PONTIFÍCIA UNIVERSIDADE CATÓLICA DE MINAS GERAIS
NÚCLEO DE EDUCAÇÃO A DISTÂNCIA
PÓS-GRADUAÇÃO LATO SENSU EM CIÊNCIA DE DADOS E BIG DATA

TAINARA GUADAGNIN MARCON

ESTUDO SOBRE O CONSUMO DE MEDICAMENTOS EM UM HOSPITAL NO RIO GRANDE DO SUL

CAXIAS DO SUL
2023

TAINARA GUADAGNIN MARCON

ESTUDO SOBRE O CONSUMO DE MEDICAMENTOS EM UM HOSPITAL NO RIO GRANDE DO SUL

Trabalho de Conclusão de Curso apresentado ao Curso de Especialização em Ciência de Dados e Big Data como requisito parcial à obtenção do título de especialista.

CAXIAS DO SUL

2023

LISTA DE FIGURAS

LISTA DE TABELAS

SUMÁRIO

1. INTRODUÇÃO

1.1 CONTEXTUALIZAÇÃO

Hospitais são instituições complexas onde o suprimento de medicamentos e materiais hospitalares são os principais desafios da administração de uma instituição de saúde (WOLKER, 2019).

Em virtude deste cenário, este estudo tem como objetivo analisar o histórico de consumo de medicamentos durante o ano de 2019 e relacionar com a quantidade de pacientes recebidos durante 2019. Após esta análise, verificar o comportamento dos modelos de *machine learning* para a classificação do consumo. O histórico dos dados foi de 2019 pois os anos de 2020 a 2022 foram anos de pandemia, portanto o cenário de medicamentos e pacientes estaria distorcido.

Para este estudo foram utilizadas as técnicas de processamento e análise de dados e os modelos de m*achine learning* de classificação utilizando o Google Colab.

1.2 PROBLEMA PROPOSTO

O estudo tem como objetivo analisar e identificar o comportamento do consumo de medicamentos em um hospital do Rio Grande do Sul. Assim, este estudo propõe uma análise exploratória e modelos de previsão para comparação.

Para facilitar o entendimento do problema proposto, foi aplicada a técnica de 5W abaixo:

1.2.1 (Why?) Por que esse problema é importante?

Este problema é importante pois a falta da análise de consumo de medicamentos afeta o ponto de reposição de compras, podendo ocasionar perda ou aumento de estoque, desperdícios e até falta de medicamentos o que afeta no atendimento prestado ao paciente.

1.2.2 (*Who?*) De quem são os dados analisados?

Os dados analisados são referentes a um hospital do Rio Grande do Sul. A base de dados do consumo de medicamentos é oriunda do setor de logística deste hospital e a base de dados da quantidade de pacientes atendidos é referente a outro setor deste mesmo hospital

chamado de setor de internações. Ambas as bases analisadas são referentes ao ano de 2019. A escolha da base de dados ser de 2019 é devido ao período de pandemia dos anos de 2020, 2021 e alguns meses de 2022, o que distorce os dados do consumo de medicamentos.

1.2.3 (*What?*) Quais os objetivos com essa análise?

O objetivo desta análise é entender o comportamento do consumo de medicamentos diante do número de pacientes por setor e tipo de medicamento. Os dados estão estratificados por mês, de janeiro a dezembro e por área de internação. As áreas de internação são: pronto atendimento, UTI adulto, UTI pediátrica, UTI neonatal, unidade coronariana, setor 200, setor 250, setor 300, setor 350, setor 400, setor 450, centro cirúrgico e centro obstétrico.

1.2.4 (*Where?*) Quais são os aspectos geográficos relacionados à análise?

Todos os dados utilizados no presente estudo são referentes a um hospital do Rio Grande do Sul.

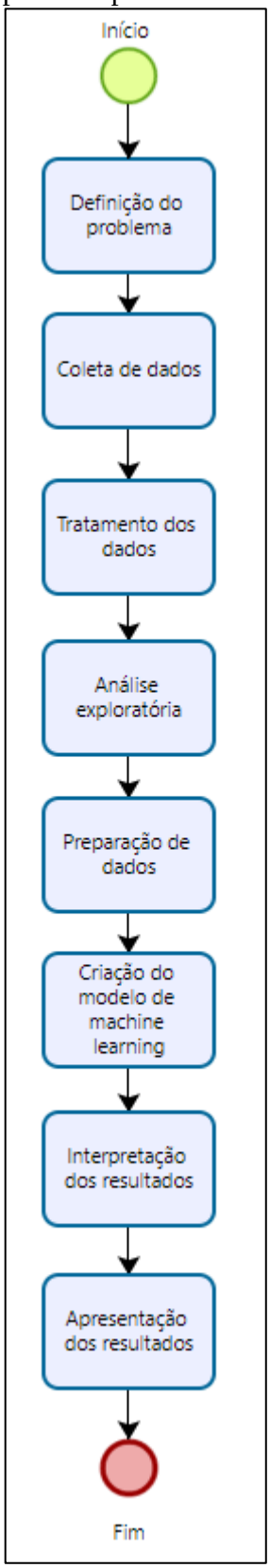
1.2.5 (*When?*) Qual o período está sendo analisado?

O período analisado é de janeiro a dezembro de 2019.

2. METODOLOGIA

O presente estudo visa identificar e classificar o consumo de medicamentos no ano de 2019 e aplicar modelos de *machine learning* para classificar o consumo de medicamentos entre baixo, moderado e alto. As etapas adotadas neste estudo são apresentadas no fluxo abaixo:

Figura 1 – Principais componentes de um sistema de IA



Fonte: Elaborado pelo autor (2023)

Foi utilizada a linguagem de programação Python desenvolvida no ambiente do Google Colab.

3. COLETA DE DADOS

Nesta etapa, foram identificadas as fontes de dados e obtidos os datasets de interesse. Os dados coletados foram extraídos de planilhas eletrônicas do setor de logística e do setor de internação de um hospital do Rio Grande do Sul que são setores distintos dentro da Instituição.

3.1 BASE DE DADOS 1: CONSUMO DE MEDICAMENTOS

A primeira parte da coleta de dados foi realizada no setor de logística, onde a área possui o histórico do consumo de medicamentos por meio de planilha eletrônica no formato Excel. Estes dados contém as seguintes informações:

Tabela 1 - Colunas da base de dados de consumo de medicamentos

Nome da coluna/campo	**Descrição**
Mês Ano	Mês e ano de consumo do medicamento
Centro de custo	Centro de custo e nome do setor que consumiu o medicamento
Conta Contábil	Conta contábil em que o medicamento está alocado
Medicamento	Tipo de medicamento
Estabelecimento	Estabelecimento do hospital
Depósito	Local de onde o medicamento está saindo
Consumo	Quantidade do medicamento consumida
Tipo de consumo	Classificação do consumo do medicamento em alto, moderado e baixo

Fonte: Elaborado pelo autor (2023)

- Mês Ano: se refere ao mês e ano do consumo do medicamento, como por exemplo jan/2019, fev/2019, mar/2019;
- Centro de custo: número e nome da área do hospital. Exemplo: 520433 - Pronto Atendimento, 520452 - Setor 200, 520472 - Centro Cirúrgico Bloco Cirúrgico;
- Conta contábil: número sequencial criado pelo setor contábil com a finalidade de registrar as movimentações geradas de determinado item;
- Medicamento: tipo de medicamento. Exemplo: analgésico, anti-inflamatório, antídotos, etc;
- Estabelecimento: pode ser qualquer unidade da instituição de saúde, como hospital, laboratório, centrais de atendimento. Porém no caso destes dados em específico todos os itens possuem estabelecimento “hospital” devido à coleta de dados ter sido realizada neste estabelecimento;
- Depósito: de qual depósito se registrou saída do medicamento;

- Consumo: quantidade do item consumido em unidades.
- Tipo de consumo: Classificação do consumo do medicamento em alto, moderado e baixo

3.2 BASE DE DADOS 2: QUANTIDADE DE PACIENTES

A segunda parte da coleta de dados foi realizada no setor de internações do mesmo hospital de Caxias do Sul, estes dados estão em formato de planilha Excel e possuem as seguintes informações:

Tabela 2 - Colunas da base de dados de quantidade de pacientes

Nome da coluna/campo	**Descrição**
Ano	Ano de internação
Mês	Mês de internação
Unidade de Atendimento	Área de internação
Quantidade total de pacientes internados	Quantidade de pacientes internados

Fonte: Elaborado pelo autor (2023)

- Ano: ano do atendimento prestado ao paciente. Nesta base de dados o ano é somente 2019 pois foi o período determinado para análise;
- Mês: meses de janeiro a dezembro de 2019;
- Unidade de atendimento: número e nome da área do hospital. Exemplo: 520433 - Pronto Atendimento, 520452 - Setor 200, 520472 - Centro Cirúrgico Bloco Cirúrgico;
- Quantidade total de pacientes internados: quantidade total de atendimentos por setor.

4. PROCESSAMENTO DE DADOS

Neste capítulo serão apresentadas as ferramentas e bibliotecas utilizadas para o processamento e o tratamento dos dados.

4.1 FERRAMENTAS UTILIZADAS

A ferramenta utilizada para o desenvolvimento dos scripts em linguagem python foi o ambiente virtual do Google Colab.

4.2 BIBLIOTECA DE DADOS

Conforme Figura 1 algumas bibliotecas foram importadas para o processamento e análise de dados

Figura 2 - Importação de bibliotecas

```
import pandas as pd
import numpy as np
import seaborn as sns
import matplotlib.pyplot as plt
import plotly.express as px
from sklearn.preprocessing import LabelEncoder
from sklearn.preprocessing import OneHotEncoder
from sklearn.compose import ColumnTransformer
```

Fonte: Google Colab (2023)

A tabela 3 abaixo mostra de forma detalhada as bibliotecas importadas.

Tabela 3 - Detalhamento das bibliotecas importadas

Biblioteca	Descrição	Comando
Pandas	Pacote de ferramentas da linguagem de programação Python para análise de dados e manipulação	import panda as pd
Numpy	Pacote de ferramentas da linguagem de programação Python para operações matemáticas em array	import numpy as np
Seaborn	Pacote de ferramentas utilizado para geração de gráficos e visualização de dados	import seaborn as sns
Matplotlib	Pacote de ferramentas da linguagem de programação Python para criação de gráficos e visualização de dados	import matplotlib.pyplot as plt
Sklearn	Pacote de ferramentas para machine learning	from sklearn.preprocessing

		import LabelEncoder from sklearn.preprocessing import OneHotEncoder from sklearn.compose import ColumnTransformer

Fonte: Elaborado pelo autor (2023)

4.3 DATASETS

Em seguida foi realizada a importação dos *datasets* escolhidos por meio do comando "pd.read_excel". A seguir o detalhamento do processamento realizado em cada um dos *datasets*.

4.3.1 Tratamento de dados *dataset* 1

O primeiro *dataset* chamado de consumo_med contém 2.613 entradas e 8 colunas.

Figura 3 – Informações *dataset* consumo_med

```
consumo_med.info()

<class 'pandas.core.frame.DataFrame'>
RangeIndex: 2613 entries, 0 to 2612
Data columns (total 8 columns):
 #   Column           Non-Null Count  Dtype
---  ------           --------------  -----
 0   MES/ANO          2613 non-null   object
 1   CENTRO DE CUSTO  2613 non-null   object
 2   CONTA CONTÁBIL   2613 non-null   object
 3   MEDICAMENTO      2613 non-null   object
 4   ESTABELECIMENTO  2613 non-null   object
 5   DEPÓSITO         2613 non-null   object
 6   CONSUMO          2613 non-null   float64
 7   TIPO CONSUMO     2613 non-null   object
dtypes: float64(1), object(7)
memory usage: 163.4+ KB
```

Fonte: Google Colab (2023)

Na fase de processamento de dados os bancos de dados foram tratados separadamente. No primeiro banco de dados de consumo de medicamentos, algumas colunas foram eliminadas do banco de dados pelos seguintes motivos:

- Conta contábil: a conta contábil não é um dado relevante para a análise de dados;

- Estabelecimento: no caso deste banco de dados o estabelecimento é somente o “hospital”, por não ser relevante foi deletado.
- Depósito: em todos os registros o depósito é sempre o mesmo, a farmácia central, portanto não é necessário avaliar este dado.

Figura 4 – Eliminando colunas desnecessárias

```
#Eliminando colunas desnecessárias
consumo_med = consumo_med.drop(['CONTA CONTÁBIL', 'ESTABELECIMENTO', 'DEPÓSITO'], axis = 1)
consumo_med.head(5)
```

Fonte: Google Colab (2023)

A seguir algumas colunas foram renomeadas para melhor identificação ao longo do estudo.

Figura 5 - Renomeando as colunas

```
#Renomeando as colunas
consumo_med = consumo_med.rename(columns={'MES/ANO':'Mes_ano','CENTRO DE CUSTO':'Setor',
                                          'MEDICAMENTO':'Tipo_medicamento',
                                          'CONSUMO':'Qtde_consumida', 'TIPO CONSUMO':'Classificacao'})
```

Fonte: Google Colab (2023)

Na sequência foi verificado se existia campos nulos no banco de dados, onde não foram identificados registros nulos.

Figura 6 – Verificação de campos nulos

```
consumo_med.isnull().sum()

Mes_ano             0
Setor               0
Tipo_medicamento    0
Qtde_consumida      0
Classificacao       0
dtype: int64
```

Fonte: Google Colab (2023)

4.4 TRATAMENTO DE DADOS BANCO DE DADOS 2

Para enriquecimento dos dados foi utilizada outra planilha em Excel de outro setor da instituição chamado de setor de internação. Essa planilha contempla a quantidade de pacientes internados por mês e por área hospitalar. este banco de dados possui 156 entradas e 4 colunas.

Figura 7 - Informações *dataset* pacientes

```
pacientes.info()

<class 'pandas.core.frame.DataFrame'>
RangeIndex: 156 entries, 0 to 155
Data columns (total 4 columns):
 #   Column                                  Non-Null Count  Dtype
---  ------                                  --------------  -----
 0   Ano                                     156 non-null    float64
 1   Mês                                     156 non-null    object
 2   Unidade de Atendimento                  156 non-null    object
 3   Quantidade Total de Pacientes Internados  156 non-null  float64
dtypes: float64(2), object(2)
memory usage: 5.0+ KB
```

No segundo banco de dados algumas colunas foram renomeadas para melhor identificação.

Figura 8 – Renomeando as colunas

```
#Renomeando as colunas base pacientes
pacientes = pacientes.rename(columns={'Mês':'Mes',
                        'Unidade de Atendimento': 'Setor',
                        'Quantidade Total de Pacientes Internados': 'Quantidade'})
```

Fonte: Google Colab (2023)

Na sequência foi verificado se existia campos nulos no banco de dados, onde não foram identificados registros nulos.

4.5 UNIÃO DOS DADOS

A junção do banco de dados de consumo de medicamentos com o banco de dados da quantidade de pacientes foi realizada utilizando a função *merge* pelo atributo “Setor” como chave.

Figura 9 - União dos dados

```
#Para juntar os dois datasets
consumo_medicamentos = pd.merge(consumo_med, pacientes, on='Setor')
consumo_medicamentos.head(5)
```

	Mes_ano	Setor	Tipo_medicamento	Qtde_consumida	Classificacao	Ano	Mes	Qtde_pacientes
0	abr/2019	520433 - Pronto Atendimento	ANALGESICOS	1388.0	Moderado	2019.0	Jan	6729.0
1	abr/2019	520433 - Pronto Atendimento	ANALGESICOS	1388.0	Moderado	2019.0	Fev	6228.0
2	abr/2019	520433 - Pronto Atendimento	ANALGESICOS	1388.0	Moderado	2019.0	Mar	7187.0
3	abr/2019	520433 - Pronto Atendimento	ANALGESICOS	1388.0	Moderado	2019.0	Abr	6847.0
4	abr/2019	520433 - Pronto Atendimento	ANALGESICOS	1388.0	Moderado	2019.0	Mai	7220.0

Fonte: Google Colab (2023)

Após a junção dos dados as informações de ano e mês ficaram repetidas pois os dois bancos de dados possuíam essas informações. Por este motivo, estas duas colunas foram eliminadas, permanecendo somente uma única coluna para o mês e ano. A ordem das colunas foi alterada para melhor análise.

Figura 10 – Tratamento de dados após união dos dados

```
#Eliminando as colunas duplicadas
consumo_medicamentos = consumo_medicamentos.drop(['Ano','Mes_y'], axis = 1)

#Renomeando as colunas base final
consumo_medicamentos = consumo_medicamentos.rename(columns={'Mes_x':'Mes'})

#Alterando a ordem das colunas
consumo_medicamentos = consumo_medicamentos[['Mes','Setor','Tipo_medicamento','Qtde_pacientes','Qtde_consumida','Classificacao']]
```

Fonte: Google Colab (2023)

5. ANÁLISE E EXPLORAÇÃO DOS DADOS

A análise e exploração de dados foi realizada de forma a analisar graficamente os dados coletados, utilizando a biblioteca Matplotlib.

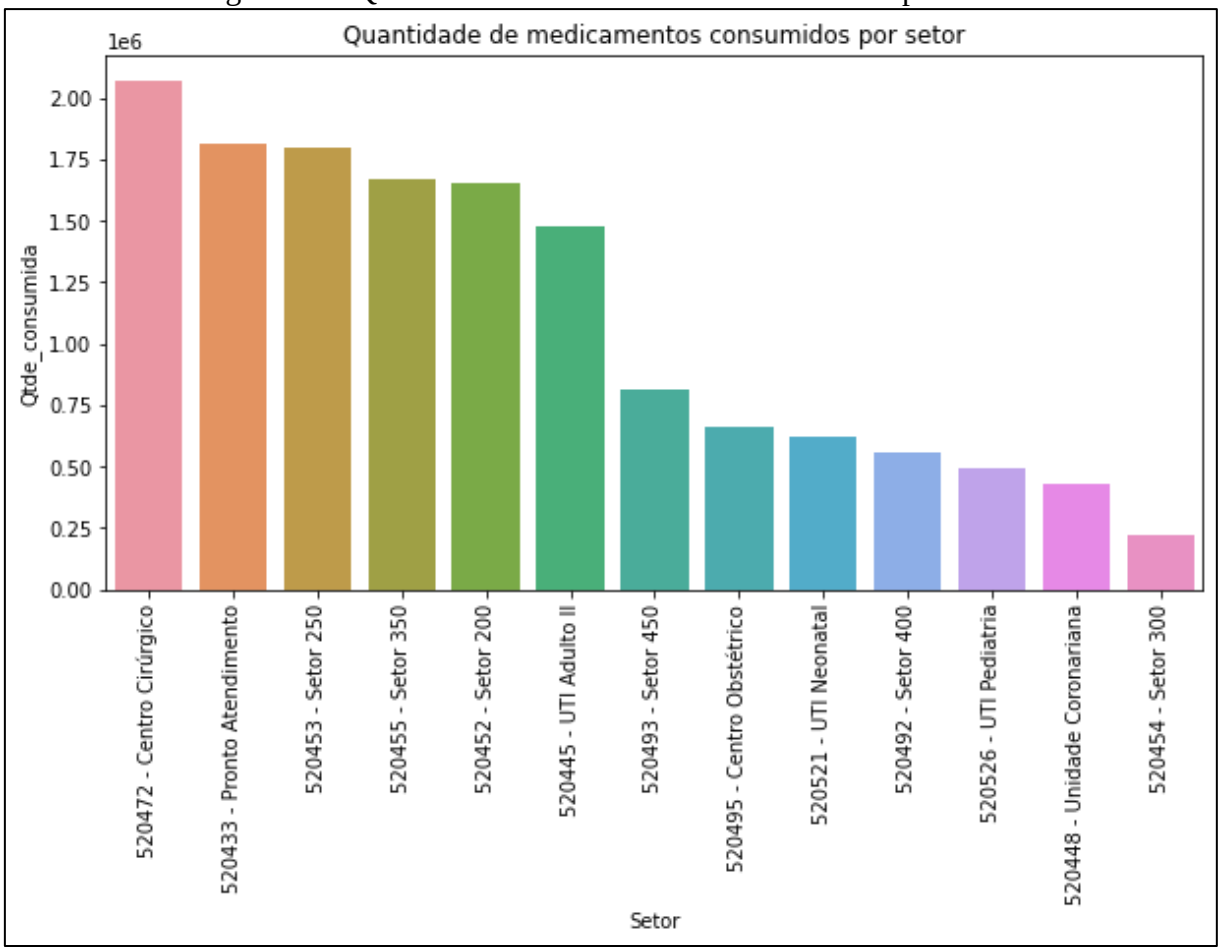
Inicialmente foi criada uma tabela com o somatório da quantidade de medicamentos consumidos por setor e assim plotado um gráfico de barras para entender qual setor possuía o maior consumo de medicamentos.

Figura 11 – Tabela do somatório da quantidade de medicamentos consumidos por setor

```
#Tabela do somatório da quantidade de medicamentos consumidos por setor
table = consumo_medicamentos.groupby('Setor')['Qtde_consumida'].sum().reset_index()
```

Fonte: Google Colab (2023)

Figura 12 – Quantidade de medicamentos consumidos por setor



Fonte: Google Colab (2023)

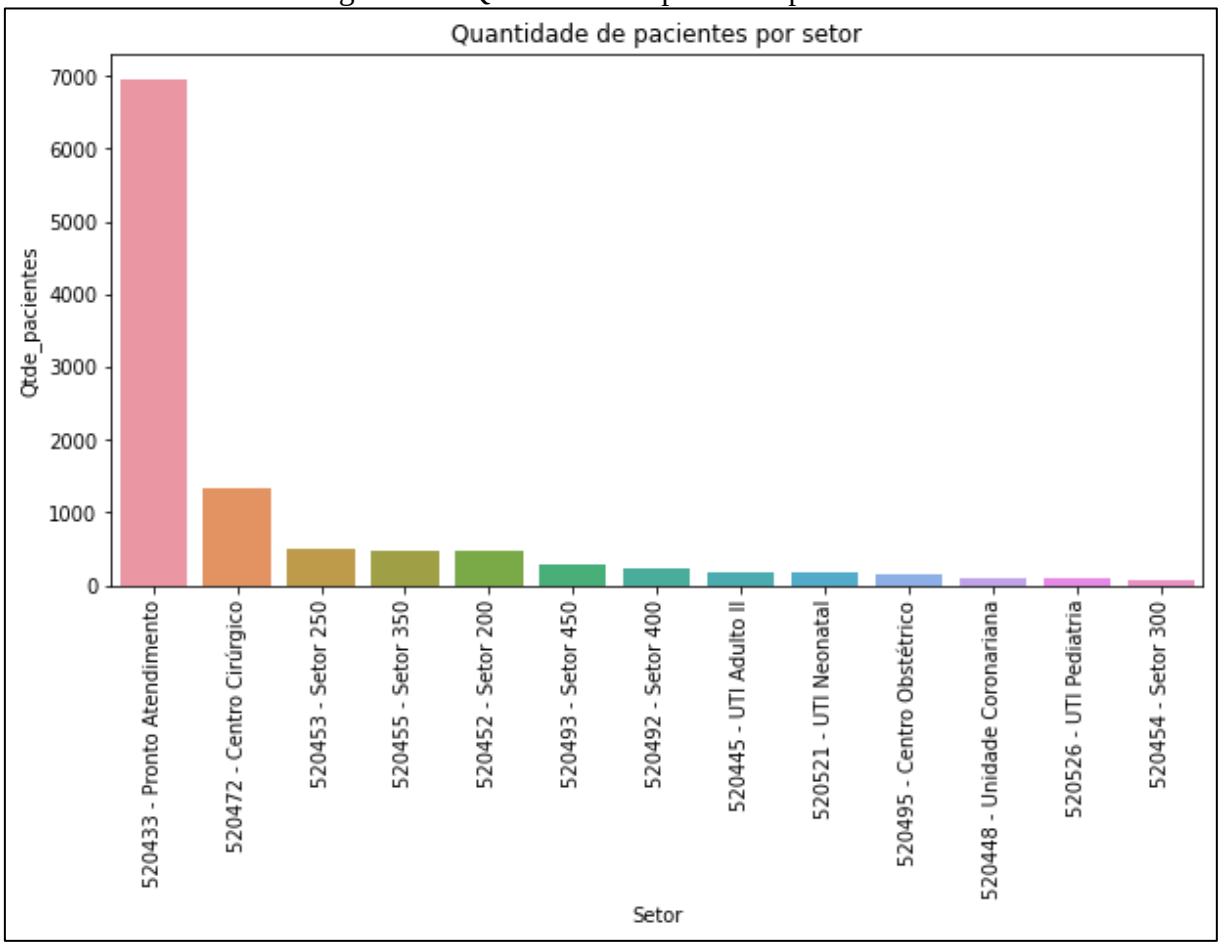
Para analisar a quantidade de medicamentos consumida por setor com a quantidade de pacientes que foram atendidos, foi criada outra tabela com a quantidade média de paciente atendidos para complementar a análise e após plotado o gráfico conforme Figura 14.

Figura 13 - Quantidade média de paciente atendidos

```
#Tabela do somatório da quantidade de pacientes atendidos por setor
table2 = consumo_medicamentos.groupby('Setor')['Qtde_pacientes'].sum().reset_index()
table2
```

Fonte: Google Colab (2023)

Figura 14 – Quantidade de pacientes por setor



Fonte: Google Colab (2023)

Por meio destes dois gráficos, é possível identificar que o setor que consome mais medicamentos é o centro cirúrgico, seguido pelo pronto atendimento, setores de internação 250, 350, 200 e UTI adulto II respectivamente. No segundo gráfico, é possível identificar que o setor que possui maior quantidade média de atendimentos é o pronto atendimento, seguido pelo centro cirúrgico e setores de internação.

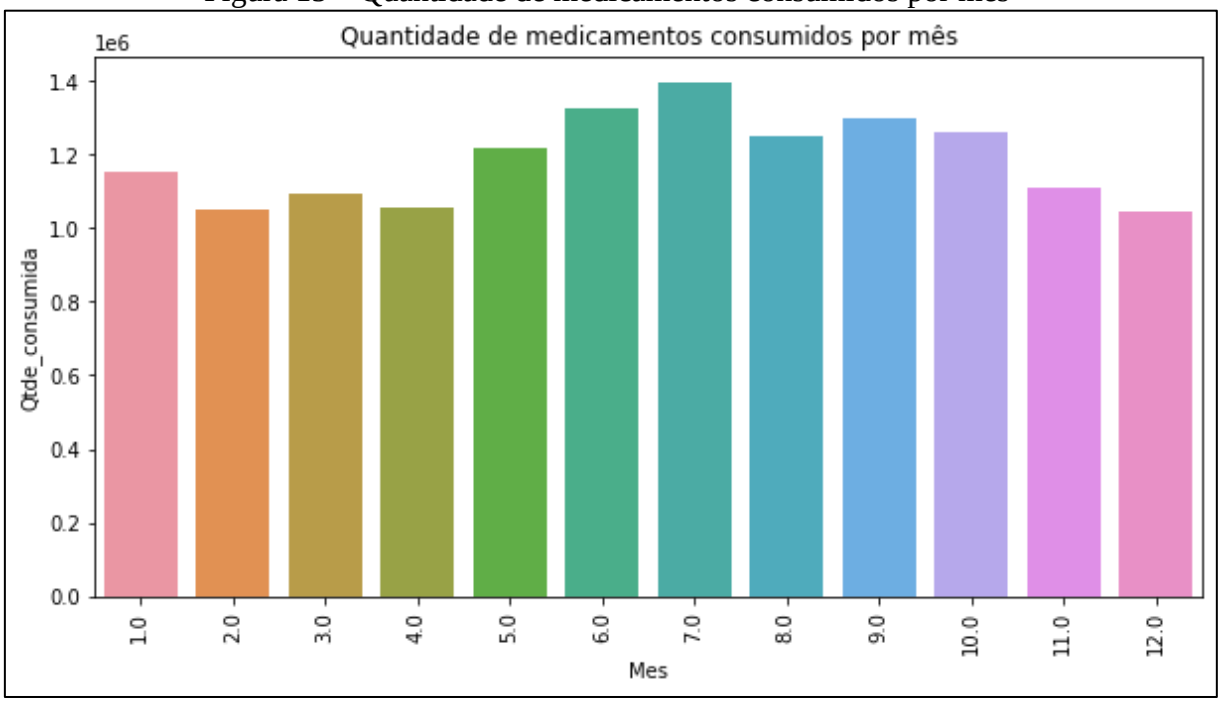
Analisando os dois gráficos em conjunto, o setor que mais consome medicamentos não é o setor que teve mais atendimentos, isso se deve pois o centro cirúrgico administra vários medicamentos em um único paciente devido aos procedimentos cirúrgicos. Já o pronto atendimento tem cerca de seis vezes mais atendimentos mensais que o centro cirúrgico, porém não consome medicamentos da mesma forma, visto que neste setor existem consultas clínicas que não requerem administração de medicamentos.

Já os setores de internação 250, 350 e 200 seguem as mesmas posições em ambos os gráficos, podendo concluir que nestes setores existe relação direta da quantidade de pacientes atendidos com o consumo de medicamentos, visto que a administração de medicamento se dá em todo o paciente que está internado.

O setor de UTI adulto II aparece no primeiro gráfico entre os setores de maior consumo, entretanto seu número de atendimentos é inferior aos demais. Analisando o contexto do setor, este é um setor de terapias intensivas, onde o número de pacientes é menor devido a disponibilidade de leitos neste setor. Todavia, por ser um setor de cuidado intensivo ao paciente, acaba consumindo mais medicamentos que os demais setores presentes no gráfico.

Após esta análise, foi iniciada uma análise de consumo por mês, conforme Figura 15.

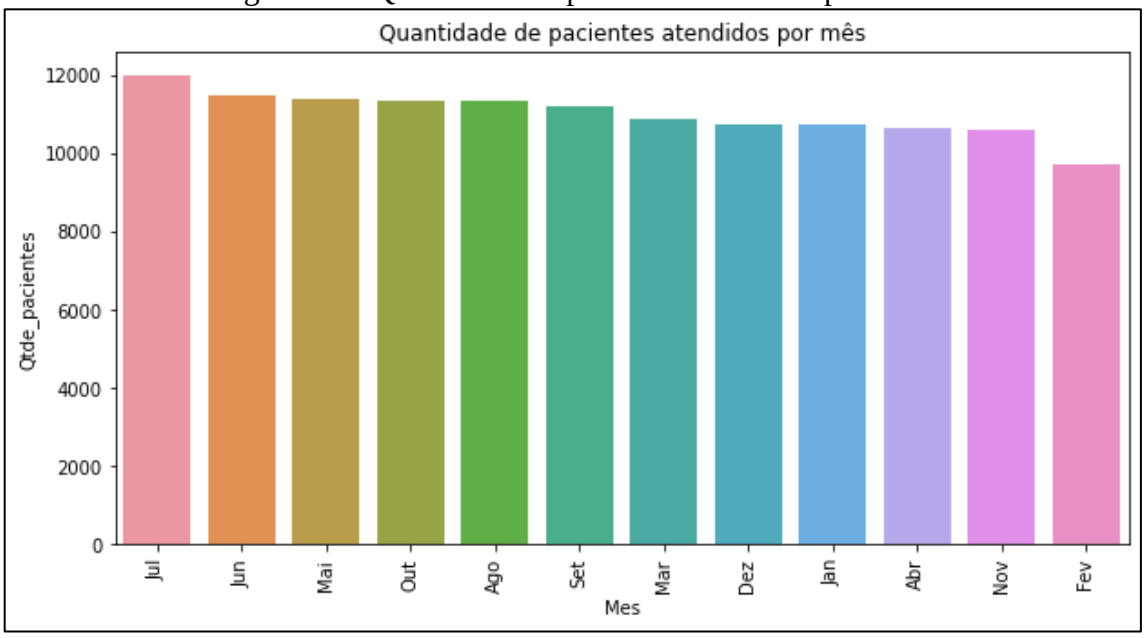
Figura 15 – Quantidade de medicamentos consumidos por mês



Fonte: Google Colab (2023)

Para complemento da análise, foi plotado a seguir um gráfico da quantidade de pacientes por mês conforme Figura 16.

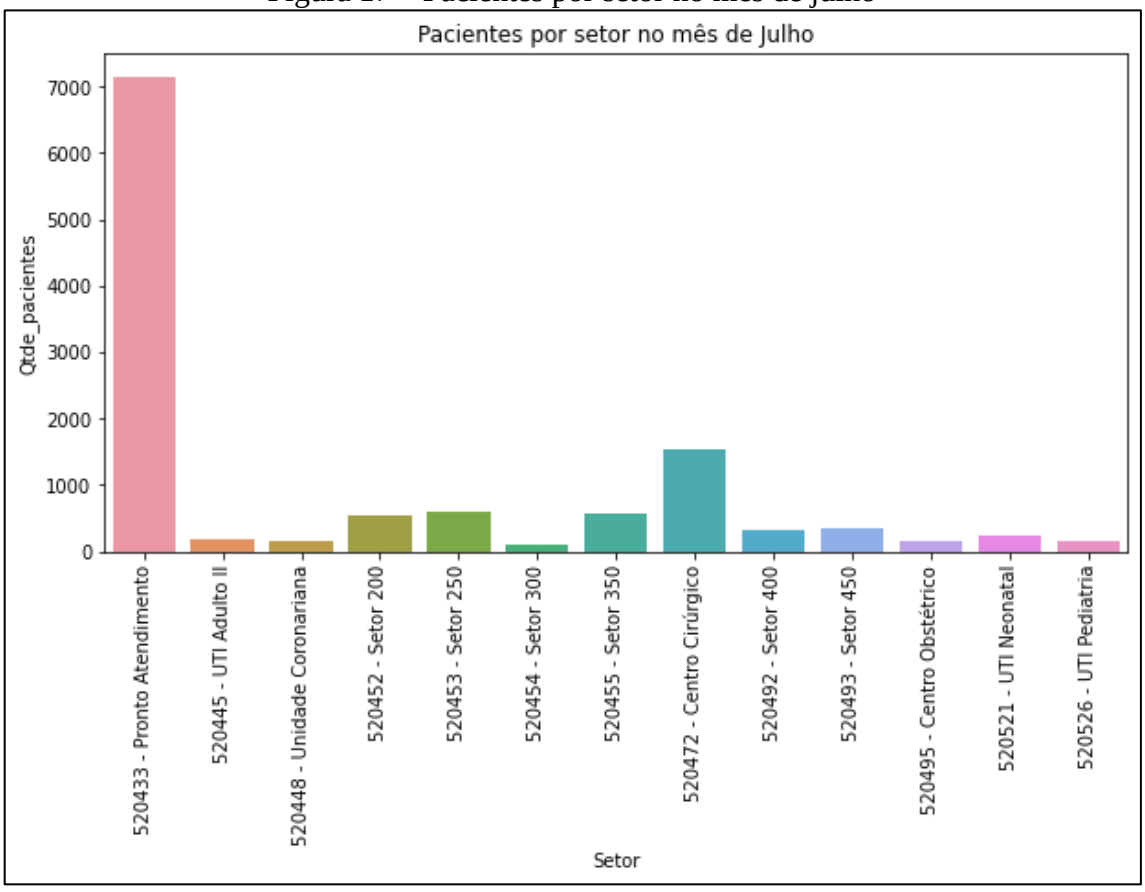
Figura 16 – Quantidade de pacientes atendidos por mês



Fonte: Google Colab (2023)

É possível analisar que em ambos os gráficos o pico de consumo e de atendimentos se deu no mês 7, referente ao mês de julho.

Figura 17 – Pacientes por setor no mês de julho

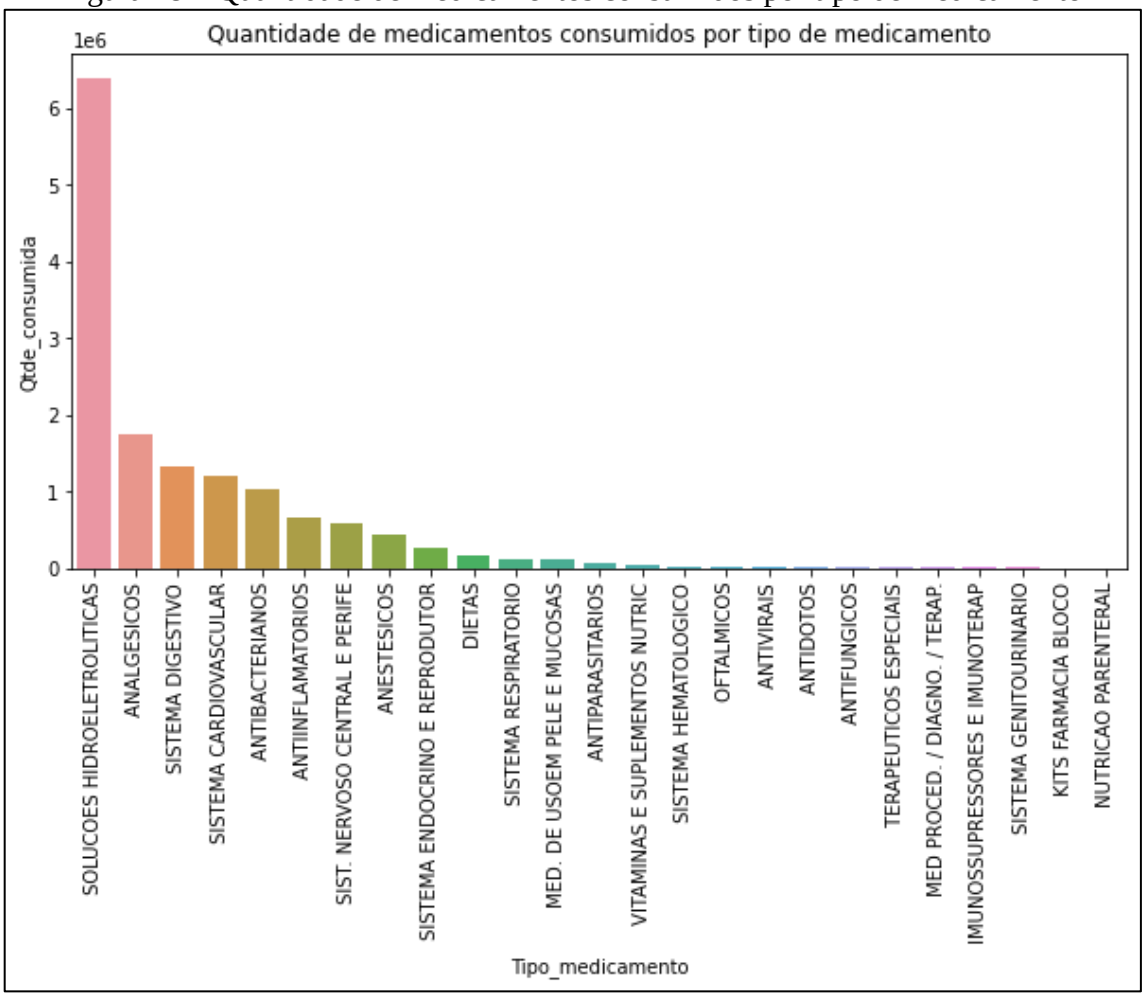


Fonte: Google Colab (2023)

Aprofundando a análise o setor que mais teve atendimentos no mês de julho foi o pronto atendimento. Este mês possui um histórico de maiores atendimentos devido a ser o mês mais frio do estado do Rio Grande do Sul, o que ocasiona resfriados e gripes fazendo com o que o setor tenha mais atendimentos que nos demais meses.

Após foi plotado um gráfico com a quantidade de medicamentos consumidos por tipo de medicamento.

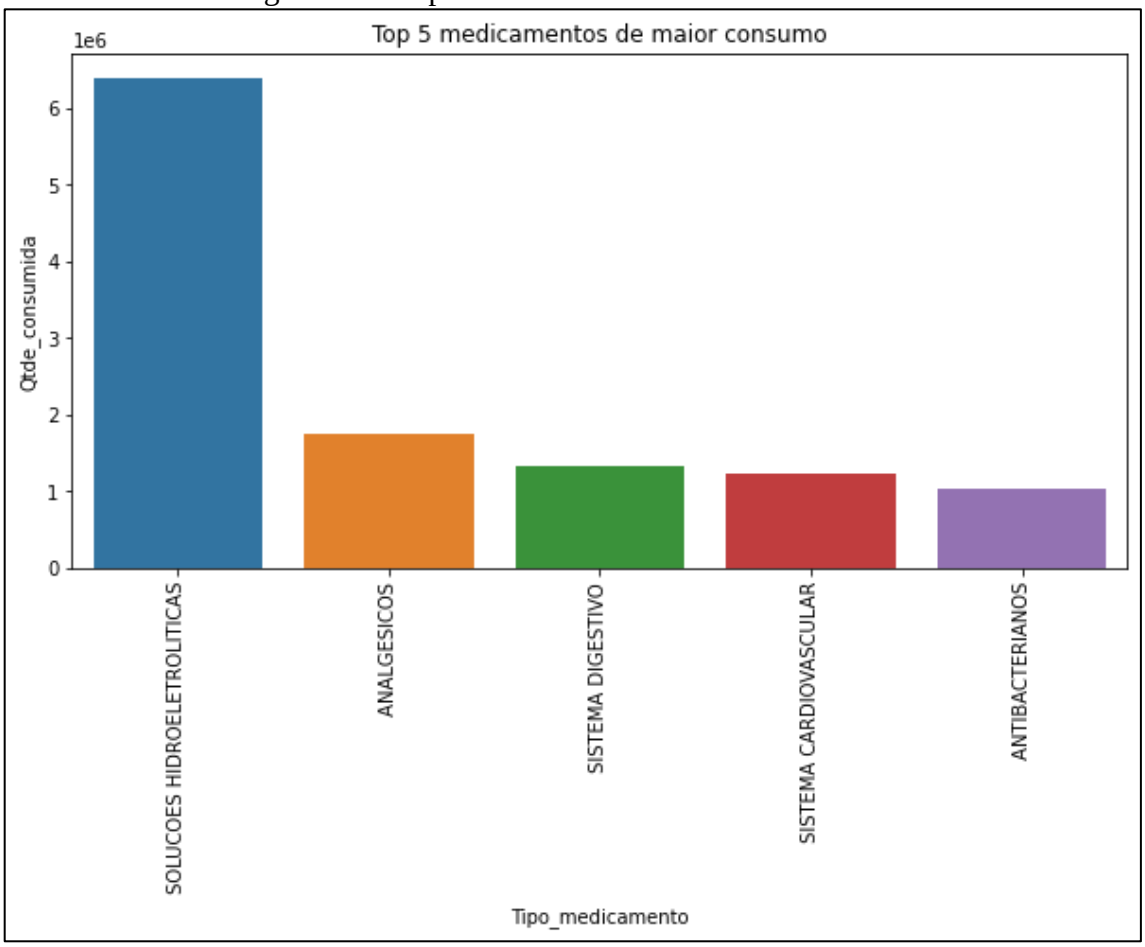
Figura 18 – Quantidade de medicamentos consumidos por tipo de medicamento



Fonte: Google Colab (2023)

Para facilitar a análise foi plotado um gráfico com os cinco tipos de medicamentos de maior consumo, onde os cinco tipos foram: soluções hidroeletrolíticas, analgésicos, medicamentos para o sistema digestivo, medicamentos para o sistema cardiovascular e antibacterianos.

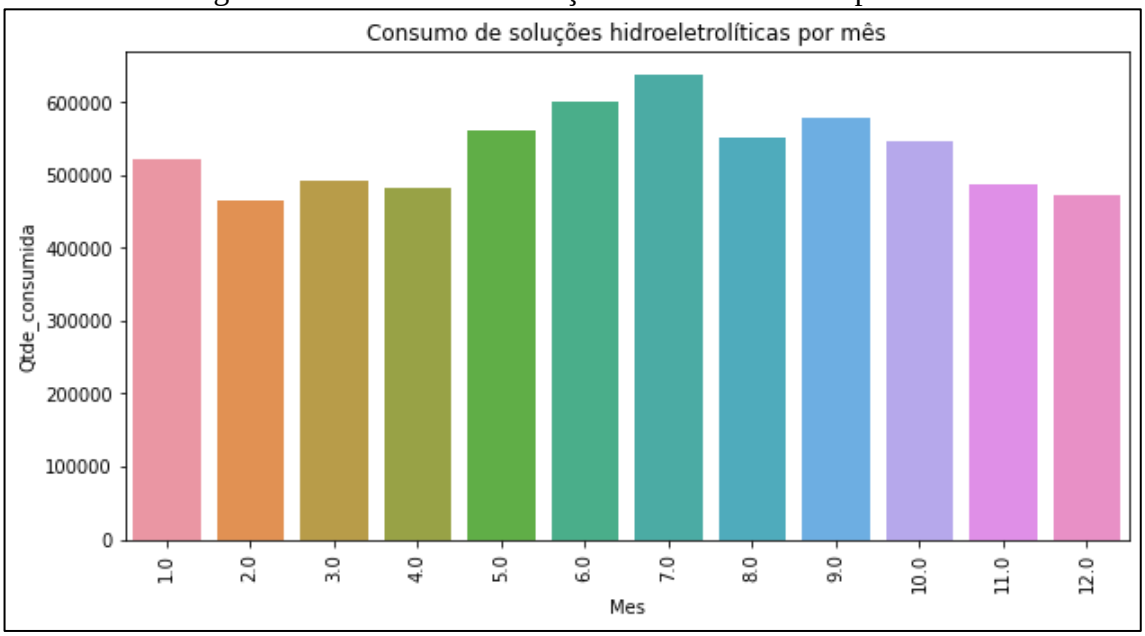
Figura 19 – Top 5 medicamentos de maior consumo



Fonte: Google Colab (2023)

As soluções hidroeletrolíticas são os medicamentos de maior consumo pois são necessárias para a diluição de medicamentos, portanto são usadas em todos os pacientes e em todos os setores do hospital. Conforme Figura 20, é possível perceber novamente a relação de maior consumo nos meses de inverno (maio a setembro), onde o mês de julho teve um resultado maior.

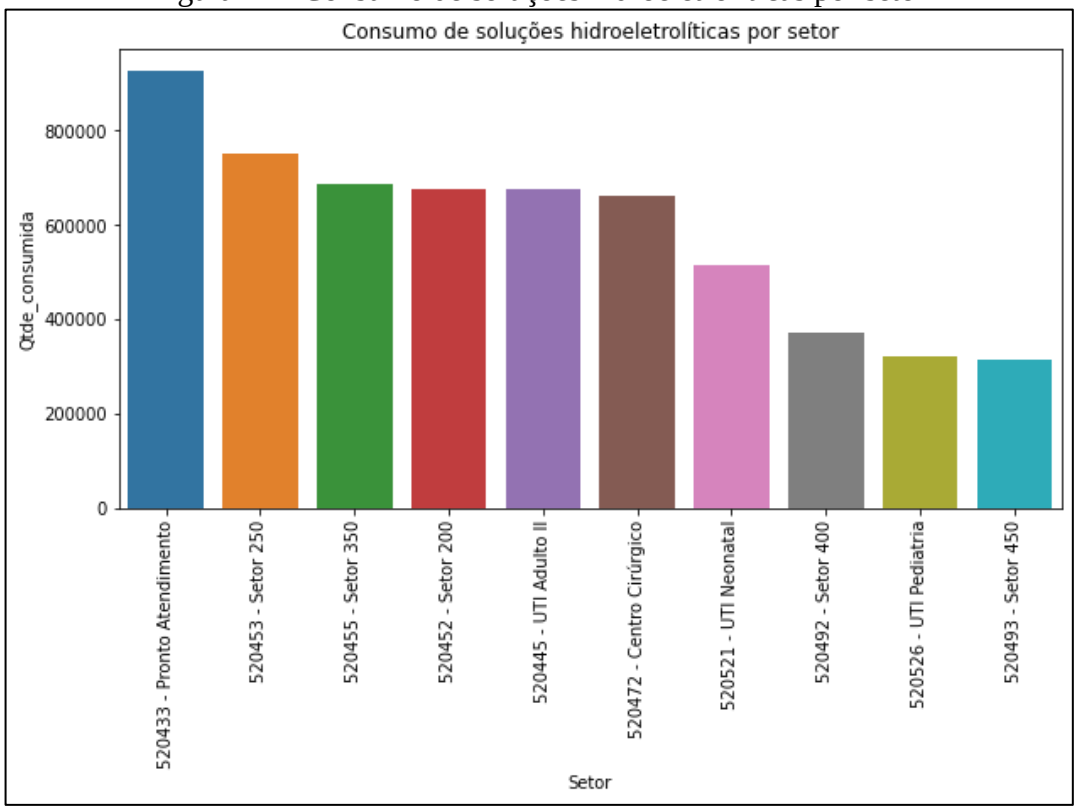
Figura 20 – Consumo de soluções hidroeletrolíticas por mês



Fonte: Google Colab (2023)

Analisando o consumo de soluções hidroeletrolíticas por setor, se percebe que o setor de pronto atendimento possui maior consumo, devido às medicações aplicadas neste setor serem diluídas em soluções hidroeletrolíticas para administração no paciente.

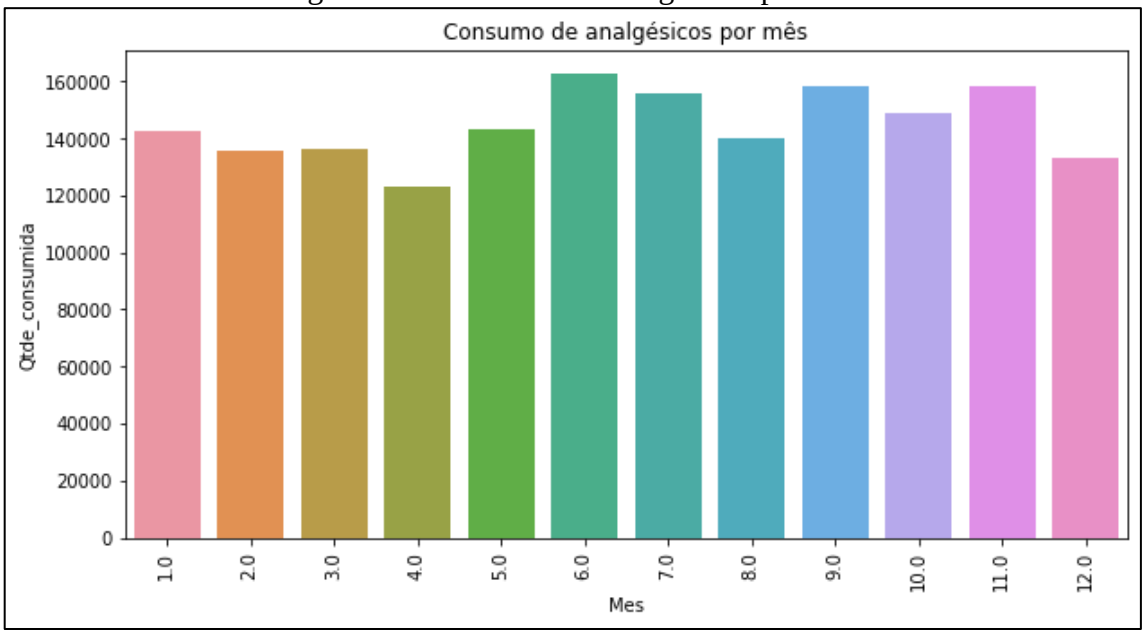
Figura 21 - Consumo de soluções hidroeletrolíticas por setor



Fonte: Google Colab (2023)

Os analgésicos também são medicamentos com elevado consumo devido a serem medicamentos para amenizar a dor do paciente. A partir da Figura 22, nota-se que não existe grande variação no consumo, existe meses como de junho a novembro onde seu consumo é sutilmente elevado, entretanto se mantém parecido ao longo do ano.

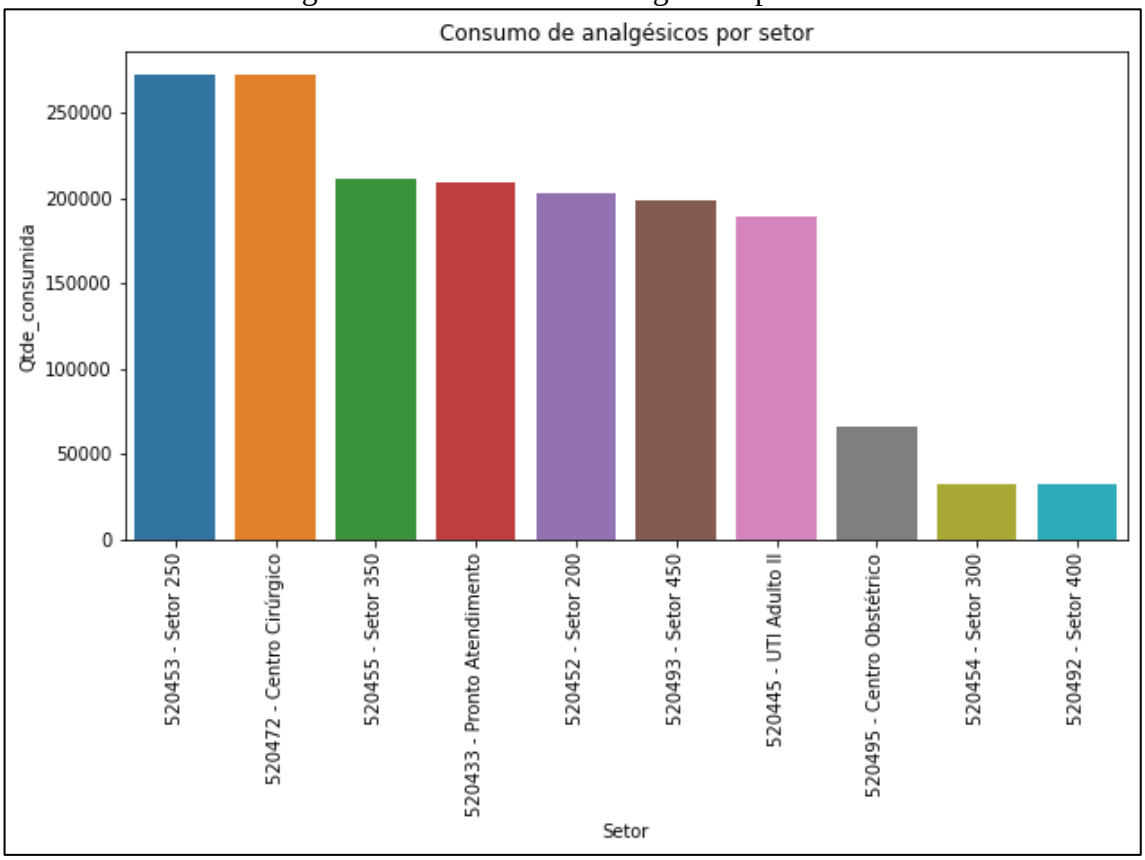
Figura 22 – Consumo de analgésicos por mês



Fonte: Google Colab (2023)

Quando se analisa o consumo de analgésicos por setor na Figura 23, o setor de internação 250 e o centro cirúrgico possuem um consumo mais elevado que os demais setores. No centro cirúrgico todo o paciente recebe analgésico para prevenir ou amenizar a dor, devido a todo procedimento cirúrgico causar desconforto ao paciente. O setor de internação 250 na maioria dos casos é um setor que recebe paciente oriundos do centro cirúrgico e, portanto, necessitam de um tratamento para dor mais prolongado. Os demais setores presentes no gráfico, como setor 350, pronto atendimento, setor 200, setor 450 e UTI adulto II, possuem um consumo de analgésicos muito similar. Já nos setores de centro obstétrico, setor 300 e setor 400 existe uma queda no consumo o que diverge dos demais. Como é possível verificar nos gráficos anteriores, estes setores possuíram um baixo número de atendimentos durante o ano, o que relaciona ao menor consumo de analgésicos.

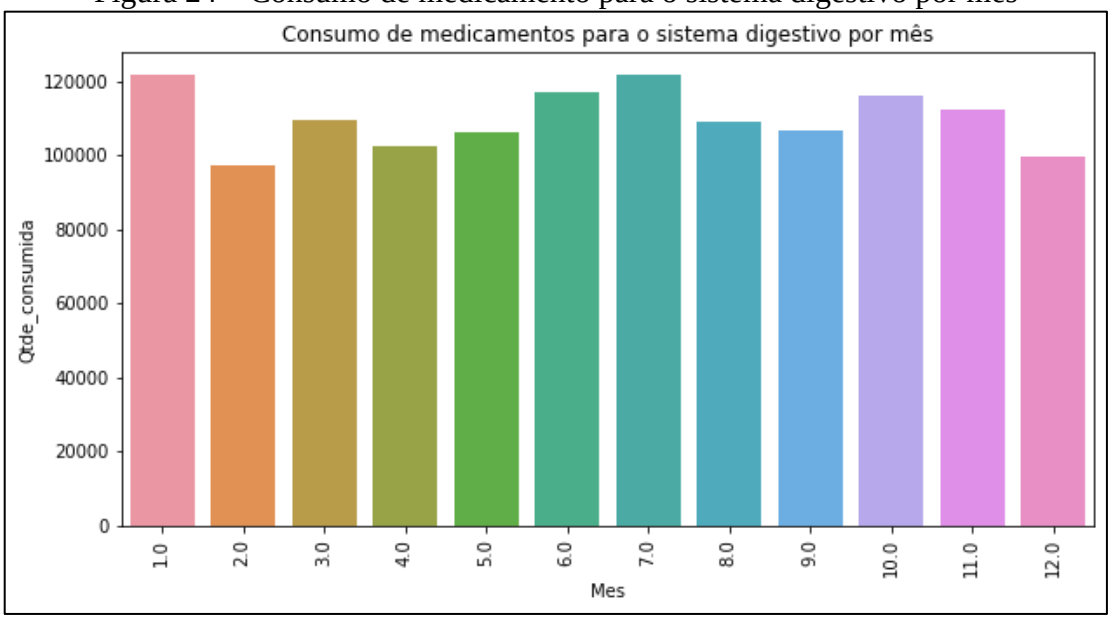
Figura 23 – Consumo de analgésicos por setor



Fonte: Google Colab (2023)

Na Figura 24 a seguir, se analisa o consumo de medicamento para o sistema digestivo, onde se constata que não existe grande sazonalidade entre os dados, entretanto o mês de janeiro teve um pico de consumo.

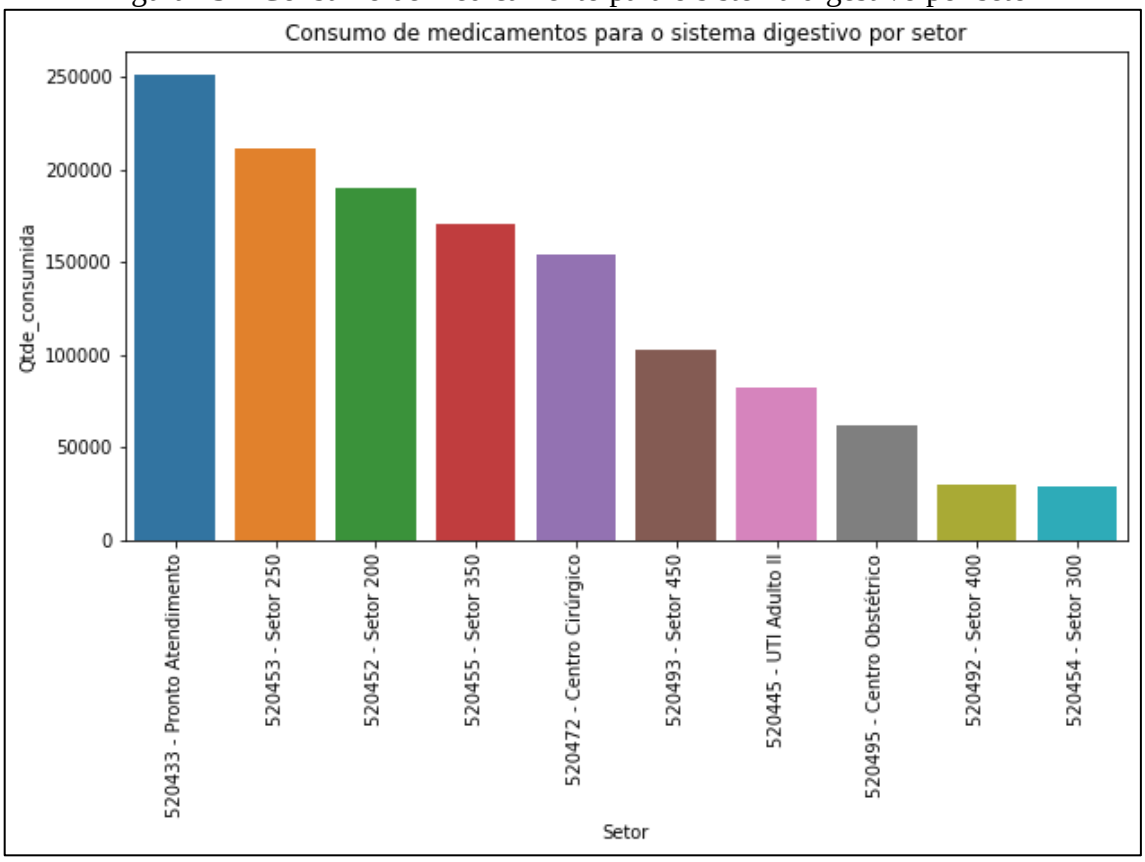
Figura 24 – Consumo de medicamento para o sistema digestivo por mês



Fonte: Google Colab (2023)

O consumo de medicamentos para o sistema digestivo por setor pode ser analisado na Figura 25 a seguir. Claramente, o setor de pronto atendimento teve o maior consumo, isto é, devido aos pacientes que fazem uso deste tipo de medicamento sofrerem com dores abdominais, vômitos, náuseas, dor e cólicas no abdômen e, portanto, usam o setor de pronto atendimento como primeiro recurso para aliviar estes sintomas. Os próximos três setores de maior consumo são setores de internação, onde se pode analisar que é devido aos pacientes que utilizaram os serviços de pronto atendimento e necessitaram de internação para tratamento.

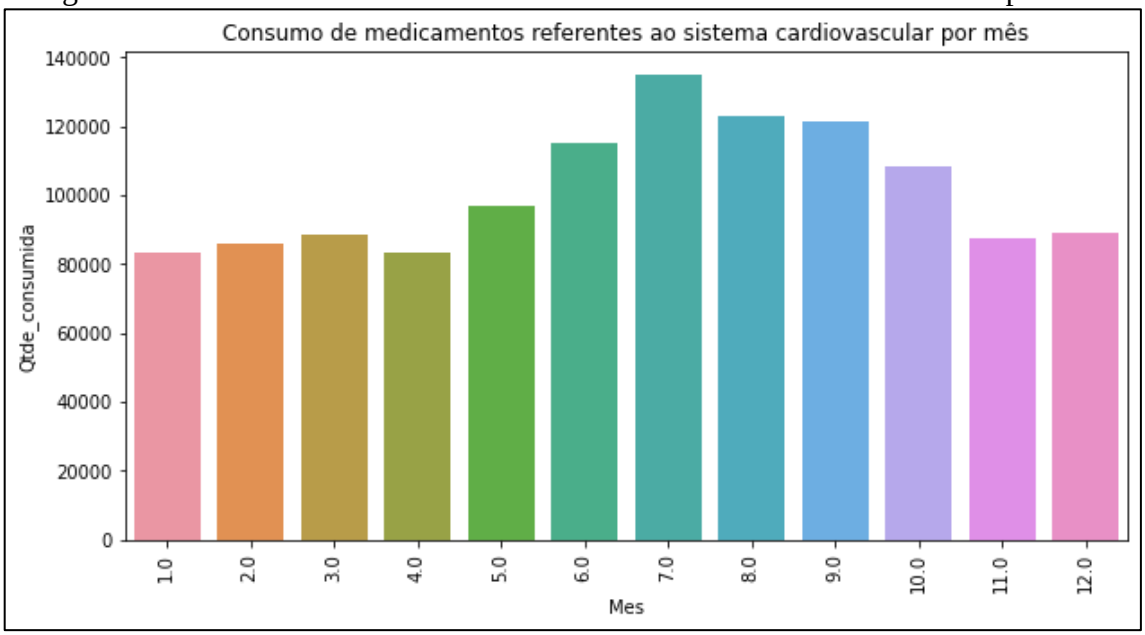
Figura 25 – Consumo de medicamento para o sistema digestivo por setor



Fonte: Google Colab (2023)

No próximo gráfico, Figura 26, se verifica o consumo de medicamento referentes ao sistema cardiovascular. Na análise mensal existe oscilação entre os meses, onde aumenta a partir do mês de maio chegando no seu pico em julho e diminuindo nos próximos meses.

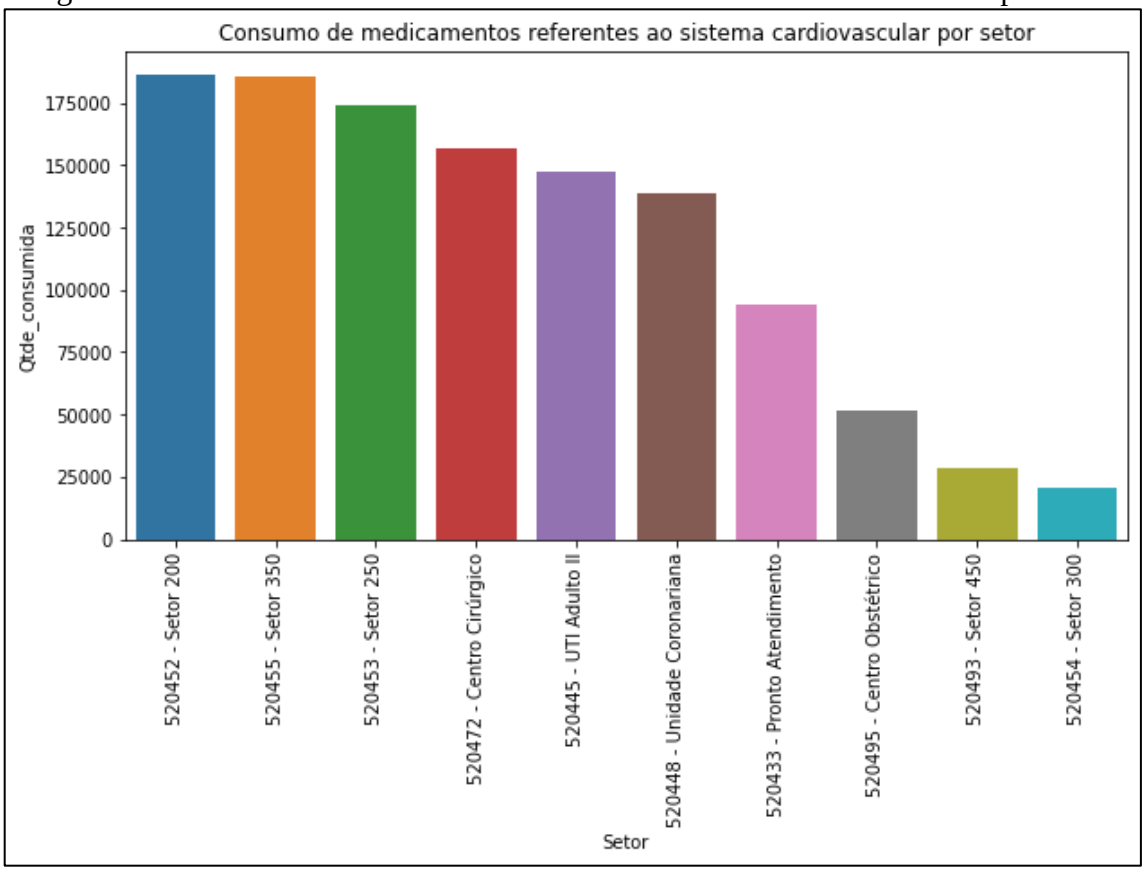
Figura 26 - Consumo de medicamento referentes ao sistema cardiovascular por mês



Fonte: Google Colab (2023)

Conforme Figura 27, os setores de internação 200, 350 e 250 possuem maior consumo de medicamentos referentes ao sistema cardiovascular. Ao se investigar a causa, se evidenciou que estes setores internam paciente com idade avançada e, portanto, mais propícios a problemas cardiovasculares como por exemplo o infarto sendo assim necessitam de internação para tratamento. O centro cirúrgico também possui um consumo considerável devido ao alto número de procedimentos cardiovasculares. A UTI adulto II e a unidade coronariana são setores de terapias intensivas, desta forma consomem esse tipo de medicamento no caso de terapias com pacientes oriundos de procedimentos cirúrgicos relacionados ao sistema cardiovascular ou pacientes que estavam nos setores de internação, mas necessitam de um cuidado mais intensivo.

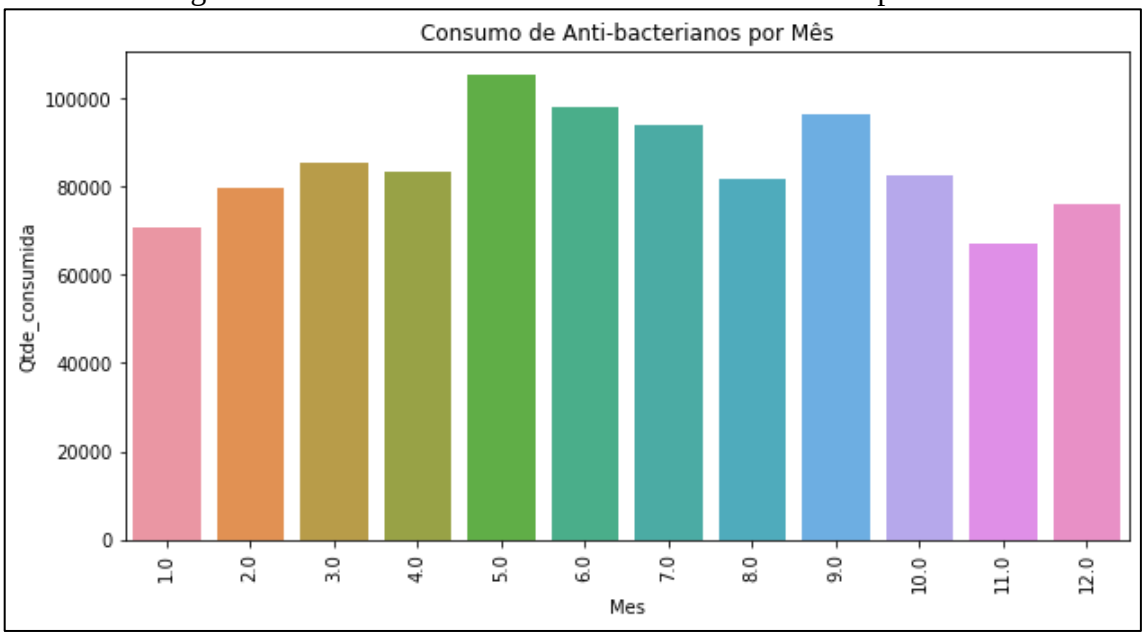
Figura 27 - Consumo de medicamento referentes ao sistema cardiovascular por setor



Fonte: Google Colab (2023)

Na Figura 28 é possível analisar o consumo de medicamentos antibacterianos por mês. Se percebe que de janeiro a abril o consumo é menor, atingindo seu pico de consumo em maio, não possuindo grande oscilação nos próximos meses e por fim tendo uma queda nos meses de novembro e dezembro.

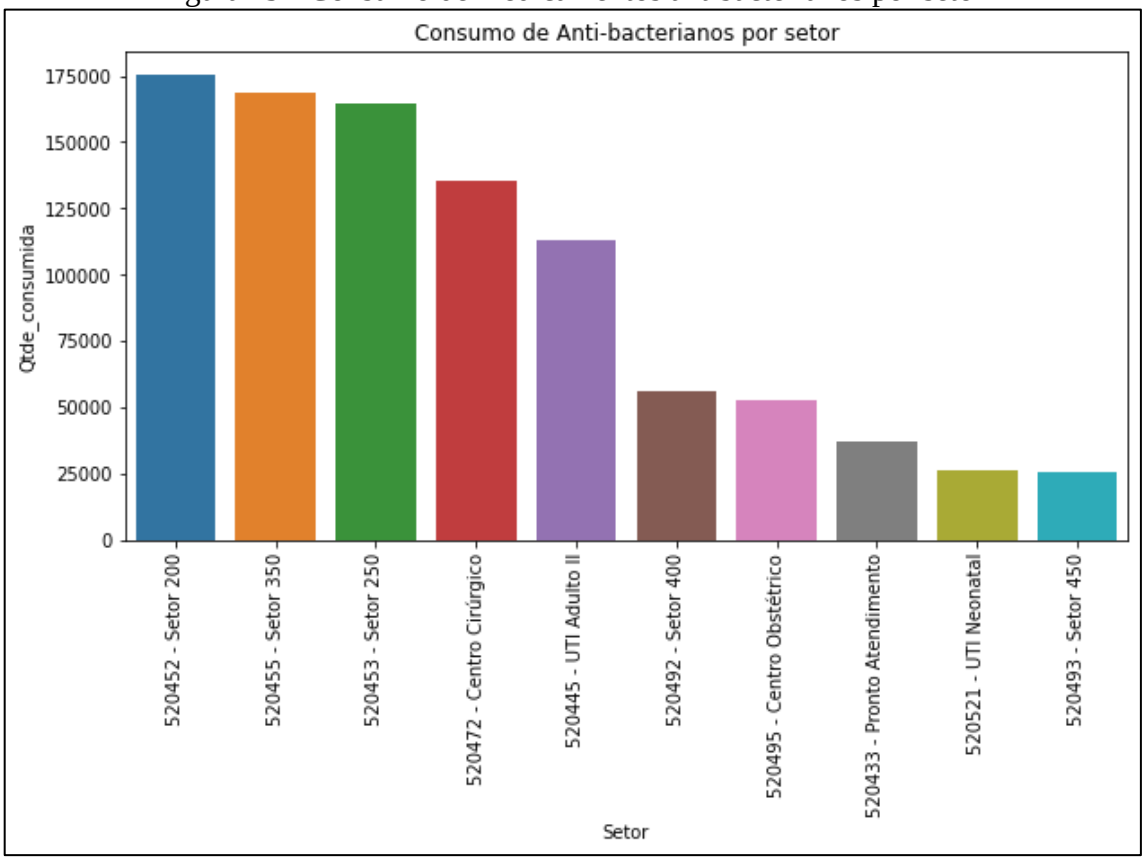
Figura 28 - Consumo de medicamentos antibacterianos por mês



Fonte: Google Colab (2023)

Analisando o consumo por setor, se evidencia na Figura 29 o maior consumo novamente nos setores de internação 200,350 e 250, seguido pelo centro cirúrgico e UTI adulto II.

Figura 29 - Consumo de medicamentos antibacterianos por setor



Fonte: Google Colab (2023)

A partir da análise dos gráficos sobre medicamento antibacterianos, se buscou o índice de infecção hospitalar com a área responsável, podendo se confirmar que o consumo deste tipo de medicamento está associado com o índice de infecção.

6. CRIAÇÃO DE MODELOS DE MACHINE LEARNING

Após a consolidação, tratamento, preparação da base de dados e da análise e exploração dos dados, foi realizada a aplicação de modelos de *machine learning* para a classificação do consumo de medicamentos em alto, moderado e baixo. O atributo “Classificacao” presente na base de dados mostra o tipo de consumo que o medicamento foi classificado que pode ser alto, moderado e baixo. Os modelos de *machine learning* irão classificar o medicamento em alto, moderado e baixo consumo sem a necessidade de análise individual dos medicamentos por uma pessoa.

Na próxima etapa foi realizada uma preparação dos dados antes da aplicação dos algoritmos de *machine learning.*

6.1 PREPARAÇÃO DOS DADOS

6.1.1 Divisão entre previsores e classe

Para melhor uso dos algoritmos, foi realizada a divisão entre previsores e classe. Uma variável para armazenar os atributos de entrada que foi chamada de x_consumo. As variáveis preditoras armazenas em x_consumo são: ‘Mês’, ‘Setor’, ‘Tipo_medicamento’, ‘Qtde_pacientes’, ‘Qtde_consumida’. A variável alvo para armazenar a classe chamada de y_consumo armazena os dados de classificação do medicamento ‘Classificacao’.

Figura 30 - Divisão entre previsores e classe

```
x_consumo = consumo_medicamentos.iloc[:, 0:5].values
x_consumo[0]

y_consumo = consumo_medicamentos.iloc[:,5].values
y_consumo[0]
```

Fonte: Google Colab (2023)

6.1.2 Tratamento de atributos categóricos

Como os atributos categóricos dificultam os cálculos matemáticos dos algoritmos a serem utilizados, foi realizada uma transformação dos dados categóricos para dados numéricos. Para esta transformação foi utilizada a função de *LabelEncoder* do módulo *preprocessing* da biblioteca *Sklearn*. A função *LabelEncoder* realiza uma codificação com um valor entre 0 e n_classes-1, onde n é o número de rótulos distintos.

Figura 31 – Transformação dos dados categóricos

```
from sklearn.preprocessing import LabelEncoder
from sklearn.compose import ColumnTransformer

label_encoder_Mes = LabelEncoder()
label_encoder_Setor = LabelEncoder()
label_encoder_Tipo_medicamento = LabelEncoder()

x_consumo[:,0] = label_encoder_Mes.fit_transform(x_consumo[:,0])
x_consumo[:,1] = label_encoder_Setor.fit_transform(x_consumo[:,1])
x_consumo[:,2] = label_encoder_Tipo_medicamento.fit_transform(x_consumo[:,2])
```

Fonte: Google Colab (2023)

6.1.3 Escalonamento

Para aplicação dos modelos de *machine learning* nem sempre é necessário realizar o escalonamento de dados, onde o método se faz necessário quando os parâmetros demonstrarem um intervalo muito distante. Conforme Figura 32, a distribuição dos dados possui diferença, portanto optou-se por efetuar o método de escalonamento. O objetivo do escalonamento é alterar os valores numéricos no conjunto de dados para uma escala comum, sem distorcer as diferenças nos intervalos de valores. Desta forma, não haverá interferência nos cálculos matemáticos realizados pelos algoritmos de *machine learning*.

Figura 32 – Distribuição dos dados

```
x_consumo
```

```
array([[3, 0, 0, 6729.0, 1388.0],
       [3, 0, 0, 6228.0, 1388.0],
       [3, 0, 0, 7187.0, 1388.0],
       ...,
       [8, 5, 24, 153.0, 33.0],
       [8, 5, 24, 70.0, 33.0],
       [8, 5, 24, 6.0, 33.0]], dtype=object)
```

Fonte: Google Colab (2023)

Conforme Figura 33 foi realizada a importação da biblioteca *StandardScaler* para escalonamento dos dados.

Figura 33 – Escalonamento dos dados

```
from sklearn.preprocessing import StandardScaler
```

```
scaler_x_consumo = StandardScaler(with_mean=False)
x_consumo = scaler_x_consumo.fit_transform(x_consumo)
```

```
x_consumo
```

```
array([[8.78068115e-01, 0.00000000e+00, 0.00000000e+00, 3.36868313e+00,
        1.41693909e+00],
       [8.78068115e-01, 0.00000000e+00, 0.00000000e+00, 3.11787168e+00,
        1.41693909e+00],
       [8.78068115e-01, 0.00000000e+00, 0.00000000e+00, 3.59796785e+00,
        1.41693909e+00],
       ...,
       [2.34151497e+00, 1.32754322e+00, 3.15861815e+00, 7.65951135e-02,
        3.36880332e-02],
       [2.34151497e+00, 1.32754322e+00, 3.15861815e+00, 3.50435160e-02,
        3.36880332e-02],
       [2.34151497e+00, 1.32754322e+00, 3.15861815e+00, 3.00372994e-03,
        3.36880332e-02]])
```

Fonte: Google Colab (2023)

6.1.4 Divisão entre bases de treinamento e teste

Após o escalonamento os dados armazenados em ‘x_consumo’ e ‘y_consumo’ foram divididos em dados de treinamento e teste: ‘x_consumo_treinamento’, ‘x_consumo_teste’, ‘y_consumo_treinamento’, ‘y_consumo_teste’. Onde os valores armazenados em ‘x_consumo_treinamento’ serão utilizados pelo algoritmo para aprender a relação entre as

variáveis preditoras. Os valores em ‘y_consumo_treinamento’ também serão utilizados para aprendizagem, porém com parte dos dados alvo.

Os valores armazenados em ‘x_consumo_teste’ serão utilizados para teste, onde o algoritmo irá exercer o aprendizado obtido nas variáveis contidas nas bases de testes. E por fim, ‘y_consumo_teste’ para avaliar o desempenho do modelo, utilizado para comparar com as previsões feitas pelo algoritmo.

Em todos os modelos utilizados foi utilizada a biblioteca *train_test_split* do *Sklearn*, onde 25% dos dados foram utilizados para teste e 75% dos dados utilizados para treinamento do modelo.

Figura 34 – train_test_split

```
from sklearn.model_selection import train_test_split

x_consumo_treinamento, x_consumo_teste, y_consumo_treinamento, y_consumo_teste = train_test_split(x_consumo, y_consumo, test_size = 0.25, random_state = 0)
```

Fonte: Google Colab (2023)

6.2 ALGORITMOS DE CLASSIFICAÇÃO

Os modelos de classificação buscam reconhecer e agrupar dados em categorias predefinidas por meio de um conjunto de dados de treinamento. O objetivo é identificar qual a classificação do consumo do medicamento, para isto foram utilizados os algoritmos de classificação: Naive Bays, Redes Neurais, Support Vector Machine (SVM) e Regressão Logística.

Para verificar a eficácia dos modelos de machine learning, foi definida a medida de avaliação ‘accuracy_score’ e ‘classification_report’ do modulo metrics da biblioteca Sklearn.

Para a aplicação de todos os algoritmos o processo será o mesmo: importação do respectivo algoritmo, treinamento nas bases de dados de treinamento por meio da função fit, previsão dos dados utilizando a função predict na base de x_consumo_teste e sua saída será comparada com ‘y_consumo_teste’ por meio das medidas de avaliação e matriz de confusão para soma a quantidade de resultados positivos verdadeiros, falsos positivos, negativos verdadeiros e falsos negativos.

6.2.1 Naive Bayes

O algoritmo de classificação Naive Bayes se baseia no teorema de Bayes, onde sua principal característica é a desconsideração completa entre a correlação das variáveis. A

vantagem do uso deste algoritmo é sua agilidade em comparação com outros métodos. A maior desvantagem é ser conhecido com mal avaliador.

Figura 35 – Naive Bayes

```
from sklearn.naive_bayes import GaussianNB

naive_consumo = GaussianNB()
naive_consumo.fit(x_consumo_treinamento, y_consumo_treinamento)
previsoes_naive = naive_consumo.predict(x_consumo_teste)
previsoes_naive

from sklearn.metrics import accuracy_score, classification_report

accuracy_score(y_consumo_teste, previsoes_naive)
```

Fonte: Google Colab (2023)

6.2.2 Redes neurais artificiais

Segundo Ferneda (2006, p. 25-30), o cérebro biológico é formado de bilhões de neurônios, onde um neurônio é uma célula composta por três segmentos com ofícios específicos e complementares: corpo, dendritos e axônio. Os dendritos têm a função de captar os estímulos obtidos em um período e transmiti-los ao corpo do neurônio, onde são executados.

No momento em que esses estímulos alcançam certo limite, o corpo da célula encaminha um novo impulso que se espalha pelo axônio e assim é difundido às células próximas por meio de sinapses. Essa execução pode se repetir em diversas camadas de neurônios. A informação que é introduzida é processada, podendo conduzir o cérebro a comandar reações físicas.

As Redes Neurais Artificiais (RNA's) visam desenvolver modelos matemáticos que se identifiquem às arquiteturas neurais biológicas. Dentro da arquitetura destas redes, os neurônios são chamados de unidades de processamento onde ocorre o processamento de informações da rede. Estas unidades de processamento possuem uma grande quantidade de conexões entre elas que são responsáveis por transmitir a informação entre as unidades.

As unidades de processamento são inseridas em uma ou mais camadas associadas por diversas conexões normalmente unidirecionais. Estas conexões estão associadas a pesos, ou

força própria, onde o conhecimento representado no modelo é armazenado com a finalidade de ponderar a entrada por cada unidade de processamento da rede.

Um parâmetro importante na concepção da RNA é sua arquitetura, visto que ela delimita o tipo de problema a ser tratado por meio da RNA. Para a definição da arquitetura são considerados o número de camadas da rede, o número de nodos em cada camada, o tipo de conexão entre os nodos e a topologia da rede (SANTOS et. al., 2005, p.117-126).

Para a aplicação da rede neural foi realizada a importação das bibliotecas necessárias, a divisão das bases em treinamento e teste, foi realizado o procedimento de escalonamento das variáveis para padronização delas, com o objetivo de não ocorrer interferência no momento da aplicação da rede, aplicação do treinamento e por fim foi realizada a previsão e o cálculo do erro médio absoluto.

Figura 36 – Redes Neurais

```
x_consumo_treinamento.shape
```

```
(23517, 5)
```

```
#Definir numero de neuronios
(5 + 1) / 2
```

```
3.0
```

```
from sklearn.neural_network import MLPClassifier
```

```
rna_consumo = MLPClassifier(activation='relu', solver='adam', max_iter=1000, hidden_layer_sizes=(3,3), tol=0.00001)
rna_consumo.fit(x_consumo_treinamento, y_consumo_treinamento.ravel())
```

```
previsoes_rna = rna_consumo.predict(x_consumo_teste).reshape(-1, 1)
previsoes_rna
```

```
from sklearn.metrics import accuracy_score, classification_report
accuracy_score(y_consumo_teste, previsoes_rna)
```

Fonte: Google Colab (2023)

6.2.3 Support Vector Machine

O algoritmo de Support Vector Machine (SVM) é um algoritmo de aprendizado de máquina supervisionado utilizado para classificação, mas também pode ser utilizado para regressão. Seu principal foco é no treinamento e classificação. A classificação encontra o hiperplano que melhor diferencia as classes. A vantagem do uso deste algoritmo é que não é influenciado por ruídos de dados e fácil de usar. A desvantagem é que pode ser lento.

Figura 37 – *Support Vector Machine*

```
from sklearn.svm import SVC

svm_consumo = SVC(kernel='linear', random_state=1)
svm_consumo.fit(x_consumo_treinamento, y_consumo_treinamento)

previsoes_svm = svm_consumo.predict(x_consumo_teste)
previsoes_svm

from sklearn.metrics import accuracy_score, classification_report
accuracy_score(y_consumo_teste, previsoes_svm)
```

Fonte: Google Colab (2023)

6.2.4 Regressão Logística

A regressão logística é um recurso que nos permite estimar a probabilidade associada à ocorrência de determinado evento em face de um conjunto de variáveis explanatórias. É uma técnica recomendada para situações em que a variável dependente é de natureza dicotômica ou binária. Quanto às independentes, tanto podem ser categóricas ou não. Ela é mais útil no entendimento da influência de diversas variáveis independentes em uma saída única variável. A desvantagem é que funciona apenas quando a variável prevista é binária, assume que todos os preditores são independentes uns dos outros e assume que os dados estão livres de valores ausentes.

Figura 38 – Regressão Logística

```
from sklearn.linear_model import LogisticRegression

logistic_consumo = LogisticRegression(random_state = 1)
logistic_consumo.fit(x_consumo_treinamento, y_consumo_treinamento)

previsoes_logistic = logistic_consumo.predict(x_consumo_teste)
previsoes_logistic

from sklearn.metrics import accuracy_score, classification_report
accuracy_score(y_consumo_teste, previsoes_logistic)
```

Fonte: Google Colab (2023)

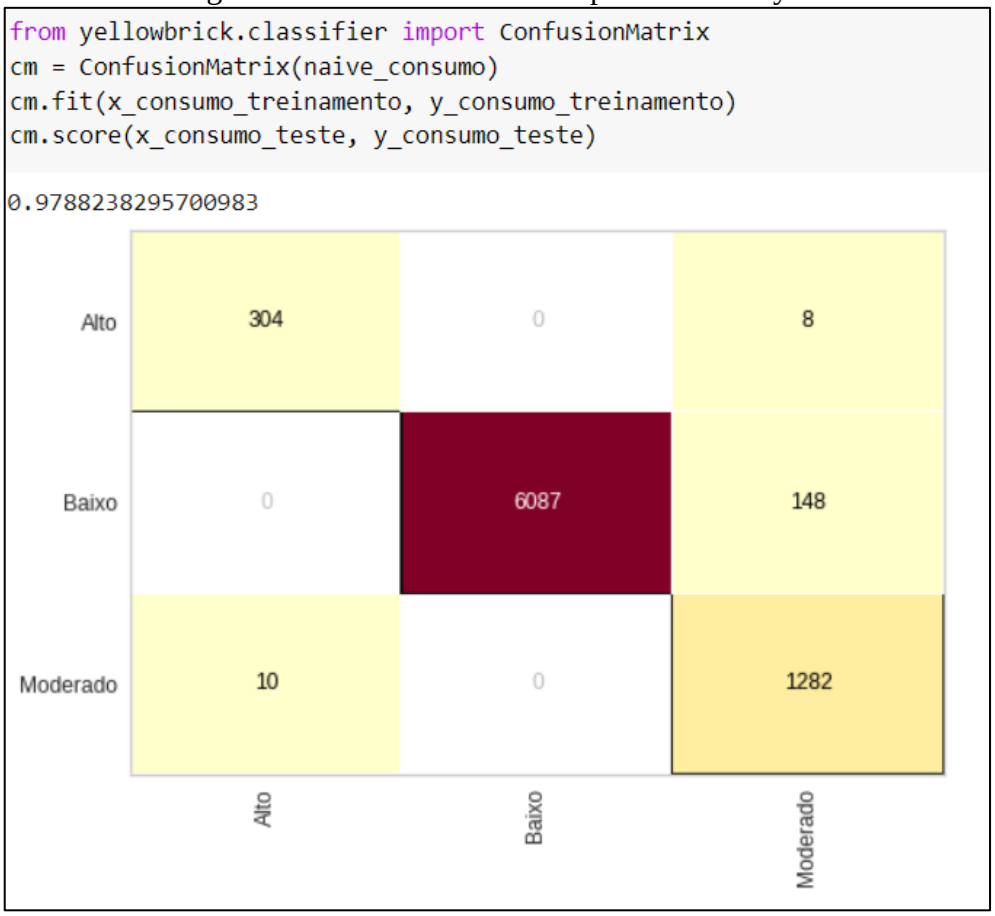
7. INTERPRETAÇÃO DOS RESULTADOS

A principal medida de avaliação dos algoritmos aplicados é a acurácia, entretanto todas as outras medidas de avaliação serão consideradas.

7.1 RESULTADO NAIVE BAYES

O algoritmo de classificação Naive Bayes obteve uma acurácia de 97,88%. Para análise da eficácia do resultado do algoritmo, pode se observar a matriz de confusão e o classification report nas figuras 39 e 40 respectivamente.

Figura 39 – Matriz de confusão para Naive Bayes



Fonte: Google Colab (2023)

Figura 40 – Classification report para Naive Bayes

```
print(classification_report(y_consumo_teste, previsoes_naive))
```

	precision	recall	f1-score	support
Alto	0.97	0.97	0.97	312
Baixo	1.00	0.98	0.99	6235
Moderado	0.89	0.99	0.94	1292
accuracy			0.98	7839
macro avg	0.95	0.98	0.97	7839
weighted avg	0.98	0.98	0.98	7839

Fonte: Google Colab (2023)

Na matriz de confusão pode se observar que o algoritmo Naive Bayes acertou a classificação de 7.673 registros do total de 7.839 registros e errou 166 registros no total. Pelo cassification report se observa o resultado de 0,95 de precisão, indica que dentre os exemplos classificados como verdadeiros, 95% são realmente verdadeiros. A medida de recall resultou em 0,98, o que indica que 98% dos dados foram classificados como verdadeiros em comparação com a quantidade real de resultados verdadeiros que existem na amostra. A F1-score 0,97, demonstra a média ponderada de precisão e recall, mostrando a qualidade geral do modelo em 97%.

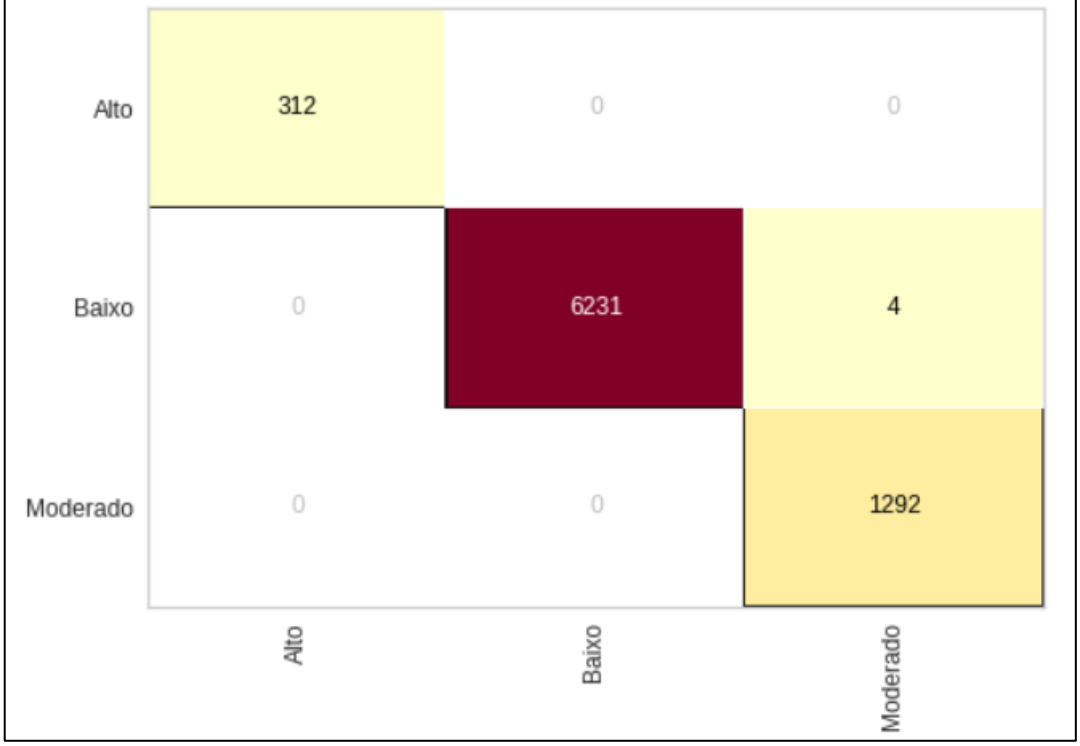
7.2 RESULTADO REDES NEURAIS

O algoritmo de redes neurais obteve 99,94% de acurácia. Para análise da eficácia do resultado do algoritmo, pode se observar a matriz de confusão e o classification report nas figuras 41 e 42 respectivamente.

Figura 41 – Matriz de confusão para redes neurais

```
from yellowbrick.classifier import ConfusionMatrix
cm = ConfusionMatrix(rna_consumo)
cm.fit(x_consumo_treinamento, y_consumo_treinamento)
cm.score(x_consumo_teste, y_consumo_teste)
```

```
0.9994897308330144
```



Fonte: Google Colab (2023)

Figura 42 – Classification report para redes neurais

```
print(classification_report(y_consumo_teste, previsoes_rna))
```

	precision	recall	f1-score	support
Alto	1.00	1.00	1.00	312
Baixo	1.00	1.00	1.00	6235
Moderado	1.00	1.00	1.00	1292
accuracy			1.00	7839
macro avg	1.00	1.00	1.00	7839
weighted avg	1.00	1.00	1.00	7839

Fonte: Google Colab (2023)

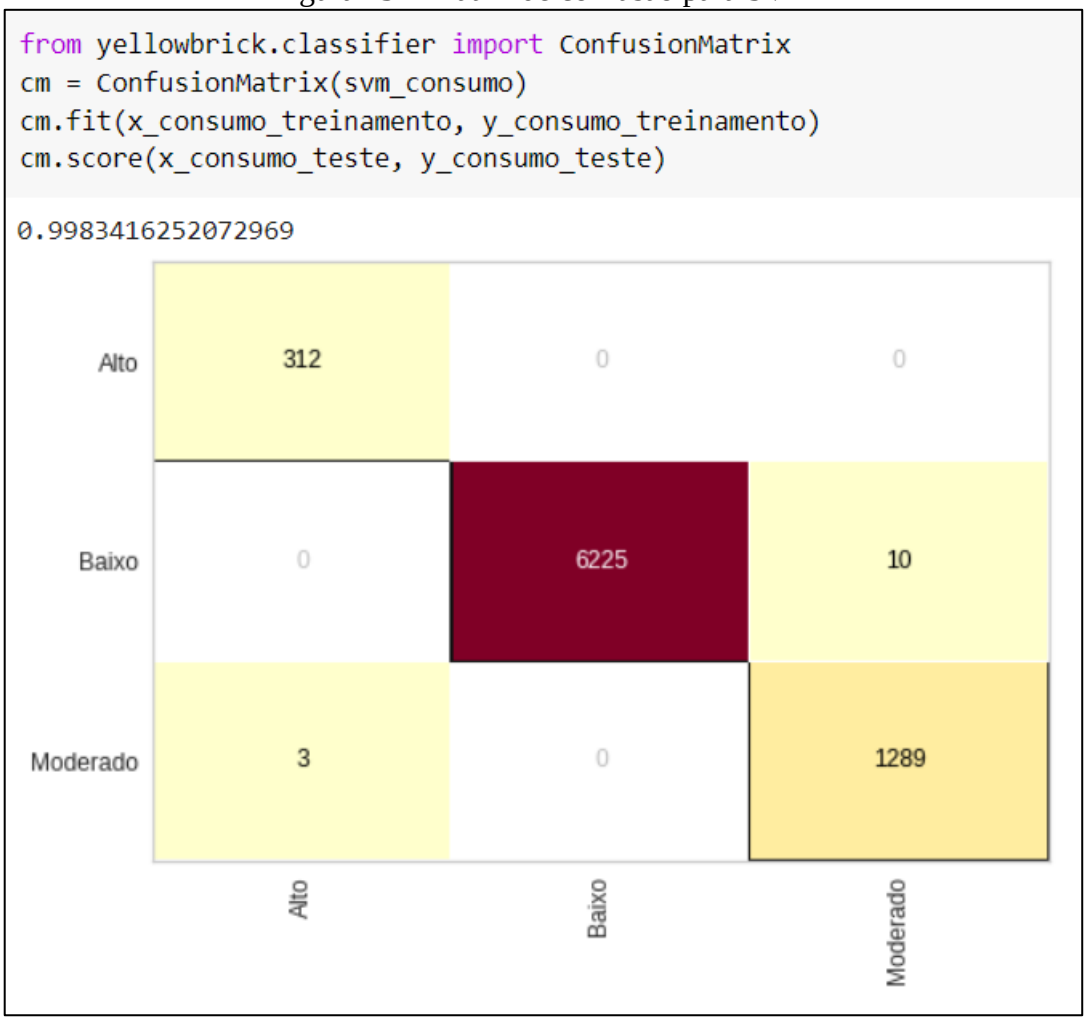
Na matriz de confusão demonstrada na Figura 41 pode se observar que o algoritmo acertou a classificação de 7.835 registros do total de 7.839 registros e errou apenas 4 registros no total. Pelo cassification report se observa o resultado de 1,00 de precisão, indicando que dentre os exemplos classificados como verdadeiros 100% são realmente verdadeiros. A medida de recall resultou em 1,00, o que indica que 100% dos dados foram classificados como

verdadeiros em comparação com a quantidade real de resultados verdadeiros que existem na amostra. A F1-score 1,00, demonstra a média ponderada de precisão e recall, mostrando a qualidade geral do modelo em 100%.

7.3 RESULTADO SVM

O algoritmo de SVM obteve 99,83% de acurácia. Para análise da eficácia do resultado do algoritmo, pode se observar a matriz de confusão e o classification report nas figuras 43 e 44 respectivamente.

Figura 43 – Matriz de confusão para SVM



Fonte: Google Colab (2023)

Figura 44 – Classification report para SVM

```
print(classification_report(y_consumo_teste, previsoes_svm))
```

	precision	recall	f1-score	support
Alto	0.99	1.00	1.00	312
Baixo	1.00	1.00	1.00	6235
Moderado	0.99	1.00	0.99	1292
accuracy			1.00	7839
macro avg	0.99	1.00	1.00	7839
weighted avg	1.00	1.00	1.00	7839

Fonte: Google Colab (2023)

Na matriz de confusão demonstrada na Figura 43 pode se observar que o algoritmo acertou a classificação de 7.826 registros do total de 7.839 registros e errou 13 registros no total. Pelo cassification report se observa o resultado de 0,99 de precisão, indicando que dentre os exemplos classificados como verdadeiros 99% são realmente verdadeiros. A medida de recall resultou em 1,00, o que indica que 100% dos dados foram classificados como verdadeiros em comparação com a quantidade real de resultados verdadeiros que existem na amostra. A F1-score 1,00, demonstra a média ponderada de precisão e recall, mostrando a qualidade geral do modelo em 100%.

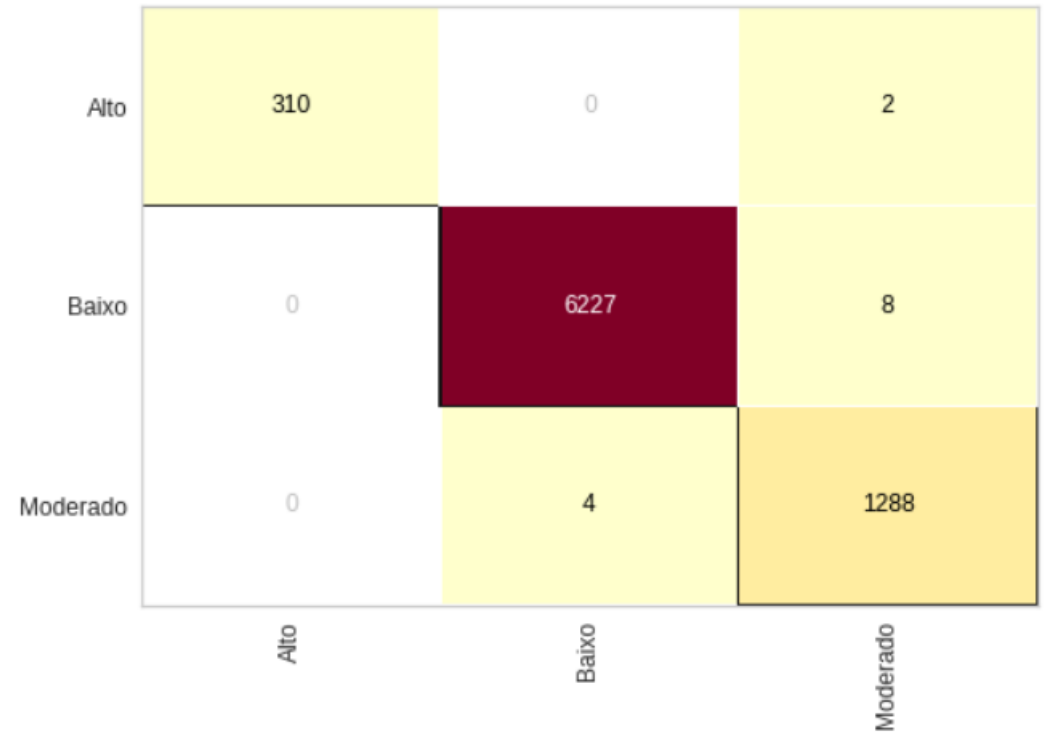
7.4 RESULTADO REGRESSÃO LOGÍSTICA

O algoritmo de Regressão logística obteve 99,82% de acurácia. Para análise da eficácia do resultado do algoritmo, pode se observar a matriz de confusão e o classification report nas figuras 45 e 46 respectivamente.

Figura 45 – Matriz de confusão para Regressão Logística

```
from yellowbrick.classifier import ConfusionMatrix
cm = ConfusionMatrix(logistic_consumo)
cm.fit(x_consumo_treinamento, y_consumo_treinamento)
cm.score(x_consumo_teste, y_consumo_teste)
```

```
0.9982140579155504
```



Fonte: Google Colab (2023)

Figura 46 – Classification report para Regressão Logística

```
print(classification_report(y_consumo_teste, previsoes_logistic))
```

	precision	recall	f1-score	support
Alto	1.00	0.99	1.00	312
Baixo	1.00	1.00	1.00	6235
Moderado	0.99	1.00	0.99	1292
accuracy			1.00	7839
macro avg	1.00	1.00	1.00	7839
weighted avg	1.00	1.00	1.00	7839

Fonte: Google Colab (2023)

Na matriz de confusão demonstrada na Figura 45 pode se observar que o algoritmo acertou a classificação de 7.825 registros do total de 7.839 registros e errou 14 registros no total. Pelo cassification report se observa o resultado de 1,00 de precisão, indicando que dentre os exemplos classificados como verdadeiros 100% são realmente verdadeiros. A medida de recall

resultou em 1,00, o que indica que 100% dos dados foram classificados como verdadeiros em comparação com a quantidade real de resultados verdadeiros que existem na amostra. A F1-score 1,00, demonstra a média ponderada de precisão e recall, mostrando a qualidade geral do modelo em 100%.

7.5 TUNING DOS PARÂMETROS

Mesmo tendo bons resultados nos algoritmos aplicado, empregou-se o método GridSearch para encontrar hiper parâmetros mais adequados para o uso do algoritmo. O algoritmo Naive Bayes não possui hiper parâmetros, portanto não consta nesta análise.

Figura 47 – Tuning dos parâmetros

```
from sklearn.model_selection import GridSearchCV

parametros_rna = {'activation': ['relu', 'logistic', 'tahn'],
              'solver': ['adam', 'sgd'],}

grid_search = GridSearchCV(estimator=MLPClassifier(), param_grid=parametros_rna)
grid_search.fit(x_consumo, y_consumo)
melhores_parametros = grid_search.best_params_
melhor_resultado = grid_search.best_score_
print(melhores_parametros)
print(melhor_resultado)

parametros_svc = {'tol': [0.001, 0.0001, 0.00001],
              'C': [1.0, 1.5, 2.0],
              'kernel': ['rbf', 'linear', 'poly', 'sigmoid']}

grid_search = GridSearchCV(estimator=SVC(), param_grid=parametros_svc)
grid_search.fit(x_consumo, y_consumo)
melhores_parametros = grid_search.best_params_
melhor_resultado = grid_search.best_score_
print(melhores_parametros)
print(melhor_resultado)

parametros_logistic = {'tol': [0.0001, 0.00001, 0.000001],
              'C': [1.0, 1.5, 2.0],
              'solver': ['lbfgs', 'sag', 'saga']}

grid_search = GridSearchCV(estimator=LogisticRegression(), param_grid=parametros_logistic)
grid_search.fit(x_consumo, y_consumo)
melhores_parametros = grid_search.best_params_
melhor_resultado = grid_search.best_score_
print(melhores_parametros)
print(melhor_resultado)
```

Fonte: Google Colab (2023)

Após aplicação deste método, se optou por utilizar os algoritmos com os parâmetros iniciais, devido aos melhores resultados encontrados.

Tabela 4 – Comparação antes e após GridSearch

Algoritmo	**Acurácia antes do GridSearch**	**Acurácia após do GridSearch**
Naive Bays	0,9788	-
Redes neurais	0,9994	0,9936
SVM	0,9983	0,9930
Regressão logística	0,9982	0,9950

Fonte: Elaborado pelo autor (2023)

8. APRESENTAÇÃO DOS RESULTADOS

É possível verificar na Tabela 5 um comparativo entre os modelos de *machine learning* aplicados no presente estudo.

Tabela 5 - Comparação entre os modelos de *machine learning*

Algoritmo	Acurácia de previsão	Precisão	Recall	F1-Score
Naive Bays	0,9788	0,95	0,98	0,97
Redes neurais	0,9994	1,00	1,00	1,00
SVM	0,9983	0,99	1,00	1,00
Regressão logística	0,9982	1,00	1,00	1,00

Fonte: Elaborado pelo autor (2022)

Analisando a tabela 5 e as figuras 41 e 42 do capítulo 7.2, o algoritmo de melhor desempenho foi o de redes neurais, possuindo melhor acurácia e demais medidas de avaliação. Para validação dos modelos aplicados foi utilizado o método de validação cruzada por meio da importação da biblioteca cross_val_score do Sklearn, realizando 30 testes de cada algoritmo.

Figura 48 – Validação cruzada

```
from sklearn.model_selection import cross_val_score, KFold

resultados_naive = []
resultados_rna = []
resultados_svm = []
resultados_logistic = []

for i in range(30):
  print(i)
  kfold = KFold(n_splits=10, shuffle=True, random_state=i)

  naive = GaussianNB()
  scores = cross_val_score(naive, x_consumo, y_consumo, cv = kfold)
  resultados_naive.append(scores.mean())

  logistic = LogisticRegression(random_state = 1)
  scores = cross_val_score(logistic, x_consumo, y_consumo, cv = kfold)
  resultados_logistic.append(scores.mean())

  svm = SVC(kernel='linear', random_state=1)
  scores = cross_val_score(svm, x_consumo, y_consumo, cv = kfold)
  resultados_svm.append(scores.mean())

  rede_neural = MLPClassifier(activation='relu', solver='adam', max_iter=1000, hidden_layer_sizes=(3,3),tol=0.00001)
  scores = cross_val_score(rede_neural, x_consumo, y_consumo, cv = kfold)
  resultados_rna.append(scores.mean())
```

Fonte: Google Colab (2023)

Após os 30 testes foi possível extrair dados estatísticos para auxiliar na definição do melhor algoritmo de *machine learning* para este estudo.

Figura 49 – Dados estatísticos

```
resultados.describe()
```

	Naive Bayes	Logistica	SVM	Rede neural
count	30.000000	30.000000	30.000000	30.000000
mean	0.976825	0.997781	0.998319	0.995699
std	0.000086	0.000046	0.000104	0.009531
min	0.976655	0.997704	0.998086	0.950185
25%	0.976759	0.997744	0.998246	0.995511
50%	0.976815	0.997768	0.998342	0.999410
75%	0.976870	0.997823	0.998374	0.999577
max	0.977006	0.997863	0.998501	0.999745

Fonte: Google Colab (2023)

A média dos resultados dos algoritmos de naive bayes, regressão logística, SVM e rede neural foram 0.97, 0.99, 0.99, 0.99 respectivamente. O desvio padrão “std” apresentado na Figura 49 mostra que o algoritmo de regressão logística teve o menor desvio padrão, de 0.000046, o que indica que o algoritmo não sofreu grande variação nos testes realizados. Entretanto, o algoritmo de redes neurais teve média de 0.9995 para 75% da amostra, além do resultado obtido no capítulo 7.2. Portanto, concluindo que para este estudo o algoritmo de machine learning de redes neurais é o que apresentou o melhor desempenho.

9. LINKS

A seguir, estão os links do vídeo de apresentação e do repositório contendo os dados utilizados no estudo.

Link para o vídeo: https://youtu.be/6eT70K1Zd6s
Link para o repositório: https://github.com/Tainara-guadagnin/TCC_PUC_Minas

REFERÊNCIAS

DEVORE, Jay L. **Probabilidade e estatística para engenharia e ciências.** Tradução: Solange A. Visconte. São Paulo: Cengage Learning, 2019. 630 p. Título original: Probability and Statistics for Engineering and the Sciences. ISBN 13: 978-1-337-09426-9.

DOWNING, Douglas; CLARK, Jeffrey. Estatística aplicada. 3° ed. São Paulo: Saraiva, 2010.

FERNEDA, Edberto. Redes neurais e sua aplicação em sistemas de recuperação de informação. **Ci. Inf**, Brasília, v. 35, n. 1, p. 25-30, jan. 2006.

HINES, William W.; MONTGOMERY, Douglas C.; GOLDSMAN, Dave; BORROR, Connie M. **Probabilidade e estatística na engenharia.** 4 ed. Rio de Janeiro:LTC, 2004.

SANTOS, Alcione Miranda de; SEIXAS, José Manoel de; PEREIRA, Basílio de Bragança; MEDRONHO, Roberto de Andrade. Usando Redes Neurais Artificiais e Regressão Logística na Predição da Hepatite A. Rev. Bras. Epidemiol, v. 8, n. 2, p. 117-126, 2005.

Wolker SL, Costa TP, Peterlini OLG. **Revisão integrativa sobre o processo de compra e distribuição de materiais médicos e hospitalares.** R. Saúde Públ. Paraná. 2019 Jul;2(Suppl 1): 103-112