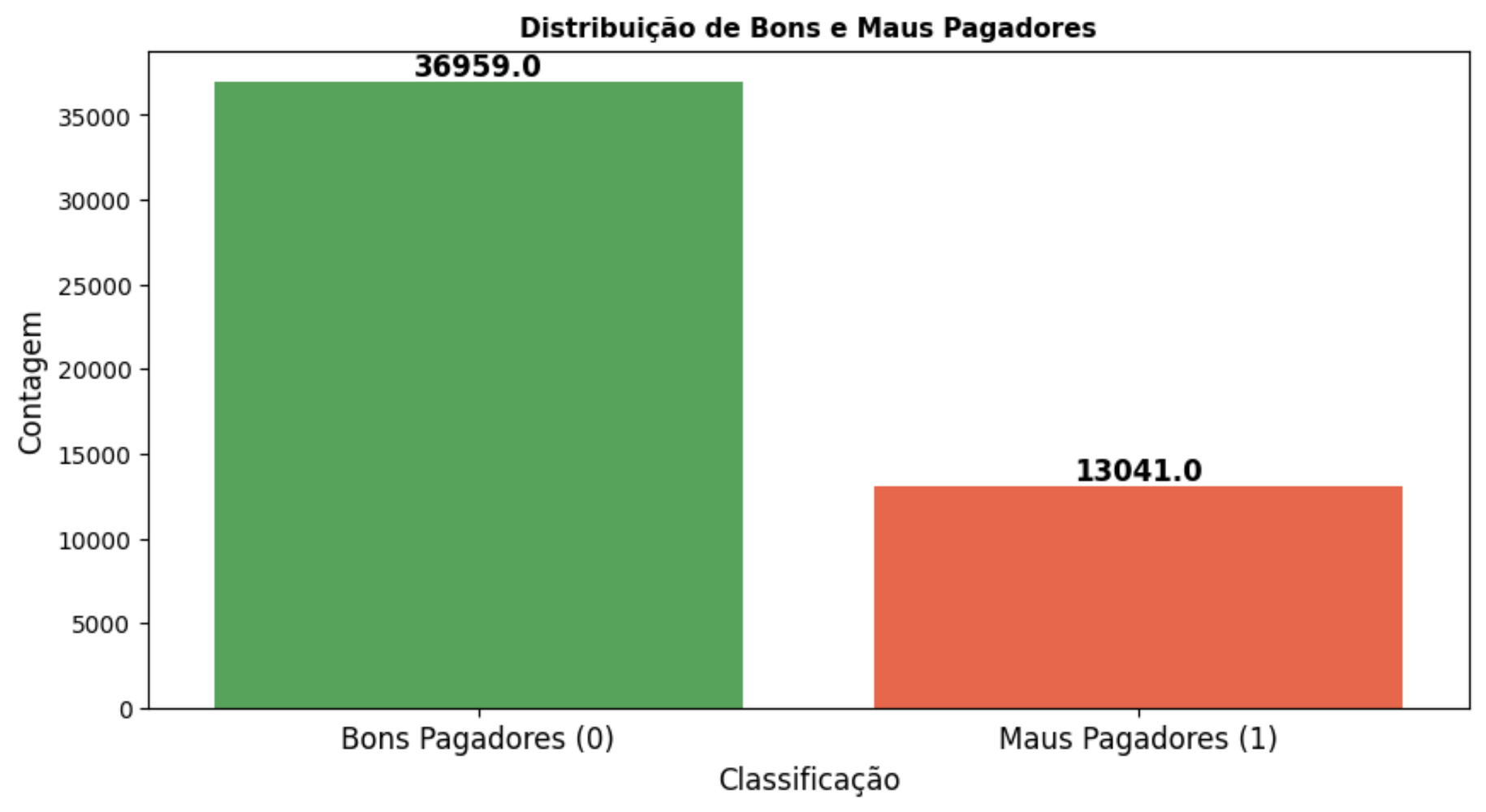
Parte 1 – Base de Dados
Construído um dicionário dos dados e realizado a análise do dataset e verificação dos tipos de variáveis
Parte 2 – Pré-processamento
Feito implantação do dicionário de dados definido.
Parte 3 – Limpeza dos dados
Identificação dos dados nulos e realizada remoção dos valores ausentes transformando para “zero”.
Parte 4 – Análise exploratória dos dados
Realizado uma análise dos dados procurando responder às principais perguntas que consideramos importante para o negócio.
Parte 5 - Análise de outlers
Foram identificados alguns outlers que foram removidos a fim de obtermos mais assertividade nas análises.
Parte 6 - Tratamento dos outliers - Z Score
Os valores que estavam muito fora também foram tratados substituindo pelos valores de mediana.
Parte 7 - Feature engineering
Identificação das colunas não categóricas, remoção dos valores nulos, utilização do Label Encoder
Parte 8 - Divisão treino e teste
Realizado o drop da variável alvo e definição de X e y para treino.
Parte 9 - Treinamento modelo
Realizado treinamento do modelo com a Random_seed fixa em 42.
Parte 10 - Classe desbalanceamento
Foi aplicado a técnica de oversampling conhecida como SMOTE com o objetivo de balancear as classes da variável alvo.
Parte 11 - Modelo machine learning: Utilizados GaussianNB, Decision Tree, Random Forest, Logistic Regression, AdsBoost, XGBoost, LightGBM, KNN, Gradient Boosting e SVC.
Parte 12 - Feature importance
Análise das várias mais utilizadas pelos Modelos.
Parte 13 – Métrica e avaliação
Feita as matrizes de confusões para cada modelo, análise comparativa e acurácias.



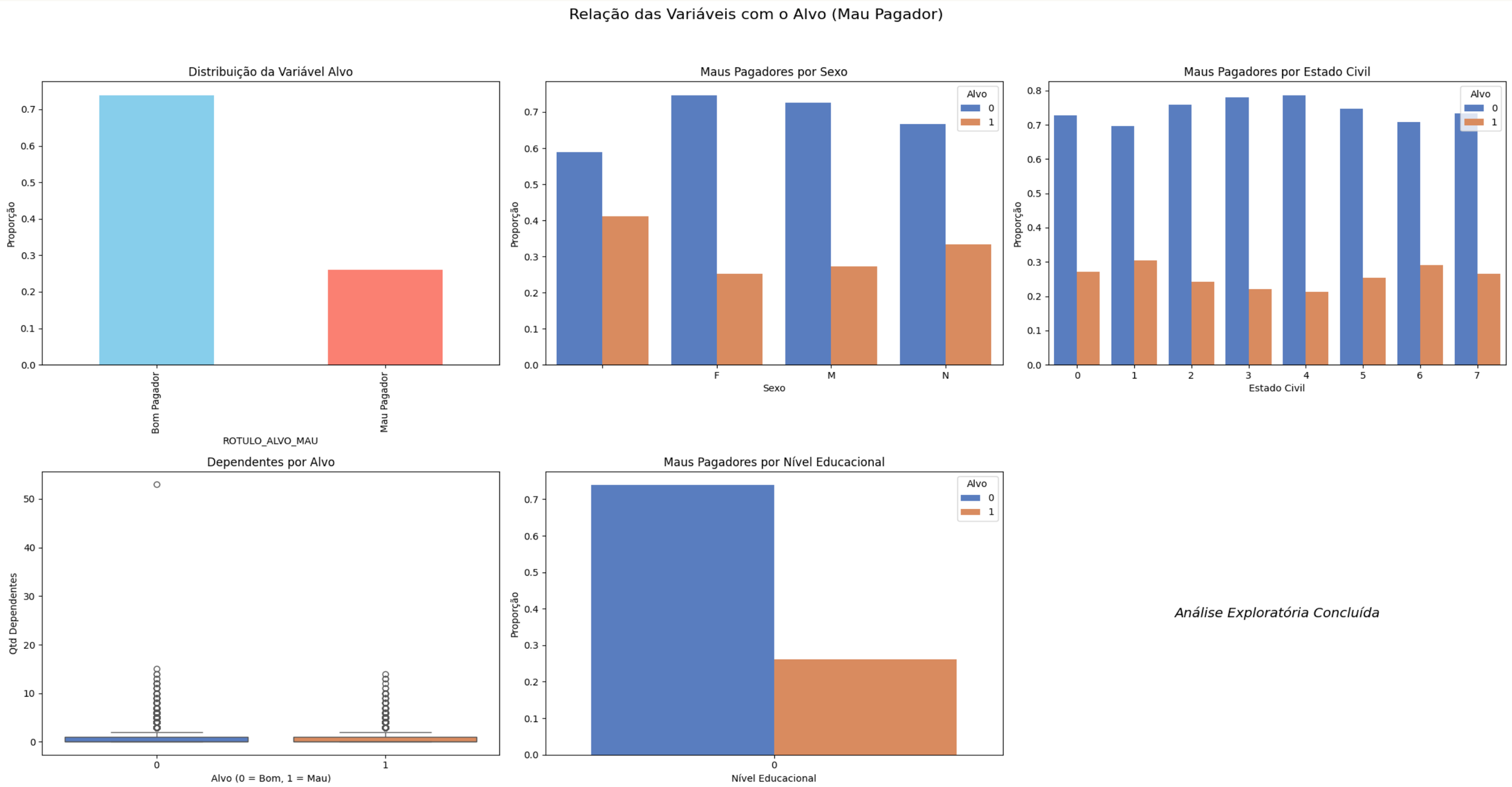
•Esse desbalanceamento de 72/28% é suficiente para que modelos de machine learning — especialmente os mais simples — prefiram prever apenas a classe majoritária (0), o que artificialmente eleva a acurácia, mas prejudica:

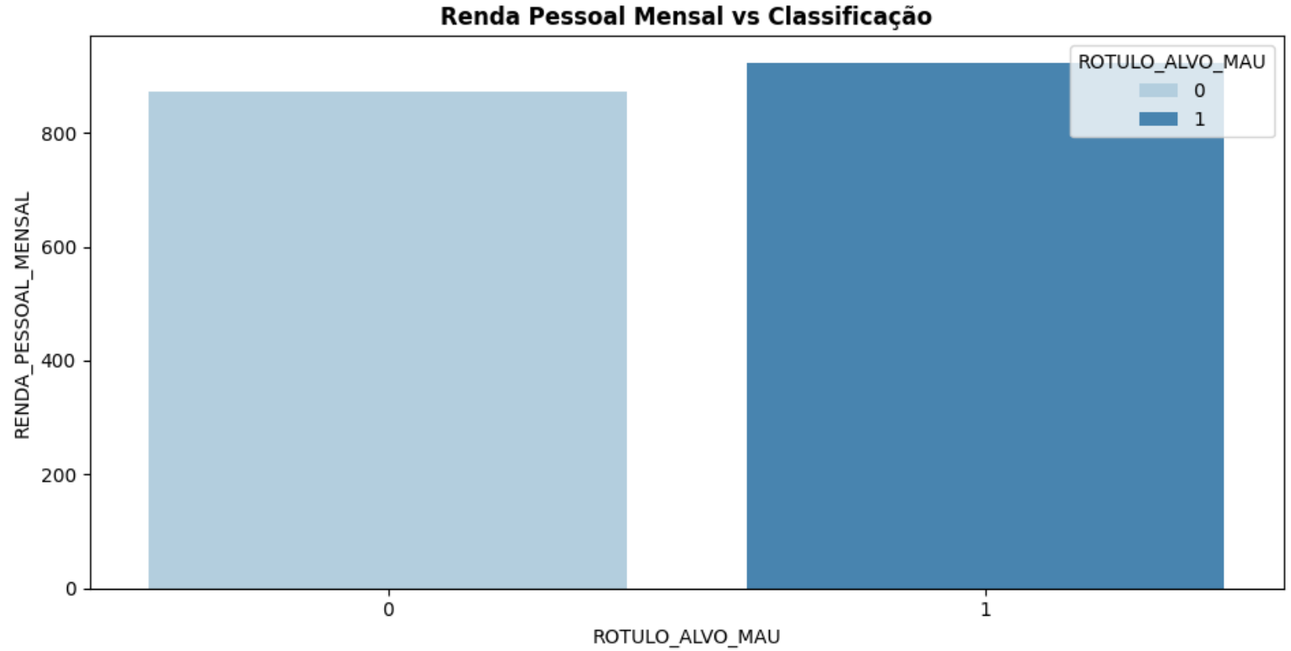
Parte 4 – Análise exploratória dos dados

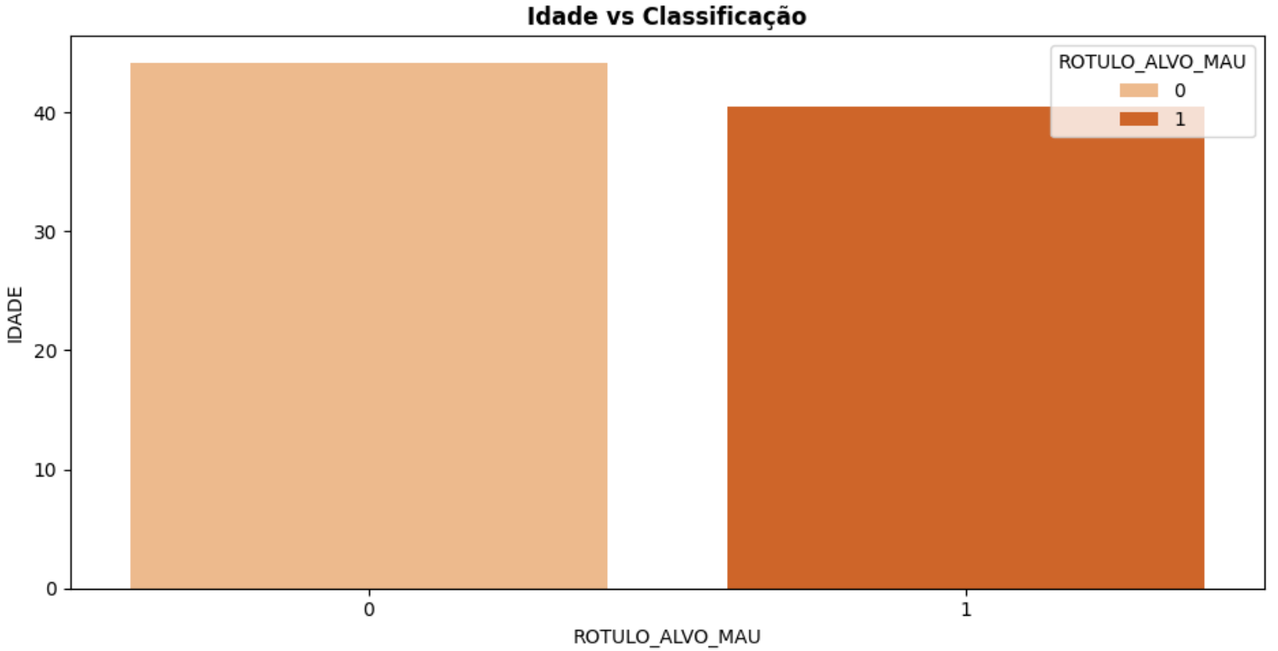
Questão 1) Explorar e analisar os dados para entender a distribuição das características e a relação com a classificação de cada cliente.



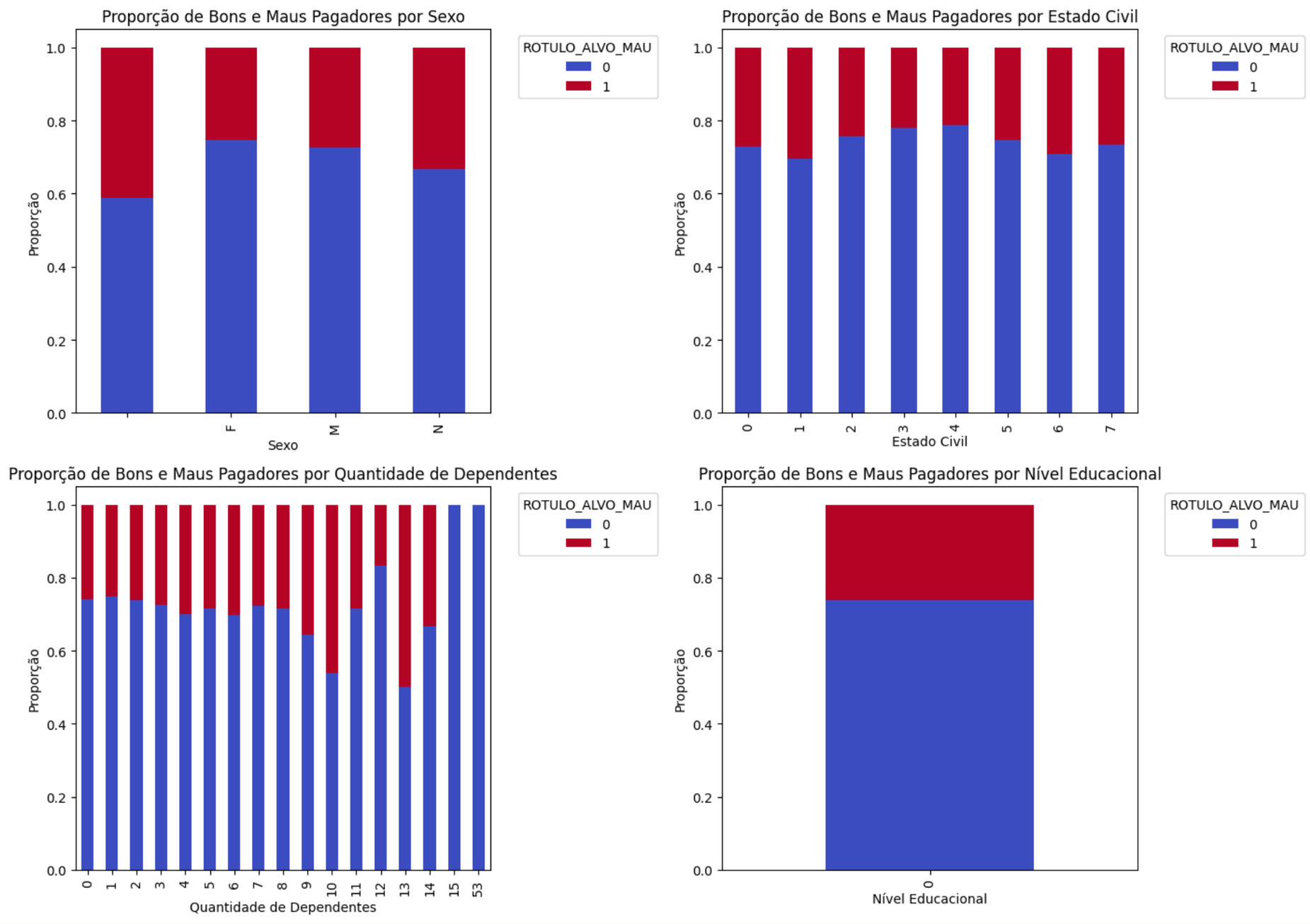
Questão 2) Como as Features **Sexo**, **Estado Civil**, **Quantidade de Dependentes** e **Nível Educacional** se Relacionam com a **Variável Meta** ?

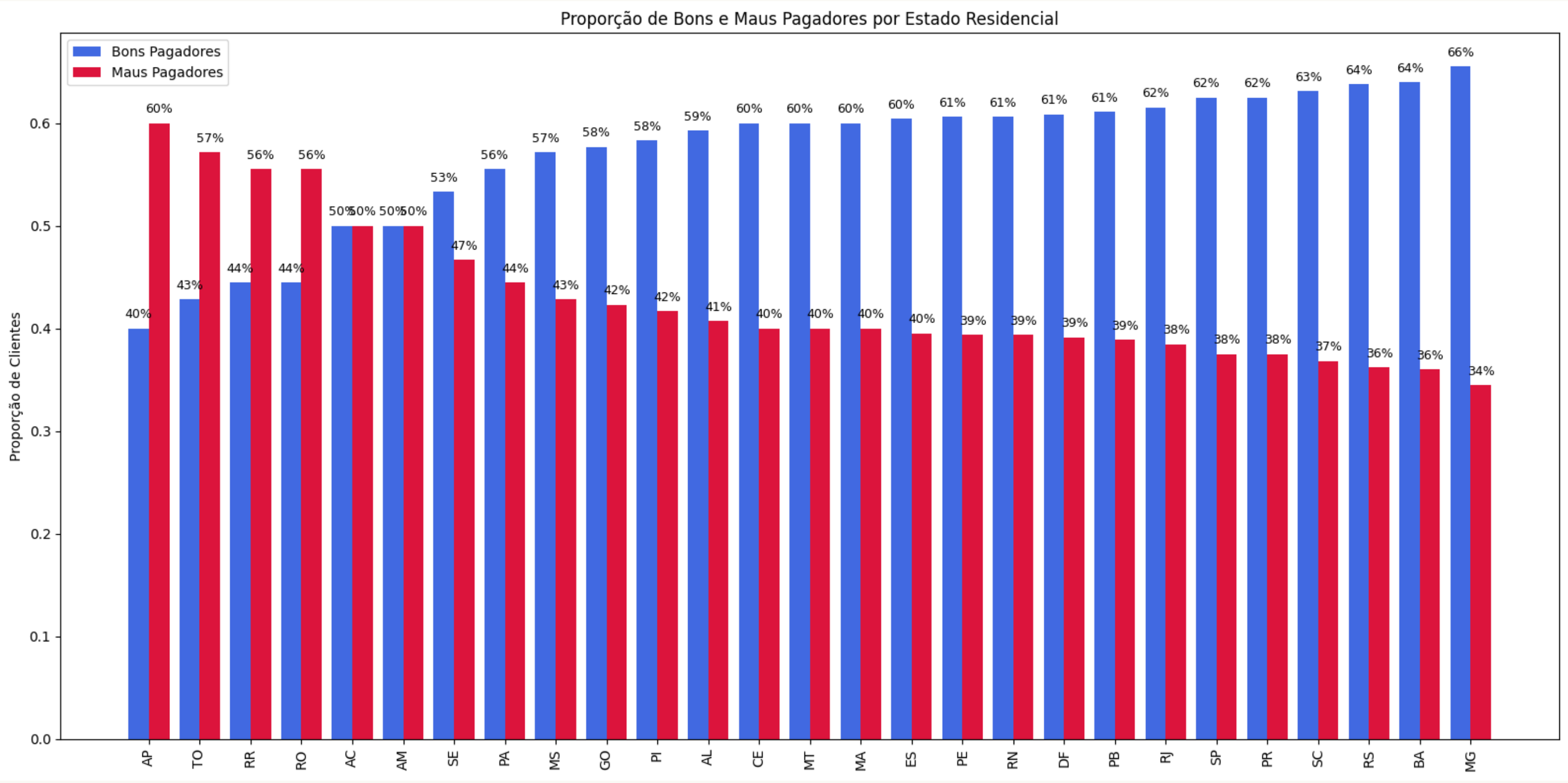


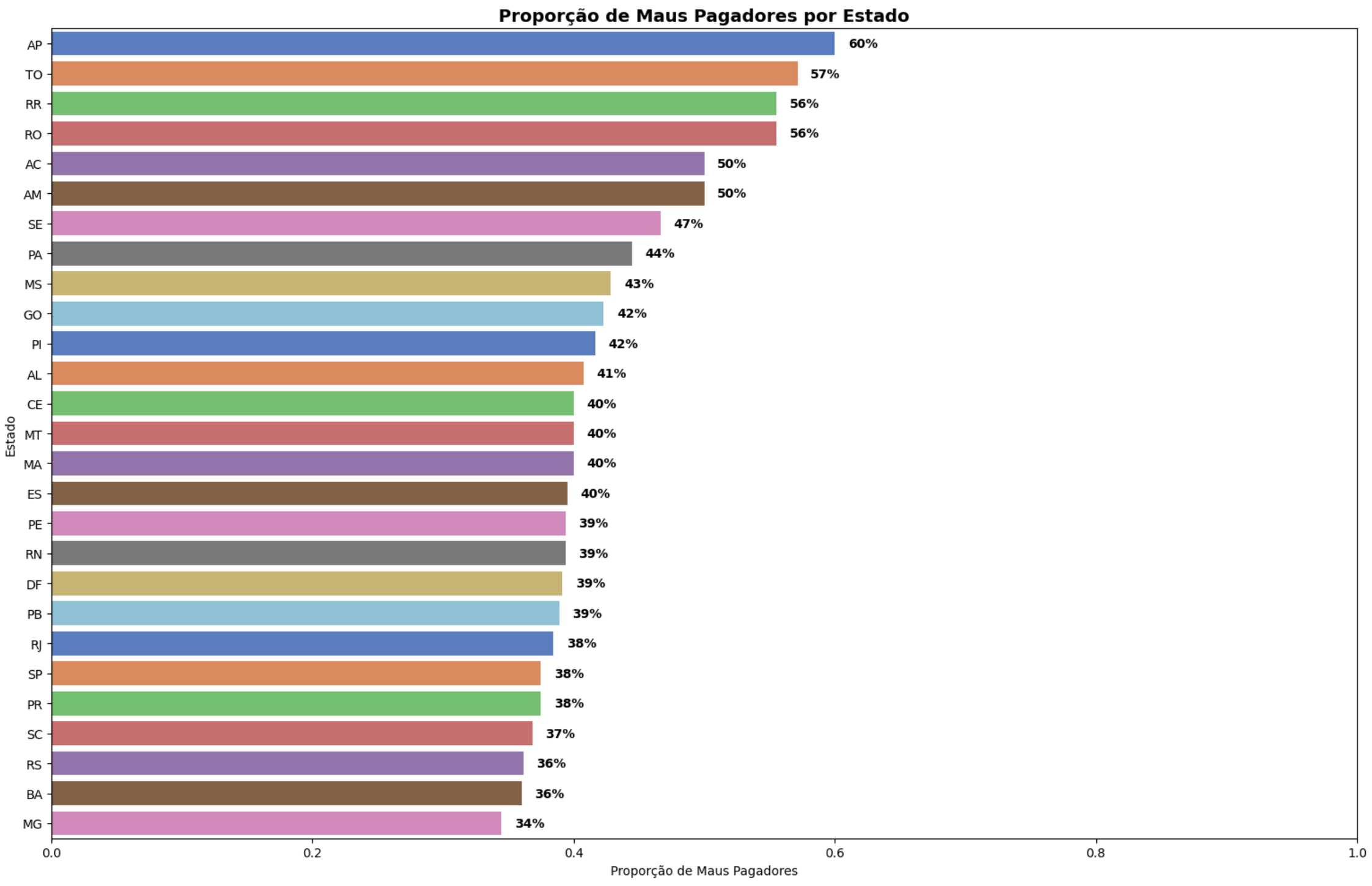

Renda Pessoal Mensal vs Classificação
ROTULO_ALVO_MAU
0
1
RENDA_PESSOAL_MENSAL
0
200
400
600
800
0
1
ROTULO_ALVO_MAU

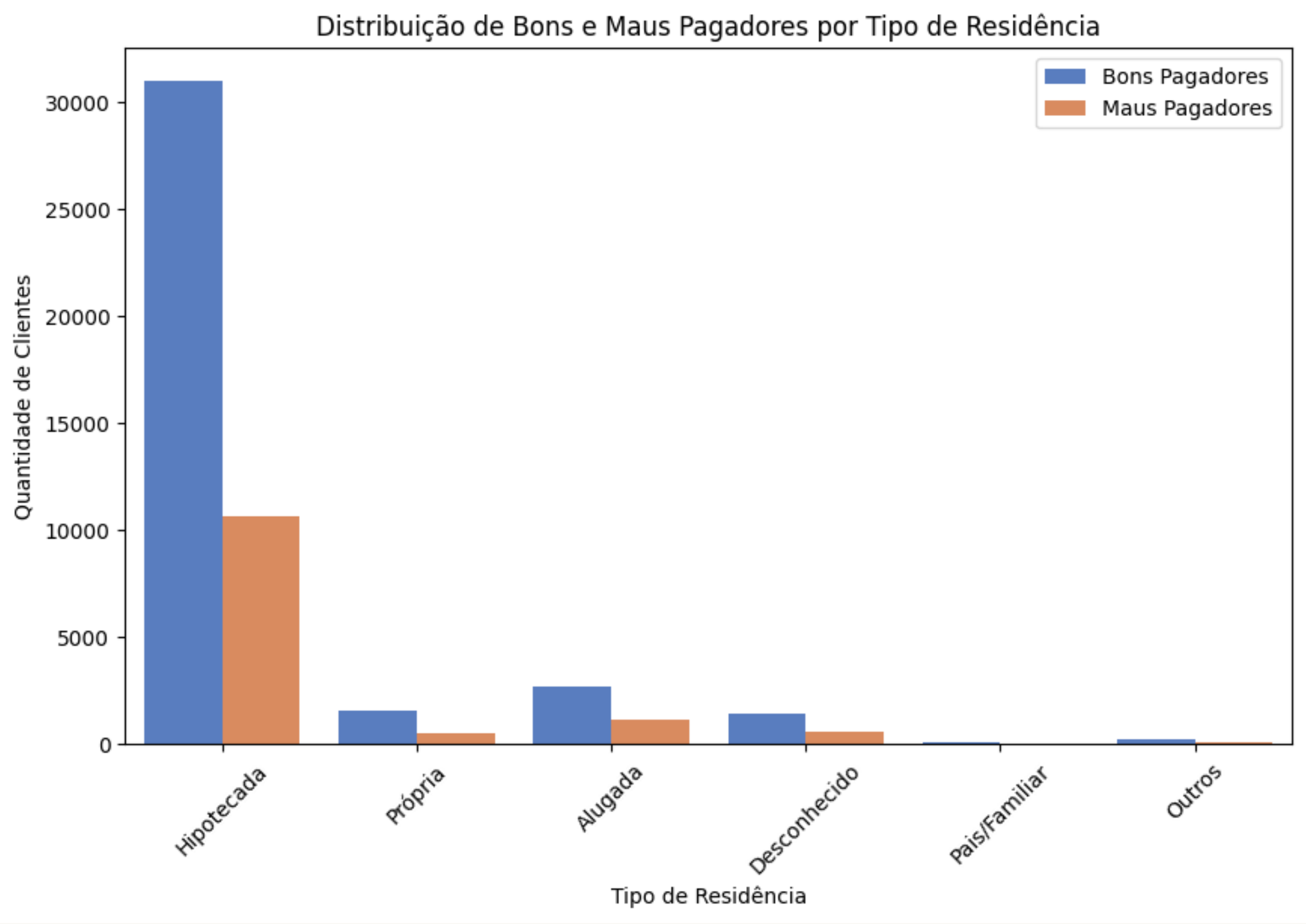

Idade vs Classificação
ROTULO_ALVO_MAU
0
1
IDADE
0
10
20
30
40
0
1
ROTULO_ALVO_MAU

Questão 3) Distribuição de Bons e Maus Pagadores por Estado (categoria) ?

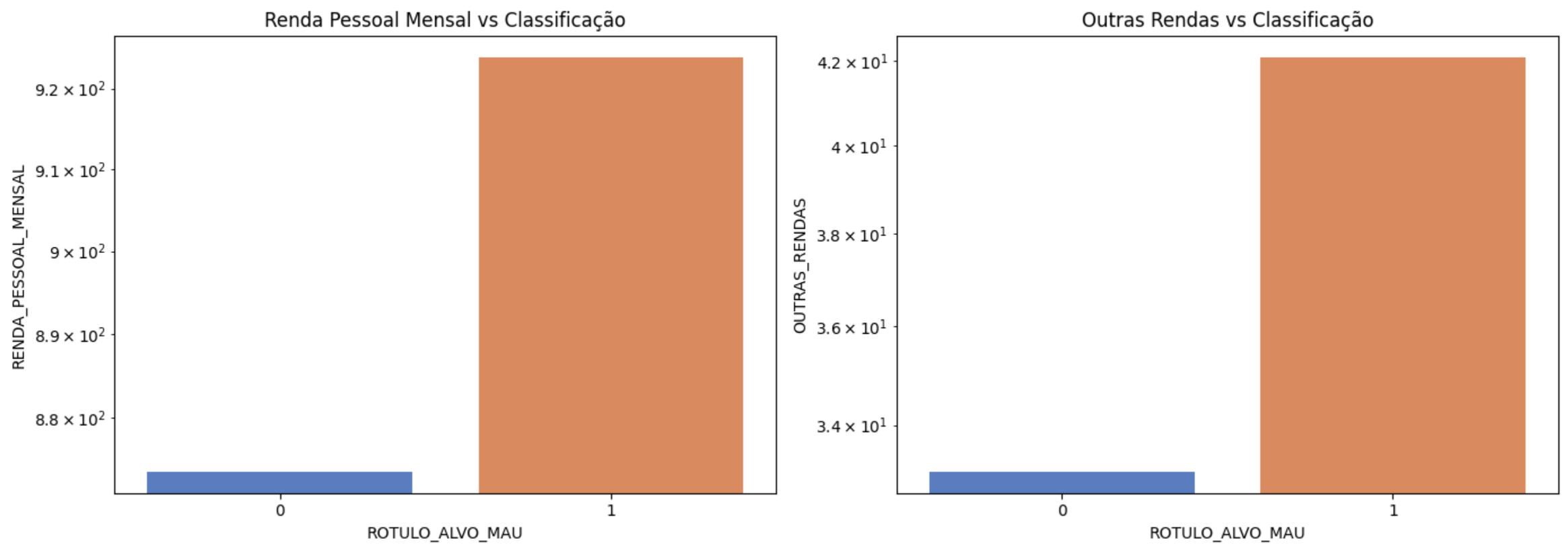



Proporção de Bons e Maus Pagadores por Estado Residencial
Bons Pagadores
Maus Pagadores
Proporção de Clientes
0.0
0.1
0.2
0.3
0.4
0.5
0.6
40%
60%
43%
57%
44%
56%
44%
56%
50%
50%
50%
50%
53%
47%
56%
44%
57%
43%
58%
42%
58%
42%
59%
41%
60%
40%
60%
40%
60%
40%
60%
40%
61%
39%
61%
39%
61%
39%
61%
39%
62%
38%
62%
38%
62%
38%
63%
37%
64%
36%
64%
36%
66%
34%
AP
TO
RR
RO
AC
AM
SE
PA
MS
GO
PI
AL
CE
MT
MA
ES
PE
RN
DF
PB
RJ
SP
PR
SC
RS
BA
MG

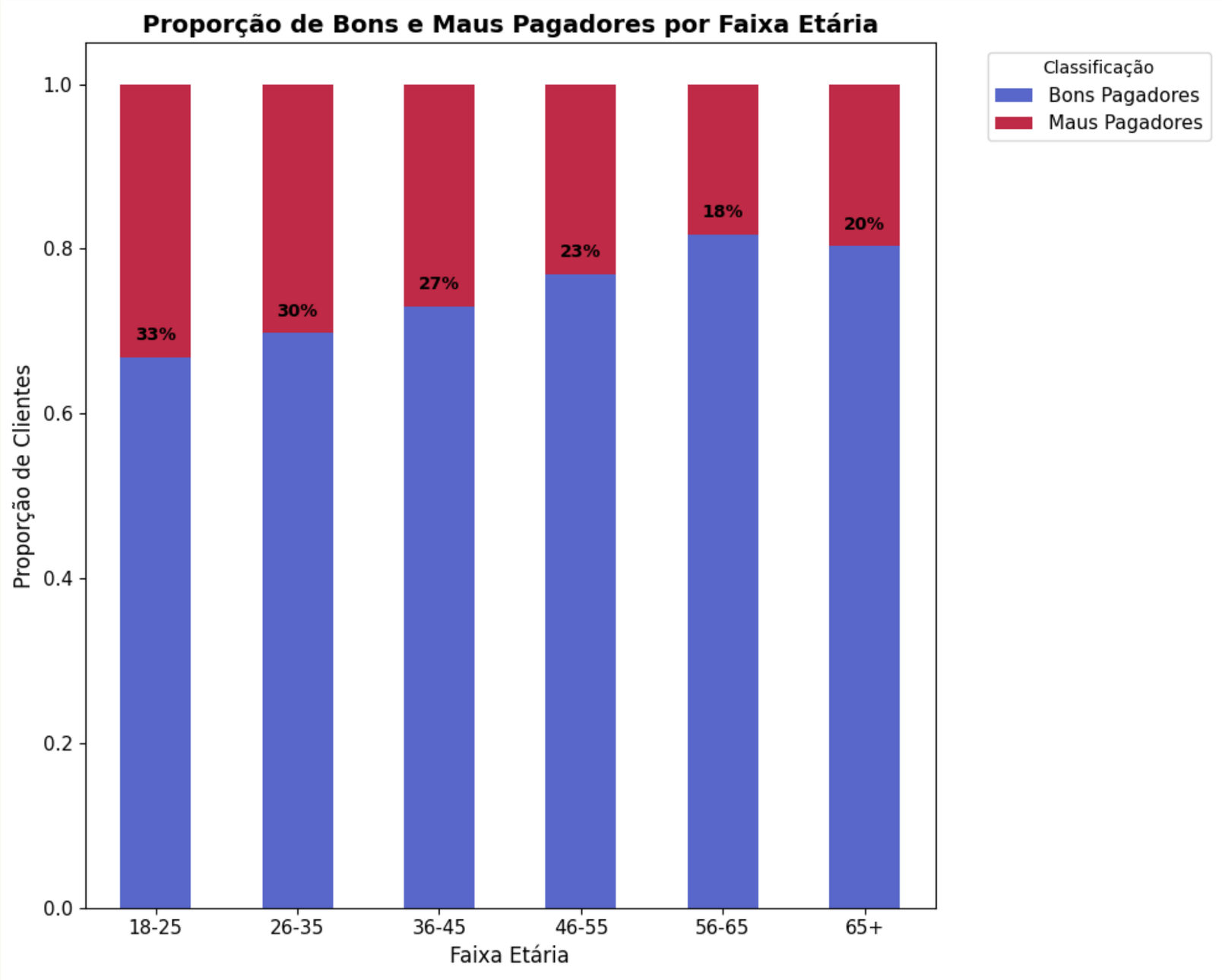

Proporção de Maus Pagadores por Estado
AP
TO
RR
RO
AC
AM
SE
PA
MS
GO
PI
AL
CE
MT
MA
ES
PE
RN
DF
PB
RJ
SP
PR
SC
RS
BA
MG
60%
57%
56%
56%
50%
50%
47%
44%
43%
42%
42%
41%
40%
40%
40%
40%
39%
39%
39%
39%
38%
38%
38%
37%
36%
36%
34%
Estado
0.0
0.2
0.4
0.6
0.8
1.0
Proporção de Maus Pagadores


Distribuição de Bons e Maus Pagadores por Tipo de Residência
Bons Pagadores
Maus Pagadores
Quantidade de Clientes
30000
25000
20000
15000
10000
5000
0
Hipotecada
Própria
Alugada
Desconhecido
Pais/Familiar
Outros
Tipo de Residência

Questão 5) Como as Features 'Renda Pessoal Mensal' e 'Outras Rendas' se Relacionam com a Variável Meta ?



Questão 6) A Idade Interfere na Classificação do Pagador?



Insights importantes:

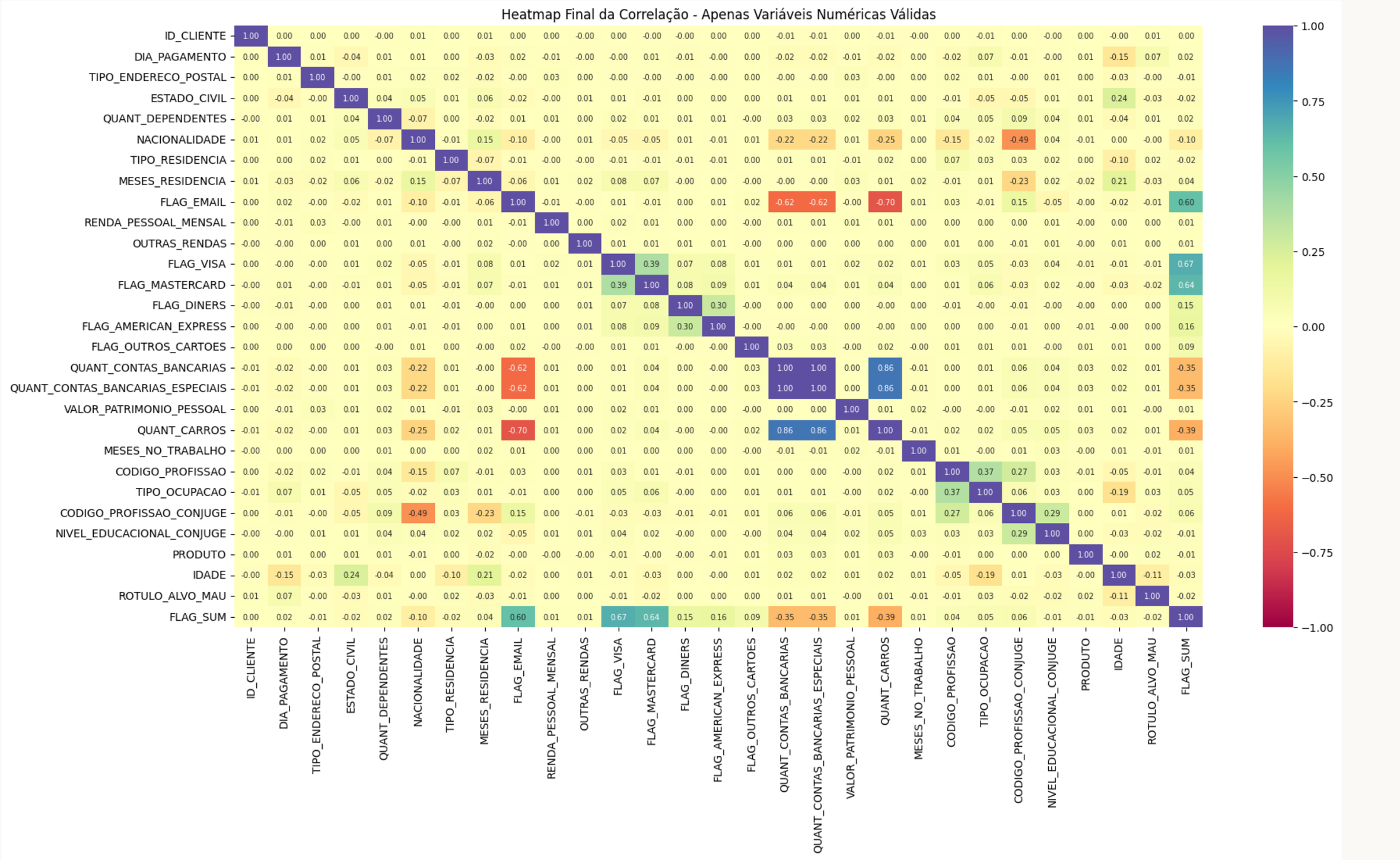
1) A idade tem um impacto significativo na classificação do pagador.

2) Jovens (18-25 anos) são os mais inadimplentes, provavelmente devido à instabilidade financeira, falta de histórico de crédito e menos experiência na gestão financeira.

3) Clientes de 46 a 65 anos possuem menor inadimplência, indicando maior maturidade financeira e comportamento de pagamento mais confiável.

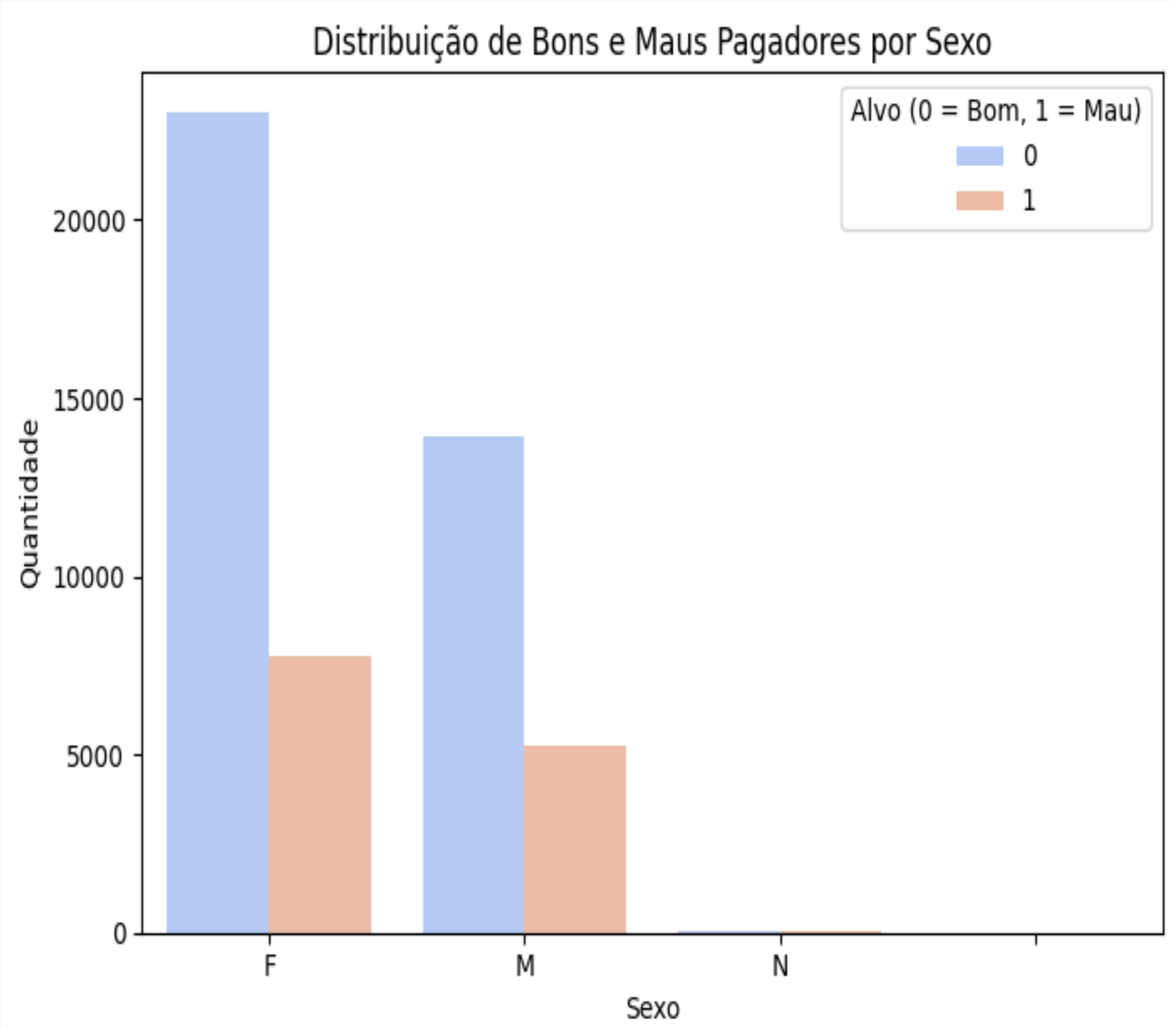
4) A inadimplência volta a subir na faixa 65+, sugerindo que a aposentadoria pode impactar a capacidade de pagamento.

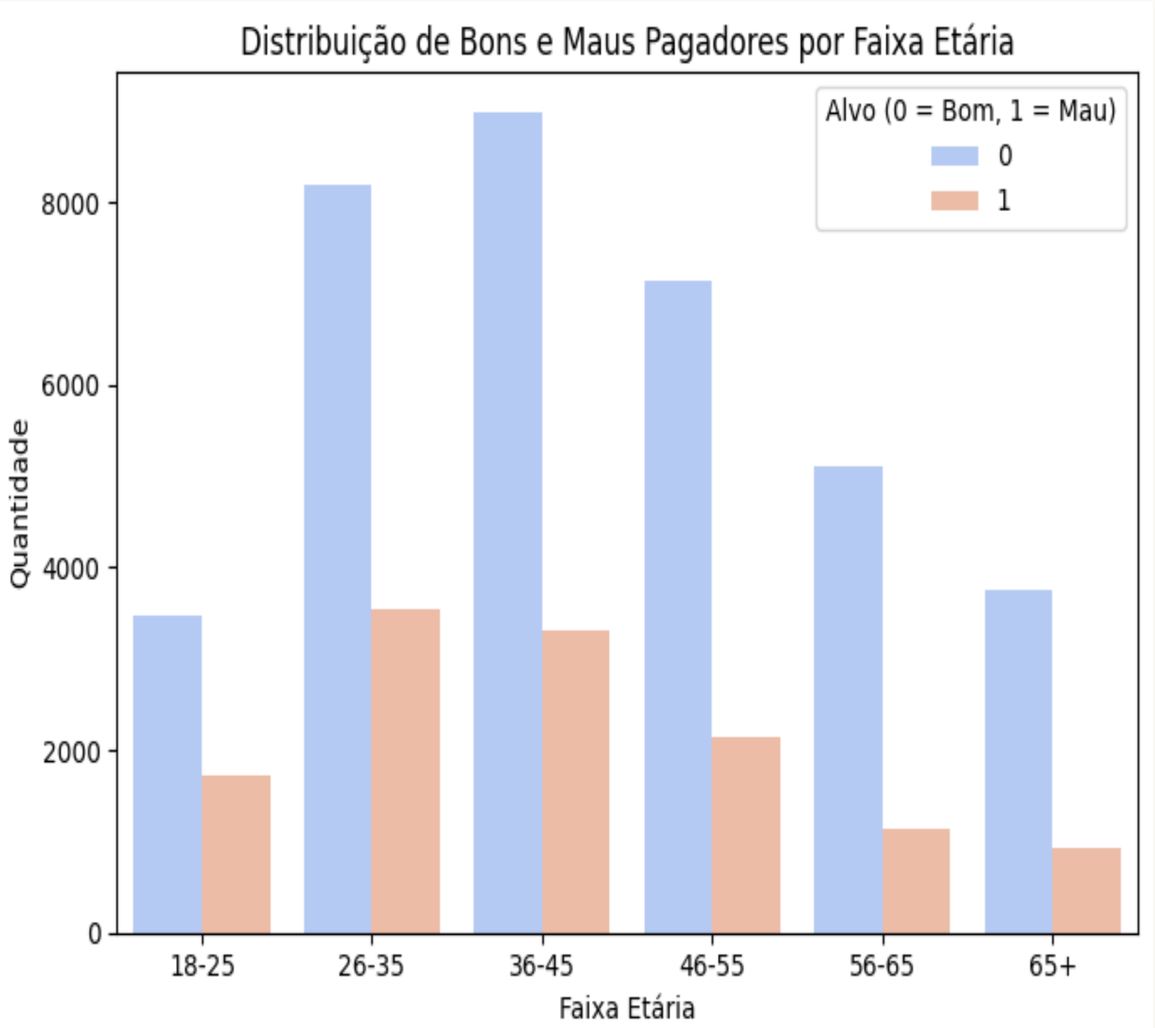
Analisar o impacto da renda e do tempo de trabalho na inadimplência por faixa etária verificar se clientes mais velhos possuem mais empréstimos ou comprometimento de renda com familiares, criar estratégias para educação financeira e crédito responsável para jovens e aposentados, ajudando a reduzir os riscos de inadimplência Esse gráfico reforça que a idade é um fator importante na análise de risco de crédito e pode ser um critério relevante na concessão de crédito e precificação de juros.


Heatmap Final da Correlação - Apenas Variáveis Numéricas Válidas
ID_CLIENTE
DIA_PAGAMENTO
TIPO_ENDERECO_POSTAL
ESTADO_CIVIL
QUANT_DEPENDENTES
NACIONALIDADE
TIPO_RESIDENCIA
MESES_RESIDENCIA
FLAG_EMAIL
RENDA_PESSOAL_MENSAL
OUTRAS_RENDAS
FLAG_VISA
FLAG_MASTERCARD
FLAG_DINERS
FLAG_AMERICAN_EXPRESS
FLAG_OUTROS_CARTOES
QUANT_CONTAS_BANCARIAS
QUANT_CONTAS_BANCARIAS_ESPECIAIS
VALOR_PATRIMONIO_PESSOAL
QUANT_CARROS
MESES_NO_TRABALHO
CODIGO_PROFISSAO
TIPO_OCUPACAO
CODIGO_PROFISSAO_CONJUGE
NIVEL_EDUCACIONAL_CONJUGE
PRODUTO
IDADE
ROTULO_ALVO_MAU
FLAG_SUM
1.00
0.75
0.50
0.25
0.00
−0.25
−0.50
−0.75
−1.00

As correlações observadas indicam que variáveis relacionadas à **estrutura familiar (como profissão e escolaridade do cônjuge)**, **posse de ativos (carros)** e **vínculo com instituições financeiras (contas e cartões)** se relacionam positivamente entre si. Isso sugere a presença de **clusters socioeconômicos**, onde determinados perfis concentram múltiplos sinais de renda e estabilidade.

Essas descobertas são valiosas para modelos preditivos, segmentações de clientes ou estratégias de crédito, pois evidenciam **comportamentos correlacionados que podem ajudar a diferenciar grupos de risco**.

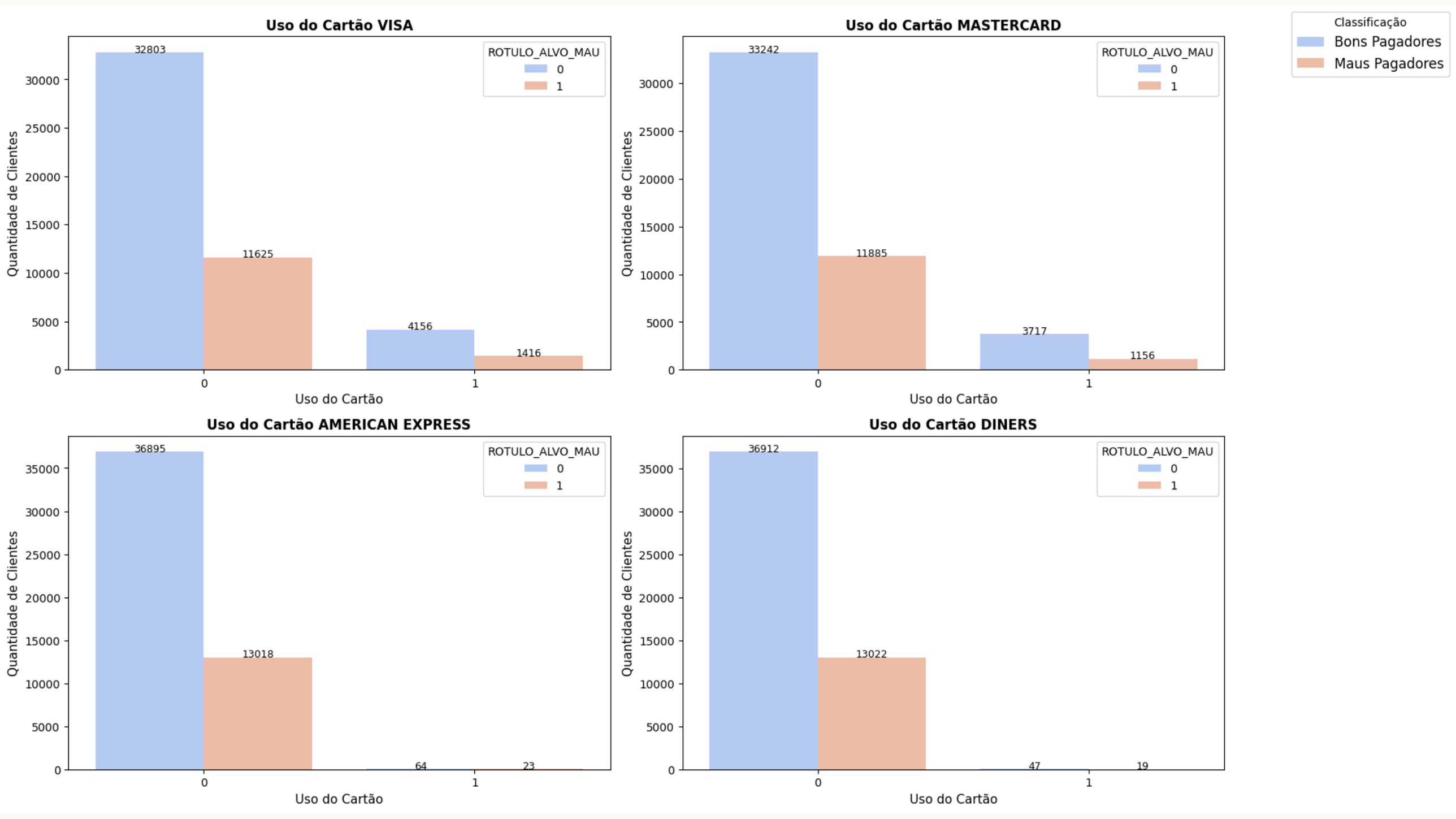

Distribuição de Bons e Maus Pagadores por Sexo
Alvo (0 = Bom, 1 = Mau)
0
1
Quantidade
0
5000
10000
15000
20000
F
M
N
Sexo


Distribuição de Bons e Maus Pagadores por Faixa Etária
Alvo (0 = Bom, 1 = Mau)
0
1
Quantidade
0
2000
4000
6000
8000
18-25
26-35
36-45
46-55
56-65
65+
Faixa Etária

Análise

Análise do Gráfico 1: Distribuição por Sexo o Sexo Masculino (M) há mais bons pagadores (cor azul) do que maus pagadores (cor laranja). O Sexo Feminino Assim como no sexo masculino, há uma quantidade maior de bons pagadores em relação aos maus pagadores, porém, a diferença parece ligeiramente maior do que no grupo masculino não informado ou outro quase não há registros, indicando que essa categoria pode não ser representativa. Tanto homens quanto mulheres têm uma maioria de bons pagadores, mas as mulheres parecem ter um número absoluto maior de bons pagadores do que os homens. A Distribuição por Faixa 18-25 anos embora a maioria seja de bons pagadores, há uma quantidade razoável de maus pagadores. As idades 26-35 anos essa é a faixa etária com o maior número de bons e maus pagadores, o que sugere que essa idade concentra o maior número de indivíduos no conjunto de dados.

As outras idades 36-45 anos ainda há uma grande quantidade de bons pagadores, mas também um número significativo de maus pagadores. Outra idade 46-55 anos o número de bons pagadores começa a diminuir, e o número de maus pagadores também é menor. Outra idade 56-65 anos e 65 a quantidade de pessoas diminui drasticamente, o que indica que essa faixa etária tem menor representação nos dados. A maior concentração de bons e maus pagadores está na faixa de 26-35 anos. Conforme a idade aumenta, o número de indivíduos diminui, o que pode indicar que menos pessoas acessam crédito em faixas etárias mais elevadas. A proporção de bons pagadores é maior do que a de maus pagadores em todas as categorias analisadas a maior incidência de maus pagadores ocorre nas faixas etárias mais jovens (18-35 anos). Mulheres possuem um número absoluto maior de bons pagadores em comparação aos homens.


Uso do Cartão VISA
Uso do Cartão MASTERCARD
Uso do Cartão AMERICAN EXPRESS
Uso do Cartão DINERS
Classificação
Bons Pagadores
Maus Pagadores
ROTULO_ALVO_MAU
0
1
Quantidade de Clientes
Uso do Cartão
32803
11625
4156
1416
33242
11885
3717
1156
36895
13018
64
23
36912
13022
47
19

Análise:

•A maioria dos clientes não utiliza cartões de crédito (VISA, MASTERCARD, AMERICAN EXPRESS e DINERS).
•Entre os não usuários, predominam os bons pagadores (azul) sobre os maus pagadores (laranja).
•Entre os usuários, a proporção de bons pagadores é maior, mas a presença de maus pagadores não é desprezível.

Análise por cartão de crédito

VISA: 3.650 bons e 1.409 maus pagadores não utilizam; 677 bons e 251 maus utilizam.
MASTERCARD: 3.713 bons e 1.470 maus não utilizam; 614 bons e 190 maus utilizam.
AMERICAN EXPRESS: 4.319 bons e 1.658 maus não utilizam; poucos usuários, com possíveis inconsistências nos dados (valores 8 e 2).
DINERS: 4.325 bons e 1.659 maus não utilizam; poucos usuários, com possíveis anomalias nos dados (valores 1 e 2).

Conclusão:

Bons pagadores predominam tanto entre os usuários quanto entre os não usuários. No entanto, a proporção de maus pagadores entre os usuários merece atenção.
Dados dos cartões AMERICAN EXPRESS e DINERS requerem revisão devido a possíveis erros de codificação.

Parte 6 - Tratamento dos outliers - Z Score

```python
from scipy.stats import zscore

# Lista de colunas com outliers identificados
num_cols = ["DIA_PAGAMENTO", "TIPO_ENDERECO_POSTAL", "ESTADO_CIVIL", "QUANT_DEPENDENTES",
                    "NACIONALIDADE", "TIPO_RESIDENCIA", "MESES_RESIDENCIA", "RENDA_PESSOAL_MENSAL",
                    "OUTRAS_RENDAS", "FLAG_VISA", "FLAG_MASTERCARD", "FLAG_DINERS",
                    "FLAG_AMERICAN_EXPRESS", "FLAG_OUTROS_CARTOES", "QUANT_CONTAS_BANCARIAS",
                    "QUANT_CONTAS_BANCARIAS_ESPECIAIS", "VALOR_PATRIMONIO_PESSOAL",
                    "QUANT_CARROS", "MESES_NO_TRABALHO", "CODIGO_PROFISSAO", "TIPO_OCUPACAO",
                    "NIVEL_EDUCACIONAL_CONJUGE", "PRODUTO", "IDADE"]

# Garantir que todas as colunas estejam numéricas
for col in num_cols:
    df[col] = pd.to_numeric(df[col], errors='coerce')  # força float, coloca NaN em inválidos

# Tratar outliers com Z-Score
for col in num_cols:
    col_zscore = zscore(df[col].dropna())  # evita erro por NaN temporário
    outliers = np.abs(zscore(df[col])) > 3
    mediana = df[col].median()
    df.loc[outliers, col] = mediana

# Lista das colunas numéricas tratadas
num_cols = ["DIA_PAGAMENTO", "TIPO_ENDERECO_POSTAL", "ESTADO_CIVIL", "QUANT_DEPENDENTES",
                    "NACIONALIDADE", "TIPO_RESIDENCIA", "MESES_RESIDENCIA", "RENDA_PESSOAL_MENSAL",
                    "OUTRAS_RENDAS", "FLAG_VISA", "FLAG_MASTERCARD", "FLAG_DINERS",
                    "FLAG_AMERICAN_EXPRESS", "FLAG_OUTROS_CARTOES", "QUANT_CONTAS_BANCARIAS",
                    "QUANT_CONTAS_BANCARIAS_ESPECIAIS", "VALOR_PATRIMONIO_PESSOAL",
                    "QUANT_CARROS", "MESES_NO_TRABALHO", "CODIGO_PROFISSAO", "TIPO_OCUPACAO",
                    "NIVEL_EDUCACIONAL_CONJUGE", "PRODUTO", "IDADE"]

# Criar boxplots para todas as colunas após o tratamento dos outliers
n_cols = 3
n_rows = (len(num_cols) + n_cols - 1) // n_cols
plt.figure(figsize=(n_cols * 5, n_rows * 4))
```

Análise Nesta etapa, foi realizada uma análise de outliers nas variáveis numéricas utilizando a técnica de Z-Score (padronização estatística). Em vez de excluir os dados extremos, optamos por um tratamento conservador, substituindo os valores considerados outliers (valores com $|z| > 3$) pela mediana da respectiva variável. Essa abordagem preserva o tamanho do conjunto de dados, reduz a influência de valores extremos nos modelos de machine learning e mantém a integridade estatística da base. Esse processo foi essencial para garantir uma distribuição mais equilibrada e evitar distorções na etapa de modelagem.

Parte 7 - Feature engineering

```python
# Remover todas as linhas que possuem pelo menos um valor nulo
df = df.dropna()

# Contar valores ausentes em cada coluna
print("\nValores ausentes por coluna:")
print(df.isnull().sum())
```

```
TIPO_FUNCIONARIO                    0
DIA_PAGAMENTO                       0
TIPO_ENVIO_APLICACAO                0
QUANT_CARTOES_ADICIONAIS            0
TIPO_ENDERECO_POSTAL                0
SEXO                                0
ESTADO_CIVIL                        0
QUANT_DEPENDENTES                   0
NIVEL_EDUCACIONAL                   0
ESTADO_NASCIMENTO                   0
CIDADE_NASCIMENTO                   0
NACIONALIDADE                       0
ESTADO_RESIDENCIAL                  0
CIDADE_RESIDENCIAL                  0
BAIRRO_RESIDENCIAL                  0
FLAG_TELEFONE_RESIDENCIAL           0
CODIGO_AREA_TELEFONE_RESIDENCIAL    0
TIPO_RESIDENCIA                     0
MESES_RESIDENCIA                    0
FLAG_TELEFONE_MOVEL                 0
FLAG_EMAIL                          0
RENDA_PESSOAL_MENSAL                0
OUTRAS_RENDAS                       0
FLAG_VISA                           0
FLAG_MASTERCARD                     0
FLAG_DINERS                         0
FLAG_AMERICAN_EXPRESS               0
```

Após a identificação e somas de todas as colunas com valor nulo.
Foi realizado o drop de todas as linhas que possuem pelo menos um valor nulo.
Através da função df.dropna().

Parte 7 - Feature engineering

Foi realizada a técnica de **Label Encoder,** conforme abaixo:

```
[ ] # Importando biblioteca
    from sklearn.preprocessing import LabelEncoder

    # Lista das variáveis categóricas
    categorical_cols = ['TIPO_FUNCIONARIO', 'TIPO_ENVIO_APLICACAO', 'SEXO', 'ESTADO_NASCIMENTO', 'CIDADE_NASCIMENTO', 'ESTADO_RESIDENCIAL', 'CIDADE_RESIDENCIAL', 'BAIRRO_RESIDENCIAL', 'CODIGO_AREA_TELEFONE_RESIDENCIAL', 'EMPRESA', 'ESTADO_PROFISSIONAL', 'CIDADE_PROFISSIONAL', 'BAIRRO_PROF

    # Dicionário para armazenar os encoders
    label_encoders = {}

    # Aplicar LabelEncoder para cada coluna na base SEM outliers
    for col in categorical_cols:
        le = LabelEncoder()
        df[col] = df[col].astype(str)  # garantir strings
        df[col] = le.fit_transform(df[col])
        label_encoders[col] = le  # guardar para uso posterior, ex: no test set

    # Visualizando
    le
```

```
LabelEncoder
LabelEncoder()
```

Nesta etapa, aplicamos técnicas de feature engineering, utilizando o Label Encoder para **transformar variáveis categóricas em variáveis numéricas**. Esse processo é essencial para tornar os dados compatíveis com algoritmos de machine learning que não operam diretamente com dados não numéricos.

Ao substituir categorias por valores inteiros, mantemos a estrutura dos dados enquanto facilitamos a análise preditiva e o treinamento dos modelos supervisionados.

Parte 8 – Divisão treino e teste

Nesta etapa, realizamos a divisão entre variáveis independentes (features) e a variável alvo **(ROTULO_ALVO_MAU)**, que representa a classe a ser prevista pelo modelo. Utilizamos o método drop() para remover a coluna alvo do conjunto de dados original e assim obter as variáveis preditoras (X).
A coluna **ROTULO_ALVO_MAU** foi então atribuída como y, representando os rótulos que servirão para treinar e avaliar os algoritmos de classificação supervisionada. Essa separação é fundamental para garantir a integridade do processo de modelagem e evitar vazamento de dados entre entrada e saída

```
[ ]  # Divisão treino e teste
     X = df.drop('ROTULO_ALVO_MAU', axis=1)
     y = df['ROTULO_ALVO_MAU']
```

```
[ ]  # Visualizando dados x
     X.shape
```

```
[ ]  # Visualizando dados y
     y.shape
```

Parte 9 – Divisão treino e teste

```
# Dividindo os dados em conjuntos de treinamento e teste
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)

# Visualizando linhas e colunas do X_train
print("Visualizando dados do trem X:", X_train.shape)

# Visualizando linhas e colunas do y_train
print("Visualizando dados do trem y:", y_train.shape)
```

Após a separação entre variáveis independentes (X) e a variável alvo (y), verificamos que o conjunto de dados de entrada possui 4.789 amostras e 53 features. A variável y, por sua vez, contém 4.789 rótulos, correspondendo ao mesmo número de amostras em X. Isso confirma que os dados estão devidamente alinhados para o treinamento de modelos supervisionados, onde cada instância de entrada possui um rótulo associado.

Parte 10 - Classe desbalanceamento

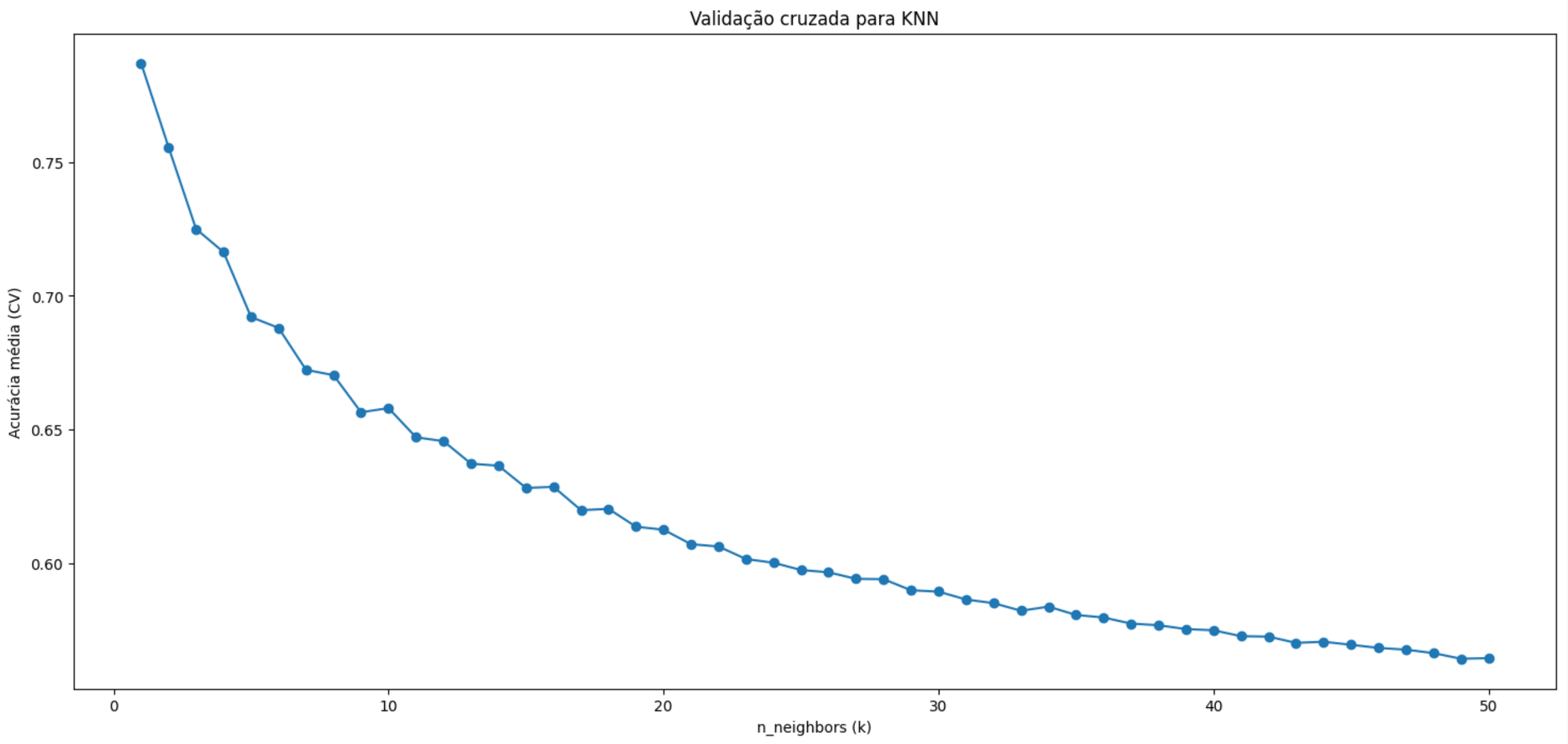
Aplicação de SMOTE (Synthetic Minority Oversampling Technique)
Foi utilizada a técnica de oversampling conhecida como SMOTE com o objetivo de balancear as classes da variável alvo. Esse método gera novas observações sintéticas da classe minoritária a partir da combinação de exemplos existentes, preservando a distribuição dos dados originais.
A aplicação foi realizada após a divisão dos dados em treino e teste, sendo aplicada apenas no conjunto de treino para evitar vazamento de informação. O balanceamento das classes contribui para reduzir o viés dos modelos em favor da classe majoritária, permitindo que algoritmos de classificação aprendam padrões mais representativos da classe com menor frequência.
Esse processo é especialmente importante quando a base apresenta desbalanceamento significativo entre as categorias da variável de saída, o que pode afetar diretamente métricas como recall e F1-score.

Como resultado, o conjunto X_train passou de 4789 para 6918 amostras, mantendo as 53 variáveis preditoras.

Dessa forma, foi realizada a equiparação do modelo.

Parte 11 – Modelo Machine Learning

Utilização do best_k para encontrar o melhor K para o modelo KNN.




```
Melhor k = 1, acurácia média (5-fold CV) = 0.7870 CPU
times: user 24min 48s, sys: 13.1 s, total: 25min 1s Wall
time: 25min 1s
```

Análise:
Nesta etapa, realizamos a varredura sistemática do hiperparâmetro k para o algoritmo KNN (K-Nearest Neighbors), com o objetivo de encontrar o valor ideal que maximize o desempenho do modelo. Utilizamos validação cruzada para testar diversos valores de k — número de vizinhos considerados — e avaliar sua influência na acurácia média.

Essa abordagem é essencial, pois a escolha inadequada de k pode levar a problemas de overfitting (quando k é muito pequeno) ou underfitting (quando k é muito grande). O valor ótimo de k é aquele que oferece o melhor equilíbrio entre viés e variância, garantindo maior capacidade de generalização do modelo

Resumo por Modelo:

Modelo	Treino	Teste	Observações Principais
1. GaussianNB	0.6220	0.4775	Desempenho fraco; provável suposição de independência incorreta.
2. Decision Tree	0.7031	0.5993	Leve overfitting; melhora possível com poda ou tuning.
3. Random Forest	1.0000	0.6945	Overfitting claro no treino; bom desempenho no teste.
4. Logistic Regression	0.5792	0.5751	Estável, mas fraco; teve *warning* de convergência — sugere normalização e mais `max_iter`.
5. AdaBoost	0.7376	0.6444	Bom equilíbrio; performance consistente.
6. XGBoost	0.9900	0.6953	Excelente no treino e muito bom no teste; atenção ao possível overfitting.
7. LightGBM	0.8575	0.6962	Melhor desempenho geral no teste; ótima escolha.
8. KNN	1.0000	0.5518	Overfitting severo; sensível a ruído e escala dos dados.
9. Gradient Boosting	0.8060	0.6828	Sólido; bom compromisso entre viés e variância.
10. SVC	0.5267	0.4866	Fraco desempenho geral; pode melhorar com `scaling` e tuning.

Desempenho dos Modelos no Conjunto de Teste
Ordem de desempenho com base na taxa de acerto:
•LightGBM: 0.6962
•XGBoost: 0.6953
•Random Forest: 0.6945
•Gradient Boosting: 0.6828
Esses métodos apresentaram os maiores resultados em termos de acerto, mantendo bom aproveitamento mesmo com variáveis diversas.

Observações sobre Ajuste Excessivo
Alguns métodos alcançaram desempenho total na fase de aprendizado, mas perderam rendimento nos dados de teste:
•Random Forest e KNN tiveram desempenho máximo durante o aprendizado, mas queda no uso com dados novos.
•XGBoost também ficou próximo do limite durante o treinamento, o que exige verificação mais cuidadosa da estabilidade com outros conjuntos.

Modelos com Resultado Inferior
•GaussianNB, SVC e LogisticRegression ficaram com taxas de acerto abaixo de 0.58.
•O resultado pode estar ligado à forma como tratam relações entre variáveis, sem considerar interações mais complexas.

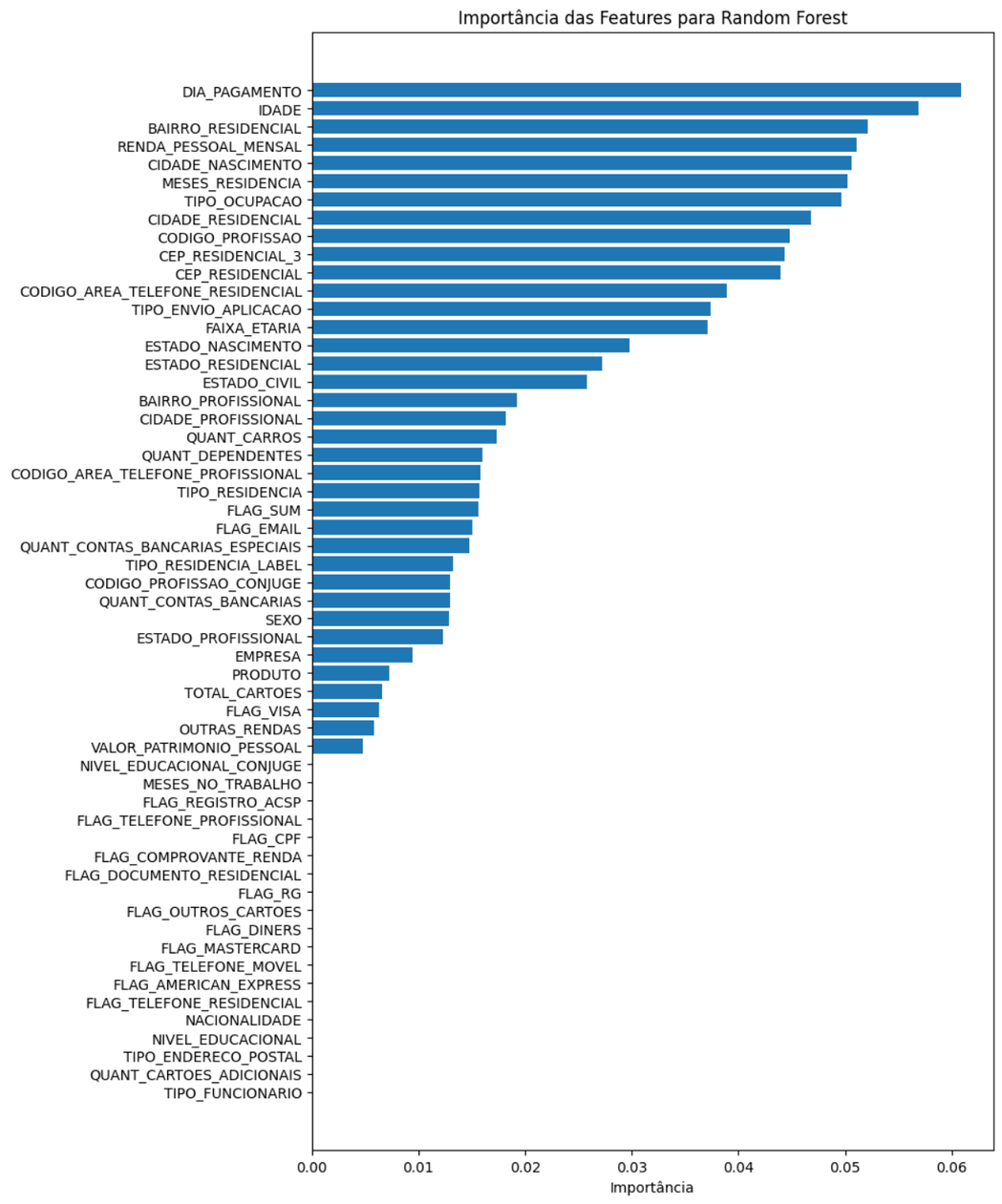
Aviso de Convergência
O aviso apresentado pelo modelo de regressão indica que os cálculos não chegaram a uma solução final. Para resolver isso, é possível:
•Aumentar o número de iterações permitidas.
•Padronizar as variáveis de entrada.
•Verificar se há dependência excessiva entre os campos usados.

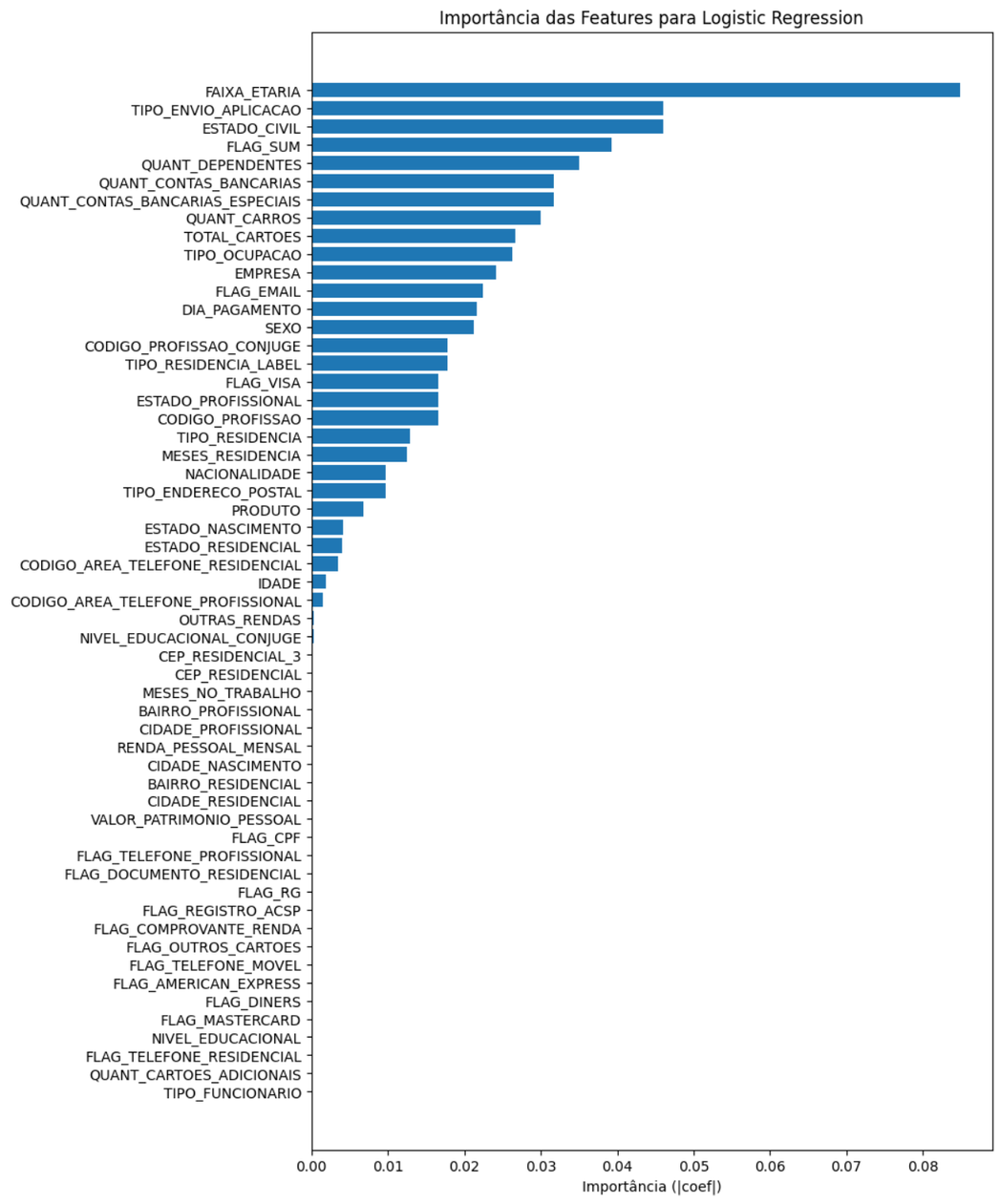
Conclusão

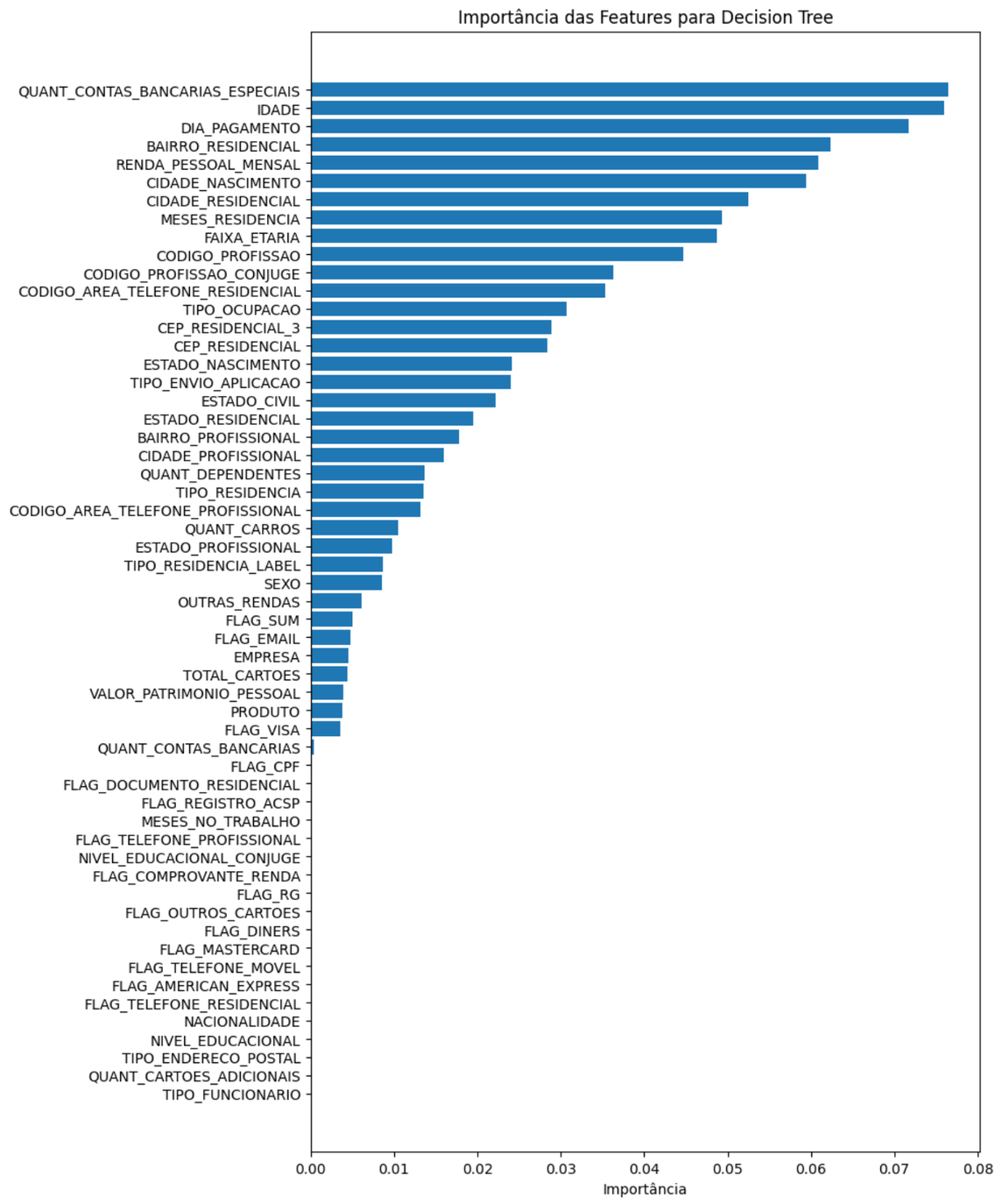
Modelos como LightGBM e XGBoost obtiveram os melhores resultados. Outros métodos, como Gradient Boosting e Random Forest, também foram consistentes. Métodos mais simples apresentaram rendimento inferior, mas são úteis como ponto de partida para comparação. Algumas abordagens mostraram excesso de ajuste aos dados iniciais e podem precisar de ajustes nos parâmetros.

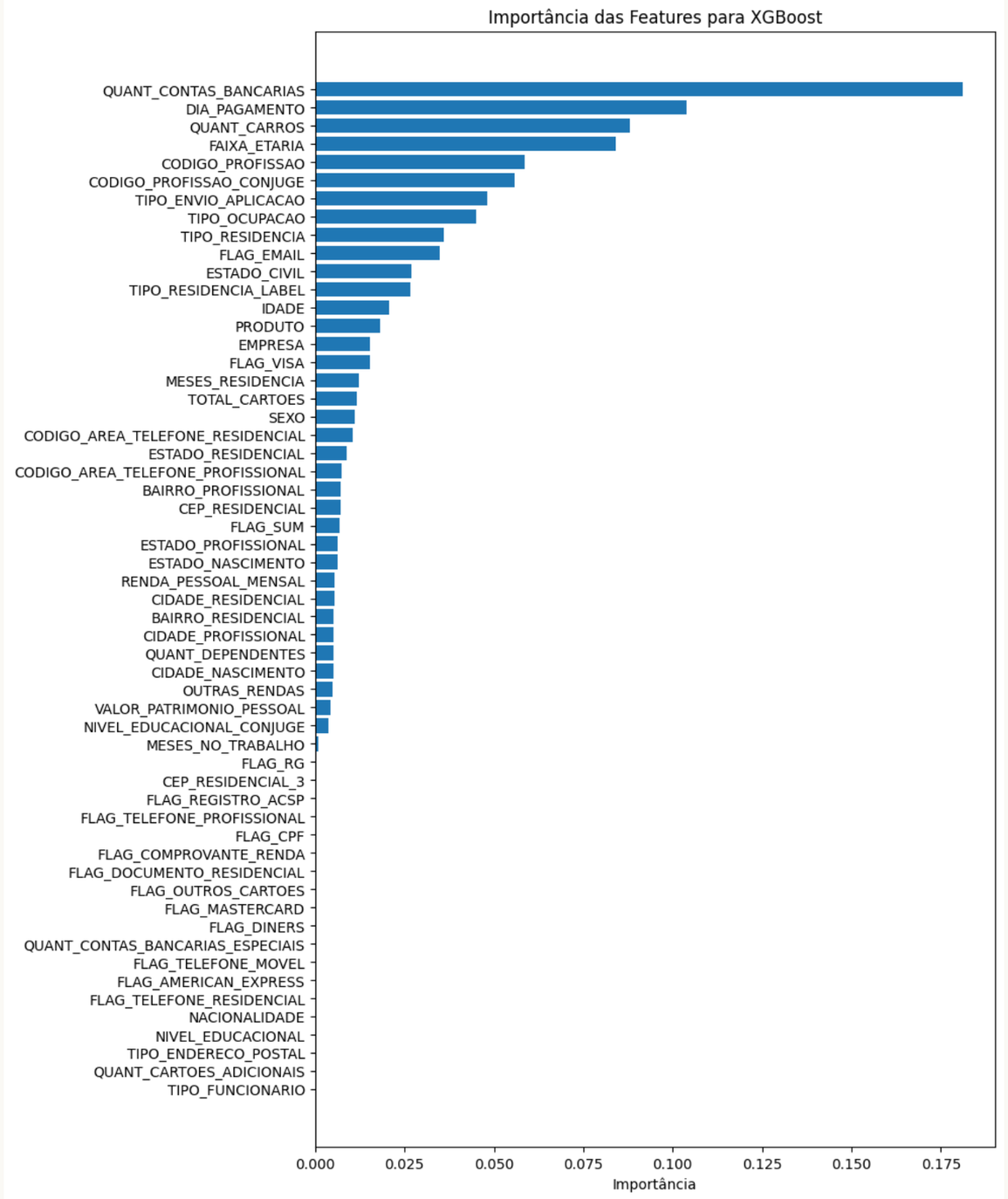
Parte 12 – Feature Importance

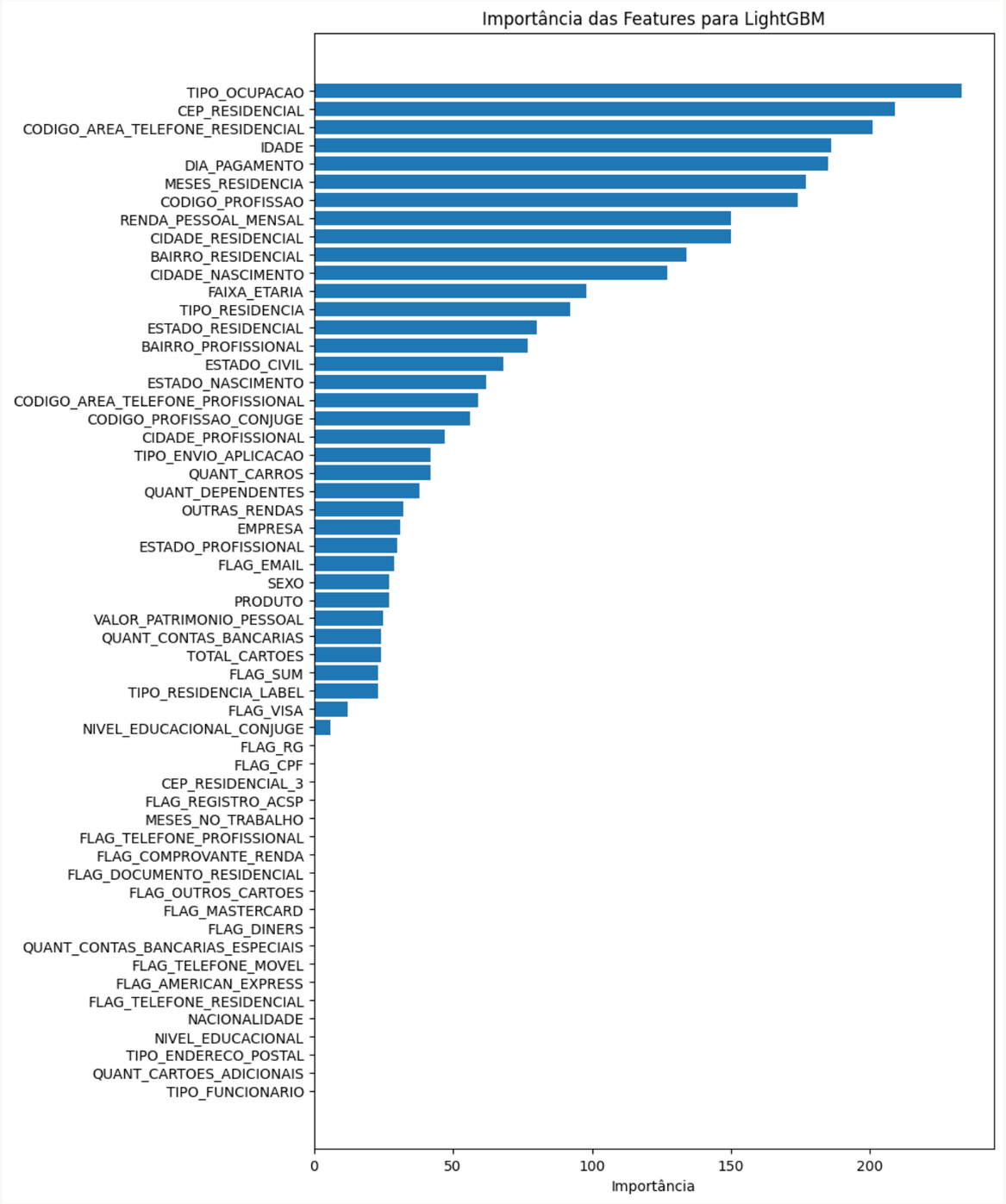
Foi a plotagem de quais features foram utilizadas por cada modelo.

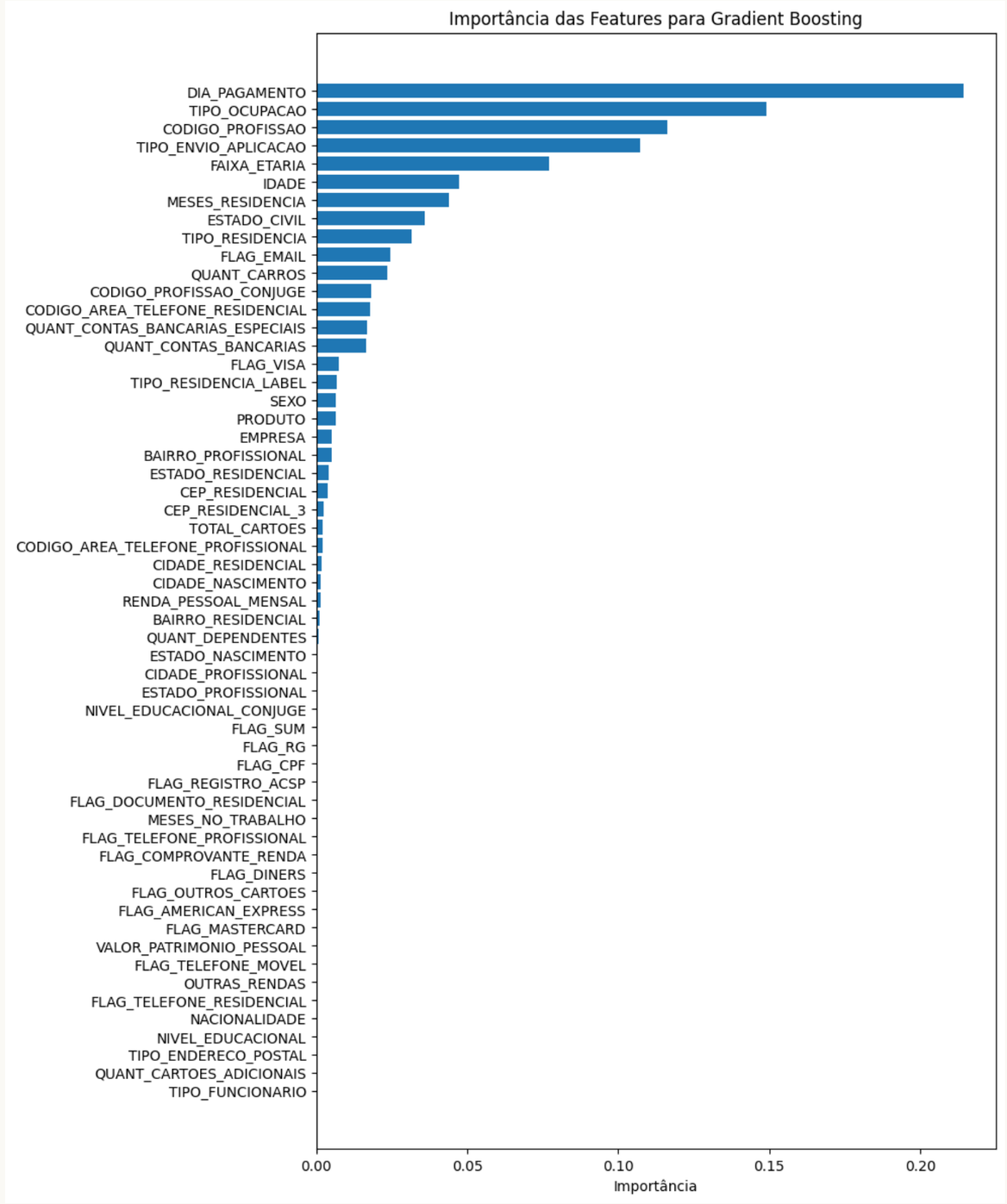





Importância das Features para Decision Tree
QUANT_CONTAS_BANCARIAS_ESPECIAIS
IDADE
DIA_PAGAMENTO
BAIRRO_RESIDENCIAL
RENDA_PESSOAL_MENSAL
CIDADE_NASCIMENTO
CIDADE_RESIDENCIAL
MESES_RESIDENCIA
FAIXA_ETARIA
CODIGO_PROFISSAO
CODIGO_PROFISSAO_CONJUGE
CODIGO_AREA_TELEFONE_RESIDENCIAL
TIPO_OCUPACAO
CEP_RESIDENCIAL_3
CEP_RESIDENCIAL
ESTADO_NASCIMENTO
TIPO_ENVIO_APLICACAO
ESTADO_CIVIL
ESTADO_RESIDENCIAL
BAIRRO_PROFISSIONAL
CIDADE_PROFISSIONAL
QUANT_DEPENDENTES
TIPO_RESIDENCIA
CODIGO_AREA_TELEFONE_PROFISSIONAL
QUANT_CARROS
ESTADO_PROFISSIONAL
TIPO_RESIDENCIA_LABEL
SEXO
OUTRAS_RENDAS
FLAG_SUM
FLAG_EMAIL
EMPRESA
TOTAL_CARTOES
VALOR_PATRIMONIO_PESSOAL
PRODUTO
FLAG_VISA
QUANT_CONTAS_BANCARIAS
FLAG_CPF
FLAG_DOCUMENTO_RESIDENCIAL
FLAG_REGISTRO_ACSP
MESES_NO_TRABALHO
FLAG_TELEFONE_PROFISSIONAL
NIVEL_EDUCACIONAL_CONJUGE
FLAG_COMPROVANTE_RENDA
FLAG_RG
FLAG_OUTROS_CARTOES
FLAG_DINERS
FLAG_MASTERCARD
FLAG_TELEFONE_MOVEL
FLAG_AMERICAN_EXPRESS
FLAG_TELEFONE_RESIDENCIAL
NACIONALIDADE
NIVEL_EDUCACIONAL
TIPO_ENDERECO_POSTAL
QUANT_CARTOES_ADICIONAIS
TIPO_FUNCIONARIO
0.00
0.01
0.02
0.03
0.04
0.05
0.06
0.07
0.08
Importância


Importância das Features para XGBoost
QUANT_CONTAS_BANCARIAS
DIA_PAGAMENTO
QUANT_CARROS
FAIXA_ETARIA
CODIGO_PROFISSAO
CODIGO_PROFISSAO_CONJUGE
TIPO_ENVIO_APLICACAO
TIPO_OCUPACAO
TIPO_RESIDENCIA
FLAG_EMAIL
ESTADO_CIVIL
TIPO_RESIDENCIA_LABEL
IDADE
PRODUTO
EMPRESA
FLAG_VISA
MESES_RESIDENCIA
TOTAL_CARTOES
SEXO
CODIGO_AREA_TELEFONE_RESIDENCIAL
ESTADO_RESIDENCIAL
CODIGO_AREA_TELEFONE_PROFISSIONAL
BAIRRO_PROFISSIONAL
CEP_RESIDENCIAL
FLAG_SUM
ESTADO_PROFISSIONAL
ESTADO_NASCIMENTO
RENDA_PESSOAL_MENSAL
CIDADE_RESIDENCIAL
BAIRRO_RESIDENCIAL
CIDADE_PROFISSIONAL
QUANT_DEPENDENTES
CIDADE_NASCIMENTO
OUTRAS_RENDAS
VALOR_PATRIMONIO_PESSOAL
NIVEL_EDUCACIONAL_CONJUGE
MESES_NO_TRABALHO
FLAG_RG
CEP_RESIDENCIAL_3
FLAG_REGISTRO_ACSP
FLAG_TELEFONE_PROFISSIONAL
FLAG_CPF
FLAG_COMPROVANTE_RENDA
FLAG_DOCUMENTO_RESIDENCIAL
FLAG_OUTROS_CARTOES
FLAG_MASTERCARD
FLAG_DINERS
QUANT_CONTAS_BANCARIAS_ESPECIAIS
FLAG_TELEFONE_MOVEL
FLAG_AMERICAN_EXPRESS
FLAG_TELEFONE_RESIDENCIAL
NACIONALIDADE
NIVEL_EDUCACIONAL
TIPO_ENDERECO_POSTAL
QUANT_CARTOES_ADICIONAIS
TIPO_FUNCIONARIO
0.000
0.025
0.050
0.075
0.100
0.125
0.150
0.175
Importância


Importância das Features para LightGBM
TIPO_OCUPACAO
CEP_RESIDENCIAL
CODIGO_AREA_TELEFONE_RESIDENCIAL
IDADE
DIA_PAGAMENTO
MESES_RESIDENCIA
CODIGO_PROFISSAO
RENDA_PESSOAL_MENSAL
CIDADE_RESIDENCIAL
BAIRRO_RESIDENCIAL
CIDADE_NASCIMENTO
FAIXA_ETARIA
TIPO_RESIDENCIA
ESTADO_RESIDENCIAL
BAIRRO_PROFISSIONAL
ESTADO_CIVIL
ESTADO_NASCIMENTO
CODIGO_AREA_TELEFONE_PROFISSIONAL
CODIGO_PROFISSAO_CONJUGE
CIDADE_PROFISSIONAL
TIPO_ENVIO_APLICACAO
QUANT_CARROS
QUANT_DEPENDENTES
OUTRAS_RENDAS
EMPRESA
ESTADO_PROFISSIONAL
FLAG_EMAIL
SEXO
PRODUTO
VALOR_PATRIMONIO_PESSOAL
QUANT_CONTAS_BANCARIAS
TOTAL_CARTOES
FLAG_SUM
TIPO_RESIDENCIA_LABEL
FLAG_VISA
NIVEL_EDUCACIONAL_CONJUGE
FLAG_RG
FLAG_CPF
CEP_RESIDENCIAL_3
FLAG_REGISTRO_ACSP
MESES_NO_TRABALHO
FLAG_TELEFONE_PROFISSIONAL
FLAG_COMPROVANTE_RENDA
FLAG_DOCUMENTO_RESIDENCIAL
FLAG_OUTROS_CARTOES
FLAG_MASTERCARD
FLAG_DINERS
QUANT_CONTAS_BANCARIAS_ESPECIAIS
FLAG_TELEFONE_MOVEL
FLAG_AMERICAN_EXPRESS
FLAG_TELEFONE_RESIDENCIAL
NACIONALIDADE
NIVEL_EDUCACIONAL
TIPO_ENDERECO_POSTAL
QUANT_CARTOES_ADICIONAIS
TIPO_FUNCIONARIO
0
50
100
150
200
Importância


Importância das Features para Gradient Boosting
DIA_PAGAMENTO
TIPO_OCUPACAO
CODIGO_PROFISSAO
TIPO_ENVIO_APLICACAO
FAIXA_ETARIA
IDADE
MESES_RESIDENCIA
ESTADO_CIVIL
TIPO_RESIDENCIA
FLAG_EMAIL
QUANT_CARROS
CODIGO_PROFISSAO_CONJUGE
CODIGO_AREA_TELEFONE_RESIDENCIAL
QUANT_CONTAS_BANCARIAS_ESPECIAIS
QUANT_CONTAS_BANCARIAS
FLAG_VISA
TIPO_RESIDENCIA_LABEL
SEXO
PRODUTO
EMPRESA
BAIRRO_PROFISSIONAL
ESTADO_RESIDENCIAL
CEP_RESIDENCIAL
CEP_RESIDENCIAL_3
TOTAL_CARTOES
CODIGO_AREA_TELEFONE_PROFISSIONAL
CIDADE_RESIDENCIAL
CIDADE_NASCIMENTO
RENDA_PESSOAL_MENSAL
BAIRRO_RESIDENCIAL
QUANT_DEPENDENTES
ESTADO_NASCIMENTO
CIDADE_PROFISSIONAL
ESTADO_PROFISSIONAL
NIVEL_EDUCACIONAL_CONJUGE
FLAG_SUM
FLAG_RG
FLAG_CPF
FLAG_REGISTRO_ACSP
FLAG_DOCUMENTO_RESIDENCIAL
MESES_NO_TRABALHO
FLAG_TELEFONE_PROFISSIONAL
FLAG_COMPROVANTE_RENDA
FLAG_DINERS
FLAG_OUTROS_CARTOES
FLAG_AMERICAN_EXPRESS
FLAG_MASTERCARD
VALOR_PATRIMONIO_PESSOAL
FLAG_TELEFONE_MOVEL
OUTRAS_RENDAS
FLAG_TELEFONE_RESIDENCIAL
NACIONALIDADE
NIVEL_EDUCACIONAL
TIPO_ENDERECO_POSTAL
QUANT_CARTOES_ADICIONAIS
TIPO_FUNCIONARIO
0.00
0.05
0.10
0.15
0.20
Importância

Parte 12 – Feature Importance - Conclusions

Foi possível verificar que certos campos se **destacaram** por aparecerem com frequência entre os modelos:

Dessa forma é possível se dar mais foco na atenção e avaliação dessas features quando necessária avaliação mais criteriosa, entendendo que estão sendo importantes para cada modelo e problema.

Features com alta relevância:

Informações financeiras:
RENDA_PESSOAL_MENSAL, IDADE

Localização:
BAIRRO_RESIDENCIAL, CEP_RESIDENCIAL, ESTADO_RESIDENCIAL

Perfil pessoal:
ESTADO_CIVIL, DIA_PAGAMENTO

Parte 13 - Métricas e avaliação para cada modelo.

Plotadas as matrizes de confusão para cada modelo e extraídas as métricas de **acurácia, precisão, revocação (recall) e F1-score**

Resultados por modelos

Naive Bayes

Matriz:

```
[[368, 500],
 [126, 204]]
```

- TP = 204, TN = 368, FP = 500, FN = 126
- Acurácia: $\frac{368+204}{1198} \approx 0.478$
- Precisão: $\frac{204}{204+500} \approx 0.290$
- Recall: $\frac{204}{204+126} \approx 0.618$
- F1: ≈ 0.396

Decision Tree

Matriz:

```
[[584, 284],
 [210, 120]]
```

- TP = 120, TN = 584, FP = 284, FN = 210
- Acurácia: $\frac{584+120}{1198} \approx 0.587$
- Precisão: $\frac{120}{120+284} \approx 0.297$
- Recall: $\frac{120}{120+210} \approx 0.364$
- F1: ≈ 0.327

Random Forest

Matriz:

```
[[786, 82],
 [284, 46]]
```

- TP = 46, TN = 786, FP = 82, FN = 284
- Acurácia: $\frac{786+46}{1198} \approx 0.694$
- Precisão: $\frac{46}{46+82} \approx 0.359$
- Recall: $\frac{46}{46+284} \approx 0.139$
- F1: ≈ 0.201

Resultados por modelos

Logistic Regression

Matriz:

```
[[529, 339],
 [170, 160]]
```

- TP = 160, TN = 529, FP = 339, FN = 170
- Acurácia: $\frac{529+160}{1198} \approx 0.576$
- Precisão: $\frac{160}{160+339} \approx 0.321$
- Recall: $\frac{160}{160+170} \approx 0.485$
- F1: ≈ 0.385

AdaBoost

Matriz:

```
[[672, 196],
 [230, 100]]
```

- TP = 100, TN = 672, FP = 196, FN = 230
- Acurácia: $\frac{672+100}{1198} \approx 0.644$
- Precisão: $\frac{100}{100+196} \approx 0.338$
- Recall: $\frac{100}{100+230} \approx 0.303$
- F1: ≈ 0.319

XGBoost

Matriz:

```
[[753, 115],
 [250, 80]]
```

- TP = 80, TN = 753, FP = 115, FN = 250
- Acurácia: $\frac{753+80}{1198} \approx 0.695$
- Precisão: $\frac{80}{80+115} \approx 0.410$
- Recall: $\frac{80}{80+250} \approx 0.242$
- F1: ≈ 0.303

LightGBM

Matriz:

```
[[779, 89],
 [275, 55]]
```

- TP = 55, TN = 779, FP = 89, FN = 275
- Acurácia: $\frac{779+55}{1198} \approx 0.696$
- Precisão: $\frac{55}{55+89} \approx 0.382$
- Recall: $\frac{55}{55+275} \approx 0.167$
- F1: ≈ 0.233

KNN

Matriz:

```
[[448, 420],
 [182, 148]]
```

- TP = 148, TN = 448, FP = 420, FN = 182
- Acurácia: $\frac{448+148}{1198} \approx 0.497$
- Precisão: $\frac{148}{148+420} \approx 0.260$
- Recall: $\frac{148}{148+182} \approx 0.448$
- F1: ≈ 0.327

Gradient Boosting

Matriz:

```
[[762, 106],
 [274, 56]]
```

- TP = 56, TN = 762, FP = 106, FN = 274
- Acurácia: $\frac{762+56}{1198} \approx 0.683$
- Precisão: $\frac{56}{56+106} \approx 0.346$
- Recall: $\frac{56}{56+274} \approx 0.170$
- F1: ≈ 0.230

Conclusão: Análise Comparativa das Matrizes de Confusão
A partir das matrizes de confusão e das métricas derivadas (acurácia, precisão, recall e F1-score), podemos tirar as seguintes conclusões:

Modelos com Melhor Desempenho Geral (Alta Acurácia)
•*Random Forest, XGBoost e LightGBM foram os modelos com maior acurácia (em torno de 0.694 a 0.696).

•No entanto, todos eles apresentaram baixa capacidade de identificar "Bons Pagadores" (classe minoritária), evidenciado pelo baixo recall e F1-score, o que pode ser problemático em aplicações onde falsos negativos (não identificar um bom pagador) são críticos.

Modelos com Alta Sensibilidade (Recall) para Bons Pagadores

•O modelo Naive Bayes teve o maior recall para bons pagadores (≈ 0.618), ou seja, foi o que melhor identificou a classe positiva.

•Contudo, isso veio ao custo de uma alta taxa de falsos positivos, gerando uma precisão muito baixa (≈ 0.290).

Modelos Balanceados

•Logistic Regression e Decision Tree mostraram métricas mais equilibradas, com F1-scores entre 0.32 e 0.38.

•Eles não são os mais precisos nem os mais sensíveis, mas podem ser úteis quando se busca equilíbrio entre evitar falsos positivos e falsos negativos.

Interpretação Estratégica

•Se o objetivo é minimizar o risco de conceder crédito a maus pagadores, priorize modelos com alta precisão (ex: XGBoost).

•Se o foco for identificar oportunidades entre bons pagadores, prefira modelos com alto recall (ex: Naive Bayes ou Logistic Regression).

•Para um cenário mais equilibrado, Decision Tree ou Logistic Regression com ajuste de limiar e reamostragem (SMOTE, class weights) podem ser uma boa estratégia inicial.

Conclusão Final

Embora modelos como Random Forest e LightGBM tenham apresentado alta acurácia, seu desempenho em relação à identificação de bons pagadores é limitado. Um modelo com melhor recall ou um ensemble balanceado pode ser mais eficaz em um cenário realista de crédito.

Análise da Curva ROC e AUC

Comparativo dos Modelos

Modelo	AUC	Interpretação
Naive Bayes	0.52	Ruim – quase aleatório
Decision Tree	0.52	Ruim – quase aleatório
Random Forest	0.59	Moderado – melhor que aleatório
Logistic Regression	0.56	Fraco
AdaBoost	0.57	Fraco
XGBoost	0.58	Fraco a Moderado
LightGBM	**0.60**	**Melhor entre os modelos testados**
K-Nearest Neighbors	0.48	Abaixo do aleatório
Gradient Boosting	0.59	Moderado

Considerações:

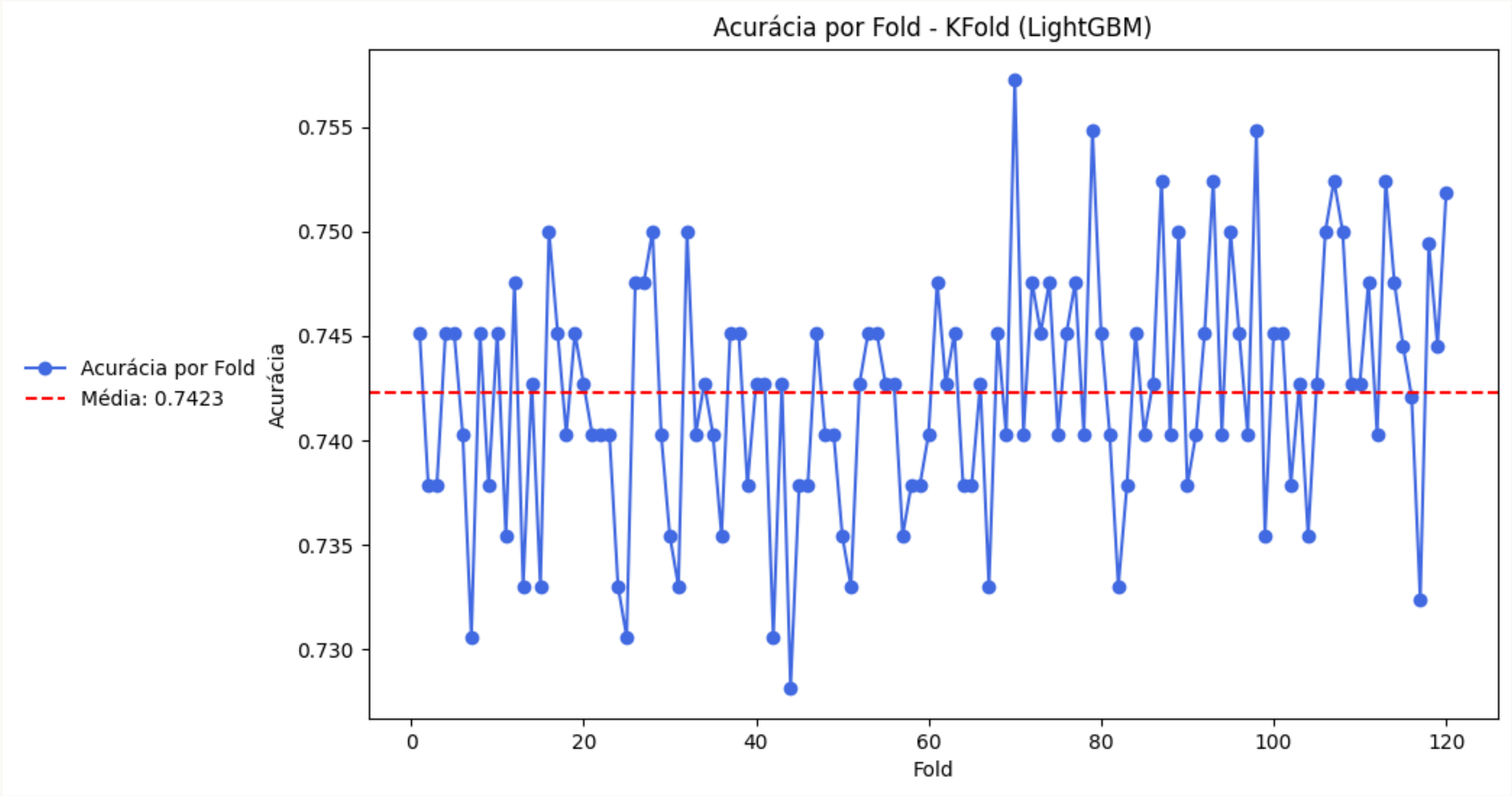
•Modelos de Ensemble (LightGBM, Gradient Boosting, XGBoost, AdaBoost) tendem a apresentar melhor capacidade discriminativa (AUC > 0.57).

•Modelos como Naive Bayes e Decision Tree puros tiveram desempenho quase aleatório, sugerindo que não são capazes de capturar bem a separação entre bons e maus pagadores com os dados fornecidos.

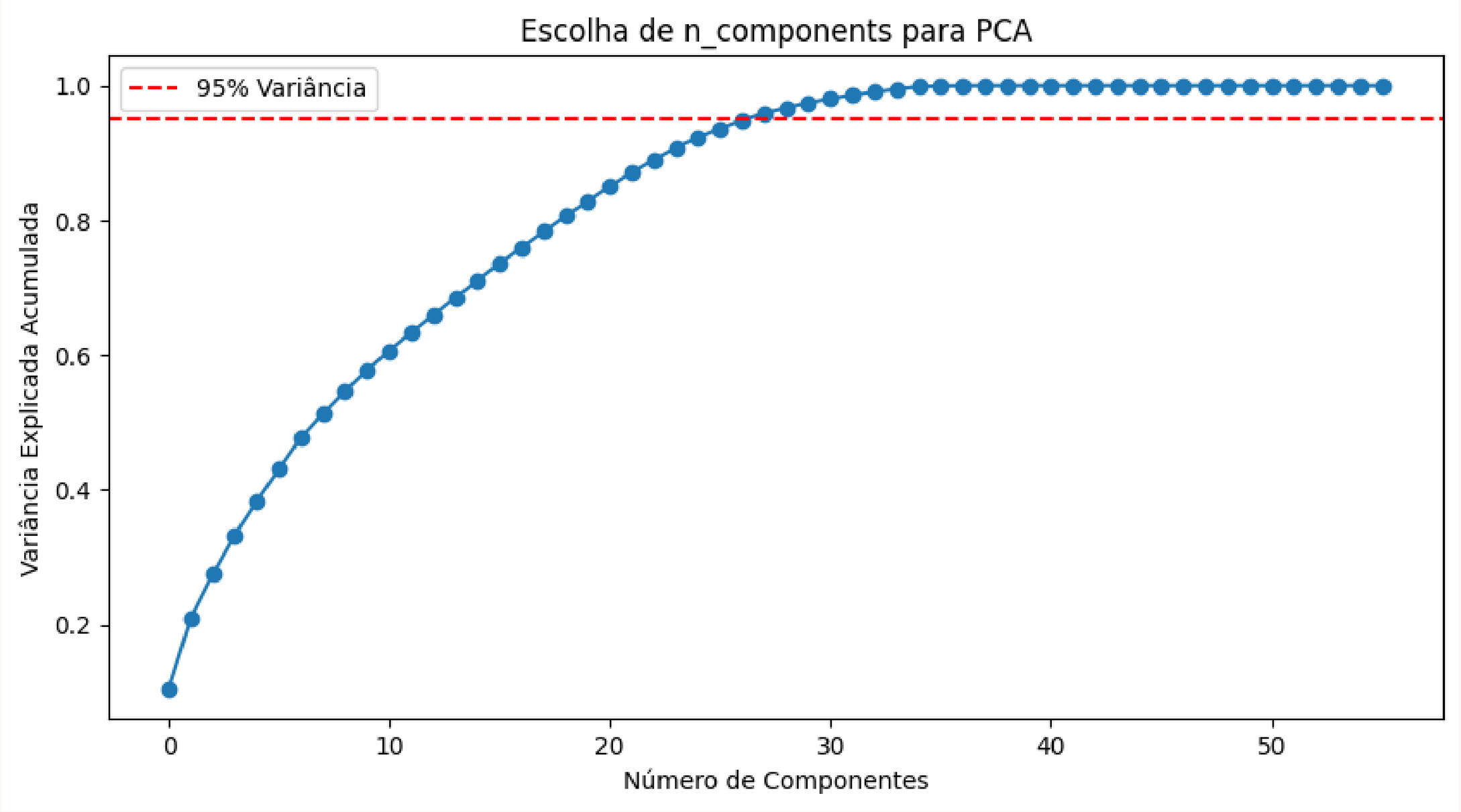
•LightGBM apresentou a melhor AUC (0.60), embora ainda modesta — indicando que há potencial de ganho com engenharia de atributos e balanceamento de classes.

•O modelo KNN teve o pior desempenho (AUC = 0.48), abaixo da linha aleatória, o que indica inadequação para este problema específico, possivelmente devido à alta dimensionalidade ou escala não apropriada.

Utilização do LightGBM



•A média de acurácia (0,7423) indica que o modelo tem um desempenho razoável, mas a variabilidade sugere que ele pode estar sensível a diferentes divisões do conjunto de dados.
•Seria interessante investigar as características dos folds com maior e menor acurácia para identificar padrões ou outliers.
•Ajustes nos hiperparâmetros ou técnicas de balanceamento de classes podem ajudar a melhorar a estabilidade do modelo.



O gráfico indica que 30 componentes principais são suficientes para atingir 95% da variância acumulada nos dados. A partir desse ponto, o acréscimo de novos componentes tem impacto mínimo na variância total explicada. Esse valor pode ser utilizado como critério para seleção de variáveis no processo de redução de dimensionalidade.

Interpretação da Visualização PCA (2 Componentes)

•**Eixos**:
- O eixo horizontal representa o primeiro componente extraído da transformação.
- O eixo vertical representa o segundo componente.

•**Cores**:
- Cada ponto é associado a uma categoria da variável de destino.
- Os grupos estão indicados com cores distintas.

•**Distribuição**:
- Os dados das duas categorias aparecem distribuídos de forma próxima.
- Há interseção significativa entre os grupos em diversas regiões do gráfico.
- A maior concentração está entre -2 e +2 no eixo horizontal, com maior dispersão vertical.

•**Sobreposição**:
- A configuração observada mostra que os dois componentes utilizados não separam as categorias de maneira visível.
- Componentes adicionais podem conter informações relevantes que não foram capturadas neste plano.

Observação Final
A projeção em duas dimensões permite visualizar a estrutura geral, mas não apresenta distinção clara entre os grupos. A separação entre as categorias pode requerer mais dimensões para ser representada de forma adequada.

Aplicação do Modelo t-SNE

O t-SNE (t-distributed Stochastic Neighbor Embedding) é uma técnica de redução de dimensionalidade projetada para representar dados de alta dimensão em um espaço de duas ou três dimensões. O principal objetivo de aplicar t-SNE é visualizar a estrutura dos dados preservando a relação local entre os pontos.
Ao contrário de métodos lineares como PCA, o t-SNE é capaz de capturar relações não lineares. Ele prioriza a manutenção da proximidade entre os dados similares, o que é útil quando há sobreposição de classes ou agrupamentos complexos no espaço original.

Esse método é especialmente útil para:
1.Analisar separação entre grupos em problemas de classificação;
2.Detectar padrões locais que não são evidentes com técnicas lineares;
3.Avaliar a distribuição dos dados após o pré-processamento.

A aplicação do t-SNE neste contexto busca entender se há algum padrão espacial entre as classes da variável alvo quando os dados são projetados em duas dimensões, complementando as análises anteriores realizadas com PCA.

```python
def testar_tsne_n_components(X_scaled, cluster_labels, n_components_list=[2, 3], perplexity=30, learning_rate=200):
    """
    Testa diferentes valores de n_components para t-SNE e retorna visualizações
    e métricas de separação.
    """

    results = {}

    for n in n_components_list:
        print(f"\nExecutando t-SNE com n_components = {n}...")
        tsne = TSNE(
            n_components=n,
            perplexity=perplexity,
            learning_rate=learning_rate,
            max_iter=1000,        # substitui n_iter
            init="pca",
            random_state=42,
            verbose=1
        )

        tsne_result = tsne.fit_transform(X_scaled)

        # Visualização apenas para 2 componentes
        if n == 2:
            df_tsne = pd.DataFrame(tsne_result, columns=['Dim1', 'Dim2'])
            df_tsne['Label'] = cluster_labels.values  # Certifique-se que tem mesmo tamanho

            plt.figure(figsize=(18, 6))
            sns.scatterplot(data=df_tsne, x='Dim1', y='Dim2', hue='Label', palette='muted', s=70)
            plt.title(f't-SNE com 2 componentes (perplexidade={perplexity})')
            plt.grid(False)
            plt.tight_layout()
            plt.show()
```

```python
        # Silhouette Score (apenas se houver mais de 1 classe)
        try:
            score = silhouette_score(tsne_result, cluster_labels)
            print(f"Silhouette Score (n_components={n}): {score:.4f}")
            results[n] = score
        except Exception as e:
            print(f"Não foi possível calcular o silhouette para n_components = {n}: {e}")

    return results

# Aplicando t_SNE
resultados_tsne = testar_tsne_n_components(X_train_scaled, y_train, n_components_list=[2, 3])
```

Considerações finais:

O projeto de machine learning iniciou com a construção do dicionário de dados e análise do dataset, seguido pelo pré-processamento com implantação do dicionário definido.
A limpeza incluiu a identificação e substituição de valores nulos por zero.
Na etapa exploratória, foram respondidas perguntas de negócio, enquanto os outliers foram identificados e tratados, utilizando o Z Score para substituir valores extremos pela mediana.
Na fase de feature engineering, identificaram-se variáveis categóricas e utilizou-se o Label Encoder.
A divisão entre treino e teste foi realizada com definição das variáveis preditoras (X e y), e o modelo foi treinado com random_seed fixa em 42.
Para balancear as classes, aplicou-se o método SMOTE. Diversos algoritmos foram testados, incluindo GaussianNB, Decision Tree, Random Forest, Logistic Regression, AdaBoost, XGBoost, LightGBM, KNN, Gradient Boosting e SVC.
A análise das features identificou as mais relevantes para os modelos, e a avaliação foi realizada com matrizes de confusão e comparações de acurácia.

Conclusão, os modelos com Melhor Desempenho Geral (Alta Acurácia):

- Random Forest, XGBoost e LightGBM foram os modelos com maior acurácia (em torno de 0.694 a 0.696).

- Toda a preparação dos dados foram totalmente importante para um melhor desempenho do modelo.