**Preparação de Dados/Engenharia de Recursos**

## 1. Visão Geral

A Fase de preparação dos dados e engenharia de recursos são as etapas mais importantes de um projecto de Machine Learning porque são essas as fases que garantem dados de qualidades para o modelo aprender e extrair características relevantes para a tarefa que deverá ser executado pelo modelo. Onde selecionamos as features mais relevantes, transformamos as features categóricas em numéricas porque os modelos lidam melhores com números, tratamos os outliers e aplicamos mais outras técnicas de manipulação de dados para garantir uma performece de qualidados no treinamento e predição dos modelos de machine learning.

## 2. Coleta de Dados

Os dados foram extraidos diretamente da plantaforma Kaggle e não foram necessária nenhum tipo de pré-processamento durante a coleta de dados.

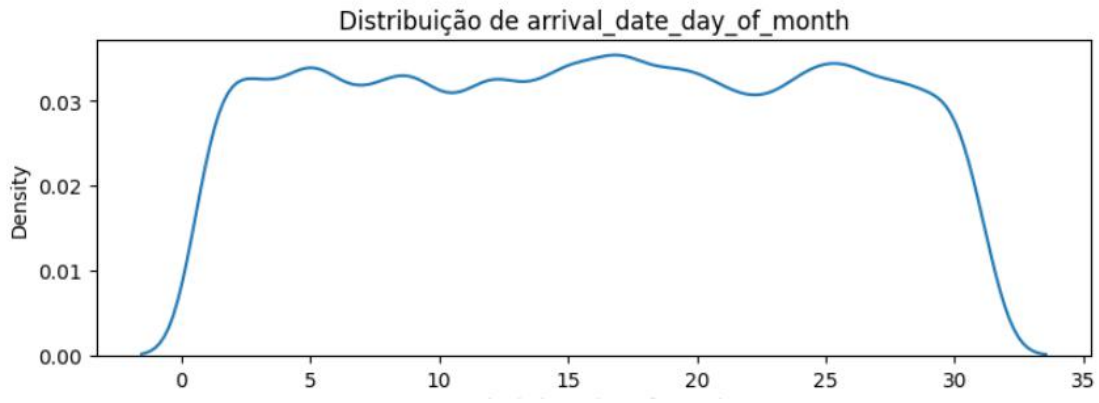
## 3. Limpeza de Dados

A limpeza de dados é a fase na qual dados imprecisos ou que não agregam valor ao conjunto de dados são descartados, para o nosso conjunto de dados foram dropados(ignorados) as colunas: "agent", "company", "arrival_date_month", "country", "reservation_status_date", "arrival_date_year", "arrival_date_week_number", "reservation_status".

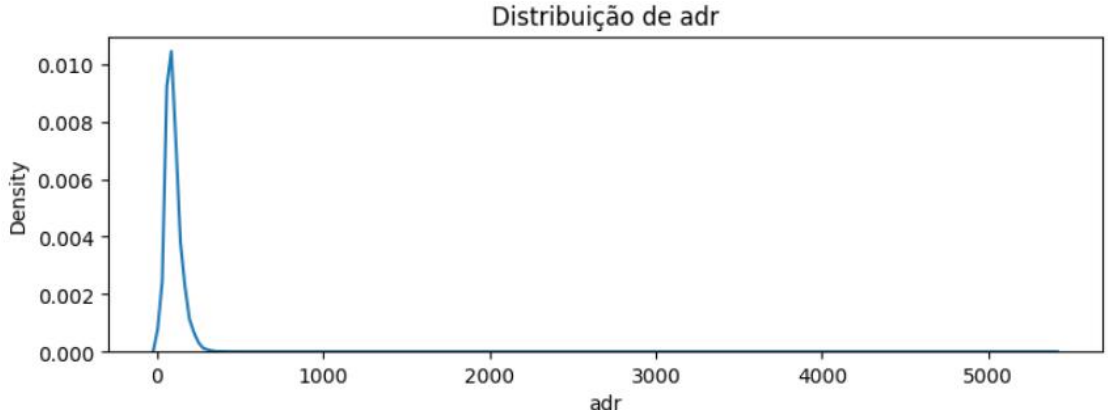
Essas colunas não contiam informações relevantes para o nosso problema, as suas correlações em comparação a variável alvo é abaixo de 50%. As outras colunas categóricas foram tratadas com a técnica LabelEncoder para transformação numérica, optamos por LabelEncoder porque ela não produz o efeito de alta dimensionalidade após a aplicação com o OneHotEncoder, por se tratar de 9 colunas com várias classes com o LabelEncoder já conseguimos uma performecer excelente.

## 4. Análise Exploratória de Dados(EDA)

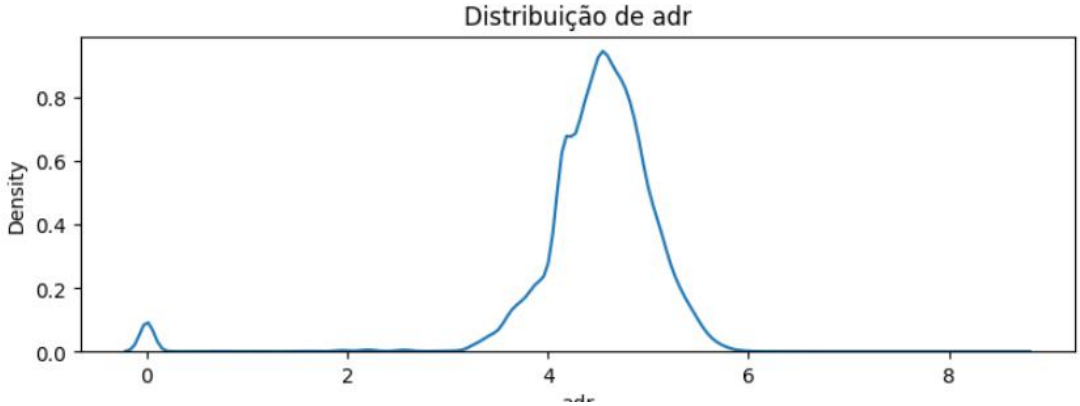
Analisamos os dados para compreender como estavam as distribuições dos dados, e podemos perceber que algumas variáveis possuem uma distribuiç ão normal, como a '**arrival_date_day_of_month'.**



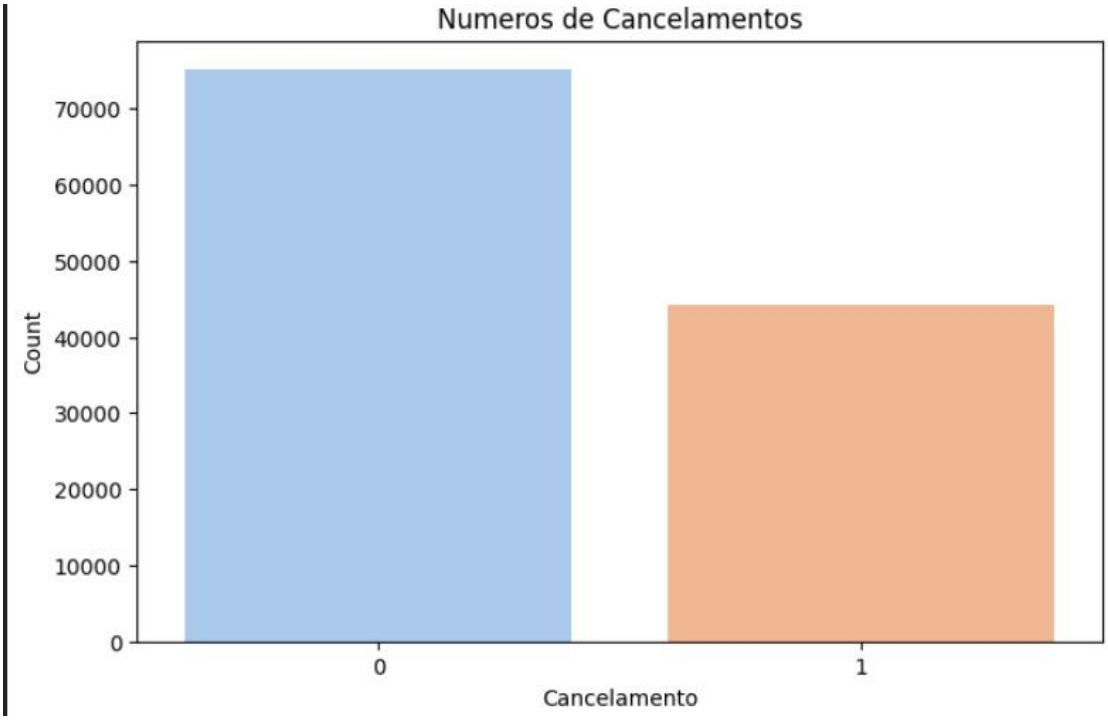
O restante das variáveis possuem uma distribuição exponencial, para e ssas variáveis foram feita a transformação logarítmica. Como o “sdr” essa trans formação é feita para dados que contêm uma cauda longa a direita, e para dad os com a cauda a esquerda é feita a transformação exponencial para normaliza r esses dados.



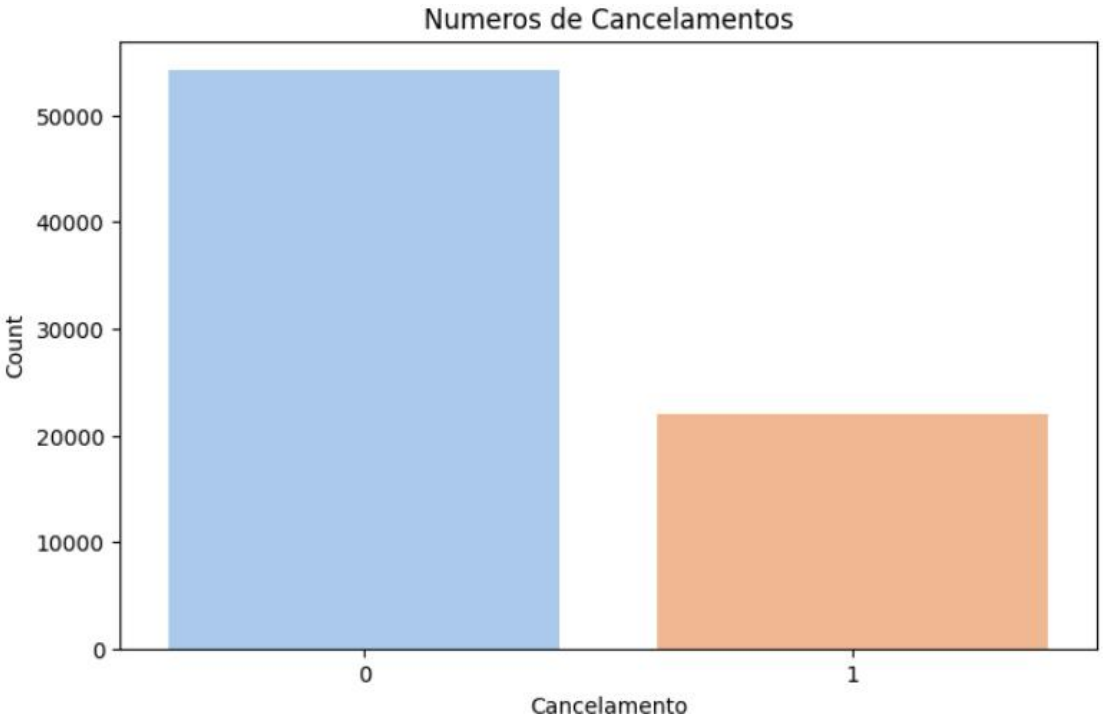

**Após a transformação Logarítmica:**



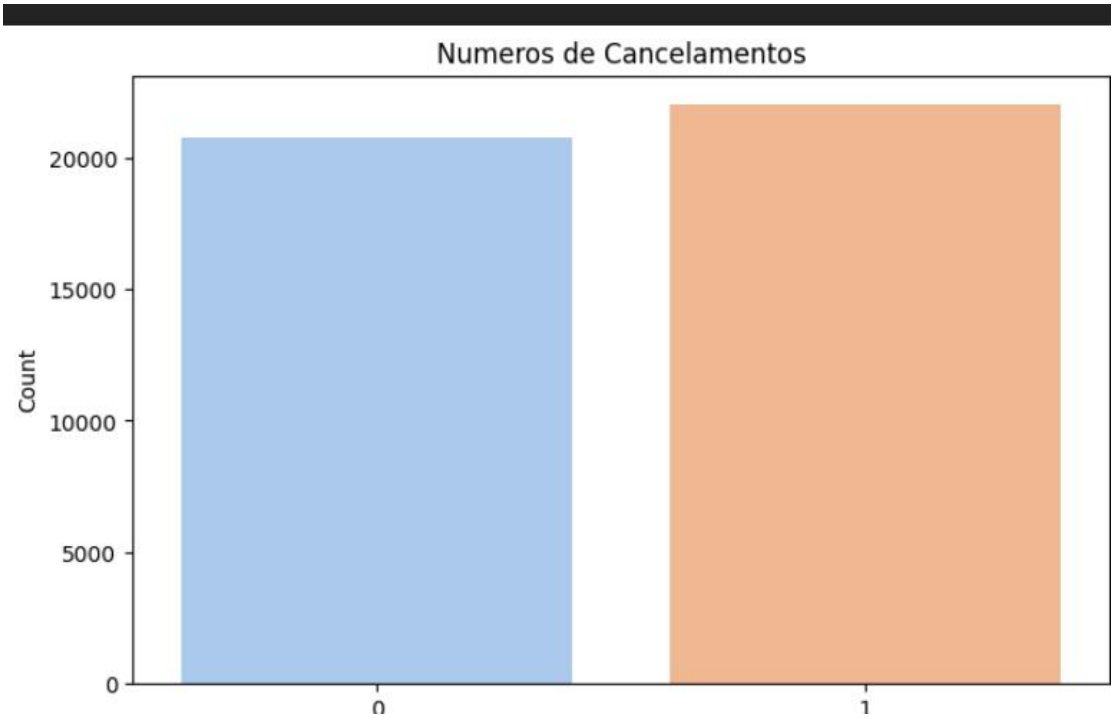

Um grande número das reservas de Hotel e Risort não são canceladas após a marcação.



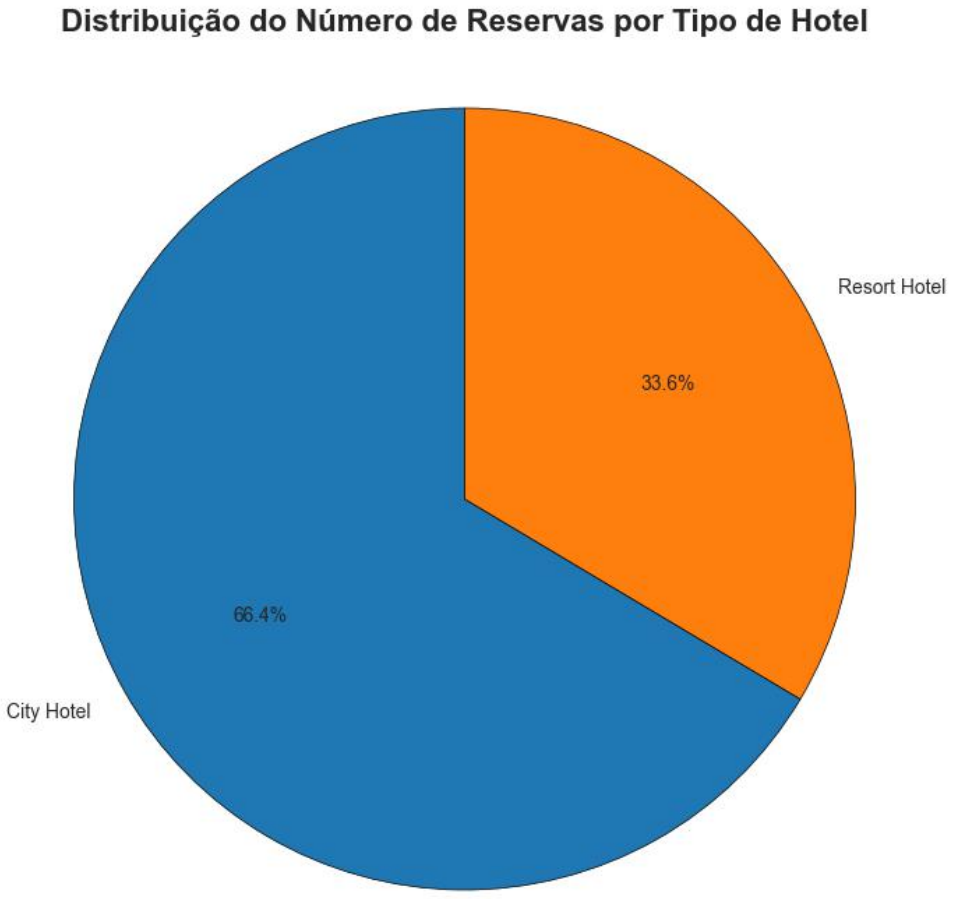

Para "lead_time" com valor < 113 as chances de cancelamento é muito reduzid a:




Para "lead_time" com valor > 113 as chances de cancelamento estão quase nas mesmas proporção:




Os visitantes têm como prefência maioritária hoteis próximo a cidade:



## 5. Engenharia de Recursos

Separamos os nosso conjunto de dados original em dois dataset, um dataset contendo somente colunas categóricas e outro dataset contendo somente colunas numéricas. Para a engenharia de recursos aplicamos a seleção de recursos(feature selection) onde aplicamos a técnica seleção stepwise para colunas numéricas, essa técnica é menos robusta e ideal para início da nossa fase de teste, ela é uma técnica de seleção passo a passo. A seleção passo a passo é uma combinação de seleção para frente e para atrás, em outras palavras, essa seleção adiciona e remove características x na tentativa de chegar a um modelo ótimo, ela faz isso verificando se há valores-p insignificantes após a adição de uma nova característica. Se encontrar uma, remove a característica x insignificante e continua adicionar/remover até que não haja mais características insignificantes para adicionar. Testamos também a seleção de recursos com a otimização de GridSearchCV que é uma técnica mais robusta que otimiza os parâmetros automaticamente mas não seguimos com essa abordagem porque ela é muito agressiva aos dados ao ponto de selecionar somente três features como relevantes, mas sabemos que até algumas features menos relevantes podem ajudar o modelo a generalizar melhor os dados e com a seleção de passo a passo o modelo já obteve bons resultados. Para as colunas categóricas usamos métodos estatísticos comparando o p-valor da coluna categórica em relação a variável alvo, e as colunas com um p-valor menor que 0.05 foram excluidas.

## 6. Transformação de Dados

Aplicamos a normalização StandardScaler para deixar os valores nos mesmos intervalos para que os modelos não coloquem um peso maior a uma feature porque possuem valores muito alto, usamos o StandardScaler ao invés de MinMaxScaler ou RobustScaler porque em geral o StandardScaler funciona muito bem para a maioria dos casos.

```
scaler = StandardScaler()
Xnorm = scaler.fit_transform(X)
```

**Exploração do Modelo**

1. Seleção de modelo

Para o nosso projecto testamos três modelos diferentes, testamos o modelo XGBoost baseado em árvores de decisão, K-Neighbors baseado em vizinhos mais próximos e o LogisticRegression baseado em probabilidade que usa por detras dos panos a função sigmoid. Em nosso projecto estabelecemos algumas métricas de negócio para nos ajudar na escolha do melhor modelo para a tarefa realizada:

> ***Minimizar reservas canceladas falsas*** **(Precision):** Quando o modelo disser que a reserva **será cancelada**, ela deverá acertar na maioria das vezes.

***Capturar o máximo de cancelamentos reais*** **(Recall):** Identificar a maioria dos clientes que realmente irão cancelar.

Com isso o modelo escolhido foi o K-Neighors por apresentar uma média armônica maior em relação aos outros modelo(f1_score, média harmónica entre Precision e Recall) de 0.716. O nosso modelo obteve um recall superior de 0.67 em relação aos outros modelos isso implica dizer que o modelo c**aptura o máximo de cancelamentos reais,** consegue Identificar a maioria dos clientes que realmente irão cancelar (Positivos Reais). Mas ela perde na precisão pela LogistcRegression, assim sendo o modelo emitirá mais alertas falsos (reservas não canceladas sendo alertadas). O impacto financeiro e operacional dos erros do modelo podem implicar o seguinte:

* **Ações Desnecessárias:** A equipa gasta tempo e recursos (e-mails, chamadas, ofertas de retenção) num cliente que não ia cancelar.

* **Experiência do Cliente:** O cliente que ia ficar pode sentir-se incomodado ou confuso com a intervenção desnecessária.

* **Perda de Tempo:** Desperdício de tempo da equipa de vendas/suporte.

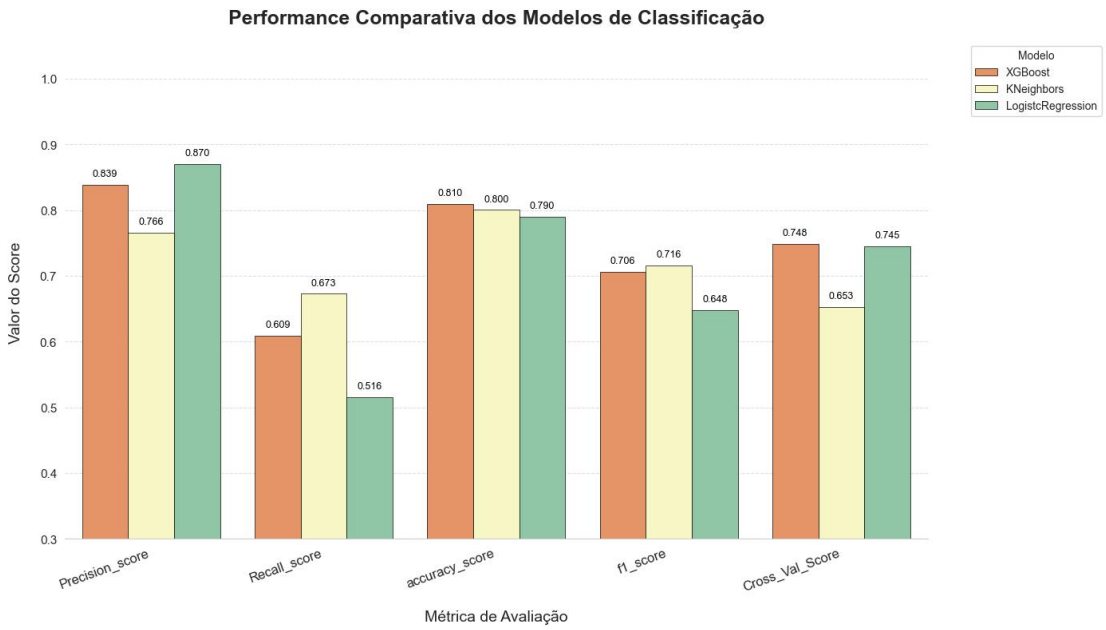
Por outro lado, com um Recall alto evitamos as p**erdas de receita** e ganha-se mais tentando salvar o máximo de reservas mesmo que isso implique aceitar mais alarmes falsos.

2. Treinamento dos Modelos

Os modelo foram treinados treinados com os hiperparâmetros optimiza dos, utilizamos o GridSearchCV para a otimização dos modelos. Para o XGBo ost esses foram os parâmetros: {'colsample_bytree': 0.9, 'learning_rate': 0.1, 'ma x_depth': 5, 'reg_lambda': 1} para o K-Neighbors: {'n_neighbors': 11, 'weights': 'distance'} e para o LogistcRegression : {'C': 10.0, 'penalty': 'l2', 'solver': 'libline ar', 'max_iter': '1000', 'random_state': '41'}, a técnica de validação cruzada aplic ada foi o cross_val_score : cross_val_score(

```
    estimator=model,
    X=X,
    y=y,
    cv=3,
)
```

3. Avaliação dos Modelos




Algumas vantangens do modelos a nível computacional são:

| **Aspeto** | **Vantagem Específica** | **Implicação Computacional/Geral** |
|---|---|---|
| **Simplicidade/Interpretabilidade** | **Fácil de entender e implementar.** | É um algoritmo baseado em distância, o que o torna intuitivo. Não tem um processo de "treinamento" complexo (é um *lazy learner*). |
| **Não Paramétrico** | **Não faz pressuposições sobre a distribuição dos dados.** | Pode capturar formas de decisão complexas e não lineares. Bom para dados onde a relação entre as variáveis é desconhecida. |
| **Aprendizagem Tardia (*Lazy Learning*)** | **Treinamento quase instantâneo.** | O tempo de "treinamento" (fase em que o modelo aprende) é basicamente zero, pois o modelo só armazena o conjunto de dados. |
| **Adaptabilidade** | **Fácil de adicionar novos dados.** | Novos pontos de dados podem ser adicionados ao conjunto de treinamento sem retreinar todo o modelo. |

Desvantagens:

| **Aspeto** | **Desvantagem Específica** | **Implicação Computacional/Geral** |
|---|---|---|
| **Custo Computacional na Previsão** | **Alto custo de tempo na fase de teste/previsão.** | Para classificar um novo ponto, o KNN precisa calcular a distância desse ponto para *todos* os pontos de treinamento. O custo computacional do algoritmo é $O(n \cdot d)$, onde $n$ é o número de pontos de treinamento e $d$ é o número de características. **Quanto maior o *dataset* ($n$), mais lenta é a previsão.** |
| **Requisito de Memória** | **Alto consumo de memória.** | Como é um *lazy learner*, o modelo precisa armazenar **todo o conjunto de dados de treinamento** na memória para fazer previsões. |
| **Escalabilidade (Maldição da Dimensionalidade)** | **Mau desempenho em dados de alta dimensão.** | O cálculo de distância torna-se menos significativo (todos os pontos ficam "igualmente distantes") à medida que o número de características ($d$) aumenta, o que pode levar a previsões imprecisas e aumentar o custo computacional. |
| **Sensibilidade à Escala** | **Sensível à escala e ruído dos dados.** | As características com valores maiores podem dominar a função de distância. Requer que os dados sejam normalizados ou padronizados. Também é sensível a *outliers* (valores atípicos). |
| **Escolha de *k*** | **Otimizar o valor de $k$ é crucial e empírico.** | A escolha do número de vizinhos ($k$) afeta drasticamente o desempenho do modelo (valores baixos são sensíveis ao ruído; valores altos suavizam demais a fronteira de decisão). |