Previsão de Posse de Computador Residencial: Uma Análise Usando o Dataset da PNS 2019

Henrique Lara*
henriquerlara@outlook.com
Pontifícia Universidade Católica de Minas Gerais (PUC-MG)
Belo Horizonte, Minas Gerais, Brasil

Rodrigo Drummond
rodrigodm2810@gmail.com
Pontifícia Universidade Católica de Minas Gerais (PUC-MG)
Belo Horizonte, Minas Gerais, Brasil

Tiago Lascasas
tiago-antunes@hotmail.com.br
Pontifícia Universidade Católica de Minas Gerais (PUC-MG)
Belo Horizonte, Minas Gerais, Brasil

Thomas Neuenschwander*
thom.nmaciel.baron@gmail.com
Pontifícia Universidade Católica de Minas Gerais (PUC-MG)
Belo Horizonte, Minas Gerais, Brasil

ABSTRACT

Este estudo utiliza dados do módulo A - Informações do Domicílio da Pesquisa Nacional de Saúde de 2019 (PNS 2019) para prever a posse de computadores em residências brasileiras com base em atributos domiciliares e socioeconômicos. Foram selecionados atributos como estrutura familiar, condições de moradia e nível educacional, entre outros, para treinar modelos de aprendizado de máquina e realizar previsões. Os atributos utilizados foram processados para gerar insights sobre a correlação entre fatores sociais e a presença de computadores nas residências. Diferentes modelos de classificação foram testados, e seu desempenho foi avaliado por meio de métricas como acurácia, precisão e F1-Score. Os resultados demonstram que os atributos domiciliares selecionados são bons indicadores da posse de computador, com modelos como árvores de decisão e KNN apresentando as melhores performances preditivas. Este trabalho reforça a utilidade de técnicas de aprendizado de máquina na análise de tendências tecnológicas em lares brasileiros.

KEYWORDS

Aprendizado de Máquina, Classificação de Dados, Inclusão Digital, Previsão de Posse de Computador, Análise Socioeconômica, PNS 2019, Modelos de Classificação, Técnicas de Pré-processamento, Desigualdade Tecnológica, Atributos Domiciliares

ACM Reference Format:
Henrique Lara, Rodrigo Drummond, Tiago Lascasas, and Thomas Neuenschwander. 2018. Previsão de Posse de Computador Residencial: Uma Análise Usando o Dataset da PNS 2019. In *Woodstock '18: ACM Symposium on Neural Gaze Detection, June 03–05, 2018, Woodstock, NY*. ACM, New York, NY, USA, 9 pages. https://doi.org/10.1145/1122445.1122456

1 INTRODUÇÃO

Nos últimos anos, o acesso à tecnologia e à internet tem se tornado um dos principais indicadores de inclusão social e econômica. A presença de computadores nas residências é um fator crucial para a inclusão digital, promovendo oportunidades de educação, trabalho remoto e acesso à informação. No entanto, a posse de computadores nas residências brasileiras ainda é desigual, com grande variação entre diferentes regiões e classes sociais [13].

Este trabalho busca investigar os fatores socioeconômicos que influenciam a posse de computadores nas residências no Brasil, utilizando dados da Pesquisa Nacional de Saúde de 2019 (PNS 2019). Com base no módulo A da PNS, que contém informações sobre características domiciliares, será realizada uma análise preditiva para identificar os principais atributos associados à presença de computadores nas casas. Através de técnicas de aprendizado de máquina, modelos de classificação serão treinados para prever a posse de computadores a partir de variáveis como renda, escolaridade, estrutura familiar, entre outros.

*Ambos os autores contribuíram igualmente para esta pesquisa.

Woodstock '18, June 03–05, 2018, Woodstock, NY

ACM ISBN 978-1-4503-9999-9/18/06…$15.00
https://doi.org/10.1145/1122445.1122456

O objetivo deste estudo é não apenas prever a posse de computadores, mas também entender como os fatores sociais contribuem para a inclusão digital no país. Este tipo de análise pode auxiliar na formulação de políticas públicas voltadas para a redução das desigualdades tecnológicas, promovendo maior acesso à tecnologia em áreas e grupos sociais que historicamente têm menos acesso a esses recursos. A análise dos dados da PNS 2019 oferece uma oportunidade única para avaliar como o cenário atual de posse de computadores reflete as desigualdades sociais e regionais no Brasil.

2 MATERIAIS E MÉTODOS

Descrição da base de dados

A base de dados utilizada neste estudo contém, inicialmente, 40 atributos que descrevem características domiciliares e socioeconômicas das residências brasileiras. Esses atributos incluem informações como estrutura familiar, material de construção, quantidade de cômodos e outros fatores relevantes. Durante o pré-processamento, alguns atributos foram removidos devido à alta quantidade de valores ausentes, como 'A01402', 'A01403', 'A018016', 'A018018', 'A018022', 'A018026', 'A018028', e 'D00202'. Além disso, o atributo 'A018024' foi removido por ser redundante, pois sua presença revela diretamente o conteúdo de 'A018023'. Como 'A018023' indica se o domicílio possui computador, qualquer valor numérico em 'A018024' já confirma essa presença, introduzindo um viés no modelo de IA. Isso permitiria que o modelo detectasse a existência de computadores apenas com base no preenchimento de 'A018024', sem precisar aprender a relação entre os atributos de forma independente.

Table 1: Tabela de contingência entre A018023 e A018024

A018023	1	2	3	4	5	6	8	NaN
1	18702	4328	1099	322	94	31	4	0
2	0	0	0	0	0	0	0	70284

A tabela acima evidencia que o atributo de contagem de computadores se torna desnecessário. Quando 'A018023' indica ausência de computadores, não há registro de quantidade, resultando em "Não informado" (NaN). Em contraste, com a presença de computadores, temos uma contagem numérica associada.

Os atributos da base possuem diferentes naturezas. Alguns são categóricos, como 'A001' (tipo do domicílio, onde 1 = Casa, 2 = Apartamento, 3 = Cortiço) ou 'A018011' (existência de televisão em cores, onde 1 = Sim, 2 = Não). Embora representados numericamente, esses atributos são nominais, já que não possuem uma ordem ou escala quantitativa entre seus valores.

Além dos atributos nominais, a base contém atributos quantitativos que são considerados discretos. Atributos como o número de cômodos ('A01001'), que pode variar entre 1 e 30, ou a quantidade de televisores ('A018012'), que varia entre 0 e 5, são exemplos de dados numéricos discretos. Para cada um desses atributos, os possíveis valores estão dentro de intervalos predefinidos. No entanto, antes de serem utilizados nos modelos de aprendizado de máquina, esses valores passaram por uma etapa de normalização, onde foram transformados para o intervalo entre 0 e 1. A normalização foi realizada utilizando a função MinMaxScaler [3], que ajusta os valores de cada atributo com base em seus valores mínimo e máximo.

A base de dados utilizada neste estudo é supervisionada, tendo a posse de computadores nas residências como variável-alvo. Foram aplicados diferentes modelos de aprendizado de máquina, incluindo Naive Bayes, Árvore de Decisão, Random Forest, K-Nearest Neighbors (KNN), Regressão Logística e Self-Organizing Map (SOM), para prever se uma residência possui ou não um computador com base nos atributos fornecidos. As etapas de pré-processamento incluíram Filtragem dos Atributos, Tratamento de Valores Ausentes, Remoção de Valores Inconsistentes e Duplicados, Conversão com One-Hot Encoding, Normalização dos Dados, Binarização de Variáveis e Balanceamento de Classes e Divisão de Dados.

Após o pré-processamento, o dataset final conta com 36.073 instâncias da classe "domicílios sem computador" e 11.785 instâncias da classe "domicílios com computador", representando um total significativo de dados que serão utilizados para análise e modelagem preditiva.

Etapas de Pré-processamento

Nesta seção, são apresentadas as etapas de pré-processamento aplicadas ao dataset da PNS 2019, utilizando os atributos selecionados. O objetivo foi garantir a qualidade e a consistência dos dados antes da aplicação dos modelos de aprendizado de máquina, otimizando os resultados.

Filtragem dos Atributos. A filtragem inicial envolveu a seleção de atributos relacionados a aspectos socioeconômicos, relevantes para a previsão da posse de computadores nas residências. Essa escolha foi guiada pelo objetivo do estudo, priorizando variáveis com maior impacto na análise.

Tratamento de Valores Ausentes. Os valores ausentes foram tratados utilizando um processo em múltiplas etapas. Inicialmente, valores como 9 e 99, indicados no dataset como "Ignorados", foram transformados em nulos para evitar que influenciassem os modelos de maneira indevida. As instâncias com valores nulos na variável de classe foram removidas, assim como atributos com mais de 60% de valores ausentes, como 'A005012', 'A01402', 'A01403', 'A018018', 'A018022', 'A018024', 'A018026', 'A018028' e 'D00202', que continham pouca informação útil. Para as demais variáveis, a imputação

Table 2: Descrição dos Atributos

Atributo	Descrição
A001 (A1)	Tipo do domicílio
A002010 (A2a)	Material predominante nas paredes
A003010 (A3a)	Material predominante na cobertura
A004010 (A4a)	Material predominante no piso
A01001 (A10a)	Quantidade de cômodos
A011	Quantidade de dormitórios
A005010 (A5a)	Principal forma de abastecimento de água
A005012 (A5b)	Ligado à rede geral de água?
A00601 (A6a)	Água canalizada
A009010 (A9a)	Água para beber
A01401 (A14a)	Quantidade de banheiros exclusivos
A01402 (A14b)	Quantidade de banheiros de uso comum
A01403 (A14c)	Uso de sanitário ou buraco para dejeções
A01501 (A15a)	Destino do esgoto
A016010 (A16a)	Destino do lixo
A018011 (A18a)	Existe televisão em cores?
A018012 (A18b)	Quantidade de televisões em cores
A018013 (A18c)	Existe geladeira?
A018014 (A18d)	Quantidade de geladeiras
A018015 (A18e)	Existe máquina de lavar roupa?
A018016 (A18f)	Quantidade de máquinas de lavar roupa
A018017 (A18g)	Existe telefone fixo?
A018018 (A18h)	Quantidade de telefones fixos
A018019 (A18i)	Existe telefone celular?
A018020 (A18j)	Quantidade de celulares
D001 (D1)	Sabe Ler e Escrever
D00201 (D2a)	Frequenta Escola ou Creche
D00202 (D2b)	Tipo de Escola Frequentada

foi realizada utilizando o *KNN Imputer*, escolhendo registros semelhantes para preencher os valores faltantes. Para os dados quantitativos, o *KNN* foi aplicado normalmente, mantendo a coerência com os padrões numéricos. Já nos dados categóricos, após a imputação, os valores foram ajustados para se limitarem às categorias válidas existentes, evitando a criação de novas classes. Isso foi essencial para evitar o aumento desnecessário da dimensionalidade no *One-Hot Encoding* e o risco de *overfitting*, além de garantir maior consistência com os dados originais [12].

Remoção de Valores Inconsistentes e Duplicados. Valores inconsistentes, como instâncias em que as mesmas características estavam associadas a classes diferentes, foram removidos para evitar impacto negativo nos modelos. Além disso, registros duplicados foram eliminados, garantindo que cada entrada no dataset fosse única, reduzindo a chance de vieses nos resultados.

Table 3: Valores Possíveis dos Atributos

Atributo	Valores Possíveis
A001 (A1)	1: Casa, 2: Apartamento, 3: Cortiço, 9: Ignorado, N/A
A002010 (A2a)	1: Alvenaria c/ revest., 2: Alvenaria s/ revest., 3: Taipa s/ revest., 4: Madeira aprop., 5: Madeira aproveit., 6: Outro, 9: Ignorado, N/A
A003010 (A3a)	1: Telha s/ laje, 2: Telha c/ laje, 3: Somente laje, 4: Madeira aprop., 5: Zinco/alumínio, 9: Ignorado, N/A
A004010 (A4a)	1: Cerâmica, lajota, 2: Madeira aprop., 3: Cimento, 4: Terra, 5: Outro, 9: Ignorado, N/A
A01001 (A10a)	01 a 30: Cômodos, 99: Ignorado, N/A
A011	01 a 15: Dormitórios, 99: Ignorado, N/A
A005010 (A5a)	1: Rede geral, 2: Poço prof., 3: Poço raso, 4: Fonte, 5: Água da chuva, 9: Ignorado, N/A
A005012 (A5b)	1: Sim, 2: Não, 9: Ignorado, N/A
A00601 (A6a)	1: Canalizada em cômodo, 2: Canalizada no terreno, 3: Não canalizada, 9: Ignorado, N/A
A009010 (A9a)	1: Filtrada, 2: Fervida, 3: Cloro, 4: Outra, 5: Mineral, 9: Ignorado, N/A
A01401 (A14a)	01 a 15: Banheiros, 0: Nenhum, 99: Ignorado, N/A
A01402 (A14b)	01 a 15: Banheiros, 00: Nenhum, 99: Ignorado, N/A
A01403 (A14c)	1: Sim, 2: Não, 9: Ignorado, N/A
A01501 (A15a)	1: Rede geral, 2: Fossa séptica, 3: Fossa não ligada, 4: Fossa rudimentar, 5: Vala, 6: Rio ou lago, 7: Outra, 9: Ignorado, N/A
A016010 (A16a)	1: Coletado por serviço, 2: Coletado em caçamba, 3: Queimado, 4: Enterrado, 5: Jogado em terreno baldio, 6: Outro, 9: Ignorado, N/A
A018011 (A18a)	1: Sim, 2: Não, 9: Ignorado, N/A
A018012 (A18b)	01 a 98: Quantidade de TVs, 99: Ignorado, N/A
A018013 (A18c)	1: Sim, 2: Não, 9: Ignorado, N/A
A018014 (A18d)	01 a 98: Quantidade de geladeiras, 99: Ignorado, N/A
A018015 (A18e)	1: Sim, 2: Não, 9: Ignorado, N/A
A018016 (A18f)	01 a 98: Quantidade de máquinas de lavar, 99: Ignorado, N/A
A018017 (A18g)	1: Sim, 2: Não, 9: Ignorado, N/A
A018018 (A18h)	01 a 98: Quantidade de telefones fixos, 99: Ignorado, N/A
A018019 (A18i)	1: Sim, 2: Não, 9: Ignorado, N/A
A018020 (A18j)	01 a 98: Quantidade de celulares, 99: Ignorado, N/A
D001 (D1)	1: Sim, 2: Não, 9: Ignorado, N/A
D00201 (D2a)	1: Sim, 2: Não, 9: Ignorado, N/A
D00202 (D2b)	1: Rede privada, 2: Rede pública, 9: Ignorado, N/A

Conversão com One-Hot Encoding. As variáveis categóricas com mais de duas categorias, como o tipo de domicílio (A001) e o material predominante nas paredes (A002010), foram transformadas utilizando a técnica de *One-Hot Encoding*. Essa abordagem permitiu que cada categoria fosse representada como uma coluna binária, facilitando o aprendizado do modelo sem introduzir uma hierarquia artificial entre as categorias.

Normalização dos Dados. Para tratar as variáveis numéricas, foi realizada a normalização utilizando o *Min-Max Scaling*, escalando os valores para o intervalo [0, 1]. Essa técnica foi aplicada, por exemplo, às variáveis 'A01001' (quantidade de cômodos), 'A011' (quantidade de dormitórios) e 'A018012' (quantidade de geladeiras). A normalização garantiu que todas as variáveis numéricas tivessem a mesma escala, prevenindo que valores em escalas maiores tivessem mais peso no aprendizado do modelo. A equação utilizada para a normalização é apresentada abaixo[3]:

$$X_{norm} = \frac{X - X_{min}}{X_{max} - X_{min}} \quad (1)$$

Aqui, X representa o valor original do atributo, enquanto X_{min} e X_{max} são, respectivamente, os valores mínimo e máximo da variável.

Binarização de Variáveis. As variáveis categóricas que possuíam apenas duas categorias, como 'A005012', foram binarizadas, transformando os valores diretamente em 0 e 1. Essa escolha foi feita para evitar que o modelo interpretasse os valores originais, como 1 e 2, como se houvesse uma relação de ordem entre eles. Como essas variáveis representam apenas categorias distintas, sem hierarquia ou gradação, a binarização assegura que o modelo as trate corretamente como informações puramente categóricas [4]. O rótulo 'A018013', que representa a posse de computadores, também foi submetido a este processo, sendo convertido de 1 (sim) e 2 (não) para 1 e 0, respectivamente, sendo as classes automaticamente interpretadas como booleanas.

Balanceamento das Classes. Foi observado um desbalanceamento na variável de classe, onde a posse de computador era menos frequente. Para corrigir isso, aplicamos a técnica *SMOTE* (*Synthetic Minority Over-sampling Technique*) [2], que gera amostras sintéticas da classe minoritária. Esse método foi escolhido como uma abordagem para equilibrar a distribuição das classes e possibilitar que o modelo tenha maior exposição a instâncias da classe minoritária durante o treinamento, sem perder informação.

Divisão dos Dados. Os dados foram divididos em conjuntos de treino e teste na proporção de 80% para treino e 20% para teste, utilizando a função *train_test_split* com estratificação para manter a proporção das classes em ambos os conjuntos. Após a divisão, aplicou-se o balanceamento com a técnica de *SMOTE*, exclusivamente no conjunto de treino (*X_train* e *y_train*). O conjunto de teste permaneceu inalterado para garantir que a validação fosse feita apenas com dados reais.

Table 4: Divisão do dataset em treino e teste (antes do balanceamento).

Conjunto	Porcentagem	Quantidade de Instâncias
Treinamento	80%	38.286
Teste	20%	9.572

Essas etapas garantiram a preparação de um dataset consistente e equilibrado, permitindo que os modelos de aprendizado de máquina extraíssem padrões relevantes e obtivessem previsões mais confiáveis sobre a posse de computadores nas residências.

Table 5: Transformações Aplicadas aos Atributos

Atributo	**Transformação Aplicada**
A001	One-Hot Encoded
A002010	One-Hot Encoded
A003010	One-Hot Encoded
A004010	One-Hot Encoded
A01001	Normalizado com Min-Max Scaling
A011	Normalizado com Min-Max Scaling
A005010	One-Hot Encoded
A005012	Binarizado
A00601	One-Hot Encoded
A009010	One-Hot Encoded
A01401	Normalizado com Min-Max Scaling
A01501	One-Hot Encoded
A016010	One-Hot Encoded
A018011	Binarizado
A018012	Normalizado com Min-Max Scaling
A018013	Binarizado
A018014	Normalizado com Min-Max Scaling
A018015	Binarizado
A018017	Binarizado
A018019	Binarizado
A018020	One-Hot Encoded
A018021	Binarizado
A018023	Binarizado
A018025	Binarizado
A018027	Binarizado
A01901	Binarizado
A02101	Binarizado
D001	Binarizado
D00201	Binarizado

Descrição dos métodos utilizados

Para a classificação, foram utilizados cinco algoritmos de aprendizado supervisionado e um modelo não supervisionado (SOM - Self-Organizing Map), com o objetivo de prever se uma residência possui ou não um computador.

Naive Bayes. Para o Naive Bayes se utilizou o modelo GaussianNB. Um hiperparâmetro foi testado para garantir uma maior estabilidade nas previsões ao adicionar uma pequena constante a cada variância, porém por não apresentar melhoras no resultado não foi adicionado ao modelo final[4].

Árvore de Decisão. Para a Árvore de Decisão, foi utilizado o *GridSearchCV* para ajustar hiperparâmetros como profundidade máxima (5, 10, 15, 20), número mínimo de amostras para divisão (2, 5, 10) e número mínimo de amostras por folha (1, 2, 4). Essa abordagem foi escolhida devido ao menor custo computacional da Árvore de Decisão, permitindo explorar todas as combinações definidas de forma viável. O ajuste automático resultou em uma profundidade máxima de 20, que se mostrou ideal para equilibrar desempenho e tempo de treinamento, evitando sobreajuste [6].

Random Forest. Para o *Random Forest*, optou-se pelo *RandomizedSearchCV*, ajustando os hiperparâmetros de forma mais eficiente devido ao custo computacional mais elevado deste modelo. Foram testadas combinações aleatórias de parâmetros como número de estimadores (100, 200, 300), profundidade máxima (5, 10, 20), número mínimo de amostras para divisão (2, 5, 10) e número mínimo de amostras por folha (1, 2, 4). Com essa estratégia, foi possível reduzir o tempo de busca e identificar que 200 estimadores e profundidade máxima de 20 eram as configurações ideais, equilibrando precisão e complexidade [5].

A escolha de ajuste automático foi aplicada apenas nos modelos mais sensíveis aos parâmetros, como Árvore de Decisão e *Random Forest*, pois para outros algoritmos os parâmetros padrões já eram suficientes, dado seu comportamento menos dependente de ajustes.

Regressão Logística. O modelo de regressão logística foi implementado com um estado randômico fixo em 42 para garantir reprodutibilidade. Este modelo foi utilizado sem a necessidade de ajustes significativos de hiperparâmetros, visto que é um algoritmo linear que já lida bem com dados separados linearmente[7].

KNN. O algoritmo K-Nearest Neighbors (KNN) foi configurado para considerar 5 vizinhos mais próximos (K=5), determinando a classe de cada instância com base na proximidade em relação aos dados de treinamento. Foi utilizada a métrica de distância euclidiana para calcular a similaridade, e pesos uniformes, garantindo que cada vizinho tivesse a mesma contribuição na predição. A escolha de K foi definida inicialmente como padrão para os testes e mantida depois de verificar um bom desempenho em termos de acurácia e das outras métricas no conjunto de teste[8].

Self-Organizing Map (SOM). Além dos modelos supervisionados, utilizou-se o Self-Organizing Map (SOM) para mapear os dados e prever a posse de computadores. O SOM foi configurado com uma grade de 2x2, um comprimento de entrada igual ao número de atributos do conjunto de treinamento e os seguintes parâmetros: *sigma* igual a 0.8, taxa de aprendizado de 0.2 e uma semente randômica fixada em 42 para reprodutibilidade. O treinamento foi realizado com 100 iterações sobre os dados de entrada. Após o treinamento, os clusters do SOM foram mapeados associando a cada nó vencedor a classe predominante entre os registros atribuídos a ele. Para os dados de teste, as predições foram realizadas com base no nó vencedor correspondente, atribuindo a classe majoritária do cluster. Em casos onde um cluster não possuía rótulo associado, foi utilizada a classe *-1* para indicar que o dado não foi classificado[7].

Ambiente. Todo o processo relatado ao longo do relatório foi realizado nos ambientes do Google Colab e de Jupyter Notebooks, ambos para a criação e desenvolvimento do código. No ambiente Colab a versão do Python utilizada foi a 3.10 e no Notebook a versão 3.12

Métricas de Avaliação de Qualidade

Para avaliar a qualidade dos modelos, foram utilizadas as métricas de acurácia, precisão, recall e F1-score [10], todas provenientes da biblioteca scikit-learn. Essas métricas fornecem uma visão abrangente do desempenho dos modelos e são definidas matematicamente como segue:

Acurácia. A acurácia mede a proporção de previsões corretas em relação ao total de previsões, fornecendo uma visão geral de quão bem o modelo classifica as amostras. Sua fórmula é dada por:

$$\text{Acurácia} = \frac{TP + TN}{TP + TN + FP + FN} \tag{2}$$

Onde:

- TP: Verdadeiros positivos
- TN: Verdadeiros negativos
- FP: Falsos positivos
- FN: Falsos negativos

Precisão (Precision). A precisão indica a proporção de previsões positivas que foram corretas. Sua fórmula é:

$$\text{Precisão} = \frac{TP}{TP + FP} \tag{3}$$

Recall (Sensibilidade). O recall, também conhecido como sensibilidade, mede a capacidade do modelo de identificar todas as amostras positivas reais. É definido como:

$$\text{Recall} = \frac{TP}{TP + FN} \tag{4}$$

F1-Score. O F1-score é a média harmônica entre precisão e recall, equilibrando os dois aspectos. Sua fórmula é:

$$\text{F1-Score} = 2 \cdot \frac{\text{Precisão} \cdot \text{Recall}}{\text{Precisão} + \text{Recall}} \tag{5}$$

Essa métrica é especialmente útil em problemas com desbalanceamento de classes, pois considera tanto os falsos positivos quanto os falsos negativos no cálculo.

Além disso, foi gerada uma matriz de confusão para cada modelo, a fim de visualizar as quantidades de verdadeiros positivos (acertos para a classe positiva), falsos negativos (amostras positivas classificadas como negativas), falsos positivos (amostras negativas classificadas como positivas) e verdadeiros negativos (acertos para a classe negativa). Para facilitar a interpretação, essa matriz foi exibida por meio de um mapa de calor, utilizando a biblioteca seaborn, que permite identificar rapidamente os padrões de erros e acertos do modelo.

3 RESULTADOS E DISCUSSÕES

Neste tópico, são apresentados os resultados obtidos durante o processo de treinamento e avaliação dos modelos. Os resultados abordam o problema de classificação: se no domicílio existe computador.

Naive Bayes

Acurácia (teste): 0.7193

	precision	recall	f1-score	support
False	0.92	0.69	0.79	10822
True	0.46	0.82	0.59	3536
accuracy			0.72	14358
macro avg	0.69	0.75	0.69	14358
weighted avg	0.81	0.72	0.74	14358

Random Forest

Melhores hiperparâmetros:

```
{'n_estimators': 200,
'min_samples_split': 10,
'min_samples_leaf': 2,
'max_depth': 20}
```

Acurácia (teste): 0.8440

	precision	recall	f1-score	support
False	0.91	0.88	0.89	10822
True	0.67	0.74	0.70	3536
accuracy			0.84	14358
macro avg	0.79	0.81	0.80	14358
weighted avg	0.85	0.84	0.85	14358

Árvore de Decisão

Melhores hiperparâmetros:

```
{'max_depth': 20,
'min_samples_leaf': 1,
'min_samples_split': 2}
```

Acurácia (teste): 0.8340

	precision	recall	f1-score	support
False	0.90	0.88	0.89	10822
True	0.65	0.70	0.68	3536
accuracy			0.83	14358
macro avg	0.78	0.79	0.78	14358
weighted avg	0.84	0.83	0.84	14358

Regressão Logística

Acurácia (teste): 0.8092

	precision	recall	f1-score	support
False	0.91	0.83	0.87	10822
True	0.59	0.74	0.65	3536
accuracy			0.81	14358
macro avg	0.75	0.78	0.76	14358
weighted avg	0.83	0.81	0.82	14358

KNN

Acurácia (teste): 0.8331

	precision	recall	f1-score	support
False	0.93	0.84	0.88	10822
True	0.63	0.80	0.70	3536
accuracy			0.83	14358
macro avg	0.78	0.82	0.79	14358
weighted avg	0.85	0.83	0.84	14358

Self-Organizing Map (SOM)

Acurácia (teste): 0.7234

	precision	recall	f1-score	support
False	0.89	0.72	0.80	10822
True	0.46	0.73	0.56	3536
accuracy			0.72	14358
macro avg	0.67	0.72	0.68	14358
weighted avg	0.78	0.72	0.74	14358

Conclusão sobre os Modelos

A análise comparativa entre os modelos de classificação Naive Bayes, Random Forest, Árvore de Decisão, KNN, Regressão Logística e Self-Organizing Map (SOM) revelou variações significativas no desempenho de acordo com as métricas de avaliação.

Figure 1: Mapa SOM com localização dos dados de treinamento.

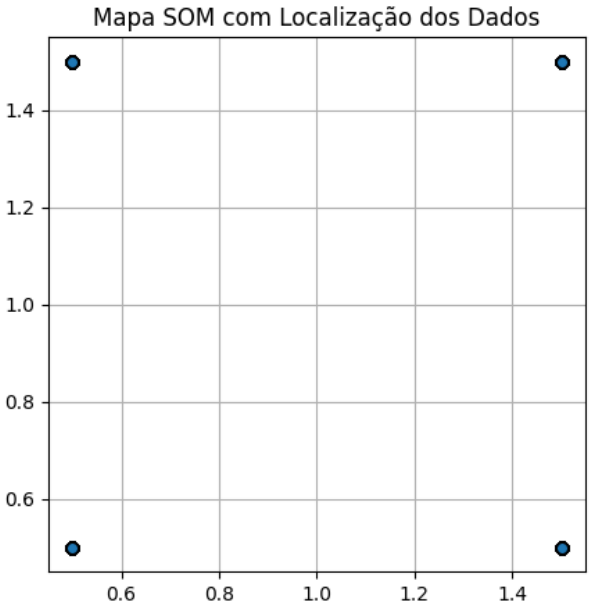
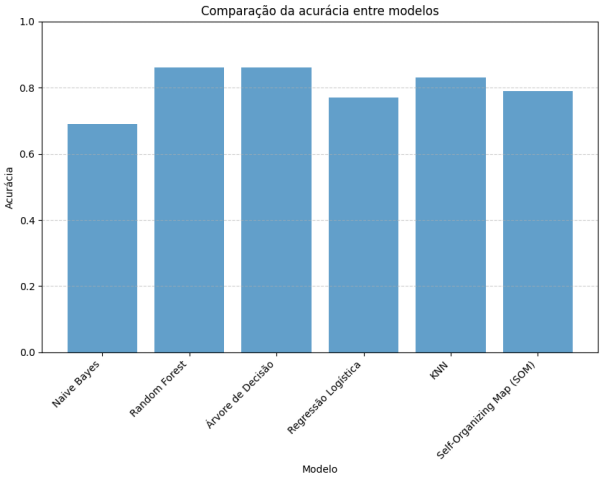


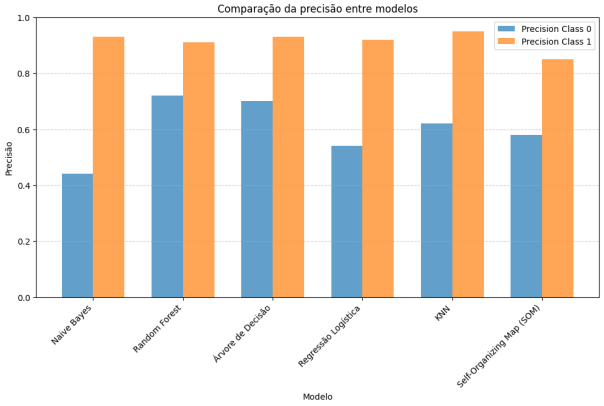
Figure 2: Comparação da acurácia entre modelos.



Desempenho Geral dos Modelos

O Random Forest destacou-se como o modelo com melhor desempenho geral, alcançando acurácia de 84,40% e F1-score de 0,73 para a classe 1 (domicílios com computador). Este modelo demonstrou robustez e equilíbrio entre precisão e recall, superando a Árvore de Decisão, que obteve acurácia de 83,40% e F1-score de 0,70 para a mesma classe. Apesar

Figure 3: Comparação da precisão entre modelos.



disso, a Árvore de Decisão mostrou-se uma alternativa interessante, oferecendo resultados competitivos com menor custo computacional, além de permitir a extração de regras interpretáveis.

O KNN apresentou uma acurácia de 83,31%, com F1-score de 0,70 para a classe 1 e 0,88 para a classe 0. Embora tenha sido competitivo, seu desempenho na identificação de domicílios com computador foi inferior aos modelos baseados em árvores, o que pode limitar sua aplicação em cenários mais variados.

A Regressão Logística, por sua vez, alcançou acurácia de 80,92% e F1-score de 0,65 para a classe 1 e 0,87 para a classe 0. Embora tenha apresentado resultados inferiores aos modelos mais complexos, destacou-se por sua simplicidade e interpretabilidade, sendo uma solução eficiente para problemas menos complexos.

O Naive Bayes obteve o pior desempenho, com acurácia de 71,93%. Seu F1-score foi de 0,59 para a classe 1 e 0,79 para a classe 0, evidenciando dificuldades em lidar com os padrões mais complexos dos dados. Este modelo mostrou uma tendência à superclassificação da classe 1, o que resultou em recall elevado (0,82), mas baixa precisão.

Por fim, o Self-Organizing Map (SOM) apresentou acurácia de 72,34%, com F1-score de 0,56 para a classe 1 e 0,80 para a classe 0. Embora tenha ficado atrás dos modelos supervisionados, o SOM demonstrou utilidade como ferramenta exploratória, permitindo uma análise visual da estrutura dos dados.

Conclusão

Com base nas métricas atualizadas, o Random Forest foi o modelo que apresentou o melhor desempenho geral, combinando alta precisão, recall e F1-score. Sua robustez se destacou por conseguir lidar de maneira eficaz com a complexidade do conjunto de dados, capturando relações importantes

entre os atributos e a variável-alvo. Além disso, sua capacidade de generalização o torna a escolha mais indicada para problemas que demandam alta confiabilidade nos resultados.

A Árvore de Decisão, por sua vez, apresentou desempenho próximo ao do Random Forest, sendo uma alternativa interessante para cenários onde a simplicidade do modelo e o menor custo computacional são fatores importantes. Apesar de sua estrutura mais simples, a Árvore de Decisão foi capaz de capturar padrões nos dados, priorizando atributos com alta relevância para a classificação, reforçando sua capacidade exploratória. Esses fatores foram refletidos nas regras geradas, evidenciando desigualdades estruturais relacionadas à inclusão digital.

O KNN também mostrou resultados competitivos, com boa precisão e recall na classe minoritária. No entanto, sua sensibilidade à distribuição dos dados pode limitar sua aplicação em casos com alta variabilidade.

A Regressão Logística, embora simples e interpretável, apresentou dificuldades para capturar os padrões mais complexos dos dados, enquanto o Naive Bayes, devido às suas suposições simplistas, mostrou-se inadequado para este problema.

O Self-Organizing Map (SOM), apesar de ficar atrás dos modelos supervisionados em desempenho, revelou-se útil para mapear e explorar a estrutura dos dados, sendo um complemento interessante para análises futuras.

No geral, o Random Forest é a melhor escolha para este conjunto de dados, com a Árvore de Decisão como uma alternativa eficiente e capaz de fornecer insights valiosos sobre os padrões capturados nos dados. O SOM, por sua vez, agrega valor ao oferecer visualizações úteis para compreender a organização dos dados.

4 CONSIDERAÇÕES FINAIS

Este trabalho utilizou técnicas de aprendizado de máquina para prever a posse de computadores em domicílios brasileiros com base em atributos socioeconômicos e domiciliares extraídos da PNS 2019. Foram testados e comparados seis modelos de classificação, sendo o Random Forest o que apresentou o melhor desempenho em termos de acurácia, precisão e F1-score.

Os resultados obtidos indicam que atributos domiciliares, como número de cômodos, tipo de material de construção e forma de abastecimento de água, têm um papel significativo na previsão da posse de computadores nas residências brasileiras. Isso reflete a correlação entre as condições de moradia e o acesso à tecnologia, evidenciando como aspectos socioeconômicos influenciam diretamente a inclusão digital no país.

No entanto, este estudo também apresenta limitações. Por exemplo, a base de dados da PNS 2019 possui alguns valores ausentes e atributos com dados incompletos, o que pode ter impactado a precisão dos modelos. Além disso, o desbalanceamento das classes foi um desafio tratado com técnicas como o SMOTE, mas ainda assim pode haver nuances não capturadas adequadamente pelos modelos. O desempenho do SOM, embora inferior aos modelos supervisionados, destacou-se como ferramenta exploratória, contribuindo para a compreensão da estrutura dos dados.

Para trabalhos futuros, sugerimos a inclusão de mais variáveis socioeconômicas, como renda per capita e grau de escolaridade, além da exploração de outros modelos mais complexos, como redes neurais. A replicação desta análise em diferentes contextos e regiões também pode contribuir para um entendimento mais profundo sobre a inclusão digital e a desigualdade tecnológica no Brasil.

Por fim, espera-se que os resultados deste estudo possam auxiliar na formulação de políticas públicas voltadas para a democratização do acesso à tecnologia em residências, promovendo maior igualdade digital no país.

5 UTILIZAÇÃO DO GPT

O ChatGPT foi usado para gerar o código tanto de pré-processamento, tanto para uso de um modelo, para gerar as tabelas no Latex, gerar o dicionário de dados presente no notebook, a partir do arquivo pns2019.csv e facilitar a busca por hiperparametros(ex: perguntar quais os hiperparametros que podem ser usados na RandomForest ao invés de buscar na documentação). Exemplos reais de uso(retirados das conversas):"Minha base de dados está desbalanceada, com a classe 0 com muito mais instâncias que a classe 1. Me de o código para usar o SMOTE para realizar um overfitting da classe 1". "Me de o código para usar uma regressão logísitca para classificar meu dados de treino". "Quais hiperparametros posso usar para melhorar a eficiência do meu Naive Bayes?".

6 CÓDIGO DESENVOLVIDO

O código desenvolvido está disponível no seguinte link: https://drive.google.com/file/d/1sG2_tnViGHX96bolJAlLA5JoplBqZnrZ/view?usp=sharing

REFERENCES

[1] Scikit-learn: Machine Learning in Python. *Documentation*, Available at: https://scikit-learn.org/stable/, Accessed: 2024-09-28.

[2] SMOTE: Synthetic Minority Over-sampling Technique. *imbalanced-learn*, Available at: https://imbalanced-learn.org/stable/references/generated/imblearn.over_sampling.SMOTE.html, Accessed: 2024-09-28.

[3] MinMaxScaler. *Scikit-learn Documentation*, Available at: https://scikit-learn.org/stable/modules/generated/sklearn.preprocessing.MinMaxScaler.html, Accessed: 2024-09-28.

[4] Naive Bayes Classifier. *Scikit-learn Documentation*, Available at: https://scikit-learn.org/stable/modules/naive_bayes.html, Accessed: 2024-09-28.

[5] Random Forest Classifier. *Scikit-learn Documentation*, Available at: https://scikit-learn.org/stable/modules/generated/sklearn.ensemble.

RandomForestClassifier.html, Accessed: 2024-09-28.
[6] Decision Tree Classifier. *Scikit-learn Documentation*, Available at: https://scikit-learn.org/stable/modules/generated/sklearn.tree.DecisionTreeClassifier.html, Accessed: 2024-09-28.
[7] Logistic Regression. *Scikit-learn Documentation*, Available at: https://scikit-learn.org/stable/modules/generated/sklearn.linear_model.LogisticRegression.html, Accessed: 2024-09-28.
[8] K-Nearest Neighbors (KNN). *Scikit-learn Documentation*, Available at: https://scikit-learn.org/stable/modules/generated/sklearn.neighbors.KNeighborsClassifier.html, Accessed: 2024-09-28.
[9] Self-Organizing Map (SOM). *MiniSom Library Documentation*, Available at: https://github.com/JustGlowing/minisom, Accessed: 2024-09-28.
[10] Accuracy, Precision, Recall, F1-Score. *Scikit-learn Metrics Documentation*, Available at: https://scikit-learn.org/stable/modules/model_evaluation.html, Accessed: 2024-09-28.
[11] Confusion Matrix. *Scikit-learn Metrics Documentation*, Available at: https://scikit-learn.org/stable/modules/generated/sklearn.metrics.confusion_matrix.html, Accessed: 2024-09-28.
[12] KNN Imputer. *Scikit-learn Documentation*, Available at: https://scikit-learn.org/stable/modules/generated/sklearn.impute.KNNImputer.html, Accessed: 2024-09-28.
[13] IBGE: Pesquisa Nacional por Amostra de Domicílios Contínua (PNAD). *Instituto Brasileiro de Geografia e Estatística*, 2019. Available at: https://www.ibge.gov.br Accessed: 2024-09-28.