Análise de Dados para *Olimpo*

Apolo

Sofia Elisabete Rosa Domingues – n°19

2°G Tech

Sumário

[Introdução 3](#_Toc181772524)

[Objetivo do aplicativo 3](#_Toc181772525)

[Objetivo do trabalho de Análise de Dados 3](#_Toc181772526)

[Levantamento dos dados 3](#_Toc181772527)

[Busca dos dados 3](#_Toc181772528)

[Descrição da base de dados encontrada 3](#_Toc181772529)

[Limpeza dos dados 4](#_Toc181772530)

[Condicionamento para alimentar o modelo de ML 6](#_Toc181772531)

[Definição dos objetivos e das classes 6](#_Toc181772532)

[Definição dos modelos mais adequados para analisar os dados 6](#_Toc181772533)

[Natureza dos dados: 6](#_Toc181772534)

[Complexidade dos Modelos: 6](#_Toc181772535)

[Diferença entre ambos: 6](#_Toc181772536)

[Descrição dos modelos selecionados 7](#_Toc181772537)

[Modelo 1: Gaussian 7](#_Toc181772538)

[Modelo 2: Tree Decision 7](#_Toc181772539)

[Modelo 3: K-Neighbours 7](#_Toc181772540)

[Análise dos Modelos Aplicados 7](#_Toc181772541)

[Divisão dos Dados 7](#_Toc181772542)

[Treinamento dos Modelos 8](#_Toc181772543)

[Métricas de Avaliação 8](#_Toc181772544)

[Análise dos Resultados 8](#_Toc181772545)

[Comparação Entre Modelos 9](#_Toc181772546)

[Ajustes Necessários 9](#_Toc181772547)

[Impacto das Modificações 9](#_Toc181772548)

[Conclusão 10](#_Toc181772549)

# Introdução

O Olimpo surgiu da iniciativa dos integrantes do primeiro ano do Apolo de resolver o problema da escassez de cooperação e integração em comunidades. Nosso público-alvo são maiores de idade. Essa escolha foi baseada no fato de que essas pessoas apresentam necessidades da vida adulta, um maior senso comunitário e um alto uso de redes sociais de acordo com o IBGE em 2022.

## Objetivo do aplicativo

O Olimpo almeja aumentar a organização comunitária dentro de comunidades, incentivando a coesão social por meio de chats e publicações dentro do próprio aplicativo. Ademais, o Olimpo visa incentivar a economia ambiental e local dentro das comunidades que poderão anunciar seus itens usados. Dessa forma, o Olimpo impactará de maneira positiva a vida de seus usuários, seja esse impacto ambiental, governamental ou social.

Recomenda-se citar aqui que os integrantes do 1° ano já fizeram a especificação do aplicativo e que os detalhes sobre essa parte podem ser encontrados no material desenvolvido por eles para documentação do trabalho.

# Objetivo do trabalho de Análise de Dados

O objetivo do trabalho de análise de dados para o Olimpo é transformar grandes volumes de informações brutas em insights acionáveis que podem orientar decisões estratégicas para determinar se uma pessoa é um cliente em potencial para o Olimpo ou não. Através da coleta, organização e interpretação de dados, a análise permite identificar padrões e tendências de perfis, proporcionando uma compreensão mais profunda dos processos e comportamentos. Essa abordagem analítica facilita a resolução de problemas, otimização de processos e a formulação de estratégias baseadas nessas análises, ajudando nossa equipe, o Apolo, a tomar decisões informadas e a alcançar seus objetivos com maior eficácia.

# Levantamento dos dados

## Busca dos dados

Variás pesquisas foram realizadas com nosso publico-alvo, moradores de comunidades, para entender suas necessidades e prioridades dentro de seu convivio social. A equipe do Apolo escolheu criar um formulário próprios com perguntas especificas com maior relevância para o grupo e treinamento de nossa IA, que visa prever possiveis clientes. O formulário recebeu aproximadamente 100 respostas.

# Descrição da base de dados encontrada

1. **Id** (*int*): Identificador único de cada resposta.
2. **Hora de início** (*datetime*): Data e hora em que a resposta começou.
3. **Hora de conclusão** (*datetime*): Data e hora em que a resposta foi concluída.
4. **Email** (*string*): Email do participante (anônimo).
5. **Nome** (*float*): Nome do participante (valores ausentes são armazenados como NaN).
6. **Termo de Anonimato** (*string*): Indica se o termo de anonimato foi aceito.
7. **Em que cidade você vive atualmente?** (*string*): Cidade onde o participante vive.
8. **Qual é seu tipo da sua moradia?** (*string*): Tipo de moradia (ex., apartamento ou casa).
9. **Com quantas pessoas você divide sua casa/apartamento?** (*string*): Número de pessoas com quem divide a moradia (contém textos como "Nenhuma").
10. **Nível de satisfação com a comunicação** (*int*): Escala numérica de satisfação com a comunicação na comunidade.
11. **Frequência de contato com vizinhos** (*string*): Frequência de contato com vizinhos (ex., "Raramente").
12. **Acredita na importância da comunicação comunitária?** (*string*): Opinião sobre a importância da comunicação na comunidade.
13. **Uso de plataforma de comunicação** (*string*): Indica se o participante usa alguma plataforma comunitária.
14. **Nível de satisfação com a plataforma** (*float*): Satisfação com a plataforma comunitária (ausências registradas como NaN).
15. **Faixa etária** (*string*): Faixa etária do participante (ex., "Maior ou igual a 43 anos").
16. **Qualidade de vida** (*string*): Avaliação subjetiva da qualidade de vida (ex., "Boa" ou "Ótima").
17. **Situação no mercado de trabalho** (*string*): Situação de trabalho atual (ex., "Empregado").
18. **Renda familiar mensal** (*string*): Faixa de renda mensal (ex., "Classe C").
19. **Animais de estimação** (*string*): Indica se o participante possui pets.
20. **Implementação do aplicativo** (*string*): Intenção de implementar o aplicativo sugerido.

A análise de dados e código de ML foi feita por um código Python próprio criado pelos integrantes do Apolo. O repositório GitHub em que o modelo está armazenado pode ser encontrado através desse link:

# Limpeza dos dados

A base de dados coletada possui um total de 20 colunas, sendo dessas:

* 12 pertencem a X (outras)
* 1 pertence a Y (implementa\_app)
* 6 não possuem relevânca para X ('ID', 'Hora de início', 'Hora de conclusão', 'Email', 'Nome', 'Termo de Anonimato', 'Nivel de Satisfação com a Plataforma Utilizada')

Após a análise macro da base, começamos a limpar e extrair somente as informações necessárias para o nosso modelo, seguindo o seguinte passo a passo:

1. Primeiro, renomeamos todas as colunas para que fosse mais fácil na hora de utiliza-las dentro do modelo. Texto

   Descrição gerada automaticamente
2. Após renomear os campos, notamos a necessidade de limpar completamente a coluna “Cidade”, pois ela não foi coletada utilizando um padrão de escrita, portanto, existiam diversas respostas iguais com formatações diferentes. Texto

   Descrição gerada automaticamente
3. Depois disso, removi a parte descritiva da coluna “Frequência de contato com vizinhos**”** e extraí somente a parte relevante com as categorias (‘Ocasionalmente’, ‘Raramente’, etc). Texto

   Descrição gerada automaticamente
4. Por fim, removi as colunas indesejadas, como descritas no início dessa seção. Interface gráfica do usuário, Texto, Site

   Descrição gerada automaticamente

Condicionamento para alimentar o modelo de ML

Os condicionamentos iniciais do modelo de ML foram:

* Normalização de dados (via zscore)
* Oversampling, pois a base de dados é muito pequena, e não faria sentido a aplicação do undersampling

É importante notar que foi testado o PCA, porém foi descartado devida a pequena quantidade de features (colunas) apresentadas na base.

# Definição dos objetivos e das classes

O objetivo do modelo é prever se uma pessoa é ou não potencial usuário do aplicativo Olimpo baseado nas informações dadas pelo usuário.

# Definição dos modelos mais adequados para analisar os dados

## Natureza dos dados:

Apesar dos dados fornecidos serem, tanto quantitativos, como qualitativos, após o tratamento de toda a base, todos os dados são transformados em dados quantitativos discretos, ou seja, são números que representam valores e quantidades, que geralmente são usados para analisar padrões, médias e correlações

## Complexidade dos Modelos

**Modelos simples:** são algoritmos de aprendizado de máquina que têm uma estrutura básica e são rápidos e fáceis de interpretar. Exemplos incluem o **Naive Bayes**. Ele é ideal para dados pequenos ou situações em que interpretabilidade e rapidez são mais importantes do que a precisão.

**Modelos complexos**: têm estruturas mais sofisticadas, como as **redes neurais profundas** ou **florestas aleatórias**. Esses modelos conseguem capturar relações complexas nos dados, mas exigem mais poder computacional e podem ser difíceis de interpretar.

## Diferença entre ambos:

* **Desempenho**: Modelos complexos tendem a ter maior precisão, especialmente em dados grandes e não lineares. Modelos simples funcionam bem em dados estruturados ou problemas onde as relações são mais diretas.
* **Interpretação**: Modelos simples são mais fáceis de entender e interpretar, enquanto modelos complexos são como uma "caixa preta", tornando difícil saber como as decisões são feitas.
* **Tempo e Recursos**: Modelos simples são rápidos e exigem menos recursos. Modelos complexos demandam mais tempo e poder computacional.

Essencialmente, escolher entre um modelo simples e um complexo depende do equilíbrio entre precisão, interpretabilidade e recursos disponíveis.

**Objetivos de Classificação:**

* Classificação Binária: Todos os modelos somente tentam prever 2 classes, “Sim” e “Não”

Parte superior do formulário

Parte inferior do formulário

# Descrição dos modelos selecionados

## Modelo 1: Gaussian

* Esse modelo pertence à família de classificadores Naive Bayes, que se baseiam no **Teorema de Bayes** e assumem independência entre as variáveis. O GaussianNB é específico para dados contínuos que seguem uma distribuição **normal (ou Gaussiana)**, o que significa que ele calcula a probabilidade de cada classe com base na média e variância dos atributos. É um modelo rápido e eficiente, especialmente em conjuntos de dados grandes e onde a suposição de independência entre variáveis faz sentido.

## Modelo 2: Tree Decision

* Esse modelo constrói uma árvore onde cada nó representa uma decisão baseada em uma **condição de um atributo**. O objetivo é dividir o conjunto de dados em partes menores e mais homogêneas em termos de classe-alvo. Ele funciona por meio de divisões sucessivas (em atributos) que maximizam a pureza das folhas da árvore, usando métricas como **gini** ou **entropia**. O Decision Tree é intuitivo e fácil de interpretar, mas pode ser suscetível a **overfitting**, especialmente em dados pequenos.

## Modelo 3: K-Neighbours

* O KNN é um **modelo de aprendizado supervisionado baseado em instâncias**, ou seja, ele faz previsões comparando novos dados com os dados de treinamento. Basicamente, para prever a classe de uma nova observação, o modelo procura os “K” exemplos mais próximos (vizinhos) e atribui a classe mais comum entre esses vizinhos. A distância entre as instâncias é geralmente medida por **métricas como a distância Euclidiana**. O KNN é simples e não exige treinamento pesado, mas pode ser mais lento com muitos dados, pois calcula distâncias em tempo real.

# Análise dos Modelos Aplicados

## Divisão dos Dados

A divisão dos dados foi realizada para garantir uma separação equilibrada entre treinamento e teste, possibilitando uma avaliação robusta e precisa dos modelos:

Divisão 80/20: O conjunto de dados foi dividido em 80% para treinamento e 20% para teste.

Validação Cruzada (opcional): Em casos em que há necessidade de minimizar o risco de overfitting, a validação cruzada (k-fold) pode ser utilizada para verificar o desempenho médio dos modelos em diferentes divisões do conjunto de dados.

## Treinamento dos Modelos

## Texto Descrição gerada automaticamente

Cada modelo foi treinado com um processo específico, incluindo ajuste de hiperparâmetros para otimização:

Texto

Descrição gerada automaticamente

GaussianNB: O modelo Naive Bayes assume independência entre as variáveis, sendo treinado diretamente nos dados de treinamento sem ajustes complexos. Ele é eficiente em termos de tempo, mas depende da suposição de normalidade dos dados.

DecisionTreeClassifier: O modelo de árvore de decisão foi treinado utilizando o critério “gini” para medir a impureza e maximizou a profundidade para melhor capturar os padrões. Ajustes em parâmetros como max\_depth e min\_samples\_split podem ser realizados para reduzir overfitting e melhorar a generalização.

KNeighborsClassifier: O KNN foi ajustado com n\_neighbors=2 e a métrica euclidiana. Ajustar o número de vizinhos é essencial, pois impacta diretamente a precisão e a sensibilidade do modelo.

## Métricas de Avaliação

As seguintes métricas foram utilizadas para avaliar a eficácia de cada modelo:

Acurácia: Proporção total de previsões corretas. É útil para uma visão geral, mas pode ser enganosa em conjuntos desbalanceados.

Precisão (Precision): Proporção de previsões corretas na classe positiva (classe “Sim”), útil para casos em que falsos positivos têm impacto significativo.

Recall: Proporção de verdadeiros positivos capturados entre todos os positivos reais, importante quando a prioridade é maximizar a detecção da classe positiva.

F1-Score: Média harmônica entre precisão e recall. Essa métrica equilibra ambos e é fundamental em contextos de classes desbalanceadas, sendo nossa métrica principal de avaliação, pois reflete melhor o equilíbrio entre os acertos das classes "Sim" e "Não".

## Análise dos Resultados

Cada modelo apresentou um desempenho único com base nas métricas. Abaixo, uma análise detalhada:

#### GaussianNB

Calendário

Descrição gerada automaticamente

Calendário

Descrição gerada automaticamente

Calendário

Descrição gerada automaticamente

Sem balanceamento: Alta precisão para “Sim” e alta para “Não”. Mesmo com SMOTE, o aumento no F1-score da classe "Não" foi limitado.

Com Random Oversampling: O modelo não obteve melhorias expressivas; o desbalanceamento dos dados impactou seu desempenho, não sendo adequado para nosso contexto de F1-score alto.

#### DecisionTreeClassifier

Texto

Descrição gerada automaticamenteTexto

Descrição gerada automaticamenteTexto

Descrição gerada automaticamente

Sem balanceamento: O desempenho para a classe "Não" foi limitado, mas com SMOTE, não só aumentou o F1-score, como ajudou em todas as outras métricas.

Com Random Oversampling: Obteve um F1-score de 0.90 em ambas as classes, sugerindo um bom equilíbrio entre precisão e recall e destacando-se como o modelo mais robusto.

#### KNeighborsClassifier

Sem balanceamento: Melhor para “Sim” , mas baixo para “Não”. O desempenho geral não foi afetado positivamente pelo balanceamento, mostrando dificuldades com classes desbalanceadas.

## Comparação Entre Modelos

#### Comparação Direta

F1-score para a classe "Não": Melhor desempenho com DecisionTreeClassifier com Random Oversampling (F1 de 0.90).

F1-score para a classe "Sim": Também com DecisionTreeClassifier com Random Oversampling (F1 de 0.90).

Precisão e Recall Balanceados: O DecisionTreeClassifier com Random Oversampling manteve o equilíbrio ideal entre precisão e recall para ambas as classes, se destacando por não favorecer uma classe em detrimento de outra.

#### Forças e Fraquezas

GaussianNB: Computacionalmente eficiente, mas limitado para dados desbalanceados, resultando em baixa robustez para a classe minoritária.

DecisionTreeClassifier: Modelo mais responsivo às técnicas de balanceamento, principalmente com Random Oversampling, onde apresentou um F1-score equilibrado e robusto.

KNeighborsClassifier: O modelo teve dificuldades em capturar bem ambas as classes, com Random Oversampling prejudicando o equilíbrio entre precisão e recall.

## Ajustes Necessários

##### Identificação de Problemas

Desbalanceamento de Dados: Prejudicou principalmente o GaussianNB e o KNeighborsClassifier, resultando em baixa captura para a classe "Não".

Baixo Recall e Precisão em Classes Minoritárias: O recall foi especialmente baixo para a classe "Não" no GaussianNB e no KNN, evidenciando a dificuldade de captura sem comprometer a precisão.

##### Mudanças na Base de Dados e Pré-processamento

Limpeza e Transformação de Dados: Revisar e eliminar outliers e aplicar normalização/padronização para beneficiar o KNN.

Engenharia de Características: Criar novas variáveis ou combinações de características para melhorar a distinção entre as classes.

Balanceamento com Oversampling: Continuar com Random Oversampling no DecisionTreeClassifier, já que o método trouxe um F1-score alto e equilibrado.

##### Ajustes nos Modelos

**Hiperparâmetros**: Ajustar max\_depth para evitar overfitting no DecisionTreeClassifier e explorar outras métricas de distância no KNN.

**Validação Cruzada**: Implementar validação cruzada (k-fold) para uma avaliação mais robusta, minimizando variações no desempenho.

## Impacto das Modificações

**DecisionTreeClassifier com Random Oversampling:** Com as modificações, o modelo atingiu o F1-score mais alto e equilibrado. O balanceamento de classes foi essencial para um desempenho robusto e generalizado.

**GaussianNB**: Mesmo com ajustes, o modelo não apresentou um desempenho adequado para ambos os grupos.

**KNeighborsClassifier**: Não atingiu F1-scores elevados mesmo após modificações, sugerindo limitações na captura de ambas as classes.

##### Comparação Final entre os Modelos (Ajustado)

Após as modificações, o **Decision Tree com Parâmetros Padrão** destacou-se como o modelo mais adequado, mantendo F1-scores altos e equilibrados em ambas as classes. Ele é ideal para o contexto, onde é essencial equilibrar o desempenho para ambas as classes.

# Conclusão

Com base nas análises realizadas, é possível observar diferenças significativas entre os modelos Gaussian Naive Bayes, Decision Tree e KNN, tanto nos parâmetros padrão quanto com os métodos de balanceamento de classes (SMOTE e Random Oversampling). Abaixo está uma análise comparativa dos resultados para identificar o modelo mais adequado e suas respectivas vantagens e limitações.Gaussian Naive Bayes

O ***Naive Bayes*** demonstrou desempenho desbalanceado ao privilegiar a classe "Sim", com baixo recall para a classe "Não". Mesmo com a aplicação de técnicas de balanceamento como SMOTE e ROS, o modelo manteve um viés acentuado, embora o SMOTE tenha mostrado um leve ganho no equilíbrio entre as classes. O Gaussian com SMOTE foi a melhor versão, com aumento de 0.63 para 0.71 no macro recall, o que trouxe um F1-score mais equilibrado. Entretanto, o Gaussian Naive Bayes é mais adequado para cenários em que um viés a favor de uma classe é tolerável e onde é aceitável que a classe minoritária receba menos atenção.

* **Ponto forte**: Simplicidade e boa performance para a classe "Sim" em dados desbalanceados.
* **Ponto fraco**: Dificuldade em captar a classe "Não" mesmo com balanceamento, sendo limitado em cenários que exigem alta sensibilidade para ambas as classes.

A ***Decision Tree*** sem ajustes apresentou um bom equilíbrio inicial entre as classes, com F1-scores de 0.67 para a classe "Não" e 0.85 para a classe "Sim", resultando em uma acurácia de 0.79. Ao aplicar SMOTE, o recall da classe "Não" foi reduzido para 0.50, embora o modelo tenha mantido acurácia geral. A aplicação de Random Oversampling (ROS) proporcionou leve melhora no recall da classe "Não", mas a acurácia geral caiu para 0.74. Isso sugere que o modelo de Decision Tree é robusto nos parâmetros padrão e perde eficiência ao tentar balancear as classes, visto que a árvore ajusta suas divisões para maximizar a precisão de ambas as classes.

* **Ponto forte**: Melhor equilíbrio inicial entre as classes com acurácia de 0.79, sem necessidade de ajustes adicionais.
* **Ponto fraco**: Diminuição no desempenho ao aplicar métodos de balanceamento, sugerindo que o modelo padrão já captura um equilíbrio razoável entre as classes.

O modelo ***KNN*** mostrou-se significativamente enviesado, com baixa precisão e recall para a classe "Não" no modelo inicial e, especialmente, no modelo com Random Oversampling, onde o desempenho permaneceu estável e pouco eficaz. O KNN com SMOTE, no entanto, trouxe uma melhoria importante: a precisão e recall para a classe "Não" chegaram a 0.50, e para a classe "Sim" permaneceram estáveis em 0.77, proporcionando maior equilíbrio e acurácia geral de 0.68. Este modelo se mostra mais adequado quando o objetivo é equilibrar classes em um modelo KNN, especialmente em problemas onde uma leve queda na acurácia é aceitável em prol de maior sensibilidade para ambas as classes.

* **Ponto forte**: Melhor equilíbrio entre as classes com SMOTE, demonstrando que o KNN pode ser uma opção viável para dados desbalanceados quando balanceado.
* **Ponto fraco**: Desempenho pobre sem balanceamento e com ROS, mostrando-se limitado em dados desbalanceados no formato inicial.

### Comparação Geral

1. **Decision Tree com Parâmetros Padrão** mostrou-se a opção mais robusta e eficiente entre os modelos, alcançando uma alta acurácia (0.79) e um bom equilíbrio entre precisão e recall para ambas as classes, sem necessidade de balanceamento adicional.
2. **Naive Bayes com SMOTE** foi eficiente ao reduzir o viés de classe do modelo, sendo uma boa escolha para dados onde um leve viés para a classe majoritária pode ser tolerável.
3. **KNN com SMOTE** demonstrou-se útil para balanceamento em dados desbalanceados, mas a acurácia foi inferior à Decision Tree. No entanto, é uma opção viável quando se busca maior sensibilidade para ambas as classes.

RPA usado no projeto:

O RPA utilizado no projeto, foi um RPA de transferência de dados, de um banco ao outro. atualizando o banco de destino com base nas mudanças do banco de origem. Ele utiliza colunas de controle (isUpdated e isDeleted): isUpdated identifica registros que precisam ser atualizados ou inseridos, enquanto isDeleted sinaliza os registros a serem removidos. O processo carrega os dados do banco de origem, separa as colunas de controle dos dados principais e, em seguida, verifica cada registro no banco de destino. Dependendo do valor das colunas de controle, o registro é atualizado, inserido ou excluído. Após a sincronização, o RPA limpa o banco de origem, removendo registros marcados para exclusão e redefinindo isUpdated para false.

Essa operação é aplicada a quatro tabelas principais (Admin, Plano, Interesse e Categoria), com ajustes de nomes de colunas para garantir a compatibilidade entre os bancos

Como o modelo foi subido no aplicativo:

O modelo de IA foi integrado ao aplicativo utilizando a biblioteca Flask e implantado na AWS EC2. O sistema consiste em um formulário que coleta informações do usuário e as processa por meio de uma pipeline pré-carregada, que inclui tanto o pré-processamento quanto o modelo de IA em si. A interface do formulário é acessada por um link e exibida no aplicativo através de uma WebView, permitindo uma experiência integrada e fluida para o usuário.

Conclusão

A análise de dados realizada para o Projeto Olimpo permitiu identificar padrões importantes sobre o perfil e as necessidades dos usuários, contribuindo para uma segmentação mais eficaz do público-alvo. Os modelos aplicados, principalmente a Decision Tree, mostrou boa capacidade de previsão, com destaque para a acurácia, f1-score e a robustez no reconhecimento de potenciais usuários. Esse processo de predição oferece uma boa pesquisa, para ver as principais localizações do público alvo do projeto, ajudando a personalizar ações que promovam maior integração e cooperação comunitária. Como próximas etapas, principalmente, pode-se aumentar a coleta de dados para aumentar a precisão dos modelos e para balancear mais a pesquisa de dados, em relação a classes ‘Sim’ e ‘Não’. Isso permitirá um impacto social ainda mais profundo, promovendo uma rede comunitária mais forte e colaborativa.