Análise de Dados – Exploratória e ML

Sumário

[Introdução 4](#_Toc176173650)

[Objetivo do aplicativo 4](#_Toc176173651)

[Objetivo da Análise Exploratória de Dados 4](#_Toc176173652)

[Objetivo da Análise de Dados (para Aprendizado de Máquina) 4](#_Toc176173653)

[Levantamento dos dados na análise exploratória 4](#_Toc176173654)

[Busca dos dados 4](#_Toc176173655)

[Justificativa de uso 4](#_Toc176173656)

[Descrição da base de dados de trabalho 5](#_Toc176173657)

[Limpeza dos dados 5](#_Toc176173658)

[Condicionamento para alimentar o modelo de ML 5](#_Toc176173659)

[Condicionamento inicial 5](#_Toc176173660)

[Definição dos objetivos e das classes 6](#_Toc176173661)

[Definição dos modelos mais adequados para analisar os dados 6](#_Toc176173662)

[Descrição dos modelos selecionados 6](#_Toc176173663)

[Modelo 1: [Nome do Modelo] 7](#_Toc176173664)

[Modelo 2: [Nome do Modelo] 7](#_Toc176173665)

[Modelo n: [Nome do Modelo] 7](#_Toc176173666)

[Aplicação dos modelos selecionados 7](#_Toc176173667)

[Análise dos Resultados 7](#_Toc176173668)

[Modelo 1: [Nome do Modelo] 7](#_Toc176173669)

[Modelo 2: [Nome do Modelo] 8](#_Toc176173670)

[Modelo n [Nome do Modelo] 8](#_Toc176173671)

[Comparação Entre Modelos: 8](#_Toc176173672)

[Ajustes Necessários 8](#_Toc176173673)

[Identificação de Problemas: 8](#_Toc176173674)

[Mudanças na Base de Dados: 8](#_Toc176173675)

[Ajustes nos Modelos: 9](#_Toc176173676)

[Impacto das Modificações 9](#_Toc176173677)

[Modelo 1: [Nome do Modelo] 9](#_Toc176173678)

[Modelo 2: [Nome do Modelo] 9](#_Toc176173679)

[Modelo n [Nome do Modelo] 9](#_Toc176173680)

[Comparação Entre Modelos: 9](#_Toc176173681)

[Conclusão 9](#_Toc176173682)

Análise de Dados para *(Delfis!)*

Golden Age

Davi Yamada Ichihara, Thiago Levi

# Introdução

Nosso aplicativo visa amenizar o risco de problemas neurodegenerativos por meio de jogos, como sudoku, caça palavras, pequenos problemas aritméticos e outras atividades comprovadas cientificamente, de uma forma diária, para a preservação do cérebro e a memória de nosso usuário. Nosso público-alvo, é principalmente pessoas de terceira idade, pois são elas as mais vulneráveis a esses tipos de doenças.

## Objetivo do aplicativo

O aplicativo irá funcionar de uma maneira em que o usuário faça pequenas atividades por dia, como simples jogos, trívias ou exercícios matemáticos, todos comprovados cientificamente que ajudam a preservação da mente.

# Objetivo da Análise Exploratória de Dados e Aprendizado de Máquina

**Montar, analisar e limpar um conjunto de dados** relacionado ao aplicativo para ser utilizado em análise de dados exploratória e para fins de aprendizado de máquina (ML). Levantar dados de múltiplas classes sociais e faixas etárias por meio de uma pesquisa realizada pelo grupo com a finalidade de alimentar uma IA que irá determinar se uma pessoa é elegível ou não, com base em seu perfil, para ser um possível cliente

Os modelos que serão utilizados no aprendizado da máquina vão ser: Gaussian Bayes, Tree Decision (Entropy & Gini) e KNN.

# Levantamento dos dados na análise exploratória

# Os dados utilizados para a criação do aplicativo serão retirados de revistas cientificas como o *nature jornal*, e artigos de universidades como a Unicamp, que serão o embasamento para a escolha de jogos no aplicativo. Além disso, fizemos um levantamento de dados no forms (base com mais de 200 respostas) para o programa de elegibilidade de usuário, via Python.

# Descrição da base de dados de trabalho

Fizemos a análise via um formulário no forms, cujas colunas são:

**Qual a sua faixa etária?: String**

**Em qual estado você mora?: String**

**Qual sua faixa de renda familiar?: String**

**Qual é o seu nível de familiaridade com o uso de aplicativos em dispositivos móveis?: String**

**Em média, quanto tempo você usa o celular diariamente?: String**

**Você se interessaria em jogar palavra cruzada, sudoku ou jogos similares a esses diariamente?: Boolean**

**Você gostaria de utilizar o nosso aplicativo?: Boolean**

# Limpeza dos dados

O formulário foi montado de maneira na qual os dados não precisam ser tratados, já que todas as perguntas eram de escolha.

# Condicionamento para alimentar o modelo de ML

Os condicionamentos iniciais do modelo de ML foram:

* Normalização de dados (via Ordinal Encoder)
* Foi testado o Undersampling e o Oversampling, porém foi optado o Oversampling por conta de poucos registros, e a grande diferença entre as classes ‘Sim’ e ‘Não’

Foram testados também, o PCA e múltiplos tratamentos de outliers (apresentados futuramente), porém nenhum deles apresentou resultados positivos, então foram retirados.

# Definição dos objetivos e das classes

Os modelos precisam prever se a pessoa perguntada algumas perguntas, se ela pode ser ou não um futuro cliente para o nosso aplicativo.

A coluna resposta do modelo é “**Você gostaria de utilizar o nosso aplicativo?:”**, que tenta prever as classes “Sim” e “Não”

# Definição dos modelos mais adequados para analisar os dados

#### **Natureza dos Dados**

* Os dados fornecidos são apenas dados quantitativos discretos, isso significa que, todas as respostas do formulário consistem em escolhas pré-definidas, em vez de perguntas abertas. Isso facilita as análises estatísticas e previsões do modelo, pois todas as respostas são numéricas e padronizadas

**Complexidade dos Modelos**

Com base na complexidade dos dados, foram considerados tanto **modelos simples** quanto **modelos mais complexos**:

* **Modelos Simples**: Oferecem a vantagem de uma fácil implementação e interpretabilidade. Modelos como Naive Bayes e árvores de decisão são bons exemplos de algoritmos que, mesmo sendo simples, podem proporcionar resultados rápidos e precisos.
* **Modelos Mais Complexos**: Esses modelos podem capturar interações complexas entre variáveis, mas muitas vezes à custa de maior tempo de treinamento e dificuldade de interpretação. O K-Nearest Neighbors (KNN), por exemplo, é um modelo que pode identificar padrões não lineares e interações mais complexas entre as variáveis, mas requer ajustes mais cuidadosos e pode ser computacionalmente mais exigente.

**Objetivos de Classificação:**

* **Classificação Binária**: O modelo somente tenta prever 2 classes, “Sim” e “Não”

# 

# Descrição dos modelos selecionados

## Modelo 1:

## **Naive Bayes**

## **Precisão e Robustez:** O Naive Bayes tende a ser robusto em cenários de dados com grande número de features categóricas e em problemas onde a independência condicional (mesmo que simplificada) fornece resultados satisfatórios. Ele pode não ser o mais preciso em relação a modelos mais complexos, mas oferece resultados consistentes para muitos tipos de dados.

## **Interpretação e Explicabilidade:** Este modelo é altamente interpretável, já que a probabilidade de cada classe pode ser analisada diretamente com base nas probabilidades das features. Isso facilita a explicação dos resultados para não especialistas.

## **Desempenho Computacional:** O Naive Bayes é extremamente eficiente em termos de tempo de treinamento e inferência. Seu baixo custo computacional o torna ideal para cenários onde o tempo de resposta é crítico.

## **Capacidade de Generalização:** Embora funcione bem com dados balanceados, pode enfrentar dificuldades quando aplicado a dados altamente desbalanceados sem ajustes específicos (como balanceamento de classes).

## Modelo 2:

## **Árvore de Decisão**

## **Natureza dos Dados**: Capaz de lidar tanto com dados categóricos quanto contínuos, sem a necessidade de transformações complexas.

## **Complexidade do Modelo**: Moderada. A árvore de decisão é mais complexa que o Naive Bayes, já que pode capturar interações entre variáveis, mas ainda assim é fácil de interpretar.

## **Objetivo de Classificação**: Funciona bem tanto para classificação binária quanto para multi-classe. A árvore de decisão oferece flexibilidade e a possibilidade de ser ajustada por meio de poda para evitar overfitting.

## Modelo 3:

## **3. K-Nearest Neighbors (KNN)**

## **Natureza dos Dados**: Pode lidar tanto com dados categóricos quanto contínuos, mas as variáveis contínuas podem precisar ser padronizadas para garantir que as distâncias calculadas sejam significativas.

## **Complexidade do Modelo**: Considerado um modelo mais complexo em termos de computação, especialmente para grandes conjuntos de dados. KNN exige a definição de k, o que afeta diretamente o desempenho do modelo.

## **Objetivo de Classificação**: Adequado para problemas binários e multi-classe. No entanto, KNN pode sofrer em problemas de classes desbalanceadas, onde a classe mais frequente pode dominar as previsões.

# Aplicação dos modelos selecionados

* Divisão dos dados:
  + A Divisão dos dados está sendo de 80/20, com a seed padrão 42

Texto

Descrição gerada automaticamente

* Os treinos dos modelos são:
  + Naive Bayes:
    - Uma validação simples:
    - Tela de computador com texto preto sobre fundo branco

      Descrição gerada automaticamente
    - E a validação cruzada, usando o comando KFOLD. O programa pega o melhor fold (com a melhor média de f1-score), e depois o treina para atingir o melhor resultado
    - Texto

      Descrição gerada automaticamente
    - Texto

      Descrição gerada automaticamente
  + Tree Decision:
    - Validação simples, tanto para entropy, quanto para gini
    - Tela de computador com texto preto sobre fundo branco

      Descrição gerada automaticamente
    - Tela de celular com aplicativo aberto

      Descrição gerada automaticamente
    - E o uso do GridSearchCV, que pega os melhores parâmetros do classificador da árvore em relação a base de dados, e depois a treina para obter os melhores resultados
    - Texto

      Descrição gerada automaticamente
    - Pego os melhores parâmetros adquiridos pelo GridSearchCV, e treino um novo modelo a partir deles
    - Texto

      Descrição gerada automaticamente
    - Tela de computador com texto preto sobre fundo branco

      Descrição gerada automaticamente
  + KNN:
    - Validação simples:
    - 
    - Usando o GridSearchCV para encontrar os melhores parâmetros, mesmo objetivo do TreeDecision:
    - 
    - Treinando um novo modelo a partir dos melhores parâmetros obtidos:
    - Tela de computador com letras e números em fundo preto

      Descrição gerada automaticamente
    - Texto

      Descrição gerada automaticamente

# Análise dos Resultados

Análise dos 3 modelos (Bayes, Tree e KNN) e comparar seus resultados.

* **Naive Bayes:**
  + **Validação simples**
    - Tela preta com letras brancas

      Descrição gerada automaticamente
    - Gráfico

      Descrição gerada automaticamente
  + **Validação cruzada**
    - Tela de celular com texto preto sobre fundo branco

      Descrição gerada automaticamente
    - Gráfico

      Descrição gerada automaticamente

**Conclusão do modelo Naive Bayes:**

**Cross Validation** (validação cruzada) e **validação simples** são métodos para avaliar a performance de um modelo de machine learning, mas são diferentes na forma como os dados são divididos.

1. **Validação Simples**:
   * Você divide o conjunto de dados em duas partes: treinamento e teste. Treina o modelo em uma parte e testa na outra.
   * É rápido e direto, mas o resultado pode variar dependendo da forma como os dados são divididos.
2. **Cross Validation (Validação Cruzada)**:
   * Os dados são divididos em várias "dobras" (partições). O modelo é treinado e testado várias vezes, cada vez com uma partição diferente como conjunto de teste e o restante como conjunto de treinamento.
   * O resultado é uma média do desempenho em todas as "dobras". Isso proporciona uma avaliação mais robusta, evitando problemas de variação na divisão dos dados.

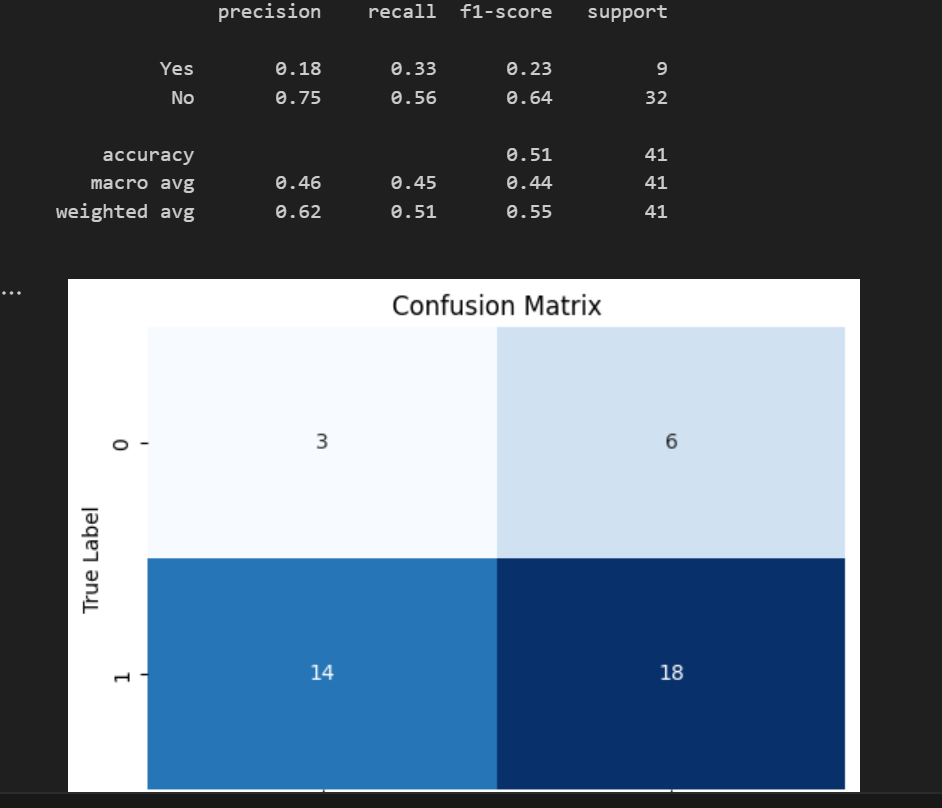
**Por que os resultados podem ser os mesmos?**

Se você está obtendo resultados semelhantes para a validação simples e a validação cruzada, isso pode acontecer em algumas situações:

* **Tamanho do conjunto de dados**: Se os dados forem muito homogêneos ou houver muitos exemplos, pode não haver muita variação nos resultados, mesmo com diferentes divisões dos dados.
* **Distribuição equilibrada**: Se os conjuntos de treinamento e teste estiverem bem equilibrados em termos de distribuição das classes (para classificadores), o resultado da validação simples pode ser próximo ao da validação cruzada.
* **Tamanho da amostra de validação cruzada**: Se estiver usando poucas "dobras" (como uma cross-validation com 2 partes, por exemplo), pode acabar replicando a divisão do conjunto de dados da validação simples.

**Naive Bayes**

**Por que "naive" (ingênuo)?** Ele é chamado de ingênuo porque faz a suposição (normalmente incorreta) de que todas as características (ou variáveis) são independentes umas das outras. Apesar dessa simplificação, o Naive Bayes é extremamente eficaz para nossa base, pois ela tem apenas 2 classes, o que ajuda na avaliação.

* **Decision Tree:**
  + **Validações simples**
    - **Gini:**
      * 
    - **Entropy:**
      * Interface gráfica do usuário

        Descrição gerada automaticamente com confiança baixa

**GridSearchCV:**

* + - * Interface gráfica do usuário, Aplicativo, Calendário

        Descrição gerada automaticamente

**Conclusão do modelo Tree Decision (Árvore de decisão):**

Árvore de decisão é um modelo de machine learning utilizado para tarefas de classificação e regressão. Ela divide os dados em subconjuntos menores com base em condições lógicas, organizando essas divisões em uma estrutura de árvore com nós e ramificações. O objetivo é criar um modelo que faça previsões a partir das condições encontradas nos dados.

**Como funciona:**

A árvore de decisão faz divisões nos dados com base nas variáveis de entrada. Para isso, ela avalia os valores de uma métrica que determina a "qualidade" da divisão, como a entropia ou o índice de Gini.

**Métricas de divisão: Entropia vs Índice de Gini**

**1. Entropia:**

A entropia mede o nível de "desordem" ou "impureza" de um conjunto de dados. O objetivo é reduzir a entropia em cada divisão, criando subconjuntos mais puros (ou seja, onde os dados de cada classe estão mais concentrados).

**2. Índice de Gini:**

O índice de Gini mede a probabilidade de uma amostra ser classificada incorretamente se ela for rotulada aleatoriamente de acordo com a distribuição de classes no conjunto. Assim como a entropia, o índice de Gini visa criar subconjuntos mais puros.

**Diferenças:**

* Entropia tende a ser mais sensível a mudanças nos dados, resultando em uma árvore ligeiramente diferente quando comparada ao Gini.
* Índice de Gini é mais simples de calcular e muitas vezes é mais rápido computacionalmente.

**GridSearchCV**

O GridSearchCV é uma ferramenta de otimização de hiperparâmetros usada em modelos de machine learning. Ele é baseado no conceito de validação cruzada (cross-validation) e funciona testando todas as combinações possíveis de um conjunto de parâmetros fornecido pelo usuário.

Como funciona:

Parâmetros do modelo: Primeiro, você define os hiperparâmetros que deseja otimizar. No caso de uma árvore de decisão, pode incluir:

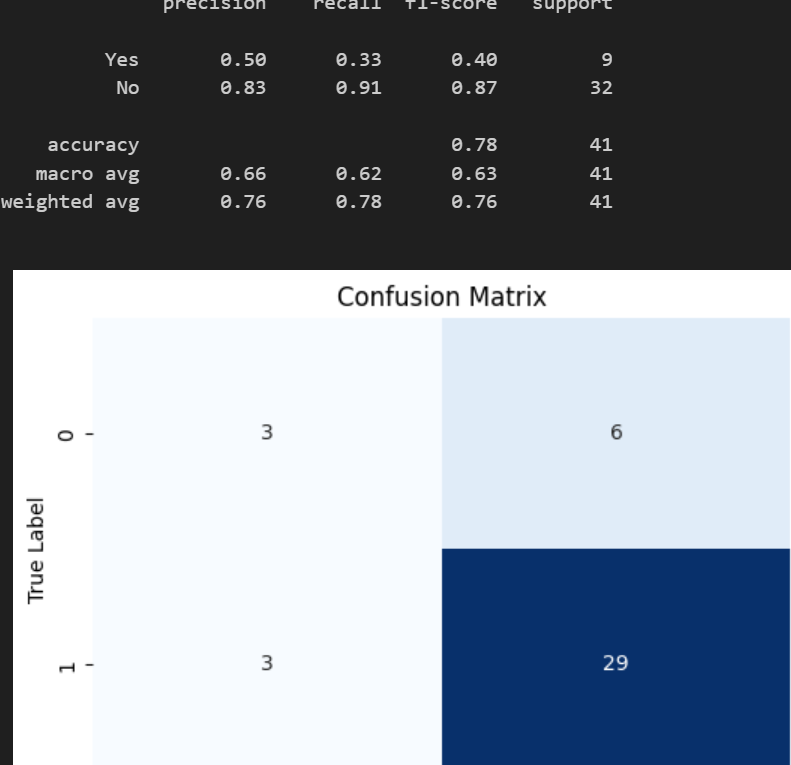
* **max\_depth**: Profundidade máxima da árvore. Controla o quanto a árvore pode crescer. Um valor menor evita o **overfitting**.
* **min\_samples\_split**: Número mínimo de amostras necessário para dividir um nó. Um valor maior ajuda a evitar divisões insignificantes, reduzindo o
* **overfitting**.
* **criterion**: Métrica usada para medir a qualidade da divisão:
  + 'gini': Usa o índice de Gini.
  + 'entropy': Usa a entropia.
* **min\_samples\_leaf**: Número mínimo de amostras que cada folha (nó terminal) deve ter. Isso ajuda a suavizar a árvore, prevenindo overfitting.
* **max\_features**: Número máximo de características a serem consideradas para cada divisão. Pode ser 'auto', 'sqrt', 'log2' ou um número específico.
* **splitter**: Estratégia de divisão. Pode ser:
  + 'best': Escolhe a melhor divisão.
  + 'random': Escolhe divisões aleatórias entre as melhores opções.

**Alguns motivos para a árvore de decisão estar dando resultados ruins podem ser:**

**Overfitting:**

* As árvores de decisão tendem a criar modelos muito complexos, capturando o ruído nos dados de treinamento, o que leva ao **overfitting**. Uma árvore de decisão não limitada pode crescer até ter folhas com apenas uma amostra, memorizando os dados de treinamento, mas generalizando mal para novos dados.

**Instabilidade:**

* As árvores de decisão são **sensíveis a pequenas mudanças nos dados**. Uma pequena alteração no conjunto de treinamento pode resultar em uma árvore completamente diferente, devido à forma como os nós são divididos.
* **KNN:**
  + **Validação simples:**
    - 
  + **GridSearchCV**
    - Uma imagem contendo Interface gráfica do usuário

      Descrição gerada automaticamente

**Conclusão sobre o modelo KNN (K-Nearest Neightbors)**

O K-Nearest Neighbors (KNN) é um algoritmo simples e intuitivo usado para classificação e regressão. Sua ideia principal é que, para prever a classe ou valor de uma amostra, o modelo olha para as amostras mais próximas (seus "vizinhos") e toma uma decisão com base nas classes ou valores desses vizinhos.

1. Distância: O KNN calcula a distância entre a amostra de teste e todas as amostras de treinamento. A distância mais comumente usada é a euclidiana, mas outras métricas como a distância manhattan também podem ser usadas.
2. Número de vizinhos (K): O valor de K determina quantos vizinhos mais próximos serão considerados para fazer a previsão. Para classificação, o modelo atribui a classe mais comum entre os vizinhos. Para regressão, ele calcula a média dos valores dos vizinhos.
3. Ponderação: Você pode dar pesos diferentes para os vizinhos, onde os vizinhos mais próximos têm mais influência. Isso pode ser feito com o parâmetro weights.

Vantagens:

* Simplicidade e fácil de entender.
* Funciona bem em dados de baixa dimensionalidade.

Desvantagens:

* O desempenho pode ser impactado por dados desbalanceados ou de alta dimensionalidade.
* É computacionalmente intensivo, pois requer o cálculo de distâncias para todas as amostras de treinamento a cada nova previsão.

Hiperparâmetros do KNN para GridSearchCV

Aqui estão os principais hiperparâmetros do KNN que podem ser ajustados usando o GridSearchCV:

**n\_neighbors**: Número de vizinhos mais próximos a serem considerados para a classificação ou regressão.

**weights**: Função de peso a ser usada na combinação dos vizinhos.

* + 'uniform': Todos os vizinhos têm o mesmo peso.
  + 'distance': Vizinhos mais próximos têm maior peso, inversamente proporcional à distância.

**algorithm**: Algoritmo a ser usado para calcular os vizinhos mais próximos:

* + 'auto': O algoritmo é selecionado automaticamente com base no tipo de dados.
  + 'ball\_tree': Usa uma estrutura de árvore para buscar vizinhos.
  + 'kd\_tree': Usa uma árvore k-d para busca de vizinhos.
  + 'brute': Busca todos os pontos.

**leaf\_size**: Tamanho da folha da árvore, que afeta o tempo de treinamento e a eficiência da consulta.

**metric**: A métrica a ser utilizada para calcular a distância entre pontos.

**p**: O parâmetro p define a norma a ser utilizada.

**Sensível ao Ruído e Outliers:**

* O KNN pode ser muito sensível a **outliers** e ao **ruído** nos dados, pois esses pontos podem alterar as distâncias e, consequentemente, a classificação.

# Conclusões sobre os modelos

É possível perceber claramente que o modelo Naive Bayes é o mais eficaz entre os três. Embora o foco esteja no F1-Score da classe "Sim", isso não significa que as outras classes devam ser desconsideradas. Nos modelos de Árvore de Decisão e KNN, o F1-Score da classe "Não" se encontra em um estado crítico, com o modelo tendendo a favorecer a classe "Sim". Isso evidencia o problema do desbalanceamento de classes durante o treinamento. Portanto, ao aprimorar a base de dados, o modelo de referência será, principalmente, o Naive Bayes.

## Mudanças na Base de Dados:

**Ajustes na base realizadas:**

**Tratamento de Outliers:**

Texto

Descrição gerada automaticamente

Gráfico, Gráfico de barras

Descrição gerada automaticamente com confiança média

Pode-se ver os possíveis outliers, apresentados nos gráficos acima, de todas as colunas

**IQR**: faz uma medida de dispersão que ajuda a identificar a distribuição dos dados. Ele é calculado subtraindo o primeiro quartil (Q1, que é o valor abaixo do qual estão 25% dos dados) do terceiro quartil (Q3, que é o valor abaixo do qual estão 75% dos dados)

Texto

Descrição gerada automaticamente

**Z-Score**: é uma métrica que indica quantos desvios padrão um valor está distante da média de um conjunto de dados. Ele é usado para padronizar e comparar dados em distribuições diferentes, facilitando a análise de como um dado específico se comporta em relação ao restante.

Tela de celular com aplicativo aberto

Descrição gerada automaticamente

**Quartil Removal**: é uma técnica de pré-processamento de dados utilizada para remover **outliers** (valores atípicos) por meio da análise de quartis. Os quartis dividem um conjunto de dados ordenados em quatro partes iguais: o **primeiro quartil (Q1)** representa o 25º percentil, a **mediana (Q2)** é o 50º percentil, e o **terceiro quartil (Q3)** corresponde ao 75º percentil. O **Intervalo Interquartil (IQR)** é calculado como

Texto

Descrição gerada automaticamente

**Isolation Forest**: é um algoritmo de detecção de outliers que utiliza árvores de decisão. Ele funciona construindo várias árvores a partir de subconjuntos aleatórios de dados, realizando divisões aleatórias para isolar pontos. A profundidade de isolamento determina a pontuação de anomalia, onde valores mais altos indicam maior probabilidade de serem outliers.

Texto

Descrição gerada automaticamente

**Local Outlier Factor**: é um algoritmo de detecção de outliers que avalia a densidade local de pontos em relação aos seus vizinhos. Ele é eficaz na identificação de outliers em conjuntos de dados com densidades variadas, permitindo ajustes na escolha do número de vizinhos. Embora seja robusto e não exija rótulos, pode ser computacionalmente intensivo e sensível à escolha do parâmetro k. Suas aplicações incluem detecção de fraudes, análise de dados e monitoramento de sistemas.

Texto

Descrição gerada automaticamente

**Mahalanobis Distance**: é uma medida que quantifica a distância entre um ponto e uma distribuição, considerando a covariância dos dados. Ela é valiosa para detectar outliers e analisar dados multivariados, oferecendo vantagens como adaptabilidade e precisão. No entanto, requer o cálculo da matriz de covariância e pode ser sensível à presença de outliers. Suas aplicações abrangem desde detecção de fraudes até reconhecimento de padrões em aprendizado de máquina.

Texto

Descrição gerada automaticamente

**Tukey’s Fences**: é uma técnica eficaz para identificar outliers em conjuntos de dados, utilizando os quartis e o intervalo interquartil (IQR). A abordagem é robusta, simples e intuitiva, sendo amplamente utilizada em análise exploratória de dados. Embora tenha algumas limitações, como a escolha do valor de k’s e eficácia em conjuntos de dados pequenos, suas aplicações em detecção de anomalias e limpeza de dados a tornam uma ferramenta valiosa para analistas e cientistas de dados.

Texto

Descrição gerada automaticamente

**Capping/Winsorization**:é uma técnica de tratamento de outliers que limita a influência de valores extremos ao substituí-los por limites definidos, em vez de removê-los completamente. Essa abordagem conserva dados, melhora a estabilidade de análises estatísticas e reduz o viés em modelos. Apesar de algumas desvantagens, como a possível perda de informação e a subjetividade na escolha dos limites, capping/winsorization é uma ferramenta valiosa em análise estatística, modelagem preditiva e ciência de dados.

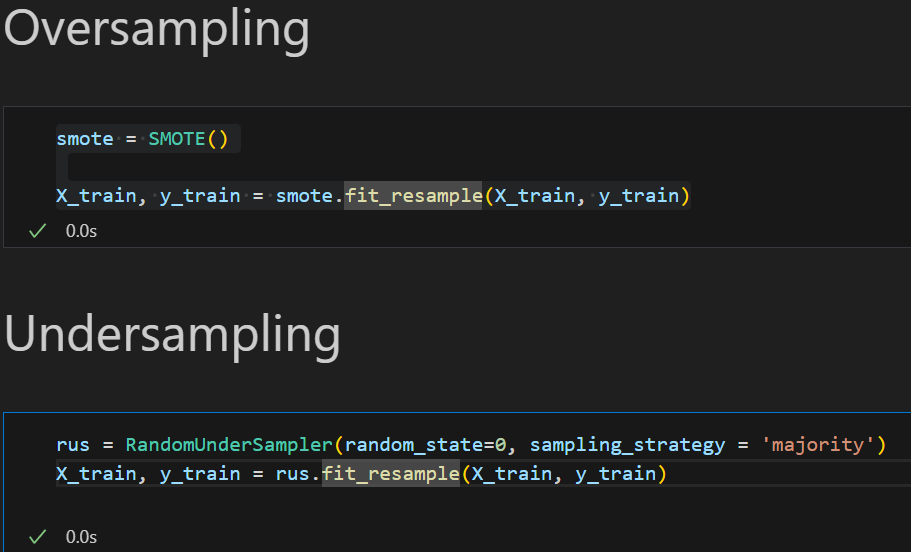
Texto

Descrição gerada automaticamente

Também foi testado o **PCA**: é uma técnica eficaz para reduzir a dimensionalidade de conjuntos de dados complexos, preservando a variabilidade dos dados. Ela funciona através da centralização dos dados, cálculo da matriz de covariância, extração de autovalores e autovetores, e projeção dos dados nas componentes principais

Texto

Descrição gerada automaticamente

O **Undersampling** e o **Oversampling**: **Undersampling** reduz a classe majoritária para equilibrar a distribuição das classes, enquanto **Oversampling** aumenta a classe minoritária. Ambas as técnicas visam melhorar o desempenho do modelo em conjuntos de dados desbalanceados, mas apresentam vantagens e desvantagens distintas. A escolha entre elas deve ser baseada nas características do problema e dos dados disponíveis. 

E o Standart Scaler: O **Standard Scaler** é uma técnica de normalização que transforma características em um conjunto de dados para que tenham média zero e desvio padrão um. É útil em algoritmos de aprendizado de máquina sensíveis à escala, melhorando a convergência e a performance do modelo

Texto

Descrição gerada automaticamente

# Impacto das Modificações:

Durante a análise dos modelos Naive Bayes, Decision Tree e KNN, foram aplicadas diversas técnicas de pré-processamento para avaliar seu impacto nos resultados. Entre as técnicas utilizadas, o **oversampling** foi a única que aumentou significativamente a performance do modelo Naive Bayes, enquanto as demais técnicas não trouxeram o mesmo impacto relevante. Abaixo está a descrição de como essas técnicas influenciaram os modelos:

**1. Oversampling:**

O oversampling, utilizado para balancear as classes desbalanceadas no conjunto de dados, que são muitas, já que a proporção da pesquisa, é aproximadamente 80% dos resultados para a classe “Sim”, e 20% para a classe “Não”, o que causa viés na hora do treino da IA. Por isso, a melhoria resultou em um aumento significativo no F1 – Score da classe “Não” do modelo Naive Bayes. Isso ocorre porque o Naive Bayes assume independência entre as features, e ao aumentar a representatividade das classes minoritárias, o modelo teve melhor capacidade de generalização.

O modelo Naive Bayes apresentou o melhor desempenho em relação ao Decision Tree e ao KNN com a aplicação dessa técnica.

**2. Undersampling:**

O undersampling foi testado para reduzir o número de exemplos das classes majoritárias, mas não apresentou melhoria significativa em nenhum dos modelos avaliados, incluindo o Naive Bayes, K-Nearest Neighbors (KNN) e Árvore de Decisão. No caso do Naive Bayes, essa técnica pode ter removido informações importantes, afetando negativamente a capacidade de predição. Para o KNN e a Árvore de Decisão, a redução das amostras pode ter levado à perda de dados representativos, resultando em um modelo menos robusto e menos capaz de capturar padrões significativos.

**3. Standard Scaler, IQR, Z-Score, Quartil Removal:**

As técnicas de normalização e remoção de outliers, como Quartil Removal, não impactaram os resultados do Naive Bayes e prejudicaram o desempenho do KNN e da Árvore de Decisão. Isso pode ser explicado pela natureza do Naive Bayes, que lida bem com dados não escalados devido à sua suposição de independência condicional entre variáveis. Para o KNN, que é sensível à escala das características, a normalização não teve o efeito desejado, pois a distribuição dos dados não foi alterada de forma benéfica. No caso da Árvore de Decisão, a normalização não é tão crucial, mas a remoção de outliers pode ter eliminado amostras importantes, prejudicando o aprendizado do modelo.

**4. Outlier Detection (Isolation Forest, Local Outlier Factor, Mahalanobis Distance, Tukey's Fences, Capping):**

As técnicas de detecção e tratamento de outliers, incluindo Isolation Forest, Local Outlier Factor, Mahalanobis Distance, Tukey's Fences e Capping, também não impactaram positivamente o desempenho do modelo Naive Bayes, KNN ou Árvore de Decisão. O Naive Bayes tende a ser robusto a outliers, dependendo da distribuição dos dados, e, portanto, não se beneficiou dessas técnicas. O KNN, por outro lado, pode ter sido afetado negativamente pela presença de outliers, mas a remoção ou tratamento inadequado desses pontos extremos pode ter levado a uma redução de informação crítica, dificultando a classificação. A Árvore de Decisão também não se beneficiou das técnicas de detecção de outliers, pois a remoção de amostras pode ter gerado divisões menos informativas e prejudicado o modelo.

**Conclusão:**

A técnica de oversampling foi a mais eficaz para melhorar a performance do modelo Naive Bayes, destacando-se como o melhor entre os três modelos. Isso se deve ao fato de o oversampling proporcionar uma maior representatividade das classes minoritárias, o que é benéfico para o Naive Bayes. Para o KNN e a Árvore de Decisão, o oversampling também pode ter contribuído para a melhoria do desempenho ao garantir que as classes minoritárias fossem adequadamente representadas. As outras técnicas, como undersampling e normalizações, não trouxeram ganhos relevantes para nenhum dos modelos, possivelmente devido à natureza dos modelos e suas suposições subjacentes.

# Conclusão Geral

A análise realizada revelou insights importantes sobre a eficácia dos modelos aplicados, incluindo Naive Bayes, K-Nearest Neighbors (KNN) e Árvore de Decisão, na tarefa de classificação. A técnica de **oversampling** demonstrou ser a mais eficaz, especialmente para o modelo Naive Bayes, ao melhorar a representatividade das classes minoritárias. Em contrapartida, técnicas como **undersampling** e várias abordagens de normalização e detecção de outliers não mostraram ganhos significativos e, em alguns casos, prejudicaram o desempenho dos modelos. O que significa, que mais em diante, com um treinamento melhor e uma pesquisa mais abrangente para o treinamento mais equilibrado do modelo ele pode ter uma melhora significativa para a predição de clientes possíveis.