

# Trabalho Final – Inteligência Artificial Sistema Inteligente com Deploy em Streamlit

Glória Xavier e Heloisa dos Santos

---

[Clique aqui para acessar a aplicação Streamlit.](#) [Clique aqui para acessar o repositório no Github.](#)

---

## 1 Problema e Objetivo

O objetivo primário deste projeto é desenvolver um sistema de previsão que auxilie na identificação de perfis de alto risco para determinadas doenças. O problema central que buscamos resolver é a avaliação de risco e diagnóstico de saúde, visando auxiliar profissionais da área a entender quais fatores (como IMC, pressão arterial, idade) realmente impulsionam o risco de um indivíduo desenvolver uma condição específica (Doença Cardíaca, AVC, etc.), dada a multiplicidade de diagnósticos possíveis.

**Abordagem de Machine Learning de Duas Etapas:**

1. **Clusterização (K-Means):** Utilizada para agrupar os indivíduos em perfis de risco latentes (ex: Alto Risco, Baixo Risco, Risco Intermediário).
2. **Classificação Multiclasse (Random Forest):** Treinada utilizando a feature de perfil de risco obtida no passo anterior, para prever o diagnóstico mais provável e as regras específicas que levam a essa condição dentro de cada grupo.

Esse fluxo garante que a predição final seja mais precisa e, principalmente, **interpretável**, facilitando a visualização e a tomada de decisão no aplicativo Streamlit.

**Dataset Utilizado:** Healthcare Risk Factors Dataset, disponível no Kaggle, com 30.000 registros e 20 variáveis relacionadas às condições de saúde.

---

## 2 Visão Geral do Dataset

O conjunto de dados é adequado para tarefas de classificação em saúde (prever diabetes, hipertensão, obesidade) e modelagem preditiva.

Característica	Detalhe
Linhas	30.000
Colunas	20
Desafios	Possui valores ausentes e colunas com ruído proposital ( <code>random_notes</code> , <code>noise_col</code> ).

### Categorias de Atributos Principais

- **Condição Médica (Target):** Diabetes, Hypertension, Asthma, Obesity, Healthy.
  - **Métricas Clínicas:** Glucose, Blood Pressure, BMI, Oxygen Saturation, Cholesterol, Triglycerides, HbA1c.
  - **Demografia:** Age, Gender.
  - **Hábitos e Estilo de Vida:** Smoking, Alcohol, Physical Activity, Diet Score, Family History.
-

### 3 Aquisição e Análise Exploratória de Dados (EDA)

#### 3.1 Obtenção e Análise Estrutural

Os dados foram carregados remotamente e lidos em um DataFrame. Foi verificado o formato dos dados (30000 linhas, 20 colunas) e identificada a presença de valores ausentes (`NaN`) em colunas-chave como `Age`, `Glucose` e `Blood Pressure`.

#### 3.2 Análise de Risco e Visualização (EDA Visual)

A fase de EDA visual foi crucial para compreender a distribuição dos dados e os perfis de risco latentes.

- **Distribuição de Variáveis Clínicas por Condição:** A análise das curvas de densidade (KDE) para as principais métricas clínicas revelou padrões distintos entre as condições.
  1. **Glicose (Glucose) x Condição Médica:** O nível de Glucose é o fator mais discriminante para **Diabetes**, que apresenta uma distribuição claramente deslocada para valores mais altos (acima de 150 mg/dL), enquanto a classe **Saudável** tem seu pico em níveis normais (abaixo de 100 mg/dL).
  2. **Idade (Age) x Condição Médica:** A Age é um fator crucial, com **Hipertensão** e **Diabetes** concentrando-se em faixas etárias mais elevadas, com picos de densidade após os 60 anos. A condição **Saudável** predomina em faixas etárias mais jovens (20-40 anos).
- **Fatores Demográficos e Comportamentais:**
  3. **Percentual de Diagnósticos por Gênero:** A análise de barras de frequência demonstrou que certas condições apresentam desequilíbrio de gênero. Por exemplo, a **Obesidade** e o **Câncer** podem ter uma prevalência maior em um dos sexos (observar a distribuição percentual).
  4. **Qualidade da Alimentação (Diet Score) x Condição Médica:** O gráfico (provavelmente um Boxplot ou Violino) revelou que a classe **Saudável** possui a melhor qualidade de alimentação (mediana mais alta), enquanto **Diabetes** e **Obesidade** apresentam medianas e dispersão em scores mais baixos (associados a hábitos ruins).
  5. **Horas Dormidas (Sleep Hours) x Condição Médica:** Observou-se que condições de risco como **Hipertensão** e **Obesidade** tendem a ter uma proporção maior de pacientes com menos de 6 horas de sono, indicando a privação do sono como um possível fator contribuinte. A maioria das condições, no entanto, tem seu pico entre 7 e 8 horas.
  6. **Nível de Estresse (Stress Level) x Condição Médica:** O nível de estresse (em escala 1-10) mostra que pacientes com **Diabetes** e **Hipertensão** apresentam maior densidade em níveis de estresse moderado a alto (acima de 6), sugerindo uma ligação entre fatores psicossociais e doenças crônicas.
- **Análise de Correlação e Outros Dados Importantes:**
  - **Matriz de Correlação:** Foi gerado um **Mapa de Calor** para confirmar a forte correlação entre Glucose e HbA1c (próxima a 0.85) e entre Cholesterol e Triglycerides.
  - **Prevalência de Tabagismo e Álcool:** A análise de contagens confirmou que o consumo de álcool e tabaco está associado a um aumento acentuado no risco, com a prevalência de **Hipertensão** subindo de 25.55% para 35.47% nos consumidores de álcool.

### 4 Pré-processamento de Dados

As seguintes etapas de limpeza e preparação foram realizadas no dataset:

- **Tratamento de Outliers:** Foi aplicada uma função que utiliza o método do Intervalo Interquartil (IQR) ( $Q1 \pm 1.5 \times IQR$ ) para **limitar** (capping) os outliers nas colunas numéricas.
- **Tratamento de Valores Ausentes:** Colunas numéricas (`Age`, `Glucose`, `Blood Pressure`) foram imputadas com a **mediana**. Colunas categóricas (`Gender`, `Medical Condition`) foram imputadas com a **moda**.

- **Remoção de Colunas Irrelevantes:** As colunas `random_notes` e `noise_col` foram removidas.
  - **Codificação de Variáveis Categóricas:** A variável alvo (`Medical Condition`) foi codificada numericamente usando `LabelEncoder`.
- 

## 5 Desenvolvimento e Treinamento do Modelo

O projeto seguiu uma abordagem de Machine Learning em duas fases para aumentar a interpretabilidade e precisão do diagnóstico.

### 5.1 Fase 1: Clusterização (K-Means)

O objetivo desta fase foi agrupar os indivíduos em perfis de risco latentes.

1. **Determinação do K:** O método do cotovelo (*Elbow Method*) foi utilizado para identificar o número ideal de clusters, que foi definido como **4**.
2. **Criação do Perfil de Risco:** O algoritmo K-Means foi aplicado, e os rótulos de cluster (0 a 3) foram adicionados ao dataset como uma nova variável categórica: `Risk_Profile_Cluster`.

### 5.2 Fase 2: Classificação (Random Forest)

O modelo de classificação utilizou as características originais **mais** o novo `Risk_Profile_Cluster` para prever a `Medical Condition`.

1. **Modelo:** Foi utilizado o classificador **Random Forest**.
2. **Divisão de Dados:** O conjunto de dados foi dividido em treino e teste (80% treino, 20% teste).
3. **Performance do Modelo Otimizado (com K-Means):**
  - **Acurácia Geral:** **90.71%**.
  - O modelo demonstrou alta performance.
4. **Comparação de Performance (Abordagem Híbrida vs. Base)**
  - Um modelo alternativo (**Modelo Base**), treinado utilizando as features originais **sem** a variável de cluster, demonstrou uma performance superior, alcançando **91.22%** de acurácia.
  - Devido ao desempenho superior, o **Modelo Base (sem Clusterização)** foi escolhido como o modelo final para o deploy em Streamlit.

### 5.3 Métricas de Avaliação do Modelo

Para a avaliação dos modelos de Random Forest, foram utilizadas métricas padrão para problemas de classificação multiclasse, com foco na robustez e no equilíbrio entre classes.

- **Acurácia (Accuracy):** Usada como métrica principal para o ranking dos modelos, representa a proporção de previsões corretas sobre o total de casos.
- **F1-Score (Ponderado):** Fundamental para avaliar o desempenho em classes individuais. O F1-Score é a média harmônica da Precisão (*Precision*) e Revocação (*Recall*), fornecendo uma visão mais equilibrada, especialmente em datasets com algum desbalanceamento de classes.
- **Precisão (Precision) e Revocação (Recall):** Analisadas para entender o tipo de erro que o modelo estava cometendo, ou seja, se estava classificando falsos positivos ou falsos negativos em cada condição médica.

## 5.4 Análise do Impacto da Clusterização

O resultado surpreendente onde a **Abordagem Híbrida** apresentou desempenho ligeiramente inferior ao **Modelo Base** sugere que a variável **Risk\_Profile\_Cluster** introduziu mais ruído do que informação útil para a classificação final. Provavelmente estas foram algumas das causas:

- **Redundância de Informação:** A clusterização (K-Means) agrupa dados com base em métricas clínicas (Glicose, BMI, Pressão Arterial), que já são features poderosas no Modelo Base. Ao adicionar o cluster, a informação de agrupamento já está inherentemente presente nas features originais. O algoritmo Random Forest pode não ter se beneficiado do cluster, pois ele já estava utilizando as features originais para criar separações semelhantes internamente.
- **Introdução de Erro de Quantização:** A clusterização transforma dados contínuos em uma variável categórica discreta (os 4 rótulos de cluster). Essa "quantização" pode ter apagado nuances sutis nos dados que o Random Forest conseguia capturar ao analisar as variáveis contínuas diretamente.
- **Otimização do Modelo Base:** O Modelo Base, apesar de não ter o benefício da interpretabilidade do cluster, conseguiu otimizar melhor os hiperparâmetros ou utilizar as correlações diretas das variáveis contínuas, resultando em uma fronteira de decisão marginalmente mais precisa.

---

## 6 Exportação do Modelo para Deploy

Ao final do processo, os artefatos de ML necessários para a implantação do sistema no Streamlit foram salvos:

- O modelo Random Forest treinado (`modelo_random_forest.pkl`).
- O objeto de escalonamento (`scaler_base.pkl`).
- O codificador de rótulos (`label_encoder.pkl`).

## 7 A partir da próxima página, está todo o desenvolvimento do projeto no Jupyter Notebooks. Abaixo, o link da aplicação Streamlit

[Clique aqui para acessar a aplicação Streamlit.](#)

# TrabalhoFinal

December 1, 2025

## 1 Trabalho Final – Inteligência Artificial

### 1.1 Sistema Inteligente com Deploy em Streamlit

**Dupla: Glória Xavier e Heloisa dos Santos** Este notebook mostra o desenvolvimento do trabalho final da disciplina de Inteligência Artificial.

Clique aqui para acessar o repositório no Github.

### 1.2 Problema e Objetivo

O objetivo primário deste projeto é desenvolver um sistema de previsão que auxilie na identificação de perfis de alto risco para determinadas doenças. O problema central que buscamos resolver é a avaliação de risco e diagnóstico de saúde, visando auxiliar profissionais da área a entender quais fatores (como IMC, pressão arterial, idade) realmente impulsoram o risco de um indivíduo desenvolver uma condição específica (Doença Cardíaca, AVC, etc.), dada a multiplicidade de diagnósticos possíveis. Pretendemos resolver isso por meio de uma abordagem de Machine Learning de duas etapas: primeiro, utilizamos a Clusterização (K-Means) para agrupar os indivíduos em perfis de risco(ex: Alto Risco, Baixo Risco, Risco Intermediário). Em seguida, treinamos um modelo de Classificação Multiclasse (Árvore de Decisão/Random Forest), utilizando a feature de perfil de risco obtida no passo anterior, para prever o diagnóstico mais provável e as regras específicas que levam a essa condição dentro de cada grupo. Esse fluxo garante que a predição final seja mais precisa e, principalmente, interpretável, facilitando a visualização e a tomada de decisão no aplicativo Streamlit.

Para esta atividade, escolhemos o dataset Healthcare Risk Factors Dataset, disponível no Kaggle.

### 1.3 Visão Geral

Este conjunto de dados contém 30.000 registros e 20 variáveis relacionadas às condições de saúde de indivíduos, incluindo algumas colunas ruidosas/aleatórias. Ele é adequado para tarefas de classificação em saúde, como prever diabetes, hipertensão e obesidade. Também é útil para modelagem preditiva, como previsão do tempo de permanência hospitalar ou análise de fatores de risco.

O dataset inclui valores ausentes e variáveis de ruído, permitindo praticar limpeza e pré-processamento de dados.

### 1.4 Detalhes Principais

- Linhas: 30.000

- Colunas: 20
- Possui valores ausentes
- Inclui colunas com ruído proposital

## **1.5 Categorias de Atributos**

### **1.5.1 Demografia**

- Age: idade do paciente (anos)
- Gender: masculino / feminino

### **1.5.2 Condição Médica**

- Medical Condition: condição reportada (Diabetes, Hypertension, Asthma, Obesity, Healthy)

### **1.5.3 Exames e Métricas Clínicas**

- Glucose: nível de glicose
- Blood Pressure: pressão arterial
- BMI: índice de massa corporal
- Oxygen Saturation: saturação de oxigênio
- Cholesterol: nível de colesterol
- Triglycerides: nível de triglicerídeos
- HbA1c: hemoglobina glicada

### **1.5.4 Hábitos e Estilo de Vida**

- Smoking: 0 = não fumante, 1 = fumante
- Alcohol: 0 = não consome, 1 = consome
- Physical Activity: horas semanais de atividade física
- Diet Score: pontuação de qualidade da dieta
- Family History: 0 = sem histórico familiar, 1 = possui

### **1.5.5 Informações Gerais**

- LengthOfStay: tempo de permanência no hospital (dias)

- Stress Level: nível de estresse
- Sleep Hours: horas de sono por dia
- random\_notes: notas aleatórias
- noise\_col: valores aleatórios sem relação

## 1.6 Obtendo o dataset

Os dados foram obtidos através da biblioteca kagglehub, permitindo que seja possível carregar os dados de forma remota e sem depender de armazenamento local.

```
[1]: import kagglehub
import glob
import pandas as pd
import os

path = kagglehub.dataset_download("abdallaahmed77/
↪healthcare-risk-factors-dataset")

caminho = glob.glob(os.path.join(path, '*.csv'))

df = pd.read_csv(caminho[0])
```

	Age	Gender	Medical Condition	Glucose	Blood Pressure	BMI	\
0	46.0	Male	Diabetes	137.04	135.27	28.90	
1	22.0	Male	Healthy	71.58	113.27	26.29	
2	50.0	NaN	Asthma	95.24	NaN	22.53	
3	57.0	NaN	Obesity	NaN	130.53	38.47	
4	66.0	Female	Hypertension	95.15	178.17	31.12	
...	...	...	...	...	...	...	
29995	54.0	Female	Healthy	86.35	127.68	28.30	
29996	41.0	Male	Asthma	89.24	154.37	24.93	
29997	83.0	Female	NaN	NaN	170.11	30.55	
29998	41.0	Male	Diabetes	242.87	152.97	23.57	
29999	46.0	Female	Hypertension	86.52	159.26	34.55	
	Oxygen Saturation	LengthOfStay	Cholesterol	Triglycerides	HbA1c	\	
0	96.04	6	231.88	210.56	7.61		
1	97.54	2	165.57	129.41	4.91		
2	90.31	2	214.94	165.35	5.60		
3	96.60	5	197.71	182.13	6.92		
4	94.90	4	259.53	115.85	5.98		
...	...	...	...	...	...		
29995	94.36	1	218.60	152.82	4.85		

```

29996          96.07        4    198.30      176.81   5.36
29997          97.83        3   210.13      168.76   5.05
29998          99.17        3   208.86      224.21   7.59
29999          96.60        4   217.37      141.28   5.77

      Smoking  Alcohol  Physical Activity  Diet Score  Family History \
0            0         0             -0.20     3.54           0
1            0         0              8.12     5.90           0
2            0         0              5.01     4.65           1
3            0         0              3.16     3.37           0
4            0         1              3.56     3.40           0
...
29995          0         0              9.15    10.35           0
29996          1         0              3.34     3.40           0
29997          0         0              4.53     4.50           0
29998          0         0              3.16     2.59           0
29999          1         0              4.91     5.82           1

      Stress Level  Sleep Hours random_notes  noise_col
0       5.07      6.05      lorem -137.057211
1       5.87      7.72      ipsum -11.230610
2       3.09      4.82      ipsum  98.331195
3       3.01      5.33      lorem  44.187175
4       6.38      6.64      lorem  44.831426
...
29995          3.65      8.71      ipsum  89.573837
29996          6.72      2.81      lorem -5.864517
29997          4.31      3.61      ### -30.934595
29998          7.10      5.80      ipsum  5.085258
29999          3.64      5.47      lorem -142.559559

```

[30000 rows x 20 columns]

## 2 EDA - Análise Exploratória de Dados

[3]: df.shape

[3]: (30000, 20)

[4]: df.info()

```

<class 'pandas.core.frame.DataFrame'>
RangeIndex: 30000 entries, 0 to 29999
Data columns (total 20 columns):
 #   Column            Non-Null Count  Dtype  
--- 
 0   Age               25500 non-null   float64

```

```

1   Gender           25500 non-null  object
2   Medical Condition 25500 non-null  object
3   Glucose          25500 non-null  float64
4   Blood Pressure   25500 non-null  float64
5   BMI              30000 non-null  float64
6   Oxygen Saturation 30000 non-null  float64
7   LengthOfStay    30000 non-null  int64
8   Cholesterol      30000 non-null  float64
9   Triglycerides    30000 non-null  float64
10  HbA1c            30000 non-null  float64
11  Smoking           30000 non-null  int64
12  Alcohol            30000 non-null  int64
13  Physical Activity 30000 non-null  float64
14  Diet Score        30000 non-null  float64
15  Family History    30000 non-null  int64
16  Stress Level      30000 non-null  float64
17  Sleep Hours       30000 non-null  float64
18  random_notes      30000 non-null  object
19  noise_col         30000 non-null  float64
dtypes: float64(13), int64(4), object(3)
memory usage: 4.6+ MB

```

[5]: df.drop\_duplicates()

	Age	Gender	Medical Condition	Glucose	Blood Pressure	BMI	\
0	46.0	Male	Diabetes	137.04	135.27	28.90	
1	22.0	Male	Healthy	71.58	113.27	26.29	
2	50.0	NaN	Asthma	95.24		NaN	22.53
3	57.0	NaN	Obesity	NaN	130.53	38.47	
4	66.0	Female	Hypertension	95.15	178.17	31.12	
...	...	...	...	...	...	...	
29995	54.0	Female	Healthy	86.35	127.68	28.30	
29996	41.0	Male	Asthma	89.24	154.37	24.93	
29997	83.0	Female		NaN	170.11	30.55	
29998	41.0	Male	Diabetes	242.87	152.97	23.57	
29999	46.0	Female	Hypertension	86.52	159.26	34.55	
	Oxygen Saturation	LengthOfStay	Cholesterol	Triglycerides	HbA1c	\	
0	96.04	6	231.88	210.56	7.61		
1	97.54	2	165.57	129.41	4.91		
2	90.31	2	214.94	165.35	5.60		
3	96.60	5	197.71	182.13	6.92		
4	94.90	4	259.53	115.85	5.98		
...	...	...	...	...	...		
29995	94.36	1	218.60	152.82	4.85		
29996	96.07	4	198.30	176.81	5.36		
29997	97.83	3	210.13	168.76	5.05		

29998		99.17		3	208.86		224.21	7.59
29999		96.60		4	217.37		141.28	5.77
	Smoking	Alcohol	Physical Activity	Diet Score	Family History	\		
0	0	0	-0.20	3.54		0		
1	0	0	8.12	5.90		0		
2	0	0	5.01	4.65		1		
3	0	0	3.16	3.37		0		
4	0	1	3.56	3.40		0		
...	...	...	...	...	...	...		
29995	0	0	9.15	10.35		0		
29996	1	0	3.34	3.40		0		
29997	0	0	4.53	4.50		0		
29998	0	0	3.16	2.59		0		
29999	1	0	4.91	5.82		1		
	Stress	Level	Sleep Hours	random_notes	noise_col			
0	5.07	6.05	lorem	-137.057211				
1	5.87	7.72	ipsum	-11.230610				
2	3.09	4.82	ipsum	98.331195				
3	3.01	5.33	lorem	44.187175				
4	6.38	6.64	lorem	44.831426				
...	...	...	...	...	...	...		
29995	3.65	8.71	ipsum	89.573837				
29996	6.72	2.81	lorem	-5.864517				
29997	4.31	3.61	###	-30.934595				
29998	7.10	5.80	ipsum	5.085258				
29999	3.64	5.47	lorem	-142.559559				

[30000 rows x 20 columns]

### Verificando outliers

```
[6]: import matplotlib.pyplot as plt
import seaborn as sns
colunas_numericas = ['Age','Glucose','Blood Pressure','BMI','Oxygen\u2192Saturation','LengthOfStay','Cholesterol','Triglycerides','HbA1c',
'Physical Activity','Diet Score','Stress Level','Sleep Hours',]

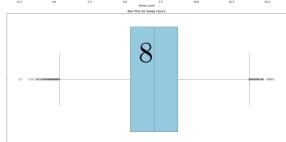
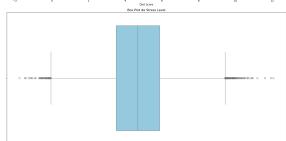
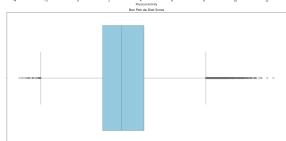
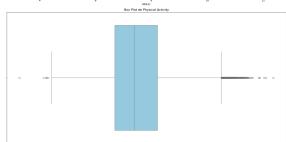
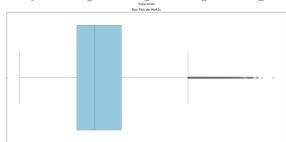
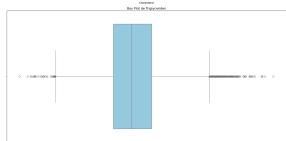
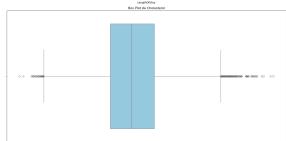
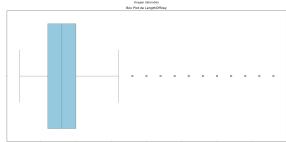
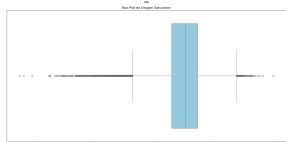
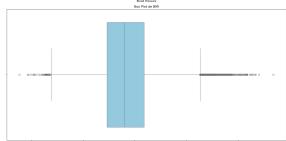
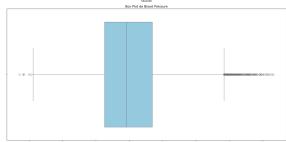
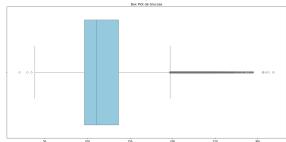
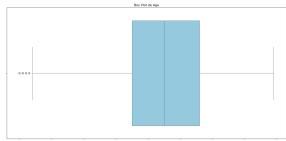
nrows = 13
ncols = 1
fig, axes = plt.subplots(nrows, ncols, figsize=(15, 100))

axes = axes.flatten()

for i, col in enumerate(colunas_numericas):
    sns.boxplot(x=df[col], ax=axes[i], color='skyblue')
    axes[i].set_title(f'Box Plot de {col}', fontsize=12)
```

```
axes[i].set_ylabel('')

plt.tight_layout()
plt.show()
```



Em todas as variáveis existe a presença de outliers. Nesse caso, aparentemente representam condições médicas extremas em vez de erros no dataset

```
[7]: print("Nulos (%):\n", df.isnull().mean().sort_values(ascending=False))
```

```
Nulos (%):
Age           0.15
Medical Condition  0.15
Glucose        0.15
Blood Pressure   0.15
Gender          0.15
Alcohol         0.00
random_notes    0.00
Sleep Hours     0.00
Stress Level    0.00
Family History   0.00
Diet Score      0.00
Physical Activity 0.00
HbA1c          0.00
Smoking         0.00
Triglycerides   0.00
Cholesterol     0.00
LengthOfStay    0.00
Oxygen Saturation 0.00
BMI             0.00
noise_col       0.00
dtype: float64
```

**Tratamento de valores nulos** As variáveis numéricas foram preenchidas com a mediana, já que praticamente todas as colunas possuem outliers que distorceriam a média

```
[13]: colunas_numericas_nulas = ['Age', 'Glucose', 'Blood Pressure']

for col in colunas_numericas_nulas:
    mediana = df[col].median()
    df[col] = df[col].fillna(mediana)
```

O gênero foi preenchido com a moda

```
[18]: moda_gender = df['Gender'].mode()[0]
df['Gender'].fillna(moda_gender, inplace=True)
```

Os registros cujo valor da variável alvo eram nulos foram removidas. Escolhemos assim para evitar que o modelo aprendesse padrões “inventados” pelo preenchimento das linhas

```
[ ]: df.dropna(subset=['Medical Condition'], inplace=True)
```

```
[19]: print("Nulos (%):\n", df.isnull().mean().sort_values(ascending=False))
```

```
Nulos (%):
Age           0.0
Gender        0.0
random_notes  0.0
Sleep Hours   0.0
Stress Level  0.0
Family History 0.0
Diet Score    0.0
Physical Activity 0.0
Alcohol        0.0
Smoking        0.0
HbA1c          0.0
Triglycerides 0.0
Cholesterol    0.0
LengthOfStay   0.0
Oxygen Saturation 0.0
BMI            0.0
Blood Pressure 0.0
Glucose         0.0
Medical Condition 0.0
noise_col       0.0
dtype: float64
```

**Realizando caping nos outliers** Para evitar que os outliers afetem o resultado do K-means, criamos outro dataframe em que os outliers estão mais próximos possível do valor interquartil mais extremo

```
[20]: def treat_outliers_iqr(df, column):

    Q1 = df[column].quantile(0.25)
    Q3 = df[column].quantile(0.75)
    IQR = Q3 - Q1

    limite_inferior = Q1 - 1.5 * IQR
    limite_superior = Q3 + 1.5 * IQR

    df[column] = df[column].apply(lambda x: limite_superior if x > limite_superior else x)

    df[column] = df[column].apply(lambda x: limite_inferior if x < limite_inferior else x)

    return df

for col in colunas_numericas:
```

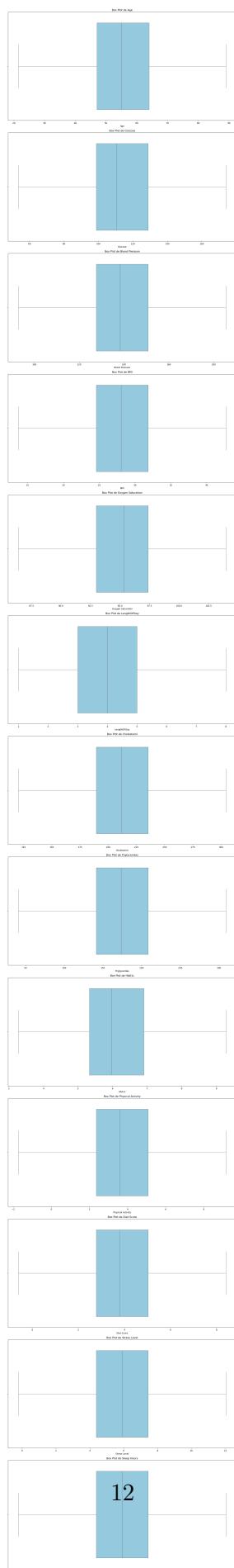
```
df_cap = treat_outliers_iqr(df, col)

[21]: nrows = 13
ncols = 1
fig, axes = plt.subplots(nrows, ncols, figsize=(15, 100))

axes = axes.flatten()

for i, col in enumerate(colunas_numericas):
    sns.boxplot(x=df_cap[col], ax=axes[i], color='skyblue')
    axes[i].set_title(f'Box Plot de {col}', fontsize=12)
    axes[i].set_ylabel('')

plt.tight_layout()
plt.show()
```



Até o momento, realizamos as etapas críticas de pré-processamento para garantir que os dados estejam limpos e estáveis para a Clusterização (K-Means). Inicialmente, removemos 4500 linhas que possuíam valores nulos na variável-alvo Medical Condition para preservar a integridade do diagnóstico final. Em seguida, tratamos os valores nulos nas features numéricas usando a mediana e a variável Gender utilizando a moda. Por último, aplicamos a técnica de Capping nas colunas numéricas, pois todas continham outliers, utilizando a regra do 1.5xIQR, para mitigar o impacto de casos médicos extremos no K-Means, garantindo que o modelo de Clusterização não seja distorcido por valores muito distantes.

### Observando correlações

```
[22]: df_eda = df.copy()

df_eda['Medical_Condition_Encoded'] = df_eda['Medical Condition'].  
    ↪astype('category').cat.codes

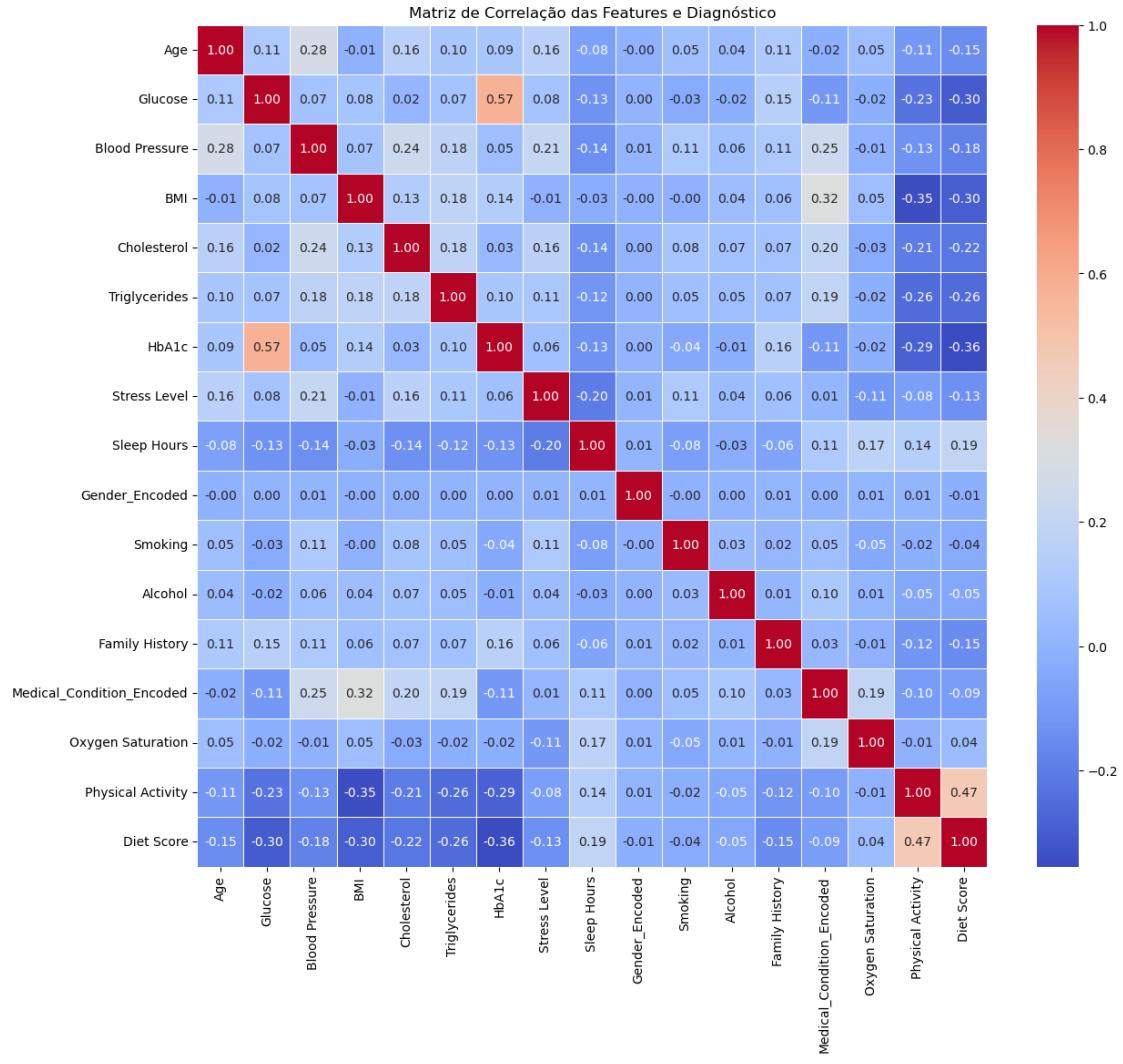
df_eda['Gender_Encoded'] = df_eda['Gender'].astype('category').cat.codes

df_eda['Smoking'] = df_eda['Smoking'].astype('int64')
df_eda['Alcohol'] = df_eda['Alcohol'].astype('int64')
df_eda['Family History'] = df_eda['Family History'].astype('int64')

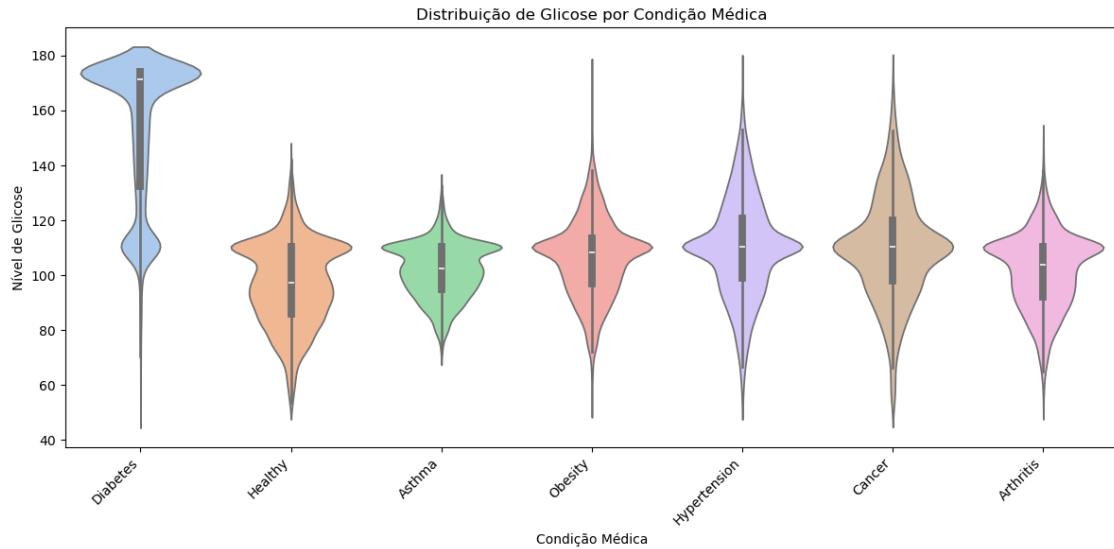
features_numericas = [  
    'Age', 'Glucose', 'Blood Pressure', 'BMI', 'Cholesterol',  
    'Triglycerides', 'HbA1c', 'Stress Level', 'Sleep Hours',  
    'Gender_Encoded', 'Smoking', 'Alcohol', 'Family History',  
    'Medical_Condition_Encoded', 'Oxygen Saturation', 'Physical Activity', 'Diet  
    ↪Score'  
]

corr_matrix = df_eda[features_numericas].corr()

[23]: plt.figure(figsize=(14, 12))  
sns.heatmap(corr_matrix, annot=True, cmap='coolwarm', fmt=".2f", linewidths=.5)  
plt.title('Matriz de Correlação das Features e Diagnóstico')  
plt.show()
```

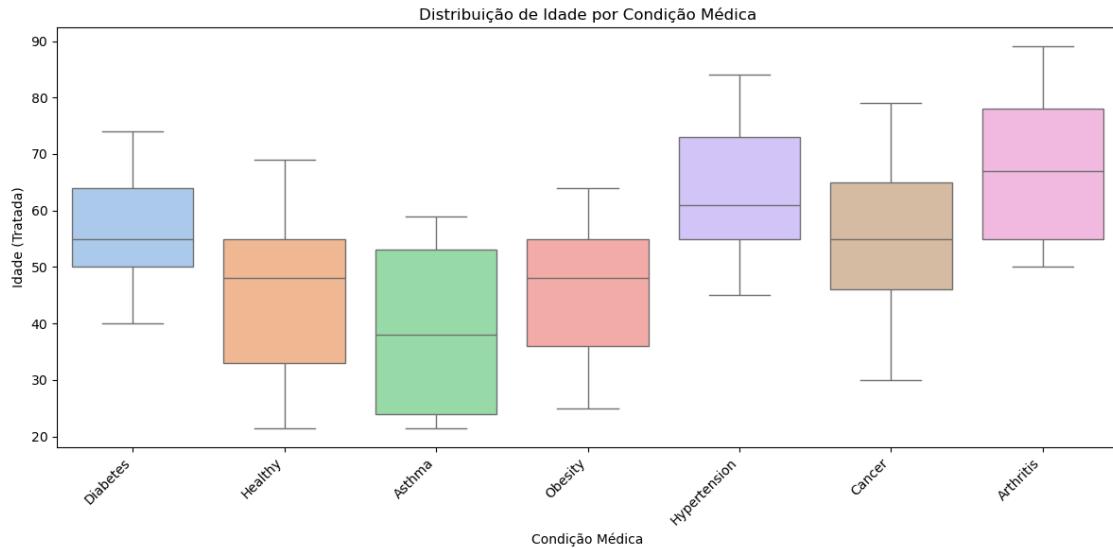


```
[24]: plt.figure(figsize=(12, 6))
sns.violinplot(x='Medical Condition', y='Glucose', data=df_eda, hue='Medical Condition', palette='pastel', legend=False)
plt.title('Distribuição de Glicose por Condição Médica')
plt.xlabel('Condição Médica')
plt.ylabel('Nível de Glicose')
plt.xticks(rotation=45, ha='right')
plt.tight_layout()
plt.show()
```



O gráfico de violino mostra que a distribuição de glicose está mais concentrada e dentro da faixa normal no grupo saudável, e mais baixa e estreita no grupo “Asthma”, enquanto o grupo “diabetes” exibe a maior amplitude e a maior concentração de níveis de glicose na parte superior, refletindo a natureza da doença. Os grupos com condições como obesidade, hipertensão, câncer,” e artrite apresentam medianas de glicose visivelmente mais elevadas e distribuições mais largas do que o grupo saudável, sugerindo uma forte correlação entre essas doenças e a prevalência de níveis de glicose elevados, possivelmente indicando risco metabólico aumentado ou a influência de comorbidades e tratamentos.

```
[60]: plt.figure(figsize=(12, 6))
sns.boxplot(x='Medical Condition', y='Age', data=df_eda, hue='Medical_Condition', palette='pastel', legend=False)
plt.title('Distribuição de Idade por Condição Médica')
plt.xlabel('Condição Médica')
plt.ylabel('Idade (Tratada)')
plt.xticks(rotation=45, ha='right')
plt.tight_layout()
plt.show()
```



O gráfico de Distribuição de Idade por Condição Médica mostra que Asma é a condição associada ao grupo mais jovem, com a mediana de Idade mais baixa, enquanto Artrite, Hipertensão e Câncer afetam predominantemente os mais velhos, exibindo as medianas mais altas, o que é esperado para doenças ligadas ao envelhecimento. Já os grupos Healthy e Obesidade possuem distribuições e medianas de Idade semelhantes, indicando uma faixa etária mais ampla, e Diabetes fica um pouco acima. Em resumo, a Idade é um fator fortemente distinto entre as Condições Médicas, com Asma no extremo mais jovem e Artrite no extremo mais velho.

```
[26]: import pandas as pd

contagem_diagnosticos_por_genero = df_eda.groupby('Gender')[['Medical Condition']].value_counts()

print("Contagem Agrupada de Diagnósticos por Gênero:")
print(contagem_diagnosticos_por_genero)
```

Contagem Agrupada de Diagnósticos por Gênero:

	Gender	Medical Condition	Count
Female	Female	Hypertension	4088
		Diabetes	3672
		Obesity	2242
		Healthy	1783
		Asthma	1189
		Arthritis	1043
		Cancer	708
Male	Male	Hypertension	3032
		Diabetes	2745
		Obesity	1615
		Healthy	1256

```

Asthma           848
Arthritis        753
Cancer           526
Name: count, dtype: int64

```

```
[27]: contagemgen=df_eda['Gender'].value_counts()
print(contagemgen)
```

```

Gender
Female    14725
Male      10775
Name: count, dtype: int64

```

```
[28]: proporcao_diagnosticos_por_genero = df_eda.groupby('Gender')[['Medical Condition']].value_counts(normalize=True)
proporcao_diagnosticos_por_genero_percentual = proporcao_diagnosticos_por_genero* 100

print("Proporção Percentual de Diagnósticos em Relação ao Total de Pessoas do Mesmo Gênero:")
print(proporcao_diagnosticos_por_genero_percentual.round(2).astype(str) + '%')
```

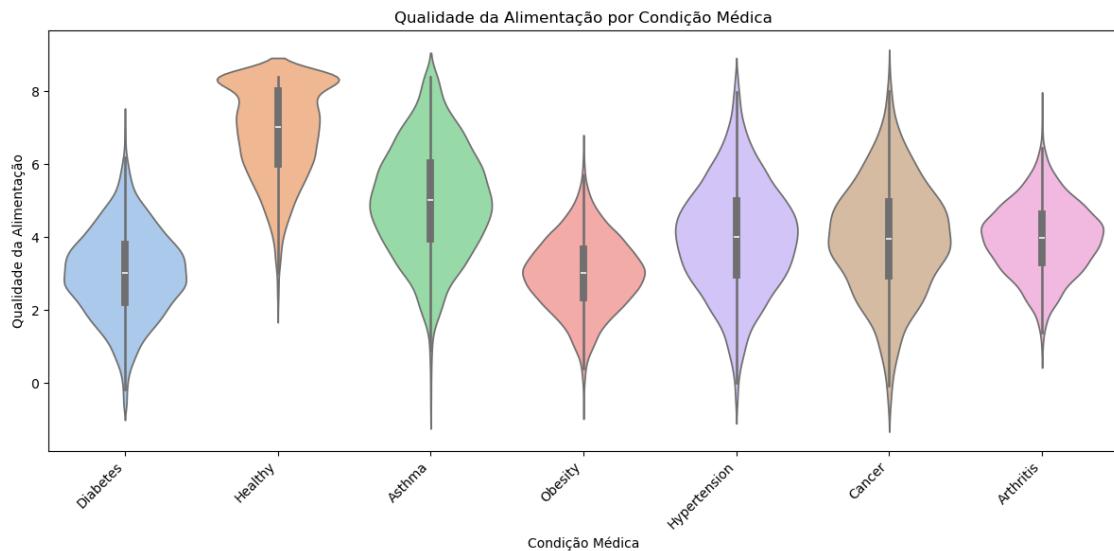
Proporção Percentual de Diagnósticos em Relação ao Total de Pessoas do Mesmo Gênero:

Gender	Medical Condition	Percentual (%)
Female	Hypertension	27.76%
	Diabetes	24.94%
	Obesity	15.23%
	Healthy	12.11%
	Asthma	8.07%
	Arthritis	7.08%
	Cancer	4.81%
Male	Hypertension	28.14%
	Diabetes	25.48%
	Obesity	14.99%
	Healthy	11.66%
	Asthma	7.87%
	Arthritis	6.99%
	Cancer	4.88%

```
Name: proportion, dtype: object
```

A análise da tabela de Proporção Percentual de Diagnósticos por Gênero mostra que tanto para o Feminino quanto para o Masculino, as condições mais prevalentes são Hipertensão e Diabetes, seguidas por Obesidade, com menos de 13% em ambos os gêneros na categoria Healthy. As proporções entre os gêneros são próximas em todas as categorias, com o Masculino apresentando prevalência ligeiramente maior em Hipertensão e Diabetes, enquanto o Feminino apresenta ligeira vantagem em Obesidade e Artrite, indicando que a carga dessas doenças crônicas é distribuída de forma quase igualitária entre homens e mulheres neste conjunto de dados.

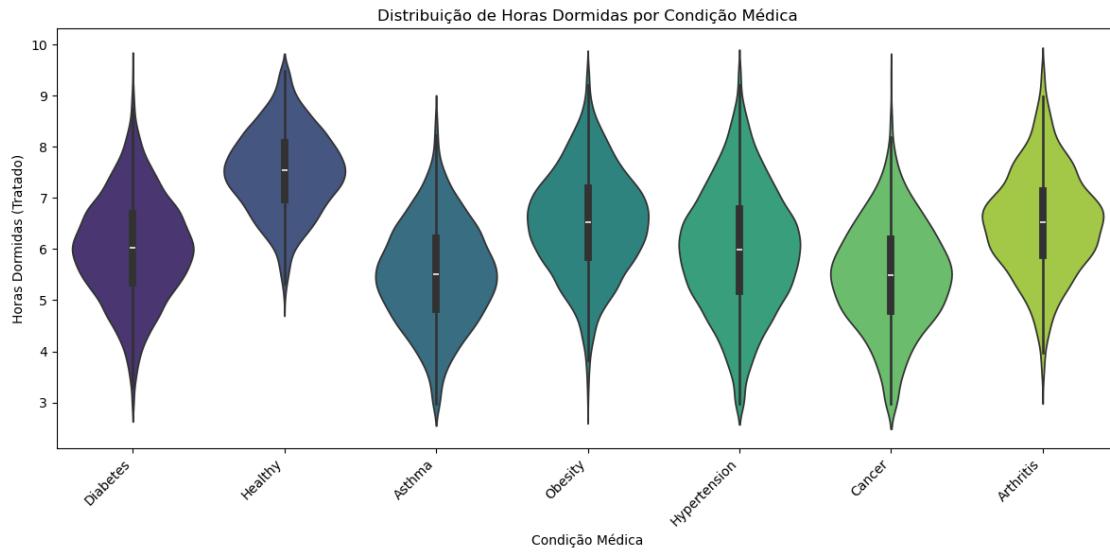
```
[61]: plt.figure(figsize=(12, 6))
sns.violinplot(x='Medical Condition', y='Diet Score', data=df_eda, hue='Medical Condition', palette='pastel', legend=False)
plt.title('Qualidade da Alimentação por Condição Médica')
plt.xlabel('Condição Médica')
plt.ylabel('Qualidade da Alimentação')
plt.xticks(rotation=45, ha='right')
plt.tight_layout()
plt.show()
```



O gráfico de violino da Distribuição de Qualidade da Alimentação por Condição Médica (onde pontuações mais altas indicam melhor qualidade) mostra que o grupo Saudável possui a melhor qualidade de alimentação, com a maior mediana e a maior concentração de dados na faixa superior da escala, sendo a distribuição mais estreita e alta. Em contraste, os grupos com condições crônicas como Diabetes e Obesidade exibem as medianas mais baixas e distribuições que se concentram na faixa intermediária-baixa (entre 2 e 4), sugerindo uma qualidade de alimentação inferior. Os grupos Asma, Hipertensão, Câncer e Artrite apresentam medianas intermediárias e distribuições mais amplas, com o Câncer e Hipertensão mostrando uma dispersão maior, mas com concentrações que indicam uma qualidade de alimentação melhor que a de Diabetes ou Obesidade, mas ainda inferior à do grupo Healthy.

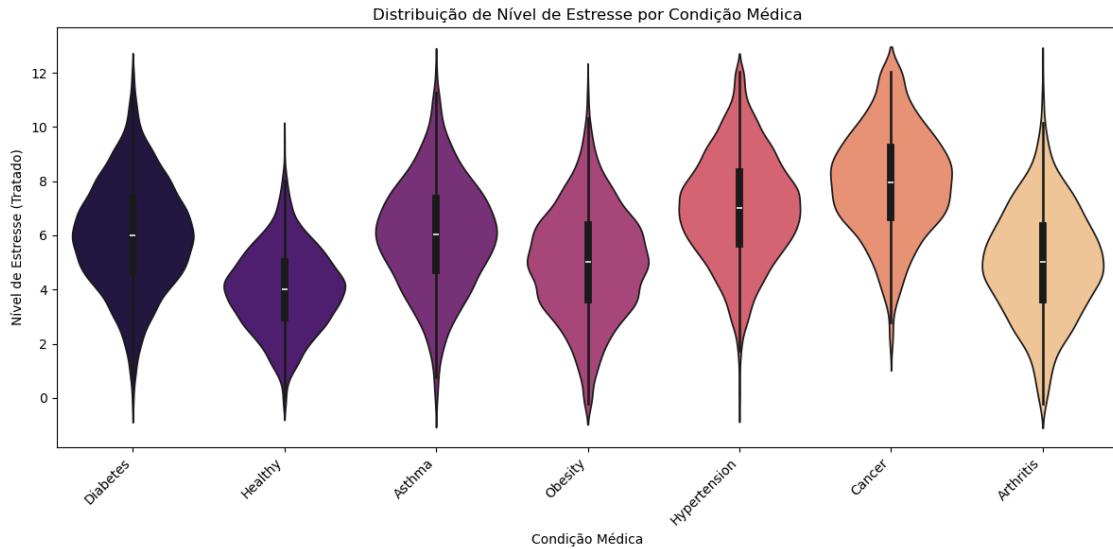
```
[30]: plt.figure(figsize=(12, 6))
sns.violinplot(x='Medical Condition', y='Sleep Hours', data=df_eda, hue='Medical Condition', palette='viridis')
plt.title('Distribuição de Horas Dormidas por Condição Médica')
plt.xlabel('Condição Médica')
plt.ylabel('Horas Dormidas (Tratado)')
plt.xticks(rotation=45, ha='right')
plt.tight_layout()
```

```
plt.show()
```



O gráfico de violino da Distribuição de Horas Dormidas por Condição Médica revela que, apesar de todos os grupos terem uma ampla faixa de horas de sono (de 3 a 10 horas), o grupo Healthy (Saudável) apresenta a mediana e a concentração de dados mais altas (perto de 7-8 horas), indicando uma média de sono melhor. Em contraste, o grupo Diabetes é o que possui a mediana mais baixa e uma distribuição notavelmente bimodal, com um pico significativo em menos de 6 horas, sugerindo que uma parcela considerável desse grupo dorme pouco. Os demais grupos, como Asma, Obesidade, Hipertensão, Câncer e Artrite, concentram suas medianas entre 6 e 7 horas de sono, com distribuições parecidas, indicando que a maioria das condições crônicas, embora apresentem medianas menores que o grupo Healthy, mantêm uma faixa de sono relativamente regular.

```
[31]: plt.figure(figsize=(12, 6))
sns.violinplot(x='Medical Condition', y='Stress Level', data=df_eda,
                hue='Medical Condition', palette='magma')
plt.title('Distribuição de Nível de Estresse por Condição Médica')
plt.xlabel('Condição Médica')
plt.ylabel('Nível de Estresse (Tratado)')
plt.xticks(rotation=45, ha='right')
plt.tight_layout()
plt.show()
```



O gráfico de violino da Distribuição de Nível de Estresse por Condição Médica demonstra que o grupo Saudável apresenta o menor Nível de Estresse. Os grupos com condições crônicas de saúde tendem a ter Níveis de Estresse mais altos, especialmente Câncer, que exibe a mediana mais elevada e uma concentração notável na faixa de alto estresse (acima de 8), seguido de perto por Hipertensão, Diabetes e Asma, cujas medianas são altas e as distribuições são largas, indicando uma alta prevalência de estresse elevado. Por fim, os grupos Obesidade e Artrite possuem medianas intermediárias, mas ainda assim visivelmente maiores que a do grupo Healthy, reforçando a conclusão de que a presença de uma condição médica está fortemente associada a um maior Nível de Estresse.

```
[63]: contagemgen=df_eda['Family History'].value_counts()
print(contagemgen)

proporcao_diagnosticos_familiar = df_eda.groupby('Family History')[['Medical Condition']].value_counts(normalize=True)
proporcao_diagnosticos_familiar_percentual = proporcao_diagnosticos_familiar * 100

print("Proporção Percentual de Diagnósticos em Relação ao Histórico Familiar:")
print(proporcao_diagnosticos_familiar_percentual.round(2).astype(str) + '%')
```

```
Family History
0    14361
1    11139
Name: count, dtype: int64
Proporção Percentual de Diagnósticos em Relação ao Histórico Familiar:
Family History  Medical Condition
0              Hypertension      24.87%
               Diabetes        18.26%
               Healthy         17.04%
```

```

          Obesity      16.27%
          Asthma       9.92%
          Arthritis    7.45%
          Cancer       6.2%
1          Diabetes     34.07%
          Hypertension 31.86%
          Obesity      13.65%
          Arthritis    6.52%
          Asthma       5.5%
          Healthy      5.31%
          Cancer       3.08%

```

Name: proportion, dtype: object

A tabela que mostra a proporção de Condições Médicas em relação ao Histórico Familiar (0 para ausente e 1 para presente) demonstra uma forte influência genética no risco de doença, pois a presença de histórico está ligada a um aumento drástico na proporção de Diabetes (subindo de 18.26% para 34.07%) e Hipertensão (de 24.87% para 31.86%), enquanto a proporção de pessoas na categoria Saudável despenca de 17.04% para apenas 5.31%. Curiosamente, as proporções de Câncer e Obesidade são menores no grupo com Histórico Familiar presente, o que pode indicar que, neste conjunto de dados, o Histórico Familiar está mais relacionado a doenças metabólicas do que a essas outras condições específicas.

```
[65]: contagemgen=df_eda['Smoking'].value_counts()
print(contagemgen)

proporcao_diagnostic_fumante = df_eda.groupby('Smoking')['Medical Condition'].
    value_counts(normalize=True)
proporcao_diagnostic_fumante_percentual = proporcao_diagnostic_fumante * 100

print("Proporção Percentual de Diagnósticos em Relação ao Total de Fumantes:")
print(proporcao_diagnostic_fumante_percentual.round(2).astype(str) + '%')
```

Smoking

0	18384
1	7116

Name: count, dtype: int64

Proporção Percentual de Diagnósticos em Relação ao Total de Fumantes:

Smoking	Medical Condition	
0	Diabetes	27.64%
	Hypertension	22.82%
	Obesity	15.92%
	Healthy	14.76%
	Arthritis	7.83%
	Asthma	7.64%
	Cancer	3.39%
1	Hypertension	41.1%
	Diabetes	18.77%
	Obesity	13.07%

Asthma	8.9%
Cancer	8.59%
Arthritis	5.0%
Healthy	4.57%

Name: proportion, dtype: object

A tabela que analisa a proporção de Condições Médicas em relação ao Tabagismo estabelece o fumo como um fator de risco, elevando a proporção de Hipertensão para 41.1% no grupo de fumantes (comparado a 22.82% nos não fumantes) e triplicando os casos de Câncer (de 3.39% para 8.59%), além de reduzir drasticamente a proporção de Healthy para apenas 4.57%. No entanto, há uma correlação inversa para o Diabetes, que é mais prevalente no grupo de Não Fumantes, o que sugere que outros fatores de risco (como a idade, o que poderia ser verificado no gráfico de idade) ou vieses no estudo podem estar mascarando a relação direta entre fumo e Diabetes.

```
[67]: contagemgen=df['Alcohol'].value_counts()
print(contagemgen)

proporcao_diagnosticos_fumante = df_eda.groupby('Alcohol')[['Medical Condition']].value_counts(normalize=True)
proporcao_diagnosticos_fumante_percentual = proporcao_diagnosticos_fumante * 100

print("Proporção Percentual de Diagnósticos em Relação ao Total de Pessoas que consomem álcool:")
print(proporcao_diagnosticos_fumante_percentual.round(2).astype(str) + '%')
```

Alcohol	Medical Condition	
0	Diabetes	26.45%
	Hypertension	25.55%
	Obesity	13.84%
	Healthy	13.27%
	Asthma	9.1%
	Arthritis	7.35%
	Cancer	4.45%
1	Hypertension	35.47%
	Diabetes	21.09%
	Obesity	19.21%
	Healthy	7.61%
	Arthritis	6.07%
	Cancer	6.07%
	Asthma	4.47%

Name: proportion, dtype: object

A análise da proporção de diagnósticos demonstra que o consumo de álcool está fortemente associado

a uma elevação drástica na prevalência de Hipertensão, que salta de 25.55% nos não consumidores para 35.47% nos consumidores, e de Obesidade, que aumenta de 13.84% para 19.21%, além de quase duplicar os casos de Câncer. A proporção de pessoas na categoria Saudável é reduzida quase pela metade no grupo consumidor de álcool, reforçando o risco. Uma exceção notável é o Diabetes, que é a condição mais prevalente entre os Não Consumidores (26.45%), mas cuja proporção cai para 21.09% nos consumidores de álcool, indicando, assim como no caso do tabagismo, uma interação complexa com outros fatores (como a idade ou a gravidade da doença) que exige análise mais aprofundada.

### 3 K-means

Após a realização da Análise Exploratória inicial, iniciamos a clusterização do nosso dataset

```
[74]: from sklearn.preprocessing import StandardScaler
from sklearn.cluster import KMeans

df['Gender_Encoded'] = df['Gender'].astype('category').cat.codes

features_para_kmeans = [
    'Age', 'Gender_Encoded', 'Glucose', 'Blood Pressure', 'BMI',
    'Oxygen Saturation', 'LengthOfStay', 'Cholesterol', 'Triglycerides',
    'HbA1c', 'Smoking', 'Alcohol', 'Physical Activity', 'Diet Score',
    'Family History', 'Stress Level', 'Sleep Hours'
]

X = df[features_para_kmeans].copy()

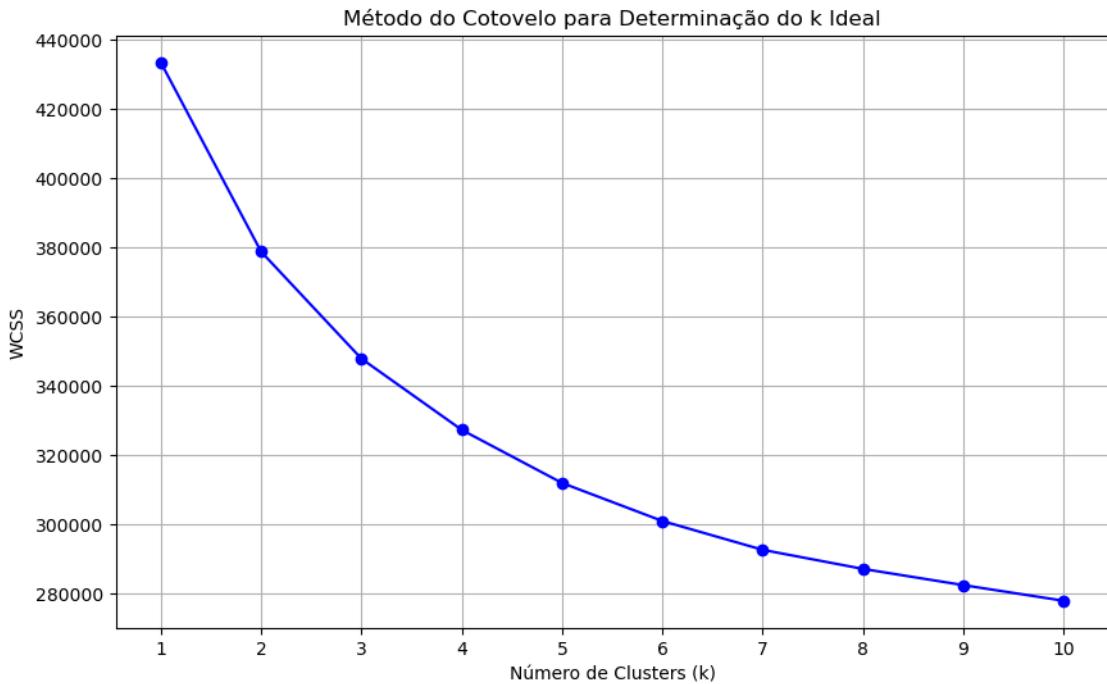
scaler = StandardScaler()
X_scaled = scaler.fit_transform(X)
```

#### Elbow Method para determinar o valor de K

```
[69]: wcss = []
k_range = range(1, 11)

for k in k_range:
    kmeans = KMeans(n_clusters=k, init='k-means++', random_state=42, n_init=10)
    kmeans.fit(X_scaled)
    wcss.append(kmeans.inertia_)

plt.figure(figsize=(10, 6))
plt.plot(k_range, wcss, marker='o', linestyle='-', color='blue')
plt.title('Método do Cotovelo para Determinação do k Ideal')
plt.xlabel('Número de Clusters (k)')
plt.ylabel('WCSS')
plt.xticks(k_range)
plt.grid(True)
plt.show()
```



No gráfico podemos observar que o momento onde a inércia deixa de variar bruscamente é quando  $k = 4$

```
[71]: k_ideal = 4

kmeans_final = KMeans(n_clusters=k_ideal, init='k-means++', random_state=42,
                       n_init=10)
cluster_labels = kmeans_final.fit_predict(X_scaled)

[70]: df['risk_profile_cluster'] = cluster_labels

print("\nDistribuição de Membros nos Perfis de Risco:")
print(df['risk_profile_cluster'].value_counts().sort_index())
```

Distribuição de Membros nos Perfis de Risco:

```
risk_profile_cluster
0    5917
1    9701
2    5863
3    4019
Name: count, dtype: int64
```

```
[76]: cluster_profile = df.groupby('risk_profile_cluster')[features_para_kmeans].mean()
```

```

cluster_profile_transposed = cluster_profile.T.round(2)

print("Tabela de Caracterização dos Perfis de Risco (Médias por Cluster):")
print(cluster_profile_transposed)

```

Tabela de Caracterização dos Perfis de Risco (Médias por Cluster):

risk_profile_cluster	0	1	2	3
Age	50.35	60.99	56.80	43.55
Gender_Encoded	0.42	0.42	0.43	0.41
Glucose	106.10	108.42	157.35	98.05
Blood Pressure	135.22	150.07	139.39	123.17
BMI	33.33	26.96	28.95	24.02
Oxygen Saturation	95.98	94.23	94.94	95.73
LengthOfStay	3.57	4.51	5.66	1.99
Cholesterol	218.09	225.13	208.62	182.75
Triglycerides	194.53	181.88	178.50	135.57
HbA1c	6.00	5.80	8.01	5.26
Smoking	0.20	0.43	0.20	0.13
Alcohol	0.28	0.29	0.19	0.14
Physical Activity	2.31	4.06	3.00	6.55
Diet Score	3.21	4.08	2.99	6.53
Family History	0.40	0.46	0.60	0.20
Stress Level	4.91	7.12	6.05	4.32
Sleep Hours	6.54	5.81	5.99	7.14

```

[109]: from sklearn.metrics import silhouette_score, silhouette_samples
from sklearn.metrics import davies_bouldin_score
from sklearn.metrics import calinski_harabasz_score
from sklearn.metrics import pairwise_distances

def avaliar_clusters(X, labels):
    print("Avaliação dos Clusters")

    unique_clusters = np.unique(labels)
    k = len(unique_clusters)
    print(f"\nQuantidade de clusters encontrados: {k}")
    print(f"Clusters: {unique_clusters}")

    print("\nTamanho de cada cluster:")
    counts = pd.Series(labels).value_counts().sort_index()
    print(counts, "\n")

    sil_score = silhouette_score(X, labels)
    print(f"Silhouette Score (geral): {sil_score:.4f} (Melhor entre -1 e 1)")

```

```

sil_amostras = silhouette_samples(X, labels)
print("\nSilhouette médio por cluster:")
for c in unique_clusters:
    media_cluster = sil_amostras[labels == c].mean()
    print(f"Cluster {c}: {media_cluster:.4f}")

db_index = davies_bouldin_score(X, labels)
print(f"\nDavies-Bouldin Index: {db_index:.4f} (Quanto menor, melhor)")

ch_score = calinski_harabasz_score(X, labels)
print(f"Calinski-Harabasz Score: {ch_score:.4f} (Quanto maior, melhor)")

print("\nDistância média intra-cluster:")
dist = pairwise_distances(X)
for c in unique_clusters:
    idx = np.where(labels == c)[0]
    if len(idx) > 1:
        dists_cluster = dist[np.ix_(idx, idx)]
        np.fill_diagonal(dists_cluster, 0)
        avg_dist = dists_cluster[np.triu_indices_from(dists_cluster, k=1)] .
        ↪mean()
        print(f"Cluster {c}: {avg_dist:.4f}")

avaliar_clusters(X_scaled, cluster_labels)

```

## Avaliação dos Clusters

Quantidade de clusters encontrados: 4

Clusters: [0 1 2 3]

Tamanho de cada cluster:

0	5917
1	9701
2	5863
3	4019

Name: count, dtype: int64

Silhouette Score (geral): 0.1141 (Melhor entre -1 e 1)

Silhouette médio por cluster:

Cluster 0:	0.0996
Cluster 1:	0.0441
Cluster 2:	0.1366

```
Cluster 3: 0.2719
```

```
Davies-Bouldin Index: 2.5710 (Quanto menor, melhor)  
Calinski-Harabasz Score: 2757.2174 (Quanto maior, melhor)
```

```
Distância média intra-cluster:
```

```
Cluster 0: 4.9552  
Cluster 1: 5.2951  
Cluster 2: 4.8818  
Cluster 3: 4.3941
```

A avaliação dos clusters mostrou resultados ruins, mas realizaremos o treino de Random Forest mesmo assim, para, posteriormente, compararmos os resultados com o modelo treinado sem utilização de clusters

### Treinamento do modelo Random Forest

```
[79]: from sklearn.model_selection import train_test_split  
from sklearn.ensemble import RandomForestClassifier  
from sklearn.preprocessing import LabelEncoder  
from sklearn.metrics import classification_report, accuracy_score  
  
features_classificacao = [  
    'Age', 'Gender_Encoded', 'Glucose', 'Blood Pressure', 'BMI',  
    'Oxygen Saturation', 'LengthOfStay', 'Cholesterol', 'Triglycerides',  
    'HbA1c', 'Smoking', 'Alcohol', 'Physical Activity', 'Diet Score',  
    'Family History', 'Stress Level', 'Sleep Hours',  
    'risk_profile_cluster'  
]  
  
X = df[features_classificacao]  
y = df['Medical Condition']  
  
le = LabelEncoder()  
y_encoded = le.fit_transform(y)
```

### Divisão em treino e teste

```
[80]: X_train, X_test, y_train, y_test = train_test_split(  
    X, y_encoded, test_size=0.2, random_state=42, stratify=y_encoded  
)  
  
rf_classifier = RandomForestClassifier(n_estimators=100, random_state=42,  
    n_jobs=-1)  
rf_classifier.fit(X_train, y_train)
```

```
[80]: RandomForestClassifier(n_jobs=-1, random_state=42)
```

### Avaliação do Modelo

```
[104]: y_pred = rf_classifier.predict(X_test)

target_names = le.classes_

print("\nAvaliação de Performance:")
print(f"Acurácia Geral: {accuracy_score(y_test, y_pred) * 100:.2f}%")
print("\nRelatório de Classificação:")
print(classification_report(y_test, y_pred, target_names=target_names))
```

Avaliação de Performance:

Acurácia Geral: 90.71%

Relatório de Classificação:

	precision	recall	f1-score	support
Arthritis	0.79	0.60	0.68	359
Asthma	0.93	0.89	0.91	407
Cancer	0.94	0.94	0.94	247
Diabetes	0.96	0.92	0.94	1284
Healthy	0.98	1.00	0.99	608
Hypertension	0.86	0.93	0.89	1424
Obesity	0.87	0.92	0.90	771
accuracy			0.91	5100
macro avg	0.90	0.88	0.89	5100
weighted avg	0.91	0.91	0.91	5100

A função accuracy\_score mede a proporção de previsões corretas feitas por um modelo de classificação em relação ao número total de previsões realizadas. Essencialmente, ela calcula a porcentagem de vezes que o modelo acertou o diagnóstico. Embora seja uma métrica simples e amplamente utilizada, ela pode ser enganosa se as classes de dados estiverem desbalanceadas, pois um modelo que sempre prevê a classe majoritária pode ainda assim apresentar uma acurácia alta, mascarando a má performance nas classes minoritárias.

O Classification Report resume o desempenho do seu modelo através de quatro métricas chave para cada classe (diagnóstico): Precisão, Recall, F1-Score e Suporte.

Precisão: a Precisão mede, de todas as vezes que o modelo previu uma classe específica, quantas vezes ele estava correto. Ela se foca nos acertos do modelo em suas próprias previsões e é calculada como  $\frac{\text{Verdadeiros Positivos}}{\text{Verdadeiros Positivos} + \text{Falsos Positivos}}$ . Por exemplo, se a Precisão para “Diabetes” for 95%, significa que de todos os pacientes que o modelo classificou como tendo Diabetes, 95% realmente tinham essa condição.

Recall: o Recall mede a capacidade do modelo de encontrar todos os casos positivos reais para uma classe específica. Ele se foca em quantos casos de uma doença existente o modelo conseguiu identificar e é calculado como  $\frac{\text{Verdadeiros Positivos}}{\text{Verdadeiros Positivos} + \text{Falsos Negativos}}$ . Um Recall de 70% para “Artrite” significa que o modelo só conseguiu detectar 70% de todos os casos de Artrite no dataset de teste, deixando 30% como Falsos Negativos.

F1-Score: o F1-Score é a média harmônica da Precisão e do Recall, combinando essas duas métricas em uma única pontuação. Ele é a métrica mais importante quando se tem um desbalanceamento significativo entre as classes, pois penaliza modelos que alcançam alta Precisão, mas baixo Recall (ou vice-versa), oferecendo uma visão mais equilibrada do desempenho geral de previsão da classe.

Suporte: o Suporte é simplesmente a contagem do número real de instâncias de cada classe que existem no seu conjunto de dados de teste. No seu caso, ele indica o número de pacientes no conjunto de teste que têm o diagnóstico real de “Diabetes”, “Hipertensão”, “Artrite”, etc., fornecendo o contexto para a robustez das outras métricas.

**Com Cap** O mesmo procedimento foi feito com o dataset após o processo de Caping dos outliers, para ver se o resultado do K-means seria afetado por essa mudança.

```
[82]: df_cap['Gender_Encoded'] = df_cap['Gender'].astype('category').cat.codes

X_cap = df_cap[features_para_kmeans].copy()
scaler_cap = StandardScaler()

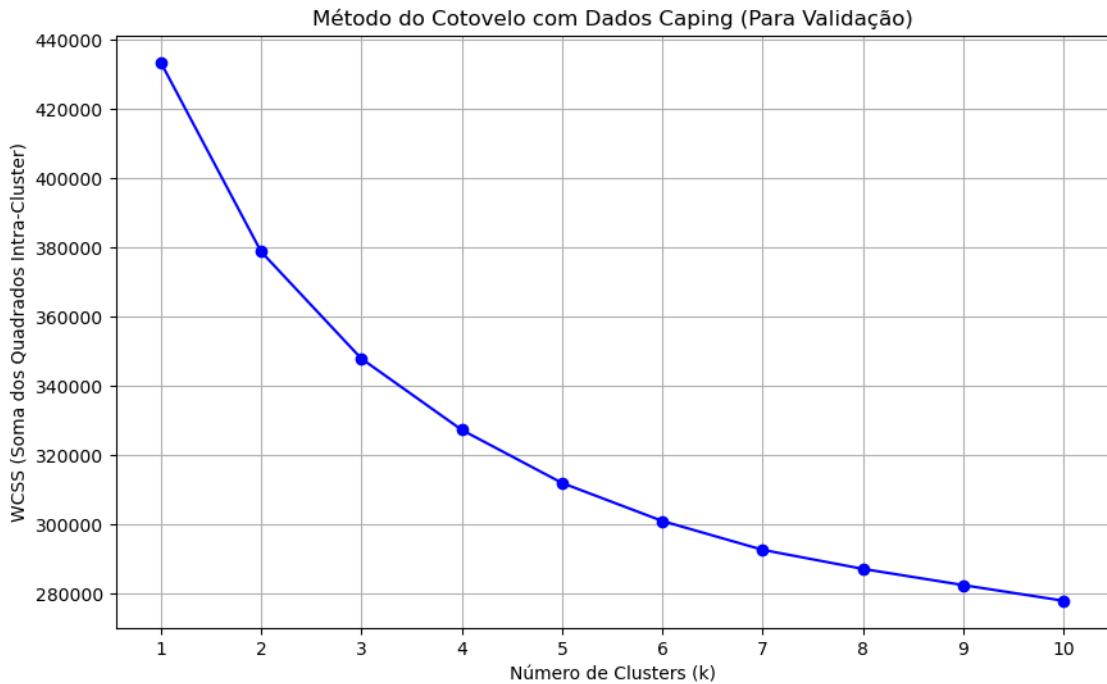
X_scaled_cap = scaler_cap.fit_transform(X_cap)
```

```
[83]: wcss_cap = []
k_range = range(1, 11)

for k in k_range:
    kmeans = KMeans(n_clusters=k, init='k-means++', random_state=42, n_init=10)
    kmeans.fit(X_scaled_cap)
    wcss_cap.append(kmeans.inertia_)

plt.figure(figsize=(10, 6))
plt.plot(k_range, wcss_cap, marker='o', linestyle='-', color='blue')
plt.title('Método do Cotovelo com Dados Caping (Para Validação)')
plt.xlabel('Número de Clusters (k)')
plt.ylabel('WCSS (Soma dos Quadrados Intra-Cluster)')
plt.xticks(k_range)
plt.grid(True)
plt.show() #

k_ideal_cap = 4
```



```
[84]: kmeans_final_cap = KMeans(n_clusters=k_ideal_cap, init='k-means++',  
    ↪random_state=42, n_init=10)  
cluster_labels_cap = kmeans_final_cap.fit_predict(X_scaled_cap)  
  
df_cap['risk_profile_cluster'] = cluster_labels_cap
```

```
[85]: features_classificacao_cap = features_para_kmeans + ['risk_profile_cluster']  
  
X_cap_rf = df_cap[features_classificacao_cap]  
y_cap_rf = df_cap['Medical Condition']  
  
le_cap = LabelEncoder()  
y_encoded_cap = le_cap.fit_transform(y_cap_rf)
```

```
[110]: X_train_cap, X_test_cap, y_train_cap, y_test_cap = train_test_split(  
    X_cap_rf, y_encoded_cap, test_size=0.2, random_state=42,  
    ↪stratify=y_encoded_cap  
)  
  
rf_classifier_cap = RandomForestClassifier(n_estimators=100, random_state=42,  
    ↪n_jobs=-1)  
  
rf_classifier_cap.fit(X_train_cap, y_train_cap)
```

```
[110]: RandomForestClassifier(n_jobs=-1, random_state=42)
```

```
[103]: y_pred_cap = rf_classifier_cap.predict(X_test_cap)

target_names_cap = le_cap.classes_

print("\nAvaliação de Performance:")
print(f"Acurácia Geral: {accuracy_score(y_test_cap, y_pred_cap) * 100:.2f}%")
print("\nRelatório de Classificação:")
print(classification_report(y_test_cap, y_pred_cap,
                            target_names=target_names_cap))
```

Avaliação de Performance:

Acurácia Geral: 90.71%

Relatório de Classificação:

	precision	recall	f1-score	support
Arthritis	0.79	0.60	0.68	359
Asthma	0.93	0.89	0.91	407
Cancer	0.94	0.94	0.94	247
Diabetes	0.96	0.92	0.94	1284
Healthy	0.98	1.00	0.99	608
Hypertension	0.86	0.93	0.89	1424
Obesity	0.87	0.92	0.90	771
accuracy			0.91	5100
macro avg	0.90	0.88	0.89	5100
weighted avg	0.91	0.91	0.91	5100

### 3.0.1 Sem K-means

Após a realização dos treinamentos utilizando as clusterizações como uma variável extra, foi feito o treinamento do modelo sem o K-means, para ver como os resultados seriam.

```
[88]: features_base = [
    'Age', 'Gender_Encoded', 'Glucose', 'Blood Pressure', 'BMI',
    'Oxygen Saturation', 'LengthOfStay', 'Cholesterol', 'Triglycerides',
    'HbA1c', 'Smoking', 'Alcohol', 'Physical Activity', 'Diet Score',
    'Family History', 'Stress Level', 'Sleep Hours'
]

X_base = df[features_base]
y_base_encoded = y_encoded
```

```
[89]: X_train_base, X_test_base, y_train_base, y_test_base = train_test_split(
    X_base, y_base_encoded, test_size=0.2, random_state=42, u
    ↪stratify=y_base_encoded
)

rf_classifier_base = RandomForestClassifier(n_estimators=100, random_state=42, u
    ↪n_jobs=-1)

rf_classifier_base.fit(X_train_base, y_train_base)
```

[89]: RandomForestClassifier(n\_jobs=-1, random\_state=42)

```
[102]: y_pred_base = rf_classifier_base.predict(X_test_base)

target_names_cap = le.classes_

print("\nAvaliação de Performance")
print(f"Acurácia Geral: {accuracy_score(y_test_base, y_pred_base) * 100:.2f}%")
print("\nRelatório de Classificação:")
print(classification_report(y_test_base, y_pred_base,
    ↪target_names=target_names_cap))
```

Avaliação de Performance  
Acurácia Geral: 91.22%

Relatório de Classificação:

	precision	recall	f1-score	support
Arthritis	0.80	0.61	0.69	359
Asthma	0.93	0.89	0.91	407
Cancer	0.95	0.94	0.95	247
Diabetes	0.96	0.94	0.95	1284
Healthy	0.99	0.99	0.99	608
Hypertension	0.86	0.94	0.90	1424
Obesity	0.90	0.90	0.90	771
accuracy			0.91	5100
macro avg	0.91	0.89	0.90	5100
weighted avg	0.91	0.91	0.91	5100

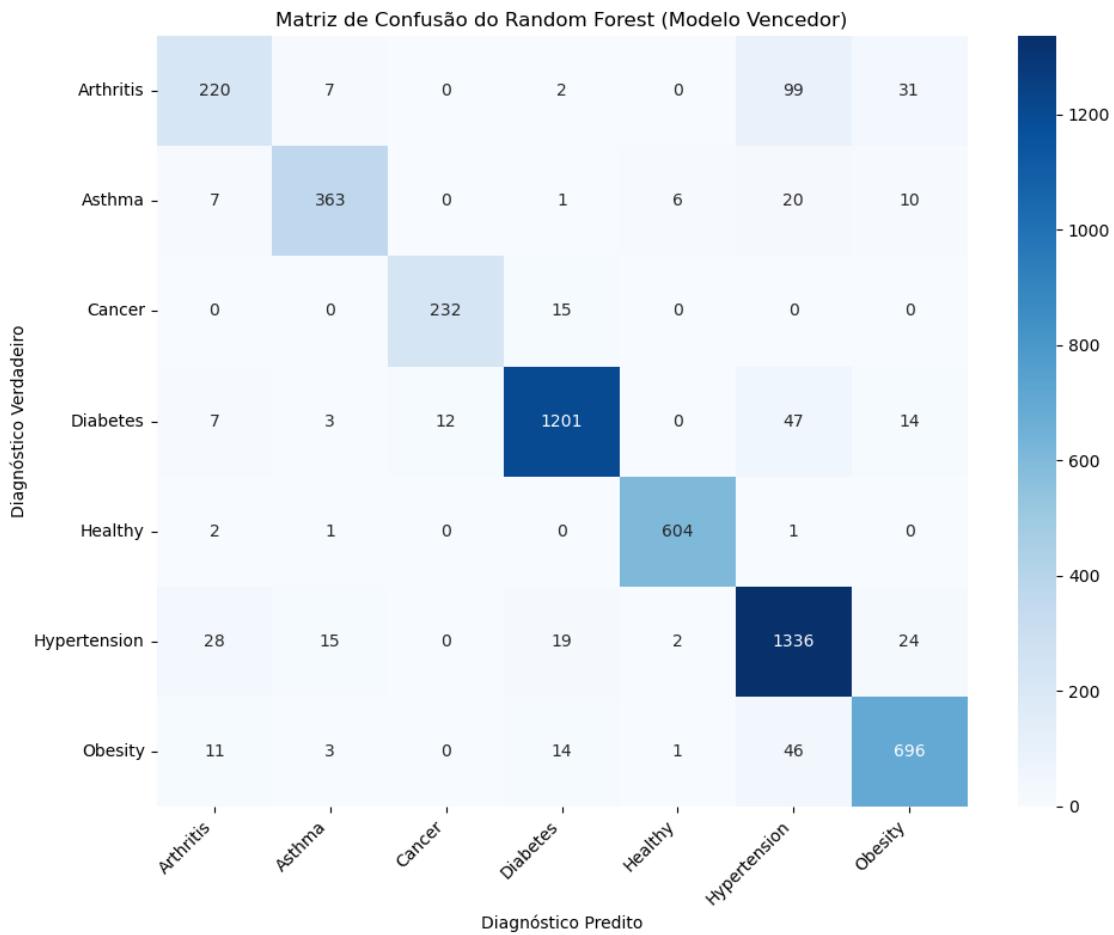
Após o treinamento do modelo sem a utilização de K-means, os resultados foram melhores

### 3.0.2 Matriz de Confusão

A Matriz de Confusão é uma tabela de resumo usada para avaliar o desempenho de modelos de classificação, comparando as previsões do modelo com os valores reais (verdadeiros) dos dados. Ela organiza os resultados em quatro categorias principais (Verdadeiros Positivos, Verdadeiros Negativos, Falsos Positivos e Falsos Negativos), onde a diagonal principal representa todos os acertos e as células fora da diagonal indicam os erros, mostrando exatamente quais classes o modelo está confundindo.

```
[100]: from sklearn.metrics import confusion_matrix
cm = confusion_matrix(y_test_base, y_pred_base)

plt.figure(figsize=(10, 8))
sns.heatmap(cm, annot=True, □
    ↪fmt='d', cmap='Blues', xticklabels=target_names_cap, yticklabels=target_names_cap)
plt.title('Matriz de Confusão do Random Forest (Modelo Vencedor)')
plt.ylabel('Diagnóstico Verdadeiro')
plt.xlabel('Diagnóstico Predito')
plt.xticks(rotation=45, ha='right')
plt.yticks(rotation=0)
plt.tight_layout()
plt.show()
```



### 3.0.3 AUC (Area Under the Curve)

A AUC (Area Under the Curve) é a métrica padrão para avaliar a performance de modelos de classificação que produzem probabilidades, como o Random Forest, fornecendo um valor entre 0 e 1. Ela mede a área total sob a Curva ROC (Receiver Operating Characteristic), que é um gráfico que plota a Taxa de Verdadeiros Positivos (Sensibilidade) versus a Taxa de Falsos Positivos ( $1 - \text{Especificidade}$ ) para todos os limiares de classificação possíveis. Essencialmente, a AUC representa a probabilidade de o modelo classificar corretamente um par aleatório de exemplos (um positivo e um negativo), sendo que valores próximos de 1.0 indicam um poder de distinção excelente entre as classes.

```
[99]: from sklearn.metrics import roc_auc_score
from sklearn.preprocessing import label_binarize
import numpy as np

classes = np.unique(y_test_base)
y_test_binarized = label_binarize(y_test_base, classes=classes)
```

```

y_proba_base = rf_classifier_base.predict_proba(X_test_base)

auc = roc_auc_score(y_test_binarized, y_proba_base, average='micro')

print(f"AUC: {auc:.4f}")

```

AUC: 0.9944

A AUC do nosso modelo é de 0.9944, um resultado excelente.

### Avaliando os atributos mais importantes

```

[111]: importances = rf_classifier_base.feature_importances_

feature_importance_df = pd.DataFrame({
    'Feature': features_base,
    'Importance': importances
})

feature_importance_df = feature_importance_df.sort_values(by='Importance', ↴
                                                               ascending=False)

print("10 Principais Fatores de Risco (Random Forest Base)")
print(feature_importance_df.head(10).round(4))

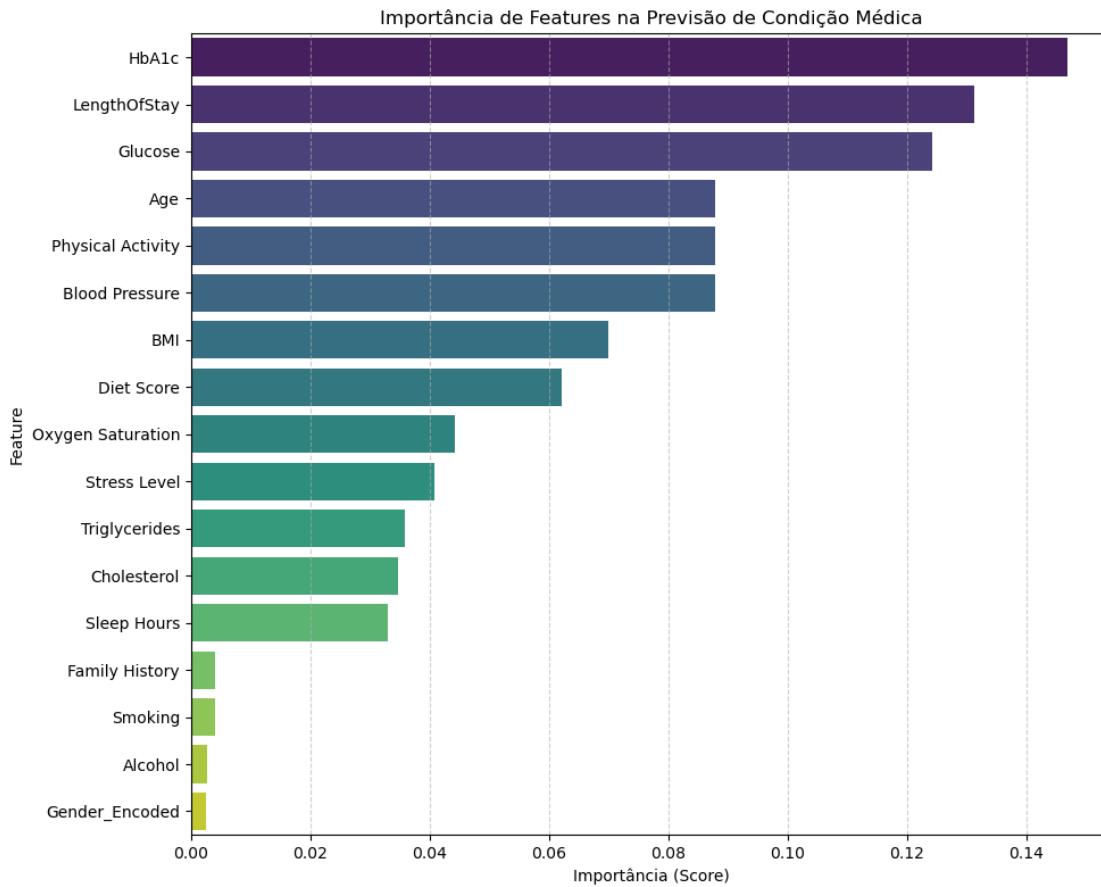
```

	Feature	Importance
9	HbA1c	0.1469
6	LengthOfStay	0.1313
2	Glucose	0.1241
0	Age	0.0879
12	Physical Activity	0.0879
3	Blood Pressure	0.0878
4	BMI	0.0699
13	Diet Score	0.0621
5	Oxygen Saturation	0.0443
15	Stress Level	0.0408

```

[96]: plt.figure(figsize=(10, 8))
sns.barplot(x='Importance', y='Feature', data=feature_importance_df, ↴
            palette='viridis', hue='Feature', legend=False)
plt.title('Importância de Features na Previsão de Condição Médica')
plt.xlabel('Importância (Score)')
plt.ylabel('Feature')
plt.grid(axis='x', linestyle='--', alpha=0.6)
plt.tight_layout()
plt.show()

```



### 3.1 Preparando os arquivos para o deploy no Streamlit

```
[108]: features_base_nova = [
    'Age', 'Gender_Encoded', 'Glucose', 'Blood Pressure', 'BMI',
    'Oxygen Saturation', 'LengthOfStay', 'Cholesterol', 'Triglycerides',
    'HbA1c', 'Smoking', 'Alcohol', 'Physical Activity', 'Diet Score',
    'Family History', 'Stress Level', 'Sleep Hours'
]

X_base_novo = df_cap[features_base_nova].copy()

scaler_base_novo = StandardScaler()
X_base_scaled_novo = scaler_base_novo.fit_transform(X_base_novo)

X_train_scaled, _, y_train, _ = train_test_split(
    X_base_scaled_novo, y_encoded_cap, test_size=0.2, random_state=42,
    stratify=y_encoded_cap
)
```

```
rf_classifier_base_novo = RandomForestClassifier(n_estimators=100,  
    ↪random_state=42, n_jobs=-1)  
rf_classifier_base_novo.fit(X_train_scaled, y_train)  
  
joblib.dump(rf_classifier_base_novo, 'modelo_random_forest.pkl')  
joblib.dump(scaler_base_novo, 'scaler_base.pkl')  
joblib.dump(le_cap, 'label_encoder.pkl')
```

[108]: ['label\_encoder.pkl']

Os arquivos contendo os modelos foram exportados para serem utilizados pelo app.py