

PONTIFÍCIA UNIVERSIDADE CATÓLICA DE MINAS GERAIS

NÚCLEO DE EDUCAÇÃO A DISTÂNCIA

Pós-graduação *Lato Sensu* em Ciência de Dados e Big Data

Gustavo Lucas Rodrigues de Nassau

**PREDIÇÃO DA NECESSIDADE DE UTI PARA PACIENTES INFECTADOS POR
COVID-19**

Belo Horizonte

2022

Gustavo Lucas Rodrigues de Nassau

**PREDIÇÃO DA NECESSIDADE DE UTI PARA PACIENTES INFECTADOS POR
COVID-19**

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.

Belo Horizonte

2022

SUMÁRIO

| | |
|--|-----------|
| 1. Introdução | 4 |
| 1.1. Contextualização | 4 |
| 1.2. O problema proposto | 5 |
| 1.3. Objetivos | 6 |
| 2. Coleta de Dados | 7 |
| 3. Processamento/Tratamento de Dados | 14 |
| 4. Análise e Exploração dos Dados | 26 |
| 4.1 Óbitos por covid | 26 |
| 4.2 Casos confirmados Covid | 32 |
| 4.3 Mortes por Covid-19 | 39 |
| 5. Criação de Modelos de Machine Learning | 40 |
| Random Forest | 47 |
| Extra Trees Classifier | 48 |
| Logistic Regression | 49 |
| KNeighbors Classifier (KNN) | 50 |
| Naive Bayes | 51 |
| SGD Classifier | 52 |
| Linear Discriminant Analysis | 53 |
| 6. Interpretação dos Resultados | 54 |
| 7. Apresentação dos Resultados | 58 |
| 8. Links | 59 |
| REFERÊNCIAS | 60 |
| APÊNDICE | 60 |

1. Introdução

1.1. Contextualização

Coronavírus são uma grande família de vírus que podem causar doenças graves ao ser humano. A primeira epidemia grave conhecida é a Síndrome Respiratória Aguda Grave (SARS), ocorrida em 2003, enquanto o segundo surto de doença grave começou em 2012 na Arábia Saudita com o Drome-Drome Respiratório do Oriente Médio (MERS). O atual surto de doença devido ao coronavírus é relatado no final de dezembro de 2019. Este novo vírus é muito contagioso e se espalhou rapidamente globalmente. Em 30 de janeiro de 2020, a Organização Mundial da Saúde (OMS) declarou o surto uma Emergência de Preocupação Internacional em Saúde Pública (PHEIC), pois havia se espalhado por 18 países. Em 11 de fevereiro de 2020, a OMS nomeou este "COVID-19". Em 11 de março, como o número de casos de COVID-19 aumentou treze vezes além da China, com mais de 118.000 casos em 114 países e mais de 4.000 mortes, a OMS declarou isso uma pandemia.

Como o surto do COVID-19 tornou-se uma pandemia mundial, são necessárias análises em tempo real dos dados epidemiológicos para preparar a sociedade com melhores planos de ação contra a doença. Desde o nascimento do romance COVID- 19 [1], o mundo está lutando inquietamente com sua causa. Até 1º de abril de 2020, com base nos dados ao vivo compartilhados globalmente pelo painel da Johns Hopkins, em todo o mundo há 932.605 casos confirmados, dos quais 193.177 foram recuperados e 46.809 perderam suas vidas [2]. COVID-19 pertence à família do SARS-CoV e MERS-CoV, onde começa com os sintomas de nível inicial do frio ao nível grave de doenças respiratórias causando dificuldade na respiração, cansaço, febre e tosse seca [3]. Muitos ensaios clínicos em andamento estão avaliando possíveis tratamentos.

Os estágios de transmissão de acordo com os relatórios da OMS, a situação da pandemia é classificada em quatro estágios. A primeira etapa começa com os casos notificados para as pessoas que viajaram em regiões já afetadas, enquanto

na segunda etapa, os casos são relatados localmente entre familiares, amigos e outros que entraram em contato com o vírus que chega das regiões afetadas. Neste momento, as pessoas afetadas são rastreáveis. Mais tarde, a terceira etapa torna a situação ainda pior à medida que a fonte transmissão se torna indetectável e se espalha pelos indivíduos que não têm histórico de viagem nem entraram em contato com os afetados pelo vírus. Essa situação exige um bloqueio imediato em todo o país para reduzir os contatos sociais entre os indivíduos e controlar a taxa de transmissão. O pior de todos, é o estágio 4 quando a transmissão se torna endêmica e incontrolável. Até agora, vários países entraram na fase 4. A China é a primeira nação que experimentou a fase 4 da transmissão COVID-19. No caso da China, observa-se que o crescimento exponencial dos casos confirmados chega à fase de saturação onde o número de casos parou de crescer. Isso se decorre do fato de que o número de pessoas suscetíveis, que são expostas ao vírus, é drasticamente reduzido. Isso foi possível devido à redução do contato social entre as pessoas, segregando os indivíduos infectados em quarentena e um bloqueio completo foi iniciado pelo governo chinês, reduzindo assim a possibilidade de maior disseminação.

Algoritmos de aprendizagem de máquina desempenham um papel importante na análise e previsão de epidemias. Na presença de dados epidêmicos maciços, as técnicas de aprendizado de máquina ajudam a encontrar os padrões epidêmicos para que a ação inicial possa ser planejada para impedir a propagação do vírus.

Modelos de aprendizagem são usados para observar o comportamento cotidiano, juntamente com a previsão de alcance futuro do COVID-2019 em todo o país, utilizando as informações em tempo real do painel da Johns Hopkins.

O resto do artigo é organizado da seguinte forma: a seção 2 contém um breve discussão sobre abordagens baseadas em aprendizado de máquina existentes, enquanto a seção 3 fala sobre o conjunto de dados usado para experimentos. A análise epidêmica e a estratégia experimental são discutidas nos trechos 4 e 5, respectivamente, enquanto os resultados são discutidos na seção 6. Observações finais e trabalhos futuros são discutidos na seção 6.

1.2. O problema proposto

Iremos passar a técnica 5-Ws para descrever o problema de forma clara e organizada.

(Why?) Por que esse problema é importante?

O coronavírus e sua rápida infecção foram assunto nos últimos dois anos, e entender como ele se comporta, e principalmente, seu tratamento que ajuda a prevenir mortes é um assunto crítico e de importância o suficiente talvez até para a sobrevivência de sociedades inteiras.

(Who?) De quem são os dados analisados? De um governo? Um ministério ou secretaria? Dados de clientes?

Os dados analisados são do governo do estado de Minas Gerais (MG) e de um portal Brasil.io que tem como objetivo facilitar o acesso a dados públicos brasileiros, no caso das Secretarias Estaduais de Saúde.

(What?) Quais os objetivos com essa análise? O que iremos analisar?

Iremos analisar utilizando os *datasets* encontrados, as variáveis dos pacientes hospitalizados que podem levá-los a necessitar de internação na UTI ou não, ajudando hospitais a prever ocupação e até mesmo otimizar os recursos para tratamentos de COVID-19.

(Where?) Trata dos aspectos geográficos e logísticos de sua análise.

No presente trabalho iremos trabalhar apenas com dados do estado de Minas Gerais (MG), onde fui nascido e criado.

(When?) Qual o período está sendo analisado? A última semana? Os últimos 6 meses? O ano passado?

O período analisado é no ano de 2020, mais especificamente entre 03/04/2020 até 12/08/2020.

1.3. Objetivos

O objetivo com o presente trabalho é identificar através de algoritmos de Machine Learning fatores que possam levar uma pessoa a ser internada na UTI com a maior assertividade possível, e dessa forma ajudando instituições de saúde a otimizar a quantidade de leitos e recursos, seja dispensando aqueles pacientes que conforme a previsão do algoritmo não precisam de internação ou internando imediatamente aqueles que forem classificados com esta necessidade.

2. Coleta de Dados

Para cumprir o objetivo citado anteriormente, serão utilizados três datasets extraídos de fontes diferentes relacionados ao COVID-19, com dados entre o período 03/04/2020 até 12/08/2020.

O primeiro dataset “obitos.csv” possui dados sobre óbitos confirmados de COVID-19 e foi extraído diretamente do Portal de Dados Abertos do Estado de Minas Gerais no formato .csv e através do link: <https://dados.mg.gov.br/dataset/obitos-confirmados-covid-19>. Ele possui a seguinte estrutura:

| Nome da coluna/campo | Descrição | Tipo |
|----------------------|---|----------------|
| _id | Index da tabela | Int64 |
| PACIENTE | Cada paciente é identificado com um único número para preservação de sua identidade | int64 |
| SEXO | Masculino ou feminino | Texto (object) |

| Nome da coluna/campo | Descrição | Tipo |
|----------------------|---|----------------|
| IDADE | Idade do paciente na data do óbito | int64 |
| MUNICIPIO_RESIDENCIA | Município do Estado de Minas Gerais onde residia o paciente que veio a óbito | Texto (object) |
| DATA_OBITO | Data do óbito, no formato YYYY-MM-DD | Texto (object) |
| COMORBIDADE | Presença de doenças preexistentes/ comorbidades (diabetes, hipertensão, etc. Dividido entre SIM e NÃO | Texto (object) |

O segundo dataset “casos confirmados.csv” possui dados de casos confirmados por Covid, também foi extraído do Portal de Dados Abertos do Estado de Minas Gerais no formato .csv e do seguinte link:

<https://dados.mg.gov.br/dataset/casos-confirmados-covid-19>.

Este, possui a seguinte estrutura:

| Nome da coluna/campo | Descrição | Tipo |
|----------------------|--|----------------|
| _id | Index da tabela | int64 |
| URS | Unidade Regional de Saúde do município de residência do paciente | Texto (object) |

| Nome da coluna/campo | Descrição | Tipo |
|----------------------|---|----------------|
| MICRO | Microrregião de Saúde do município de residência do paciente | Texto (object) |
| MACRO | Macrorregião de Saúde do município de residência do paciente | Texto (object) |
| ID | Cada paciente é identificado com um número para preservação de sua identidade. O ID não representa contagem numérica dos casos confirmados. | int64 |
| DATA_NOTIFICACAO | Data de notificação do caso | Texto (object) |
| CLASSIFICACAO_CASO | Classificação do caso | Texto (object) |
| SEXO | Masculino ou feminino | Texto (object) |
| IDADE | Idade do paciente. Valores ausentes estão codificados como NA. | float64 |
| FAIXA_ETARIA | Faixa Etária do paciente. | Texto (object) |
| MUNICIPIO_RESIDENCIA | Município de residência do paciente. O valor OUTROS representa município de | Texto (object) |

| Nome da coluna/campo | Descrição | Tipo |
|----------------------|--|----------------|
| | residência do paciente externo ao Estado de Minas Gerais. | |
| CODIGO | Código IBGE do município de residência do paciente | float64 |
| COMORBIDADE | Presença de doenças preexistentes/ comorbidades (diabetes, hipertensão, etc.), com possibilidade de preenchimento SIM ou NÃO | Texto (object) |
| EVOLUCAO | Evolução do paciente | Texto (object) |
| INTERNACAO | Informa se o paciente com caso confirmado de COVID 19 foi internado, com possibilidade de preenchimento SIM ou NÃO. Valores ausentes estão codificados como NA. | Texto (object) |
| UTI | Informa se o paciente com caso confirmado de COVID 19 internado, precisou de UTI (Unidade de Terapia Intensiva) com possibilidade de preenchimento SIM ou NÃO. Valores ausentes estão codificados como NA. | Texto (object) |

| Nome da coluna/campo | Descrição | Tipo |
|----------------------|--|----------------|
| RACA | Raça do paciente | Texto (object) |
| DATA_ATUALIZACAO | Data de upload do arquivo | Texto (object) |
| ORIGEM_DA_INFORMACAO | Fonte do dado correspondente a cada linha do arquivo | Texto (object) |
| ORIGEM_DA_INFORMACAO | Fonte do dado correspondente a cada linha do arquivo | Texto (object) |

Por fim, o terceiro dataset, também nos dá informações sobre casos de óbito, porém além disso, a taxa de morte por município e foi extraído em formato csv do link: <https://brasil.io/dataset/covid19/caso/>, com a seguinte estrutura:

| Nome da coluna/campo | Descrição | Tipo |
|----------------------|--|----------------|
| date | data de coleta dos dados no formato YYYY-MM-DD. | Texto (object) |
| state | sigla da unidade federativa, exemplo: SP. | Texto (object) |
| city | nome do município (pode estar em branco quando o registro é referente ao estado, pode ser preenchido com Importados/Indefinidos também). | Texto (object) |
| place_type | tipo de local que esse registro descreve, pode ser city ou state. | Texto (object) |
| confirmed | número de casos confirmados. | int64 |
| deaths | número de mortes. | int64 |
| order_for_place | número que identifica a ordem do registro para este local. O registro referente ao primeiro boletim em que esse local aparecer será contabilizado como 1 e os demais | int64 |

| Nome da coluna/campo | Descrição | Tipo |
|--------------------------------|--|---------|
| | boletins incrementarão esse valor. | |
| is_last | campo pré-computado que diz se esse registro é o mais novo para esse local, pode ser True ou False (caso filtre por esse campo, use is_last=True ou is_last=False) | bool |
| estimated_population_2019 | população estimada para esse município/estado em 2020, segundo o IBGE. | float64 |
| estimated_population | população estimada para esse município/estado em 2019, segundo o IBGE. ATENÇÃO: essa coluna possui valores desatualizados, prefira usar a coluna estimated_population. | float64 |
| city_ibge_code | código IBGE do local. | float64 |
| confirmed_per_100k_inhabitants | número de casos confirmados por 100.000 habitantes (baseado em estimated_population). | float64 |
| death_rate | taxa de mortalidade (mortes / confirmados). | float64 |

3. Processamento/Tratamento de Dados

Para realizar o presente trabalho, utilizei o ambiente Jupyter Notebook na versão 6.1.12 e a linguagem Python na versão 3.9.12.

Para o processamento e tratamento de dados, foram utilizadas três bibliotecas, “pandas”, “numpy” e “sklearn.utils”.

```
1 import pandas as pd
2 import numpy as np
3 from sklearn.utils import resample
.
```

Para o dataset “obitos.csv” o primeiro passo foi fazer sua importação através da função “read_csv” do pandas.

Importando CSV

```
In [55]: 1 obitos = pd.read_csv('obitos.csv', index_col=False)
```

Após a importação, identificamos que o dataset continha 2.894 linhas e 7 colunas e também verificamos os tipos de dados contidos para entender se seria necessário alguma modificação.

```
In [76]: 1 obitos
```

Out[76]:

| | _id | PACIENTE | SEXO | IDADE | MUNICIPIO_RESIDENCIA | DATA_OBITO | COMORBIDADE |
|------|------|----------|------|-------|-----------------------|---------------------|---------------|
| 0 | 1 | 1 | M | 79 | Patos de Minas | 2020-03-28T00:00:00 | SIM |
| 1 | 2 | 2 | F | 82 | Belo Horizonte | 2020-03-29T00:00:00 | SIM |
| 2 | 3 | 3 | M | 66 | Belo Horizonte | 2020-03-30T00:00:00 | SIM |
| 3 | 4 | 4 | M | 44 | Mariana | 2020-03-30T00:00:00 | NÃO |
| 4 | 5 | 5 | M | 80 | Uberlândia | 2020-03-30T00:00:00 | SIM |
| ... | ... | ... | ... | ... | ... | ... | ... |
| 2889 | 2890 | 2890 | M | 86 | Andradas | 2020-07-30T00:00:00 | SIM |
| 2890 | 2891 | 2891 | F | 68 | Borda da Mata | 2020-07-30T00:00:00 | SIM |
| 2891 | 2892 | 2892 | M | 55 | Iturama | 2020-07-31T00:00:00 | Não informado |
| 2892 | 2893 | 2893 | F | 81 | Conceição das Alagoas | 2020-01-08T00:00:00 | Não informado |
| 2893 | 2894 | 2894 | M | 68 | Pouso Alegre | 2020-01-08T00:00:00 | SIM |

```
In [57]: 1 #Entendendo tipo dos dados
          2 obitos.info()

<class 'pandas.core.frame.DataFrame'>
RangeIndex: 2894 entries, 0 to 2893
Data columns (total 7 columns):
#   Column                                Non-Null Count  Dtype
---  -
0   _id                                    2894 non-null   int64
1   PACIENTE                             2894 non-null   int64
2   SEXO                                  2894 non-null   object
3   IDADE                                 2894 non-null   int64
4   MUNICIPIO_RESIDENCIA                 2894 non-null   object
5   DATA_OBITO                          2823 non-null   object
6   COMORBIDADE                          2894 non-null   object
dtypes: int64(3), object(4)
memory usage: 158.4+ KB
```

Como não foi necessária nenhuma alteração nos tipos dos dados, o próximo passo foi verificar a existência de valores nulos.

```
In [58]: 1 #verificando existência de valores nulos
          2 obitos.isnull().sum()

Out[58]: _id                                0
          PACIENTE                         0
          SEXO                             0
          IDADE                            0
          MUNICIPIO_RESIDENCIA             0
          DATA_OBITO                      71
          COMORBIDADE                      0
          dtype: int64
```

Foi identificado que a coluna “DATA_OBITO” possuía 71 linhas com valores nulos e com isso removemos esses valores utilizando um filtro com a função “notnull()”.

```
In [59]: 1 #removendo valores nulos
          2 obitos = obitos[obitos['DATA_OBITO'].notnull()]
```

Continuando o tratamento dos dados, o objetivo neste dataset era coletar a média de idade dos óbitos por município e por isso as outras colunas que não seriam utilizadas foram removidas, restando apenas aquelas que usaremos no fim.

```
In [83]: 1 #retirando colunas que não serão usadas
2 obitos = obitos.loc[:,['IDADE','MUNICIPIO_RESIDENCIA']]
3 obitos.head()
```

```
Out[83]:
```

| | IDADE | MUNICIPIO_RESIDENCIA |
|---|-------|----------------------|
| 0 | 79 | Patos de Minas |
| 1 | 82 | Belo Horizonte |
| 2 | 66 | Belo Horizonte |
| 3 | 44 | Mariana |
| 4 | 80 | Uberlândia |

Para realizar um join no fim do processamento dos três datasets, foi necessária uma normalização nos nomes dos municípios que foram formatados em letra maiúscula e sem acentuação.

```
In [84]: 1 #corrigindo coluna MUNICIPIO_RESIDENCIA, deixando maiuscula e sem acentuação
2
3 obitos['MUNICIPIO_RESIDENCIA'] = obitos['MUNICIPIO_RESIDENCIA'].str.normalize('NFKD')\
4     .str.encode('ascii', errors='ignore')\
5     .str.decode('utf-8').astype(str).str.upper()
6
7 obitos
```

```
Out[84]:
```

| | IDADE | MUNICIPIO_RESIDENCIA |
|------|-------|-----------------------|
| 0 | 79 | PATOS DE MINAS |
| 1 | 82 | BELO HORIZONTE |
| 2 | 66 | BELO HORIZONTE |
| 3 | 44 | MARIANA |
| 4 | 80 | UBERLANDIA |
| ... | ... | ... |
| 2889 | 86 | ANDRADAS |
| 2890 | 68 | BORDA DA MATA |
| 2891 | 55 | ITURAMA |
| 2892 | 81 | CONCEICAO DAS ALAGOAS |
| 2893 | 68 | POUSO ALEGRE |

2894 rows × 2 columns

Foi feito um agrupamento pela coluna “MUNICIPIO_RESIDENCIA” pegando a média da coluna “IDADE” que também foi renomeada para “MEDIA_IDADE_POR_MUNICIPIO”, restando assim 398 linhas.

```
In [103]: 1 #agrupando por municipios... média de idade e soma de óbitos por municipio
2 obitos = obitos.groupby(by = 'MUNICIPIO_RESIDENCIA', as_index = False).agg( {'IDADE': 'mean'}).round()
3 obitos = obitos.rename({'IDADE': 'MEDIA_IDADE_POR_MUNICIPIO'}, axis=1) #renomeando coluna idade
4 obitos
```

Out[103]:

| | MUNICIPIO_RESIDENCIA | MEDIA_IDADE_POR_MUNICIPIO |
|-----|------------------------|---------------------------|
| 0 | ABADIA DOS DOURADOS | 60.0 |
| 1 | ACAICA | 85.0 |
| 2 | ACUCENA | 60.0 |
| 3 | AGUANIL | 63.0 |
| 4 | AGUAS FORMOSAS | 74.0 |
| ... | ... | ... |
| 393 | VERISSIMO | 84.0 |
| 394 | VERMELHO NOVO | 64.0 |
| 395 | VESPASIANO | 57.0 |
| 396 | VISCONDE DO RIO BRANCO | 77.0 |
| 397 | VOLTA GRANDE | 71.0 |

398 rows x 2 columns

Por fim, foi verificada a existência de linhas duplicadas na coluna “MUNICIPIO_RESIDENCIA” com o comando “drop_duplicates()”.

```
In [104]: 1 #Verificando e Removendo duplicatas
2 print('Linhas antes de remover duplicatas:', obitos['MUNICIPIO_RESIDENCIA'].shape[0])
3 obitos = obitos.drop_duplicates()
4 print('Linhas depois de remover duplicatas:', obitos['MUNICIPIO_RESIDENCIA'].shape[0])
```

Linhas antes de remover duplicatas: 398
Linhas depois de remover duplicatas: 398

Partindo agora para o dataset “casos confirmados.csv” buscamos seguir o mesmo padrão de tratamento de dados em todos os datasets, então os passos são bastante semelhantes salvo algumas poucas exceções que serão comentadas a seguir.

Importando Dataset, nos deparamos com um dataset de 46.446 linhas e 19 colunas.

In [4]:

```
1 casos_confirmados = pd.read_csv('casos confirmados.csv')
2 casos_confirmados
```

| | | | | | | | | | | |
|-------|-------|---------------|---------------------------|----------|-------|---------------------|-----------------|-----------|------|--------------|
| 3 | 4 | NaN | NaN | NaN | 4 | 2020-06-14T00:00:00 | CASO CONFIRMADO | MASCULINO | 36.0 | 30 A 39 ANOS |
| 4 | 5 | TEOFILO OTONI | TEOFILO OTONI/MALACACHETA | NORDESTE | 5 | 2020-06-14T00:00:00 | CASO CONFIRMADO | MASCULINO | 51.0 | 50 A 59 ANOS |
| ... | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... |
| 46441 | 46442 | UBA | UBA | SUDESTE | 46442 | 2020-07-24T00:00:00 | CASO CONFIRMADO | MASCULINO | 20.0 | 20 A 29 ANOS |
| 46442 | 46443 | NaN | NaN | NaN | 46443 | 2020-07-28T00:00:00 | CASO CONFIRMADO | FEMININO | 39.0 | 30 A 39 ANOS |
| 46443 | 46444 | ALFENAS | GUAXUPE | SUL | 46444 | 2020-08-03T00:00:00 | CASO CONFIRMADO | FEMININO | 39.0 | 30 A 39 ANOS |
| 46444 | 46445 | DIVINOPOLIS | BOM DESPACHO | OESTE | 46445 | 2020-08-06T00:00:00 | CASO CONFIRMADO | MASCULINO | 27.0 | 20 A 29 ANOS |
| 46445 | 46446 | DIVINOPOLIS | DIVINOPOLIS | OESTE | 46446 | 2020-08-08T00:00:00 | CASO CONFIRMADO | MASCULINO | 50.0 | 50 A 59 ANOS |

46446 rows x 19 columns

Logo após, buscamos entender os tipos de dados presentes no dataset através da função “info()”.

In [544]:

```
1 #Entendendo tipo dos dados
2 casos_confirmados.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 46446 entries, 0 to 46445
Data columns (total 19 columns):
#   Column                                Non-Null Count  Dtype
---  -
0   _id                                   46446 non-null  int64
1   URS                                  45238 non-null  object
2   MICRO                               45238 non-null  object
3   MACRO                               45238 non-null  object
4   ID                                   46446 non-null  int64
5   DATA_NOTIFICACAO                   39302 non-null  object
6   CLASSIFICACAO_CASO                  46446 non-null  object
7   SEXO                                46446 non-null  object
8   IDADE                               45581 non-null  float64
9   FAIXA_ETARIA                        45552 non-null  object
10  MUNICIPIO_RESIDENCIA                 46446 non-null  object
11  CODIGO                               45238 non-null  float64
12  COMORBIDADE                          46446 non-null  object
13  EVOLUCAO                             46446 non-null  object
14  INTERNACAO                           46446 non-null  object
15  UTI                                  46446 non-null  object
16  RACA                                  46446 non-null  object
17  DATA_ATUALIZACAO                    46446 non-null  object
18  ORIGEM_DA_INFORMACAO                 46446 non-null  object
dtypes: float64(2), int64(2), object(15)
memory usage: 6.7+ MB
```

Neste caso, também não foi necessário nenhum tipo de alteração. Então partimos para verificação de dados nulos nas colunas através da função “isnull()”.

```
In [545]: 1 #verificando existência de valores nulos
          2 casos_confirmados.isnull().sum()
```

```
Out[545]: _id          0
          URS          1208
          MICRO        1208
          MACRO        1208
          ID           0
          DATA_NOTIFICACAO  7144
          CLASSIFICACAO_CASO  0
          SEXO         0
          IDADE        865
          FAIXA_ETARIA  894
          MUNICIPIO_RESIDENCIA  0
          CODIGO       1208
          COMORBIDADE   0
          EVOLUCAO      0
          INTERNACAO    0
          UTI           0
          RACA          0
          DATA_ATUALIZACAO  0
          ORIGEM_DA_INFORMACAO  0
          dtype: int64
```

Foram identificadas várias colunas com diversos valores nulos, tendo em vista que estes valores poderiam atrapalhar nosso resultado final, eles foram retirados do dataset, restando 37.274 linhas, uma quantidade consideravelmente inferior ao dataset original.

```
In [546]: 1 #retirando valores nulos da coluna 'DATA_OBITO', 'FAIXA_ETARIA' e 'MACRO'
2 casos_confirmados = casos_confirmados[(casos_confirmados['DATA_NOTIFICACAO'].notnull() &
3                                           (casos_confirmados['FAIXA_ETARIA'].notnull() &
4                                           (casos_confirmados['MACRO'].notnull()))]
5 casos_confirmados.isnull().sum()

Out[546]: _id                0
URS                  0
MICRO                0
MACRO                0
ID                   0
DATA_NOTIFICACAO     0
CLASSIFICACAO_CASO   0
SEXO                 0
IDADE                0
FAIXA_ETARIA         0
MUNICIPIO_RESIDENCIA 0
CODIGO               0
COMORBIDADE          0
EVOLUCAO              0
INTERNACAO           0
UTI                  0
RACA                 0
DATA_ATUALIZACAO     0
ORIGEM_DA_INFORMACAO 0
dtype: int64
```

Seguindo a mesma lógica do último dataset, foram selecionadas apenas as colunas de interesse que foram selecionadas como possíveis variáveis que podem ser relevantes na justificativa de internação.

```
In [547]: 1 #retirando colunas que não serão usadas
2 casos_confirmados = casos_confirmados.loc[:,['MACRO', 'DATA_NOTIFICACAO',
3        'CLASSIFICACAO_CASO', 'SEXO', 'IDADE',
4        'MUNICIPIO_RESIDENCIA', 'COMORBIDADE', 'EVOLUCAO',
5        'INTERNACAO', 'UTI', 'RACA']]
6 casos_confirmados.head()

Out[547]:
```

| | MACRO | DATA_NOTIFICACAO | CLASSIFICACAO_CASO | SEXO | IDADE | MUNICIPIO_RESIDENCIA | COMORBIDADE | EVOLUCAO | INTERNACAO |
|---|----------|---------------------|--------------------|-----------|-------|----------------------|--------------------|--------------|---------------|
| 1 | SUDESTE | 2020-05-08T00:00:00 | CASO CONFIRMADO | MASCULINO | 33.0 | JUIZ DE FORA | NAO INFORMADO | RECUPERADO | NAC |
| 2 | LESTE | 2020-06-19T00:00:00 | CASO CONFIRMADO | FEMININO | 25.0 | GOVERNADOR VALADARES | NAO ACOMPANHAMENTO | EM | NAC |
| 4 | NORDESTE | 2020-06-14T00:00:00 | CASO CONFIRMADO | MASCULINO | 51.0 | TEOFILO OTONI | NAO INFORMADO | INVESTIGACAO | SIN |
| 5 | LESTE | 2020-06-05T00:00:00 | CASO CONFIRMADO | MASCULINO | 41.0 | GOVERNADOR VALADARES | NAO | RECUPERADO | NAC |
| 6 | CENTRO | 2020-05-15T00:00:00 | CASO CONFIRMADO | FEMININO | 27.0 | BELO HORIZONTE | NAO INFORMADO | RECUPERADO | NAC INFORMADO |

Como próximo passo, a fim de normalizar também neste dataset a coluna “MUNICIPIO_RESIDENCIA”, os itens dessa coluna foram alterados para letras maiúsculas e sem acentuação.

```
In [548]: 1 #corrigindo coluna MUNICIPIO_RESIDENCIA, deixando maiuscula e sem acentuação
2 casos_confirmados = casos_confirmados.apply(lambda x: x.astype(str).str.upper())
3 casos_confirmados['MUNICIPIO_RESIDENCIA'] = casos_confirmados['MUNICIPIO_RESIDENCIA'].str.normalize('NFKD')\
4     .str.encode('ascii', errors='ignore')\
5     .str.decode('utf-8')
```

Por fim, verificamos a quantidade de linhas duplicadas existentes no dataset e as removemos. Neste caso, haviam 2.503 linhas duplicadas nos restando no final de todas as transformações 34.701 linhas.

```
In [549]: 1 #Verificando e Removendo duplicatas
          2 print('Casos confirmados')
          3 print('Linhas antes de remover duplicatas:', casos_confirmados.shape[0])
          4 casos_confirmados = casos_confirmados.drop_duplicates()
          5 print('Linhas depois de remover duplicatas:', casos_confirmados.shape[0])
```

Casos confirmados
 Linhas antes de remover duplicatas: 37274
 Linhas depois de remover duplicatas: 34701

E com isso, chegamos ao final do tratamento do dataset “casos confirmados.csv” onde temos diversas colunas que se tornarão variáveis que podem ser utilizadas para entender a necessidade de internação na UTI ou não.

Temos agora o dataset “caso.csv” que foi retirado do site <https://brasil.io/dataset/covid19/files/>, que dentre outros dados possui coluna de interesse “death_rate” por municípios de todo o Brasil.

Assim como nos últimos dois tratamentos, o primeiro passo é importar o arquivo. Ao fazer isso, vemos que este é o maior de todos eles com 2.838.003 linhas e 13 colunas.

```
In [5]: 1 casos_e_mortes_confirmadas = pd.read_csv('caso.csv', index_col=False)
```

```
In [6]: 1 casos_e_mortes_confirmadas
```

Out[6]:

| | date | state | city | place_type | confirmed | deaths | order_for_place | is_last | estimated_population_2019 | estimated_population | city_ibge_code | confirmer |
|---------|------------|-------|------|------------|-----------|--------|-----------------|---------|---------------------------|----------------------|----------------|-----------|
| 0 | 2022-03-27 | AP | NaN | state | 160328 | 2122 | 734 | True | 845731.0 | 861773.0 | 16.0 | |
| 1 | 2022-03-26 | AP | NaN | state | 160321 | 2122 | 733 | False | 845731.0 | 861773.0 | 16.0 | |
| 2 | 2022-03-25 | AP | NaN | state | 160314 | 2122 | 732 | False | 845731.0 | 861773.0 | 16.0 | |
| 3 | 2022-03-24 | AP | NaN | state | 160301 | 2122 | 731 | False | 845731.0 | 861773.0 | 16.0 | |
| 4 | 2022-03-23 | AP | NaN | state | 160288 | 2122 | 730 | False | 845731.0 | 861773.0 | 16.0 | |
| ... | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... |
| 2837998 | 2020-06-23 | SP | Óleo | city | 1 | 0 | 5 | False | 2496.0 | 2471.0 | 3533809.0 | |
| 2837999 | 2020-06-22 | SP | Óleo | city | 1 | 0 | 4 | False | 2496.0 | 2471.0 | 3533809.0 | |
| 2838000 | 2020-06-21 | SP | Óleo | city | 1 | 0 | 3 | False | 2496.0 | 2471.0 | 3533809.0 | |
| 2838001 | 2020-06-20 | SP | Óleo | city | 1 | 0 | 2 | False | 2496.0 | 2471.0 | 3533809.0 | |
| 2838002 | 2020-06-19 | SP | Óleo | city | 1 | 0 | 1 | False | 2496.0 | 2471.0 | 3533809.0 | |

2838003 rows x 13 columns

Em seguida, o comando “info()” foi utilizado para se entender os tipos de dados que estávamos lidando

```
In [67]: 1 #Entendendo tipo dos dados
          2 casos_e_mortes_confirmadas.info()

<class 'pandas.core.frame.DataFrame'>
RangeIndex: 2838003 entries, 0 to 2838002
Data columns (total 13 columns):
#   Column                                Dtype
---  ---
0   date                                  object
1   state                                object
2   city                                  object
3   place_type                           object
4   confirmed                             int64
5   deaths                                int64
6   order_for_place                       int64
7   is_last                               bool
8   estimated_population_2019             float64
9   estimated_population                  float64
10  city_ibge_code                        float64
11  confirmed_per_100k_inhabitants         float64
12  death_rate                            float64
dtypes: bool(1), float64(5), int64(3), object(4)
memory usage: 262.5+ MB
```

Nenhuma alteração nos tipos de dados foi necessária. Esse dataset especificamente não continha apenas os dados do mesmo período que os outros, por isso, foi necessário filtrar entre as datas 04/03/2020 e 12/08/2020 que é o exato período dos datasets “obitos.csv” e “casos confirmados.csv” para que a comparação com este dataset fizesse sentido.

```
In [7]: 1 #Filtrando coluna DATE entre 04/03/2020 e 12/08/2020
          2 casos_e_mortes_confirmadas =
          3 casos_e_mortes_confirmadas[(casos_e_mortes_confirmadas['date'] >= '2020-04-03')
          4                                & (casos_e_mortes_confirmadas['date'] <= '2020-08-12')]
          5 .sort_values(by=['date'], ascending= True )
```

Além disso, como o dataset em questão abrange dados de cidades e estados de todo o Brasil, foram filtradas apenas as cidades do estado de Minas Gerais, e logo em seguida já selecionamos apenas as colunas de interesse “city” e “death_rate”.

```
In [69]: 1 #filtrando apenas por cidades
          2 casos_e_mortes_confirmadas = casos_e_mortes_confirmadas[casos_e_mortes_confirmadas['place_type'] == 'city']
          3 #filtrando apenas estado de interesse (MG)
          4 casos_e_mortes_confirmadas = casos_e_mortes_confirmadas[casos_e_mortes_confirmadas['state'] == 'MG']
          5 #filtrando apenas colunas de interesse
          6 casos_e_mortes_confirmadas = casos_e_mortes_confirmadas.loc[:,['city','death_rate']]
```

Neste momento, foi realizado o agrupamento pela coluna “city” calculando-se a média da coluna “death_rate” e como o resultado desse agrupamento possuía várias casas decimais, arredondamos para apenas duas facilitando assim a interpretação das informações.

```
In [70]: 1 #Tabela final com taxa de mortalidade arredondada pelas cidades de MG, no período especificado
2 casos_e_mortes_confirmadas =
3 (casos_e_mortes_confirmadas.groupby(by=['city'], as_index=False).mean())
4 .sort_values('death_rate', ascending = False)
5
6 casos_e_mortes_confirmadas['death_rate'] = (casos_e_mortes_confirmadas['death_rate']*100).round(2)
```

Foi verificada também a quantidade de valores nulos no dataset.

```
In [71]: 1 #verificando existência de valores nulos
2 casos_e_mortes_confirmadas.isnull().sum()

Out[71]: city      0
death_rate  0
dtype: int64
```

Como não haviam valores nulos, assim como nos outros datasets foram normalizados os nomes das cidades e verificando a quantidade de linhas restantes.

```
In [557]: 1 #Normalizando a coluna city, deixando maiuscula e sem acentuação
2
3 casos_e_mortes_confirmadas['city'] = casos_e_mortes_confirmadas['city'].str.normalize('NFKD')\
4 .str.encode('ascii', errors='ignore')\
5 .str.decode('utf-8').astype(str).str.upper()
```


Logo em seguida, unimos o dataframe “df” resultante com o dataset “caso.csv”.

```
In [565]: 1 #join dos casos de morte
          2
          3 df = df.merge(casos_e_mortes_confirmadas,
          4                  how = 'left',
          5                  left_on='MUNICIPIO_RESIDENCIA',
          6                  right_on='city',
          7                  suffixes=('_df', '_mortes'))
          8 df
```

Out[565]:

| | MACRO | DATA_NOTIFICACAO | CLASSIFICACAO_CASO | SEXO | IDADE | MUNICIPIO_RESIDENCIA |
|-------|----------|---------------------|--------------------|-----------|-------|----------------------|
| 0 | SUDESTE | 2020-05-08T00:00:00 | CASO CONFIRMADO | MASCULINO | 33.0 | JUIZ DE FORA |
| 1 | LESTE | 2020-06-19T00:00:00 | CASO CONFIRMADO | FEMININO | 25.0 | GOVERNADOR VALADARES |
| 2 | NORDESTE | 2020-06-14T00:00:00 | CASO CONFIRMADO | MASCULINO | 51.0 | TEOFILO OTONI |
| 3 | LESTE | 2020-06-05T00:00:00 | CASO CONFIRMADO | MASCULINO | 41.0 | GOVERNADOR VALADARES |
| 4 | CENTRO | 2020-05-15T00:00:00 | CASO CONFIRMADO | FEMININO | 27.0 | BELO HORIZONTE |
| ... | ... | ... | ... | ... | ... | ... |
| 34696 | LESTE | 2020-08-07T00:00:00 | CASO CONFIRMADO | MASCULINO | 25.0 | GOVERNADOR VALADARES |
| 34697 | SUDESTE | 2020-07-24T00:00:00 | CASO CONFIRMADO | MASCULINO | 20.0 | GUIRICEMA |
| 34698 | SUL | 2020-08-03T00:00:00 | CASO CONFIRMADO | FEMININO | 39.0 | MUZAMBINHO |
| 34699 | OESTE | 2020-08-06T00:00:00 | CASO CONFIRMADO | MASCULINO | 27.0 | BOM DESPACHO |
| 34700 | OESTE | 2020-08-08T00:00:00 | CASO CONFIRMADO | MASCULINO | 50.0 | CLAUDIO |

34701 rows × 14 columns

Como resultado final do nosso tratamento e processamento de dados, obtivemos uma tabela de 34.701 linhas e 14 colunas, sendo elas:

```
In [119]: 1 df.info()

<class 'pandas.core.frame.DataFrame'>
Int64Index: 34701 entries, 0 to 34700
Data columns (total 12 columns):
#   Column                                Non-Null Count  Dtype
---  -
0   MACRO                                34701 non-null  object
1   DATA_NOTIFICACAO                    34701 non-null  object
2   CLASSIFICACAO_CASO                  34701 non-null  object
3   SEXO                                 34701 non-null  object
4   IDADE                                34701 non-null  object
5   MUNICIPIO_RESIDENCIA                 34701 non-null  object
6   COMORBIDADE                          34701 non-null  object
7   EVOLUCAO                             34701 non-null  object
8   INTERNACAO                           34701 non-null  object
9   UTI                                  34701 non-null  object
10  RACA                                  34701 non-null  object
11  MEDIA_IDADE_POR_MUNICIPIO            34701 non-null  float64
dtypes: float64(1), object(11)
memory usage: 3.4+ MB
```

4. Análise e Exploração dos Dados

Nessa seção você deve mostrar como foi realizada a análise e exploração dos dados do seu trabalho. Mostre as hipóteses levantadas durante essa etapa e os padrões e *insights* identificados.

4.1 Óbitos por covid

Iremos analisar o primeiro dataset importado “obitos.csv” conforme importamos originalmente e para nos ajudar precisamos importar a biblioteca “Counter” de “Collections”.

```
1 from collections import Counter
```

O primeiro passo então, foi utilizar o comando `.describe()` para obter informações gerais sobre nosso dataset, e arredondamos os valores resultantes para apenas 2 casas decimais.

```
1 obitos.describe().round(2)
```

| | _id | PACIENTE | IDADE |
|--------------|------------|-----------------|--------------|
| count | 2894.00 | 2894.00 | 2894.00 |
| mean | 1447.50 | 1447.50 | 69.86 |
| std | 835.57 | 835.57 | 15.12 |
| min | 1.00 | 1.00 | 0.00 |
| 25% | 724.25 | 724.25 | 61.00 |
| 50% | 1447.50 | 1447.50 | 72.00 |
| 75% | 2170.75 | 2170.75 | 81.00 |
| max | 2894.00 | 2894.00 | 107.00 |

As colunas “_id” e “PACIENTE” podem ser desconsideradas nessa parte da análise pois não nos trazem dados significativos. Nos restando então a a coluna

“IDADE” com 2.894 linhas preenchidas, média de aproximadamente 70 anos, mínima de zero e máxima de 107 anos. Comparando a média com a mediana, elas possuem valores próximos, mas que não chegam a ser iguais, isso pode significar que possuímos alguns valores *outliers*.

Agora, vamos entender o período de data que o dataset abrange, e para isso vamos precisar antes transformar as colunas de data para facilitar a filtragem.

```
1 #Convetendo coluna DATA_OBITO para datetime
2 obitos["DATA_OBITO"] = pd.to_datetime(obitos["DATA_OBITO"], format="%Y/%m/%d")
3 obitos['DATA_OBITO'].info()
4 obitos.head()
```

```
<class 'pandas.core.series.Series'>
RangeIndex: 2894 entries, 0 to 2893
Series name: DATA_OBITO
Non-Null Count  Dtype
-----
2823 non-null   datetime64[ns]
dtypes: datetime64[ns](1)
memory usage: 22.7 KB
```

| | _id | PACIENTE | SEXO | IDADE | MUNICIPIO_RESIDENCIA | DATA_OBITO | COMORBIDADE |
|---|-----|----------|------|-------|----------------------|------------|-------------|
| 0 | 1 | 1 | M | 79 | Patos de Minas | 2020-03-28 | SIM |
| 1 | 2 | 2 | F | 82 | Belo Horizonte | 2020-03-29 | SIM |
| 2 | 3 | 3 | M | 66 | Belo Horizonte | 2020-03-30 | SIM |
| 3 | 4 | 4 | M | 44 | Mariana | 2020-03-30 | NÃO |
| 4 | 5 | 5 | M | 80 | Uberlândia | 2020-03-30 | SIM |

Agora que temos a coluna de data com o tipo datetime64, vamos pegar a data máxima e mínima.

```
1 #Maior e menor data da coluna DATA_OBITO
2 print(obitos['DATA_OBITO'].max())|
3 print(obitos['DATA_OBITO'].min())
```

```
2020-12-07 00:00:00
2020-01-04 00:00:00
```

O período de datas abrange exatamente o período analisado durante todo este trabalho.

Agora, explorando mais cada coluna individualmente, vamos entender de quantos municípios do estado de Minas Gerais estamos lidando.

```
1 #Quantidade de Municipios do estado de MG
2 obitos.groupby('MUNICIPIO_RESIDENCIA').MUNICIPIO_RESIDENCIA.nunique().sum()
```

405

O dataset nos mostra um total de 405 municípios com registros de óbitos no estado de Minas Gerais.

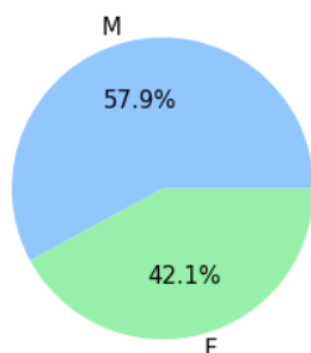
Agora, vamos analisar os gêneros que compõem o arquivo e lotar em um gráfico de pizza para melhor visualização.

```
1 #Quantidade de pessoas em cada genero
2
3 from collections import Counter
4 genero_obitos = Counter(obitos['SEXO'])
5 genero_obitos
```

Counter({'M': 1676, 'F': 1218})

```
1 #Quantidade de pessoas em cada genero
2
3 from matplotlib import pyplot as plt
4 import numpy as np
5
6 plt.style.use('seaborn-pastel')
7 plt.pie(genero_obitos.values(), labels = genero_obitos.keys(),
8 autopct = '%1.1f%%', textprops= {'fontsize':15})
9 plt.title('Gênero dos pacientes que foram a óbito', fontsize=18, pad = 40)
10 plt.axis('image')
11 plt.show()
```

Gênero dos pacientes que foram a óbito



Analisando o gráfico percebe-se que a 60% dos pacientes registrados que foram a óbito eram do sexo feminino, enquanto apenas 42% eram pertencentes ao sexo masculino.

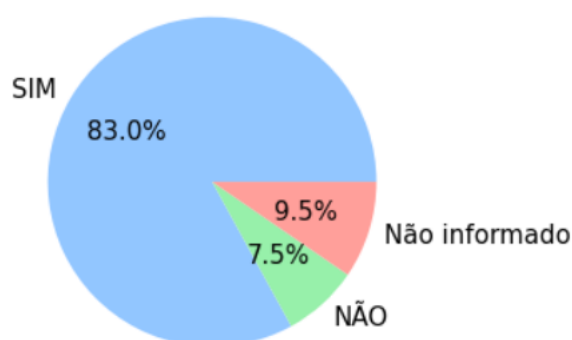
Veremos agora o total de pessoas que apresentavam comorbidades.

```
1 #Quantidade de pessoas que apresentavam comorbidades
2
3 from collections import Counter
4 genero_obitos = Counter(obitos['COMORBIDADE'])
5 genero_obitos
```

```
Counter({'SIM': 2402, 'NÃO': 216, 'Não informado': 276})
```

```
1 #Quantidade de pessoas que apresentavam comorbidades
2
3 from matplotlib import pyplot as plt
4 import numpy as np
5
6 plt.style.use('seaborn-pastel')
7 plt.pie(genero_obitos.values(), labels = genero_obitos.keys(),
8 autopct = '%1.1f%', textprops= {'fontsize':15})
9 plt.title('Quantidade de pessoas que apresentavam comorbidades', fontsize=18, pad = 40)
10 plt.axis('image')
11 plt.show()
```

Quantidade de pessoas que apresentavam comorbidades



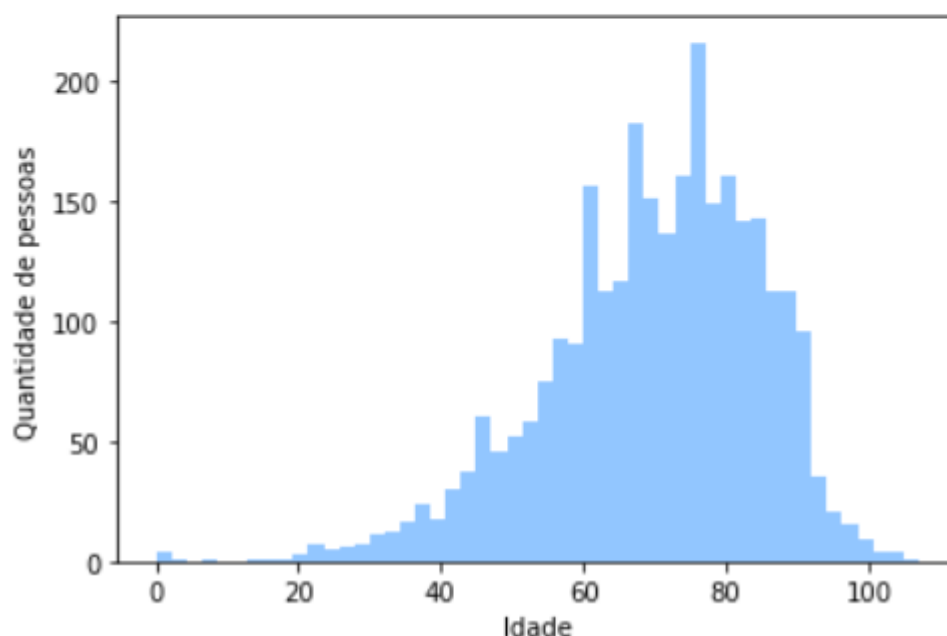
Indicando um possível padrão, vemos que 83% das pessoas que chegaram a óbito apresentavam algum tipo de comorbidade, em contrapartida, apenas 7,5% não tinham, e 9,5% não foi informado

Indo agora para a coluna “IDADE” vamos plotar um histograma para que possamos entender a distribuição das idades.

```

: 1 #Histograma de idade a óbito
  2
  3 from matplotlib import pyplot as plt
  4 x = obitos.IDADE
  5 # plt.style.use('ggplot')
  6 plt.xlabel('Idade')
  7 plt.ylabel('Quantidade de pessoas')
  8 plt.hist(x, bins= 50)
  9
10 plt.show()

```



Analisando o histograma vemos que as idades se concentram mais a partir dos 40 anos até os 85 aproximadamente, porém temos casos de óbitos de pacientes com menos de 1 ano e mais de 100 também.

Além disso, vemos três picos de idade, sendo elas 74 anos com 95 óbitos, 68 anos com 93 óbitos e 67 anos com 89 óbitos, conforme a imagem abaixo.

```

1 obitos.IDADE.value_counts()

74      95
68      93
67      89
70      82
84      82
..
2         1
17         1
107        1
24         1
14         1
Name: IDADE, Length: 92, dtype: int64

```

Para finalizar a análise deste dataset, vamos tentar achar as correlações entre as colunas, e para isso vamos transformar as colunas “RACA” e “COMORBIDADE” para dados numéricos.

```

1 #Convertendo categorias em 0 e 1 para análise de correlação
2 obitos = pd.read_csv('obitos.csv', index_col=False)
3 obitos["COMORBIDADE"] = np.where(obitos["COMORBIDADE"] == "SIM", 1, 0)
4 obitos["SEXO"] = np.where(obitos["SEXO"] == "M", 1, 0)
5 obitos.info()

<class 'pandas.core.frame.DataFrame'>
RangeIndex: 2894 entries, 0 to 2893
Data columns (total 7 columns):
#   Column                                Non-Null Count  Dtype
---  -
0   _id                                    2894 non-null   int64
1   PACIENTE                             2894 non-null   int64
2   SEXO                                  2894 non-null   int32
3   IDADE                                 2894 non-null   int64
4   MUNICIPIO_RESIDENCIA                 2894 non-null   object
5   DATA_OBITO                          2823 non-null   object
6   COMORBIDADE                          2894 non-null   int32
dtypes: int32(2), int64(3), object(2)
memory usage: 135.8+ KB

```

```

1 #Plotando correlação entre
2 import pandas as pd
3 import numpy as np
4
5 obitos = obitos.drop(labels= ['_id','PACIENTE'], axis = 1)
6
7 rs = np.random.RandomState(0)
8 corr = obitos.corr()
9 corr.style.background_gradient(cmap='coolwarm').format(precision=2)
10

```

| | SEXO | IDADE | COMORBIDADE |
|-------------|-------|-------|-------------|
| SEXO | 1.00 | -0.09 | -0.06 |
| IDADE | -0.09 | 1.00 | 0.14 |
| COMORBIDADE | -0.06 | 0.14 | 1.00 |

Analisando a tabela de correlações, vemos que as únicas colunas que se correlacionam positivamente, ou seja, uma impacta diretamente na outra são “IDADE” e “COMORBIDADE” o que faz sentido.

4.2 Casos confirmados Covid

Vamos agora explorar o que temos no dataset “casos confirmados.csv”. Assim como no último dataset, no primeiro passo, utilizamos o comando `.describe()` para entender os valores numéricos de nossa base de dados.

```
1 casos_confirmados.describe().round(2)
```

| | _id | ID | IDADE | CODIGO |
|-------|----------|----------|----------|-----------|
| count | 46446.00 | 46446.00 | 45581.00 | 45238.00 |
| mean | 23223.50 | 23223.50 | 43.58 | 313516.00 |
| std | 13407.95 | 13407.95 | 18.48 | 2294.36 |
| min | 1.00 | 1.00 | 0.00 | 310010.00 |
| 25% | 11612.25 | 11612.25 | 31.00 | 311330.00 |
| 50% | 23223.50 | 23223.50 | 41.00 | 313190.00 |
| 75% | 34834.75 | 34834.75 | 55.00 | 315460.00 |
| max | 46446.00 | 46446.00 | 220.00 | 317220.00 |

Podemos desconsiderar as colunas “_id”, “ID” e “CODIGO” pois elas não agregam nada nas conclusões deste trabalho.

Restando assim a coluna “IDADE” onde temos um mínimo de 0 e máximo de 220 anos, deve ter havido algum engano no preenchimento de dados pois sabe-se que não é possível viver até esta idade. Além disso, vemos que a idade média é de 43 anos enquanto a mediana nos indica 41 anos de idade.

Continuando a análise, vamos agora converter a coluna de data para que possamos entender qual período ela abrange.

```

1 #Convetendo coluna DATA_NOTIFICACAO para datetime
2 casos_confirmados["DATA_NOTIFICACAO"] = pd.to_datetime(\
3 casos_confirmados["DATA_NOTIFICACAO"], format="%Y/%m/%d")
4
5 casos_confirmados['DATA_NOTIFICACAO'].info()

```

```

<class 'pandas.core.series.Series'>
RangeIndex: 46446 entries, 0 to 46445
Series name: DATA_NOTIFICACAO
Non-Null Count  Dtype
-----
39302 non-null  datetime64[ns]
dtypes: datetime64[ns](1)
memory usage: 363.0 KB

```

```

1 #Maior e menor data da coluna DATA_NOTIFICACAO
2 print(casos_confirmados['DATA_NOTIFICACAO'].max())
3 print(casos_confirmados['DATA_NOTIFICACAO'].min())

```

```

2020-08-12 00:00:00
2020-03-04 00:00:00

```

Assim como no dataset “obitos.csv” estamos trabalhando com o período entre 04/03/2020 e 12/08/2020.

Iremos agora analisar a quantidade de municípios que registraram casos de COVID-19 no período especificado acima.

```

1 #Quantidade de Municipios únicos
2 ## 400 a mais que o obitos por covid, o que faz sentido
3 casos_confirmados.groupby('MUNICIPIO_RESIDENCIA')['MUNICIPIO_RESIDENCIA'].nunique().sum()

```

868

Chegamos ao resultado de 868 municípios, mais que o dobro da quantidade de municípios registrados na tabela “obitos.csv” o que faz sentido, pois nem toda infecção pelo vírus resulta em óbito.

Vamos utilizar a função importada Counter() e com a ajuda de um gráfico de pizza avaliar a distribuição por gênero no dataset de casos confirmados.

```

1 #Quantidade de pessoas em cada genero
2
3 from collections import Counter
4 genero_obitos = Counter(casos_confirmados['SEXO'])
5 genero_obitos

```

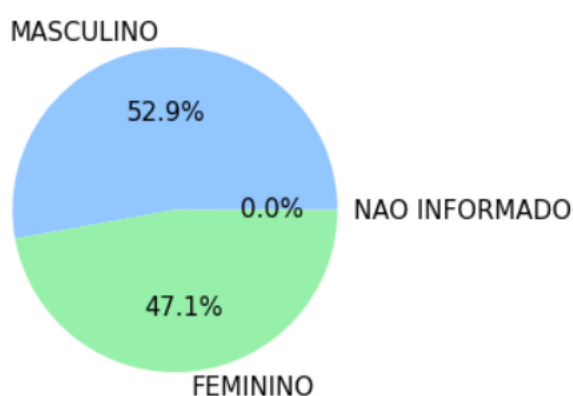
```
Counter({'MASCULINO': 24547, 'FEMININO': 21889, 'NAO INFORMADO': 10})
```

```

1 #Quantidade de pessoas em cada genero
2
3 from matplotlib import pyplot as plt
4 import numpy as np
5
6 plt.style.use('seaborn-pastel')
7 plt.pie(genero_obitos.values(), labels = genero_obitos.keys(),
8 autopct = '%1.1f%%', textprops = {'fontsize':15})
9 plt.title('Gênero dos pacientes com casos Confirmados Covid-19', fontsize=18, pad = 40)
10 plt.axis('image')
11 plt.show()

```

Gênero dos pacientes com casos Confirmados Covid-19



Com 53% dos casos de infecção em pessoas do sexo masculino, e 47% do sexo feminino podemos comparar com os dados de óbitos e vemos que apesar de menos mulheres serem infectadas a mortalidade do vírus, aparentemente, é maior para elas.

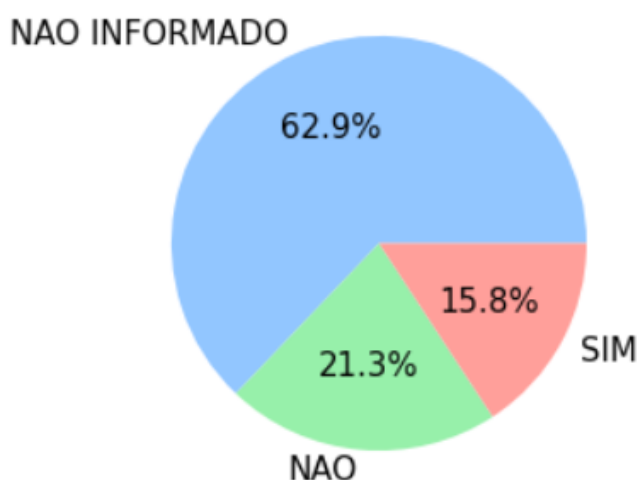
Olhando agora para os dados de comorbidade, utilizamos também do gráfico de pizza para uma melhor análise.

```

1  #Quantidade de pessoas que apresentavam comorbidades
2
3  from collections import Counter
4  comorbidade = Counter(casos_confirmados['COMORBIDADE'])
5  comorbidade
6
7  from matplotlib import pyplot as plt
8  import numpy as np
9
10 plt.style.use('seaborn-pastel')
11 plt.pie(comorbidade.values(), labels = comorbidade.keys(),
12 autopct = '%1.1f%%', textprops= {'fontsize':15})
13 plt.title('Pacientes possuíam comorbidade?', fontsize=18, pad = 40)
14 plt.axis('image')
15 plt.show()

```

Pacientes possuíam comorbidade?



Infelizmente neste dataset, 63% dos registros não informavam, enquanto 21,3% não possuíam e 15,8% apresentavam algum tipo de comorbidade. Esta

coluna será retirada mais tarde para que não impacte nos algoritmos de Machine Learning.

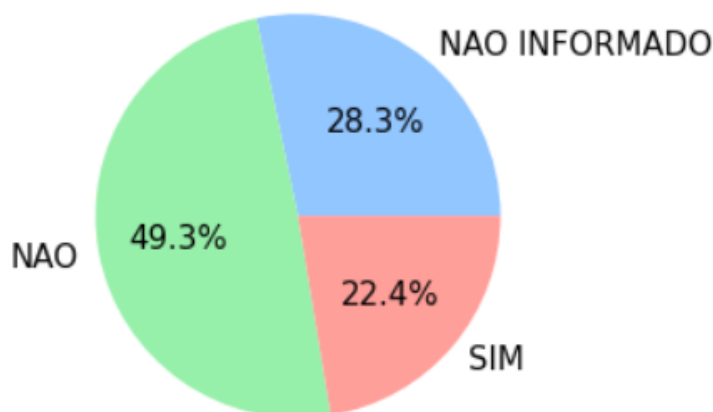
Continuando nossa análise, vamos entender o que está acontecendo com a coluna “INTERNACAO”, ou seja, quantos dos pacientes registrados foram internados ou não.

```

1  #Interação SIM/NÃO
2
3  from collections import Counter
4  internacao = Counter(casos_confirmados['INTERNACAO'])
5  internacao
6
7  from matplotlib import pyplot as plt
8  import numpy as np
9
10 plt.style.use('seaborn-pastel')
11 plt.pie(internacao.values(), labels = internacao.keys(),
12 autopct = '%1.1f%%', textprops= {'fontsize':15})
13 plt.title('Interação dos casos registrados', fontsize=18, pad = 40)
14 plt.axis('image')
15 plt.show()

```

Interação dos casos registrados



Vemos que 49% não precisou de internação enquanto 22% precisou e 28% infelizmente não foi informado.

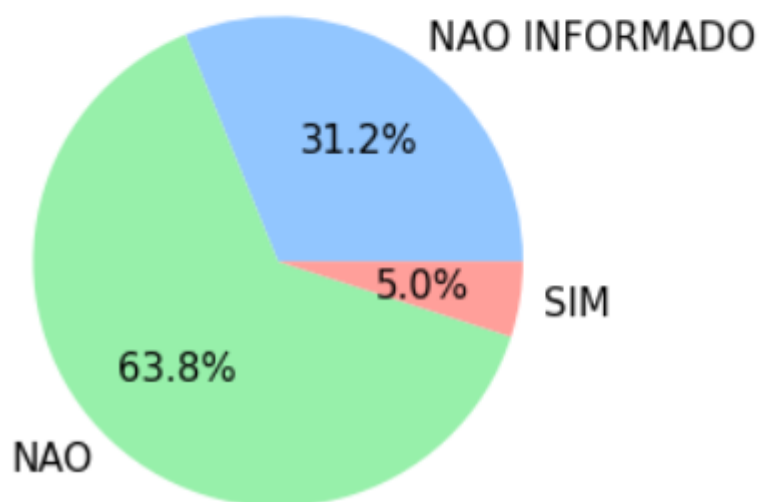
Chegamos então à nossa coluna alvo “UTI”. Essa coluna nos diz se foi preciso encaminhar o paciente para UTI ou não.

```

1 #Foi para UTI?
2
3 from collections import Counter
4 uti = Counter(casos_confirmados['UTI'])
5
6 from matplotlib import pyplot as plt
7 import numpy as np
8
9 plt.style.use('seaborn-pastel')
10 plt.pie(uti.values(), labels = uti.keys(),
11 autopct = '%1.1f%%', textprops= {'fontsize':15})
12 plt.title('Foi para UTI?', fontsize=18, pad = 40)
13 plt.axis('image')
14 plt.show()

```

Foi para UTI?



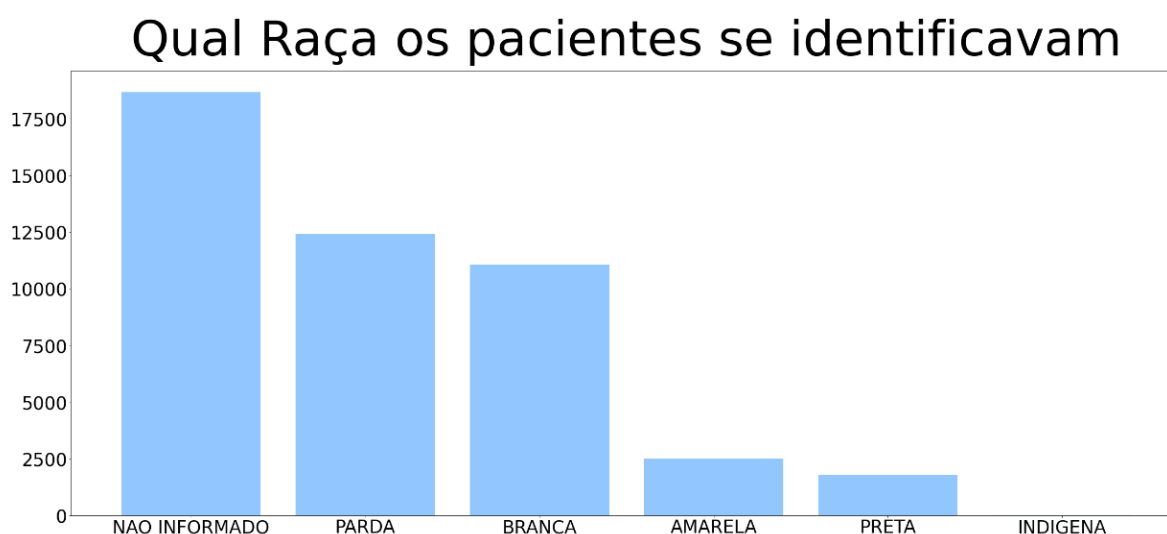
Conforme vemos no gráfico acima, 63,8% dos pacientes não precisaram ir para a UTI, enquanto 5% precisaram e 31% não foram informados pelo dataset.

Vamos analisar a coluna "RACA", onde os pacientes se identificavam entre as raças parda, branca, preta, amarela e indígena, e dessa vez com o auxílio de um gráfico de colunas.

```

1  #Qual a Raça os pacientes se identificavam
2
3
4  import json
5  #Convertendo em Json
6  json = json.loads(casos_confirmados['RACA'].value_counts().to_json())
7
8  a = [i for i in json] #Pegando Keys em lista
9  b = [json[i] for i in json] #Pegando Values em lista
10
11 import numpy as np
12 import pandas as pd
13 import matplotlib.pyplot as plt
14 cmap = plt.cm.tab10
15 import matplotlib.pyplot as plt
16 fig = plt.figure()
17 ax = fig.add_axes([0,0,8,5])
18 langs = a
19 students = b
20 ax.bar(langs,students)
21 plt.xticks(fontsize=35)
22 plt.yticks(fontsize=35)
23 plt.title('Qual a Raça os pacientes se identificavam', fontsize=65, pad = 30)
24 plt.show()

```



A maioria dos dados não continham tal informação, mas vemos a predominância através do gráfico das raças parda e branca, sendo preta e indígena minorias nos casos de contaminação confirmada do COVID-19.

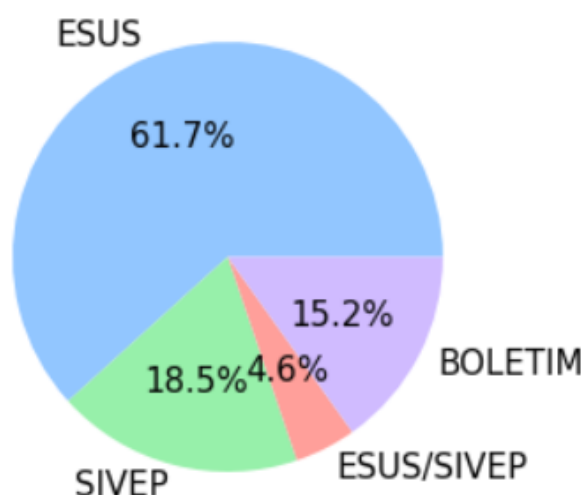
Por fim e não menos importante, vamos analisar a fonte dados correspondente a cada linha do arquivo através da coluna “ORIGEM_DA_INFORMACAO”.

```

1 casos_confirmados['ORIGEM_DA_INFORMACAO'].value_counts()
2
3 #Origem da informação
4
5 from collections import Counter
6 origem_info = Counter(casos_confirmados['ORIGEM_DA_INFORMACAO'])
7 origem_info
8
9 from matplotlib import pyplot as plt
10 import numpy as np
11
12 plt.style.use('seaborn-pastel')
13 plt.pie(origem_info.values(), labels = origem_info.keys(),
14 autopct = '%1.1f%%', textprops= {'fontsize':15})
15 plt.title('Origem da informação', fontsize=18, pad = 40)
16 plt.axis('image')
17 plt.show()

```

Origem da informação



4.3 Mortes por Covid-19

O último dataset a ser analisado é o “caso.csv” que possui uma coluna complementar muito importante para nosso estudo, “death_rate” que indica a taxa de mortalidade por município dos estados de Minas Gerais.

Assim como nos outros dois datasets, o primeiro passo é utilizar o comando `.describe()` para entender como as colunas numéricas se comportam.

```
1 casos_e_mortes_confirmadas.describe().round(2)
```

| death_rate | |
|------------|--------|
| count | 854.00 |
| mean | 3.53 |
| std | 9.21 |
| min | 0.00 |
| 25% | 0.00 |
| 50% | 0.14 |
| 75% | 2.14 |
| max | 76.12 |

Analisando o resultado, vemos que houveram municípios no período analisado que não houveram mortes registradas, enquanto a média geral da taxa de mortes é de 3,53%, bem baixa em comparação aos índices que atingiram futuramente. A taxa máxima de mortes foi no município de Senador Modestino Gonçalves com 76,12%.

É interessante notar pelos quartis apresentados que até 75% dos municípios registrados neste dataset apresentavam 2,14% apenas de taxa de mortalidade, enquanto apenas alguns deles neste período realmente foram muito impactados pela pandemia no primeiro momento.

5. Criação de Modelos de Machine Learning

O objetivo deste estudo é identificar a necessidade de internação de um paciente com Covid-19 em uma UTI e ajudar os hospitais a controlarem recursos e quantidade de leitos necessários para dar o melhor suporte para a população.

Neste tópico, iremos percorrer pelos modelos de Machine Learning escolhidos para atingir o resultado proposto acima.

Antes de aplicar os modelos, é preciso primeiramente realizar alguns tratamentos de dados.

No *dataframe* final, obtido pela união dos datasets iniciais descritos nos tópicos anteriores, o primeiro passo foi o tratamento da coluna alvo “UTI” que dentre os valores contidos nesta, possuía um valor absurdamente maior de “NÃO” em relação a “SIM”.

```
1 #verificando a maior quantidade de itens na coluna
2 from sklearn.utils import resample
3 df_maior = df[df['UTI'] == 'NAO']
4 print('linhas:' , df_maior.shape[0])
```

linhas: 22771

```
1 #verificando a menor quantidade de itens na coluna
2 df_menor = df[df['UTI'] == 'SIM']
3 print('linhas:' ,df_menor.shape[0])
```

linhas: 881

Para corrigir isso, utilizamos a biblioteca *resample* da *sklearn.utils* que possui uma função que consegue equalizar os valores da coluna alvo para que nossos algoritmos não acabem sendo enviesados por tamanha diferença de amostras.

```

1 #Igualando a quantidade de entradas de dos dfs
2 df_unsampled = resample(df_menor, replace = True, n_samples=19372, random_state=123)
3 df_unsampled.info()

```

```

<class 'pandas.core.frame.DataFrame'>
Int64Index: 19372 entries, 17110 to 610
Data columns (total 14 columns):
#   Column                                Non-Null Count  Dtype
---  -
0   MACRO                                19372 non-null  object
1   DATA_NOTIFICACAO                    19372 non-null  object
2   CLASSIFICACAO_CASO                   19372 non-null  object
3   SEXO                                 19372 non-null  object
4   IDADE                                 19372 non-null  object
5   MUNICIPIO_RESIDENCIA                  19372 non-null  object
6   COMORBIDADE                           19372 non-null  object
7   EVOLUCAO                              19372 non-null  object
8   INTERNACAO                            19372 non-null  object
9   UTI                                   19372 non-null  object
10  RACA                                   19372 non-null  object
11  MEDIA_IDADE_POR_MUNICIPIO             19372 non-null  float64
12  city                                   19332 non-null  object
13  death_rate                            19332 non-null  float64
dtypes: float64(2), object(12)
memory usage: 2.2+ MB

```

Após o tratamento, fizemos a concatenação do resultado ajustado:

```

1 #Concatenando resultados
2 df = pd.concat([df_unsampled, df_maior])
3 df.info()

```

```

<class 'pandas.core.frame.DataFrame'>
Int64Index: 42143 entries, 17110 to 34700
Data columns (total 14 columns):
#   Column                                Non-Null Count  Dtype
---  -
0   MACRO                                42143 non-null  object
1   DATA_NOTIFICACAO                    42143 non-null  object
2   CLASSIFICACAO_CASO                   42143 non-null  object
3   SEXO                                 42143 non-null  object
4   IDADE                                 42143 non-null  object
5   MUNICIPIO_RESIDENCIA                  42143 non-null  object
6   COMORBIDADE                           42143 non-null  object
7   EVOLUCAO                              42143 non-null  object
8   INTERNACAO                            42143 non-null  object
9   UTI                                   42143 non-null  object
10  RACA                                   42143 non-null  object
11  MEDIA_IDADE_POR_MUNICIPIO             42143 non-null  float64
12  city                                   42103 non-null  object
13  death_rate                            42103 non-null  float64
dtypes: float64(2), object(12)
memory usage: 10.0+ MB

```

Continuando nossos tratamentos, avaliamos quais colunas não faziam sentido permanecer no *dataframe* e quais dados inválidos ainda existiam.

Foram retiradas algumas colunas conforme o código a seguir, pois essas não contribuem em nada com o atingimento do objetivo:

```

1 #Retirando colunas desnecessárias
2 df = df.drop(['MACRO', 'DATA_NOTIFICACAO', 'CLASSIFICACAO_CASO', 'city'], axis = 1)
3

```

Depois, foram verificados os conteúdos das colunas restantes através do código:

```

1 #Verificando dados das colunas
2 for column in df:
3     print(df[column].value_counts())
4     print('-----')
5

```

```

MASCULINO      23575
FEMININO       18567
NAO INFORMADO      1
Name: SEXO, dtype: int64

```

```

39.0      955
38.0      948
50.0      935
42.0      923
36.0      916

```

...

```

97.0        3
101.0       2
103.0       1
100.0       1
108.0       1

```

```

Name: IDADE, Length: 103, dtype: int64

```

```

BELO HORIZONTE      4399
UBERLANDIA          2786
GOVERNADOR VALADARES 2753
JUIZ DE FORA        2271
IPATINGA            1450

```

...

```

CORACAO DE JESUS      1
JORDANIA              1
CARVALHOPOLIS         1
SAO SEBASTIAO DO RIO PRETO 1
UMBURATIBA            1

```

```

Name: MUNICIPIO_RESIDENCIA, Length: 640, dtype: int64

```

Percebe-se que nas colunas 'COMORBIDADE' e 'RACA' temos um grande valor de linhas classificados como "NAO INFORMADO" que se equivale a *null*... como eles representam mais de 33% do total de linhas, vamos retirar essas colunas do *dataframe*:

```
NAO INFORMADO    23863
SIM              10675
NAO              7605
Name: COMORBIDADE, dtype: int64
```

```
NAO INFORMADO    15075
BRANCA           11881
PARDA            10829
AMARELA          2385
PRETA            1962
INDIGENA          11
Name: RACA, dtype: int64
```

```
1 #Retirando colunas que possuem muitos dados inválidos
2 df = df.drop(['COMORBIDADE', 'RACA'], axis = 1)
```

Depois, foram tratadas as colunas que continham ainda algumas linhas que podem ser desprezadas:

```
1 #retirando dados inválidos
2 df = df[df['INTERNACAO'] != 'NAO INFORMADO']
3 df['INTERNACAO'].value_counts()
```

```
SIM    21401
NAO    20721
Name: INTERNACAO, dtype: int64
```

```

1 #retirando dados inválidos
2 df = df[df['SEXO'] != 'NAO INFORMADO']
3 df['SEXO'].value_counts()

```

```

MASCULINO    23562
FEMININO     18559
Name: SEXO, dtype: int64

```

```

1 #removendo valores nulos
2 df = df[df['death_rate'].notnull()]
3 df.isna().sum()

```

```

SEXO          0
IDADE         0
MUNICIPIO_RESIDENCIA  0
EVOLUCAO      0
INTERNACAO    0
UTI           0
MEDIA_IDADE_POR_MUNICIPIO  0
death_rate    0
dtype: int64

```

Continuando os tratamentos, no *dataframe* final, existem ainda muitos dados categóricos, e estes não são bem recebidos por algoritmos de Machine Learning pois estes trabalham com cálculos matemáticos, logo existem algumas técnicas como Label Encoding, One Hot Encoding e Dummy Encoding para contornar essa situação e transformar dados categóricos em numéricos, sem prejudicar a lógica de segmentação fornecida pelo próprio dado categórico.

Primeiro, tratamos as colunas que haviam poucas categorias manualmente, substituindo por valores binários.

```

1 #Convertendo categorias em 0 e 1
2 df["SEXO"] = np.where(df["SEXO"] == "FEMININO", 1, 0)
3 df["INTERNACAO"] = np.where(df["INTERNACAO"] == "SIM", 1, 0)
4 df["UTI"] = np.where(df["UTI"] == "SIM", 1, 0)
5 df

```

| | SEXO | IDADE | MUNICIPIO_RESIDENCIA | EVOLUCAO | INTERNACAO | UTI | MEDIA_IDADE_POR_MUNICIPIO | death_rate |
|-------|------|-------|----------------------|-------------------|------------|-----|---------------------------|------------|
| 17110 | 0 | 55.0 | PATOS DE MINAS | EM ACOMPANHAMENTO | 1 | 1 | 69.0 | 3.45 |
| 10790 | 1 | 42.0 | SAO JOAO DEL REI | RECUPERADO | 1 | 1 | 73.0 | 0.99 |
| 11329 | 1 | 53.0 | BELO HORIZONTE | EM ACOMPANHAMENTO | 1 | 1 | 72.0 | 2.38 |
| 8434 | 1 | 65.0 | EXTREMA | EM ACOMPANHAMENTO | 1 | 1 | 63.0 | 2.39 |
| 1327 | 1 | 73.0 | UBERLANDIA | RECUPERADO | 1 | 1 | 70.0 | 3.11 |
| ... | ... | ... | ... | ... | ... | ... | ... | ... |
| 34696 | 0 | 25.0 | GOVERNADOR VALADARES | RECUPERADO | 0 | 0 | 69.0 | 6.71 |
| 34697 | 0 | 20.0 | GUIRICEMA | EM ACOMPANHAMENTO | 0 | 0 | 0.0 | 0.16 |
| 34698 | 1 | 39.0 | MUZAMBINHO | RECUPERADO | 0 | 0 | 85.0 | 0.32 |
| 34699 | 0 | 27.0 | BOM DESPACHO | RECUPERADO | 0 | 0 | 70.0 | 2.99 |
| 34700 | 0 | 50.0 | CLAUDIO | EM ACOMPANHAMENTO | 0 | 0 | 73.0 | 0.56 |

Depois, utilizando a função “pd.get_dummies()” do próprio Pandas normalizamos as colunas “MUNICIPIO_RESIDENCIA” e “EVOLUCAO” para dados numéricos.

```

1 #Convertendo em números as categorias da coluna 'MUNICIPIO_RESIDENCIA' e 'EVOLUCAO'
2 df = pd.get_dummies(df, columns=['MUNICIPIO_RESIDENCIA', 'EVOLUCAO'])

```

Transformando assim, os dados categóricos de linhas para colunas com linhas binárias, consequentemente expandindo muito a quantidade total de colunas do *dataframe*.

Imagem do dataframe após o uso da função pd.get_dummies().

| | SEXO | IDADE | INTERNACAO | UTI | MEDIA_IDADE_POR_MUNICIPIO | death_rate | MUNICIPIO_RESIDENCIA_ABADIA DOS DOURADOS | MUNICIPIO_RESIDENCIA_ABAETE |
|-------|------|-------|------------|-----|---------------------------|------------|--|-----------------------------|
| 17110 | 0 | 55.0 | 1 | 1 | 69.0 | 3.45 | 0 | 0 |
| 10790 | 1 | 42.0 | 1 | 1 | 73.0 | 0.99 | 0 | 0 |
| 11329 | 1 | 53.0 | 1 | 1 | 72.0 | 2.38 | 0 | 0 |
| 8434 | 1 | 65.0 | 1 | 1 | 63.0 | 2.39 | 0 | 0 |
| 1327 | 1 | 73.0 | 1 | 1 | 70.0 | 3.11 | 0 | 0 |
| ... | ... | ... | ... | ... | ... | ... | ... | ... |
| 34696 | 0 | 25.0 | 0 | 0 | 69.0 | 6.71 | 0 | 0 |
| 34697 | 0 | 20.0 | 0 | 0 | 0.0 | 0.16 | 0 | 0 |
| 34698 | 1 | 39.0 | 0 | 0 | 85.0 | 0.32 | 0 | 0 |
| 34699 | 0 | 27.0 | 0 | 0 | 70.0 | 2.99 | 0 | 0 |

Com o dataframe pronto para ser trabalhado com os algoritmos, vamos agora começar a aplicá-los, finalmente. Para tal, o primeiro passo é definir quais as variáveis preditoras e qual a variável alvo. No caso, a variável alvo é a coluna "UTI", enquanto as outras seriam as variáveis utilizadas para prever o alvo. Para ajudar, foi utilizada a biblioteca "sklearn.model_selection".

```
1 from sklearn.model_selection import train_test_split
2 X_train = df.drop('UTI', axis = 1)
3 Y_train = df['UTI']
```

O

Depois, foram criados os conjuntos de dados de teste e treino, sendo que o tamanho da base de teste é 30%, restando assim 70% para treino do algoritmo.

```
1 #Criando conjuntos de dados de teste e treino
2 x_treino, x_teste, y_treino, y_teste = train_test_split(X_train, Y_train, test_size = 0.3)
```

Random Forest

O Random Forest é um algoritmo que utiliza de algoritmos mais básicos como regressão linear, árvore de decisão, entre outros, porém sua principal diferença em relação a estes mais simples é que ele combina os resultados de todos eles para se obter apenas um resultado final mais eficiente.

Para utilizá-la, importamos a classe RandomForestClassifier do pacote sklearn.ensemble.

```

1  #Random Forest
2  from sklearn.ensemble import RandomForestClassifier
3  #Criação do modelo
4  random_forest = RandomForestClassifier(random_state= 42)
5  random_forest.fit(x_treino, y_treino)
6
7  #imprimindo resultados
8  resultado = random_forest.score(x_teste, y_teste)
9  print('Acurácia:', resultado)
10
11 #Outras métricas de eficiência do modelo
12 Train_predict = random_forest.predict(x_teste)
13 print(classification_report(y_teste, Train_predict))

```

Acurácia: 0.9780524522621028

| | precision | recall | f1-score | support |
|--------------|-----------|--------|----------|---------|
| 0 | 1.00 | 0.96 | 0.98 | 6768 |
| 1 | 0.95 | 1.00 | 0.98 | 5853 |
| accuracy | | | 0.98 | 12621 |
| macro avg | 0.98 | 0.98 | 0.98 | 12621 |
| weighted avg | 0.98 | 0.98 | 0.98 | 12621 |

Extra Trees Classifier

Muito parecido com o Random Forest, este algoritmo trabalha da mesma forma, porém com duas principais diferenças: 1 - O Random Forest utiliza subamostras dos dados enquanto o Extra Trees utiliza as amostras originais, e 2 - Enquanto o Random Forest escolhe o caminho ótimo para definir os nós, o Extra Trees escolhe aleatoriamente onde defini-los.

Para utilizá-lo, importamos a classe ExtraTreesClassifier do pacote sklearn.ensemble.


```

1  #ExtraTreesClassifier
2  from sklearn.ensemble import ExtraTreesClassifier
3  #Criação do modelo
4  modelo = ExtraTreesClassifier(random_state= 42)
5  modelo.fit(x_treino, y_treino)
6
7  #imprimindo resultados
8  resultado = modelo.score(x_teste, y_teste)
9  print('Acurácia:', resultado)
10
11 #Outras métricas de eficiência do modelo
12 Train_predict = modelo.predict(x_teste)
13 print(classification_report(y_teste, Train_predict ))

```

Acurácia: 0.9781316852864274

| | precision | recall | f1-score | support |
|--------------|-----------|--------|----------|---------|
| 0 | 1.00 | 0.96 | 0.98 | 6768 |
| 1 | 0.95 | 1.00 | 0.98 | 5853 |
| accuracy | | | 0.98 | 12621 |
| macro avg | 0.98 | 0.98 | 0.98 | 12621 |
| weighted avg | 0.98 | 0.98 | 0.98 | 12621 |

Logistic Regression

A regressão logística é um método estatístico bastante utilizado para realizar previsões. Um dos seus principais objetivos é entender como algumas variáveis de interesse influenciam outras que seriam os alvos, e através de um modelo matemático encontrar uma fórmula que consiga prever corretamente os valores alvo com base em novas variáveis de interesse.

Para utilizá-lo, importamos a classe LogisticRegression do pacote `sklearn.linear_model`.

```

1  #Regressão Logística
2  from sklearn.linear_model import LogisticRegression
3  #Criação do modelo
4  logistic_regression = LogisticRegression(random_state= 42, max_iter=2000)
5  logistic_regression.fit(x_treino, y_treino)
6
7  #imprimindo resultados
8  resultado = logistic_regression.score(x_teste, y_teste)
9  print('Acurácia:', resultado)
10
11 Train_predict = logistic_regression.predict(x_teste)
12 print(classification_report(y_teste, Train_predict ))

```

```

Acurácia: 0.9516678551620316
              precision    recall  f1-score   support

         0           1.00      0.91      0.95         6823
         1           0.91      1.00      0.95         5798

 accuracy                   0.95         12621
 macro avg              0.95      0.96      0.95         12621
 weighted avg           0.96      0.95      0.95         12621

```

KNeighbors Classifier (KNN)

Este é um dos algoritmos mais utilizados em meio a área de Machine Learning. Ele utiliza cálculos matemáticos e através dos resultados desses cálculos define se aqueles dados pertencem à classe X ou Y, a grosso modo, ele compara as características entre os elementos das classes e realiza a classificação com base na semelhança entre as características das variáveis.

Para utilizá-lo, importamos a classe KNeighborsClassifier do pacote sklearn.neighbors.

```

1  #KNeighborsClassifier
2  from sklearn.neighbors import KNeighborsClassifier
3  #Criação do modelo
4  neighbors = KNeighborsClassifier()
5  neighbors.fit(x_treino, y_treino)
6
7  #imprimindo resultados
8  resultado = neighbors.score(x_teste, y_teste)
9  print('Acurácia:', resultado)
10
11 Train_predict = neighbors.predict(x_teste)
12 print(classification_report(y_teste, Train_predict ))

```

Acurácia: 0.9512716900404088

| | precision | recall | f1-score | support |
|--------------|-----------|--------|----------|---------|
| 0 | 1.00 | 0.91 | 0.95 | 6812 |
| 1 | 0.90 | 1.00 | 0.95 | 5809 |
| accuracy | | | 0.95 | 12621 |
| macro avg | 0.95 | 0.95 | 0.95 | 12621 |
| weighted avg | 0.96 | 0.95 | 0.95 | 12621 |

Naive Bayes

Diferentemente dos outros, neste caso trata-se de um algoritmo baseado em probabilidade. Foi baseado no “Teorema de Bayes” com o objetivo de tentar provar a existência de Deus. Normalmente este algoritmo é utilizado para classificação de textos desconsiderando totalmente a correlação entre variáveis.

Para utilizá-lo, importamos a classe GaussianNB do pacote `sklearn.naive_bayes`.

```

1 #Naive Bayes
2 from sklearn.naive_bayes import GaussianNB
3 #Criação do modelo
4 naive_bayes = GaussianNB()
5 naive_bayes.fit(x_treino, y_treino)
6
7 #imprimindo resultados
8 resultado = naive_bayes.score(x_teste, y_teste)
9 print('Acurácia:', resultado)
10
11 Train_predict = naive_bayes.predict(x_teste)
12 print(classification_report(y_teste, Train_predict ))

```

Acurácia: 0.5728547658664132

| | precision | recall | f1-score | support |
|--------------|-----------|--------|----------|---------|
| 0 | 1.00 | 0.21 | 0.35 | 6812 |
| 1 | 0.52 | 1.00 | 0.68 | 5809 |
| accuracy | | | 0.57 | 12621 |
| macro avg | 0.76 | 0.60 | 0.51 | 12621 |
| weighted avg | 0.78 | 0.57 | 0.50 | 12621 |

SGD Classifier

É um algoritmo que realiza ajuste de parâmetros de maneira iterativa buscando sempre a melhor reta que se ajusta aos dados estudados. À medida que o algoritmo vai processando os dados ele se aproxima mais da reta ideal. O problema é que este algoritmo necessita de um diversos hiperparâmetros, apesar de fácil implementação.

Para utilizá-lo, importamos a classe SGDClassifier do pacote `sklearn.linear_model`.

```

1  #SGDClassifier
2  from sklearn.linear_model import SGDClassifier
3  #Criação do modelo
4  sgdc = SGDClassifier()
5  sgdc.fit(x_treino, y_treino)
6
7  #imprimindo resultados
8  resultado = sgdc.score(x_teste, y_teste)
9  print('Acurácia:', resultado)
10
11 Train_predict = sgdc.predict(x_teste)
12 print(classification_report(y_teste, Train_predict ))

```

Acurácia: 0.9187069170430235

| | precision | recall | f1-score | support |
|--------------|-----------|--------|----------|---------|
| 0 | 0.92 | 0.93 | 0.92 | 6812 |
| 1 | 0.91 | 0.91 | 0.91 | 5809 |
| accuracy | | | 0.92 | 12621 |
| macro avg | 0.92 | 0.92 | 0.92 | 12621 |
| weighted avg | 0.92 | 0.92 | 0.92 | 12621 |

Linear Discriminant Analysis

Conhecido também como “LDA”, um modelo altamente estatístico, é um algoritmo que reduz características redundantes buscando calcular médias dos grupos de dados e, para cada indivíduo desses grupos, calcula a probabilidade de pertencer a um grupo específico.

Para utilizá-lo, importamos a classe `LinearDiscriminantAnalysis` do pacote `sklearn.discriminant_analysis`.

```

1 #LinearDiscriminantAnalysis
2 from sklearn.discriminant_analysis import LinearDiscriminantAnalysis
3 #Criação do modelo
4 disc = LinearDiscriminantAnalysis()
5 disc.fit(x_treino, y_treino)
6
7 #imprimindo resultados
8 resultado = disc.score(x_teste, y_teste)
9 print('Acurácia:', resultado)
10
11 Train_predict = disc.predict(x_teste)
12 print(classification_report(y_teste, Train_predict ))

```

Acurácia: 0.9512716900404088

| | precision | recall | f1-score | support |
|--------------|-----------|--------|----------|---------|
| 0 | 1.00 | 0.91 | 0.95 | 6812 |
| 1 | 0.90 | 1.00 | 0.95 | 5809 |
| accuracy | | | 0.95 | 12621 |
| macro avg | 0.95 | 0.95 | 0.95 | 12621 |
| weighted avg | 0.96 | 0.95 | 0.95 | 12621 |

6. Interpretação dos Resultados

Nessa seção você deve interpretar os resultados obtidos na análise e exploração de dados e também interpretar os resultados da aplicação dos algoritmos de Machine Learning, descobrindo insights importantes para responder o problema proposto.

Utilizamos algumas métricas para avaliar se o modelo foi fiel ao resultado esperado ou não. A primeira delas é a acurácia, que basicamente mede a quantidade de acertos e divide pelo total de amostras.

$$acurácia = \frac{VP + VN}{VP + FN + VN + FP}$$

A segunda métrica utilizada é a precisão (*precision*). Ela se caracteriza por ser uma medida de exatidão ou até mesmo qualidade, e verifica quais classificações

positivas são realmente positivas, muito utilizada em casos em que o falso positivo é mais prejudicial que o falso negativo.

$$\textit{precisão} = \frac{VP}{VP + FP}$$

A terceira métrica utilizada é o *recall* que se caracteriza por mostrar a capacidade do modelo em questão de prever assertivamente os positivos dos positivos reais. Neste caso, é a situação em que os falsos negativos podem ser considerados mais prejudiciais que os falsos positivos.

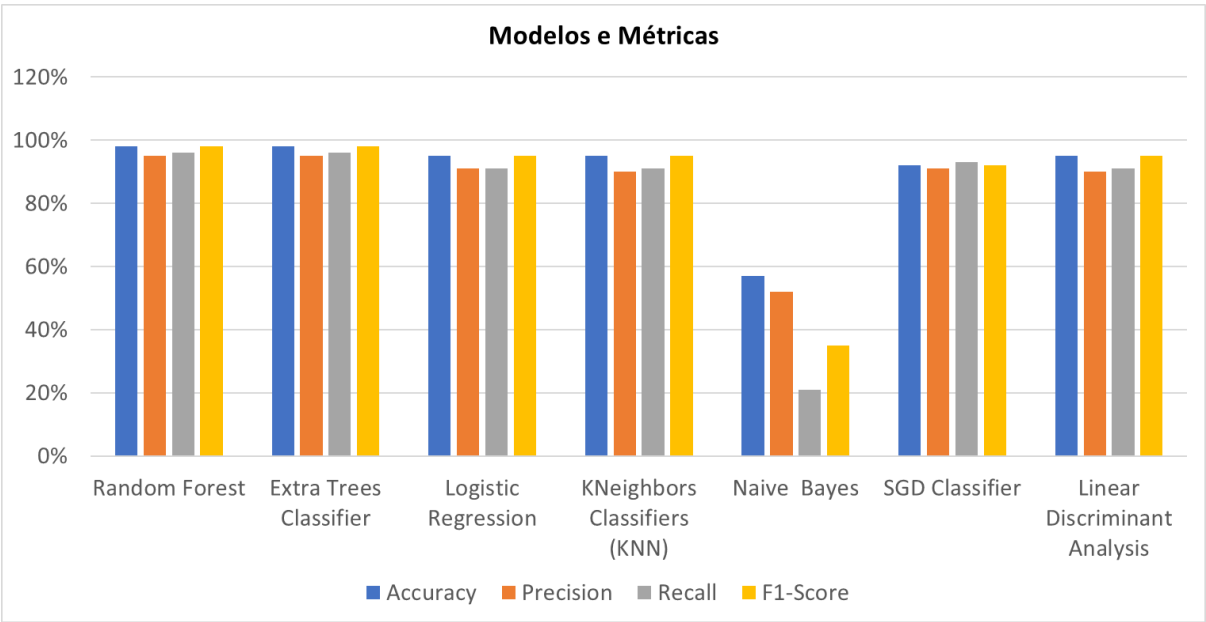
$$\textit{Recall} = \frac{TP}{TP + FN}$$

F1-score é uma métrica interessante pois consegue resumir as métricas *precision* e *recall* retornando uma média harmônica entre essas duas medidas. Muito útil quando se tenta chegar ao melhor valor entre precisão e *recall*.

$$\textit{F1 score} = 2 * \frac{\textit{Precision} * \textit{Recall}}{\textit{Precision} + \textit{Recall}}$$

Agora que entendemos o que vamos analisar e como faremos isso, foi feito o resumo de todos os resultados para facilitar nossa análise final:

| Modelo/Métricas | Accuracy | Precision | Recall | F1-Score |
|------------------------------|----------|-----------|--------|----------|
| Random Forest | 98% | 95% | 96% | 98% |
| Extra Trees Classifier | 98% | 95% | 96% | 98% |
| Logistic Regression | 95% | 91% | 91% | 95% |
| KNeighbors Classifiers (KNN) | 95% | 90% | 91% | 95% |
| Naive Bayes | 57% | 52% | 21% | 35% |
| SGD Classifier | 92% | 91% | 93% | 92% |
| Linear Discriminant Analysis | 95% | 90% | 91% | 95% |



Analisando todos os modelos através das mesmas métricas, fica muito mais simples determinar aqueles que fazem sentido serem usados. De acordo com a tabela acima, vemos que o pior modelo em todas as métricas analisadas é o de

Naive Bayes, o que faz total sentido, visto que, conforme dito no tópico 5 deste trabalho, normalmente este algoritmo é utilizado para classificação de textos desconsiderando totalmente a correlação entre variáveis.

Enquanto o melhor algoritmo, por 1 (um) centésimo de diferença do segundo colocado é o Extra Trees Classifier, fato que também é muito interessante, pois a principal diferença entre o modelo vencedor e o Random Forest, que ficou na segunda posição, é justamente que o modelo vencedor utiliza as amostras originais e não subgrupos como o Random Forest, o que pode ter causado essa pequena diferença entre eles.

Outro ponto importante a se observar é que em todas as aplicações de modelos, a distribuição de variáveis ficou em torno de 50% para a classe 1 e 50% para a classe 2. É muito importante que haja esse tipo de balanceamento pois, caso contrário, pode ser que o resultado final seja muito influenciado, enviesando nossas conclusões. Esse balanço pode ser observado na coluna Support nas imagens do tópico 5.

Através dos resultados obtidos, é possível dizer que com 98% de acurácia e 96% de Recall do modelo que melhor soube prever a necessidade de internação na UTI por pacientes com Covid-19 seriam muito relevantes no cenário pandêmico que se iniciou no Brasil no ano de 2020, e que com certeza fariam total diferença na organização de recursos hospitalares para melhor atender a sociedade.

7. Apresentação dos Resultados

THE MACHINE LEARNING CANVAS (V1.0)

Designed for:

Designed by:











Date:

Iteration:

| | | | | |
|---|---|--|--|---|
| <div>PREDICTION TASK</div> <div>?</div> <div>Type of task? Input object? Output: definition, parameters (e.g. prediction horizon), possible values?</div> | <div>DECISIONS</div> <div>⚙</div> <div>Process for turning predictions into proposed value for the end-user? Mention decision-making parameters.</div> | <div>VALUE PROPOSITION</div> <div>📦</div> <div>Who is the end-user? What are their objectives? How will they benefit from the ML system? Mention workflow/interfaces.</div> | <div>DATA COLLECTION</div> <div>⬇</div> <div>Strategy for initial train set, and continuous update. Collection rate? Holdout on prod inputs? Output acquisition cost?</div> | <div>DATA SOURCES</div> <div>🗄</div> <div>Which raw data sources can we use (internal, external)? Mention databases and tables, or APIs and methods of interest.</div> |
| <div>OFFLINE EVALUATION</div> <div>✓</div> <div>Simulation of the impact of decisions/predictions? Which test data? Cost/gain values? Deployment criteria (min performance value, fairness)?</div> | <div>MAKING PREDICTIONS</div> <div>⏮</div> <div>When do we make real-time / batch pred? Time available for this + featurization + post-processing? Compute target?</div> | | <div>BUILDING MODELS</div> <div>⚙</div> <div>How many prod models are needed? When would we update? Time available for this (including featurization and analysis)?</div> | <div>FEATURES</div> <div>📊</div> <div>Input representations available at prediction time, extracted from raw data sources.</div> |
| | <div>LIVE MONITORING</div> <div>📶</div> <div>Metrics to quantify value creation and measure the ML system's impact in production (on end-users and business)?</div> | | | |

machinelearningcanvas.com by Louis Dorard, Ph.D.

Licensed under a Creative Commons Attribution-ShareAlike 4.0 International License.

| | | | | |
|--|--|---|---|---|
| PREDICTION TASK  O valor alvo é a coluna UTI, onde o algoritmo retorna 1 ou 0, sendo 1 positivo para internação e zero negative. | DECISIONS  O resultado do modelos é a necessidade de internação dos infectados por Covid-19, possibilitando otimização de recursos nos hospitais. | VALUE PROPOSITION  O usuário final são os hospitais que utilizariam do ML para otimizar seus recursos. A cada nova entrada de paciente os dados dessa pessoa passaria pelo Sistema da instituição e o algoritmo determinaria em tempo real se haveria necessidade de internação na UTI ou não. | DATA COLLECTION  Os novos dados poderão ser coletados diretamente dos sites governamentais de forma gratuita à medida que forem sendo atualizados. | DATA SOURCES  Sites do governo e sites com bases de dados referents ao tema Covid-19. |
| IMPACT SIMULATION  Os modelos podem ser colocados em produção a um baixo custo considerando o volume de dados utilizado inicialmente. | MAKING PREDICTIONS  As predições são realizadas após o tratamento e o processamento dos dados e de forma rápida. | BUILDING MODELS  Fizemos avaliação em sete modelos diferentes para determinar qual era mais adequado ao problema proposto. | | FEATURES  As variáveis preditoras X são: 'SEXO','IDADE','MUNICIPIO_RESIDENCIA','EVOLUCAO','INTERNAcao', 'UTI','MEDIA_IDADE_POR_MUNICIPIO','DEATH_RATE' |
| | MONITORING  As métricas utilizadas para analisar e entender os resultados foram Accuracy, Recall, F1-score e Precision. | | | |

8. Links

Aqui você deve disponibilizar os links para o vídeo com sua apresentação de 5 minutos e para o repositório contendo os dados utilizados no projeto, scripts criados, etc.

Link para o vídeo:

https://www.youtube.com/watch?v=2ds1-u6kXu4&ab_channel=GustavoNassau

Link para o repositório:

<https://github.com/gnassau/TCC-P-s-gradua-o-PUC-MINAS-ciencia-de-dados-e-big-data>

REFERÊNCIAS

BROWNLEE, J. **How to Calculate Precision, Recall, and F-Measure for Imbalanced Classification**. Disponível em:

<<https://machinelearningmastery.com/precision-recall-and-f-measure-for-imbalanced-classification/#:~:text=For%20example%2C%20a%20perfect%20precision>>.

Acesso em: 23 jul. 2022.

RODRIGUES, V. **Métricas de Avaliação: acurácia, precisão, recall... quais as diferenças?** Disponível em:

<<https://vitorborbarodrigues.medium.com/m%C3%A9tricas-de-avalia%C3%A7%C3%A3o-acur%C3%A1cia-precis%C3%A3o-recall-quais-as-diferen%C3%A7as-c8f05e0a513c>>. Acesso em: 15 jun. 2022.

Accuracy, Precision, Recall & F1-Score - Python Examples. Disponível em:

<<https://vitalflux.com/accuracy-precision-recall-f1-score-python-example/>>.

105 Evaluating A Classification Model 6 Classification Report | Creating Machine Learning Models. Disponível em:

<https://www.youtube.com/watch?v=XiUIqN1Ay0U&ab_channel=MachineLearning>.

Acesso em: 12 jun. 2022.

ESCOVEDO, T. **Machine Learning — Conceitos e Modelos**. Disponível em:

<<https://tatianaesc.medium.com/machine-learning-conceitos-e-modelos-f0373bf4f445>>. Acesso em: 5 mai. 2022.

APÊNDICE

Programação/Scripts

```
In [1]: 1 import pandas as pd
2 import numpy as np
3 from sklearn.utils import resample
4 from collections import Counter
5 import json
6
7 #Para ML
8 from sklearn.metrics import classification_report
9 from sklearn.model_selection import train_test_split
10 from sklearn.ensemble import ExtraTreesClassifier
11 from sklearn.ensemble import RandomForestClassifier
12 from sklearn.linear_model import LogisticRegression
13 from sklearn.naive_bayes import GaussianNB
14 from sklearn.neighbors import KNeighborsClassifier
15 from sklearn.linear_model import SGDClassifier
16 from sklearn.discriminant_analysis import LinearDiscriminantAnalysis
```

Importando CSV

```
In [2]: 1 obitos = pd.read_csv('obitos.csv', index_col=False)
```

Analizando datasets

1- Óbitos por Covid

```
In [3]: 1 obitos
```

Out[3]:

| | _id | PACIENTE | SEXO | IDADE | MUNICIPIO_RESIDENCIA | DATA_OBITO | COMORBIDADE |
|------|------|----------|------|-------|-----------------------|---------------------|---------------|
| 0 | 1 | 1 | M | 79 | Patos de Minas | 2020-03-28T00:00:00 | SIM |
| 1 | 2 | 2 | F | 82 | Belo Horizonte | 2020-03-29T00:00:00 | SIM |
| 2 | 3 | 3 | M | 66 | Belo Horizonte | 2020-03-30T00:00:00 | SIM |
| 3 | 4 | 4 | M | 44 | Mariana | 2020-03-30T00:00:00 | NÃO |
| 4 | 5 | 5 | M | 80 | Uberlândia | 2020-03-30T00:00:00 | SIM |
| ... | ... | ... | ... | ... | ... | ... | ... |
| 2889 | 2890 | 2890 | M | 86 | Andradas | 2020-07-30T00:00:00 | SIM |
| 2890 | 2891 | 2891 | F | 68 | Borda da Mata | 2020-07-30T00:00:00 | SIM |
| 2891 | 2892 | 2892 | M | 55 | Iturama | 2020-07-31T00:00:00 | Não informado |
| 2892 | 2893 | 2893 | F | 81 | Conceição das Alagoas | 2020-01-08T00:00:00 | Não informado |
| 2893 | 2894 | 2894 | M | 68 | Pouso Alegre | 2020-01-08T00:00:00 | SIM |

2894 rows × 7 columns

```
In [4]: 1 #Entendendo tipo dos dados
        2 obitos.info()

<class 'pandas.core.frame.DataFrame'>
RangeIndex: 2894 entries, 0 to 2893
Data columns (total 7 columns):
#   Column                Non-Null Count  Dtype
---  -
0   _id                    2894 non-null   int64
1   PACIENTE              2894 non-null   int64
2   SEXO                  2894 non-null   object
3   IDADE                 2894 non-null   int64
4   MUNICIPIO_RESIDENCIA  2894 non-null   object
5   DATA_OBITO           2823 non-null   object
6   COMORBIDADE           2894 non-null   object
dtypes: int64(3), object(4)
memory usage: 158.4+ KB
```

```
In [5]: 1 #verificando existência de valores nulos
        2 obitos.isnull().sum()
```

```
Out[5]: _id                0
        PACIENTE         0
        SEXO             0
        IDADE            0
        MUNICIPIO_RESIDENCIA  0
        DATA_OBITO      71
        COMORBIDADE      0
        dtype: int64
```

```
In [6]: 1 #removendo valores nulos
        2 obitos = obitos[obitos['DATA_OBITO'].notnull()]
```

```
In [7]: 1 #retirando colunas que não serão usadas
        2 obitos = obitos.loc[:, ['IDADE', 'MUNICIPIO_RESIDENCIA']]
        3 obitos.head()
```

```
Out[7]:
```

| | IDADE | MUNICIPIO_RESIDENCIA |
|---|-------|----------------------|
| 0 | 79 | Patos de Minas |
| 1 | 82 | Belo Horizonte |
| 2 | 66 | Belo Horizonte |
| 3 | 44 | Mariana |
| 4 | 80 | Uberlândia |

```
In [8]: 1 #corrigindo coluna MUNICIPIO_RESIDENCIA, deixando maiuscula e sem acentuação
        2
        3 obitos['MUNICIPIO_RESIDENCIA'] = obitos['MUNICIPIO_RESIDENCIA'].str.normalize('NFKD')\
        4     .str.encode('ascii', errors='ignore')\
        5     .str.decode('utf-8').astype(str).str.upper()
        6
        7 obitos
```

```
Out[8]:
```

| | IDADE | MUNICIPIO_RESIDENCIA |
|------|-------|-----------------------|
| 0 | 79 | PATOS DE MINAS |
| 1 | 82 | BELO HORIZONTE |
| 2 | 66 | BELO HORIZONTE |
| 3 | 44 | MARIANA |
| 4 | 80 | UBERLANDIA |
| ... | ... | ... |
| 2889 | 86 | ANDRADAS |
| 2890 | 68 | BORDA DA MATA |
| 2891 | 55 | ITURAMA |
| 2892 | 81 | CONCEICAO DAS ALAGOAS |
| 2893 | 68 | POUSO ALEGRE |

```
In [9]: 1 #agrupando por municipios... média de idade e soma de óbitos por município
2 obitos = obitos.groupby(by = 'MUNICIPIO_RESIDENCIA', as_index = False).agg( {'IDADE':'mean'}).round()
3 obitos = obitos.rename({'IDADE': 'MEDIA_IDADE_POR_MUNICIPIO'}, axis=1) #renomeando coluna idade
4 obitos
```

```
Out[9]:
```

| | MUNICIPIO_RESIDENCIA | MEDIA_IDADE_POR_MUNICIPIO |
|-----|------------------------|---------------------------|
| 0 | ABADIA DOS DOURADOS | 60.0 |
| 1 | ACAIACA | 85.0 |
| 2 | ACUCENA | 60.0 |
| 3 | AGUANIL | 63.0 |
| 4 | AGUAS FORMOSAS | 74.0 |
| ... | ... | ... |
| 393 | VERISSIMO | 84.0 |
| 394 | VERMELHO NOVO | 64.0 |
| 395 | VESPASIANO | 57.0 |
| 396 | VISCONDE DO RIO BRANCO | 77.0 |
| 397 | VOLTA GRANDE | 71.0 |

```
In [10]: 1 #Verificando e Removendo duplicatas
2 print('Linhas antes de remover duplicatas:', obitos['MUNICIPIO_RESIDENCIA'].shape[0])
3 obitos = obitos.drop_duplicates()
4 print('Linhas depois de remover duplicatas:', obitos['MUNICIPIO_RESIDENCIA'].shape[0])
```

Linhas antes de remover duplicatas: 398
Linhas depois de remover duplicatas: 398

2- Casos confirmados por Covid

```
In [11]: 1 casos_confirmados = pd.read_csv('casos_confirmados.csv')
2 casos_confirmados
```

| 3 | 4 | NaN | NaN | NaN | 4 | 2020-06-14T00:00:00 | CASO CONFIRMADO | MASCULINO | 36.0 | 30 A 39 ANOS | | | | | | | | |
|-------|-------|---------------|---------------------------|----------|-------|---------------------|-----------------|-----------|------|--------------|--|--|--|--|--|--|--|--|
| 4 | 5 | TEOFILO OTONI | TEOFILO OTONI/MALACACHETA | NORDESTE | 5 | 2020-06-14T00:00:00 | CASO CONFIRMADO | MASCULINO | 51.0 | 50 A 59 ANOS | | | | | | | | |
| ... | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... | | | | | | | | |
| 46441 | 46442 | UBA | UBA | SUDESTE | 46442 | 2020-07-24T00:00:00 | CASO CONFIRMADO | MASCULINO | 20.0 | 20 A 29 ANOS | | | | | | | | |
| 46442 | 46443 | NaN | NaN | NaN | 46443 | 2020-07-28T00:00:00 | CASO CONFIRMADO | FEMININO | 39.0 | 30 A 39 ANOS | | | | | | | | |
| 46443 | 46444 | ALFENAS | GUAXUPE | SUL | 46444 | 2020-08-03T00:00:00 | CASO CONFIRMADO | FEMININO | 39.0 | 30 A 39 ANOS | | | | | | | | |
| 46444 | 46445 | DIVINOPOLIS | BOM DESPACHO | OESTE | 46445 | 2020-08-06T00:00:00 | CASO CONFIRMADO | MASCULINO | 27.0 | 20 A 29 ANOS | | | | | | | | |
| 46445 | 46446 | DIVINOPOLIS | DIVINOPOLIS | OESTE | 46446 | 2020-08-08T00:00:00 | CASO CONFIRMADO | MASCULINO | 50.0 | 50 A 59 ANOS | | | | | | | | |

46446 rows x 19 columns

```
In [12]: 1 #Entendendo tipo dos dados
2 casos_confirmados.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 46446 entries, 0 to 46445
Data columns (total 19 columns):
#   Column                Non-Null Count  Dtype
---  -
0   _id                    46446 non-null  int64
1   URS                    45238 non-null  object
2   MICRO                  45238 non-null  object
3   MACRO                  45238 non-null  object
4   ID                     46446 non-null  int64
5   DATA_NOTIFICACAO      39302 non-null  object
6   CLASSIFICACAO_CASO     46446 non-null  object
7   SEXO                   46446 non-null  object
8   IDADE                  45581 non-null  float64
9   FAIXA_ETARIA           45552 non-null  object
10  MUNICIPIO_RESIDENCIA   46446 non-null  object
11  CODIGO                  45238 non-null  float64
12  COMORBIDADE             46446 non-null  object
13  EVOLUCAO                46446 non-null  object
14  INTERNACAO              46446 non-null  object
15  UTI                     46446 non-null  object
16  RACA                    46446 non-null  object
17  DATA_ATUALIZACAO       46446 non-null  object
18  ORIGEM_DA_INFORMACAO   46446 non-null  object
dtypes: float64(2), int64(2), object(15)
memory usage: 6.7+ MB
```

```
In [13]: 1 #verificando existência de valores nulos
         2 casos_confirmados.isnull().sum()
```

```
Out[13]: _id                0
         URS                1208
         MICRO              1208
         MACRO              1208
         ID                 0
         DATA_NOTIFICACAO  7144
         CLASSIFICACAO_CASO  0
         SEXO               0
         IDADE              865
         FAIXA_ETARIA       894
         MUNICIPIO_RESIDENCIA 0
         CODIGO             1208
         COMORBIDADE        0
         EVOLUCAO           0
         INTERNACAO         0
         UTI                0
         RACA               0
         DATA_ATUALIZACAO  0
         ORIGEM_DA_INFORMACAO 0
         dtype: int64
```

```
In [14]: 1 #retirando valores nulos da coluna 'DATA_OBITO', 'FAIXA_ETARIA' e 'MACRO'
         2 casos_confirmados = casos_confirmados[(casos_confirmados['DATA_NOTIFICACAO'].notnull()) &
         3                                           (casos_confirmados['FAIXA_ETARIA'].notnull()) &
         4                                           (casos_confirmados['MACRO'].notnull())]
         5 casos_confirmados.isnull().sum()
```

```
Out[14]: _id                0
         URS                0
         MICRO              0
         MACRO              0
         ID                 0
         DATA_NOTIFICACAO  0
         CLASSIFICACAO_CASO 0
         SEXO               0
         IDADE              0
         FAIXA_ETARIA       0
         MUNICIPIO_RESIDENCIA 0
         CODIGO             0
         COMORBIDADE        0
         EVOLUCAO           0
         INTERNACAO         0
         UTI                0
         RACA               0
         DATA_ATUALIZACAO  0
         ORIGEM_DA_INFORMACAO 0
         dtype: int64
```

```
In [16]: 1 #retirando colunas que não serão usadas
         2 casos_confirmados = casos_confirmados.loc[:,['MACRO', 'DATA_NOTIFICACAO',
         3                                           'CLASSIFICACAO_CASO', 'SEXO', 'IDADE',
         4                                           'MUNICIPIO_RESIDENCIA', 'COMORBIDADE', 'EVOLUCAO',
         5                                           'INTERNACAO', 'UTI', 'RACA']]
         6 casos_confirmados
```

| | | | | | | | | | |
|-------|---------|---------------------|-----------------|-----------|------|----------------------|-----|----------------|------|
| 5 | LESTE | 2020-06-05T00:00:00 | CASO CONFIRMADO | MASCULINO | 41.0 | VALADARES | NAO | RECUPERADO | |
| 6 | CENTRO | 2020-05-15T00:00:00 | CASO CONFIRMADO | FEMININO | 27.0 | BELO HORIZONTE | NAO | RECUPERADO | INFO |
| ... | ... | ... | ... | ... | ... | ... | ... | ... | ... |
| 46440 | LESTE | 2020-08-07T00:00:00 | CASO CONFIRMADO | MASCULINO | 25.0 | GOVERNADOR VALADARES | NAO | RECUPERADO | |
| 46441 | SUDESTE | 2020-07-24T00:00:00 | CASO CONFIRMADO | MASCULINO | 20.0 | GUIRICEMA | NAO | ACOMPANHAMENTO | EM |
| 46443 | SUL | 2020-08-03T00:00:00 | CASO CONFIRMADO | FEMININO | 39.0 | MUZAMBINHO | NAO | RECUPERADO | |
| 46444 | OESTE | 2020-08-06T00:00:00 | CASO CONFIRMADO | MASCULINO | 27.0 | BOM DESPACHO | NAO | RECUPERADO | |
| 46445 | OESTE | 2020-08-08T00:00:00 | CASO CONFIRMADO | MASCULINO | 50.0 | CLAUDIO | NAO | ACOMPANHAMENTO | EM |

37274 rows x 11 columns


```
In [17]: 1 #corrigindo coluna MUNICIPIO_RESIDENCIA, deixando maiuscula e sem acentuação
2 casos_confirmados = casos_confirmados.apply(lambda x: x.astype(str).str.upper())
3 casos_confirmados['MUNICIPIO_RESIDENCIA'] = casos_confirmados['MUNICIPIO_RESIDENCIA'].str.normalize('NFKD')\
4 .str.encode('ascii', errors='ignore')\
5 .str.decode('utf-8')
```

```
In [18]: 1 #Verificando e Removendo duplicatas
2 print('Casos confirmados')
3 print('Linhas antes de remover duplicatas:', casos_confirmados.shape[0])
4 casos_confirmados = casos_confirmados.drop_duplicates()
5 print('Linhas depois de remover duplicatas:', casos_confirmados.shape[0])
```

Casos confirmados
 Linhas antes de remover duplicatas: 37274
 Linhas depois de remover duplicatas: 34701

```
In [19]: 1 #deixei passar? remover comorbidade = nao informado
2 casos_confirmados['COMORBIDADE'].value_counts()
```

```
Out[19]: NAO INFORMADO    25060
NAO                6255
SIM                3386
Name: COMORBIDADE, dtype: int64
```

```
In [20]: 1 casos_confirmados.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 34701 entries, 1 to 46445
Data columns (total 11 columns):
#   Column                Non-Null Count  Dtype
---  -
0   MACRO                  34701 non-null  object
1   DATA_NOTIFICACAO      34701 non-null  object
2   CLASSIFICACAO_CASO     34701 non-null  object
3   SEXO                   34701 non-null  object
4   IDADE                  34701 non-null  object
5   MUNICIPIO_RESIDENCIA   34701 non-null  object
6   COMORBIDADE            34701 non-null  object
7   EVOLUCAO               34701 non-null  object
8   INTERNACAO            34701 non-null  object
9   UTI                    34701 non-null  object
10  RACA                   34701 non-null  object
dtypes: object(11)
memory usage: 3.2+ MB
```

3- Mortes COVID-19

```
In [21]: 1 casos_e_mortes_confirmadas = pd.read_csv('caso.csv', index_col=False)
```

```
In [22]: 1 casos_e_mortes_confirmadas
```

```
Out[22]:
```

| | date | state | city | place_type | confirmed | deaths | order_for_place | is_last | estimated_population_2019 | estimated_population | city_ibge_code | confirm |
|---------|------------|-------|------|------------|-----------|--------|-----------------|---------|---------------------------|----------------------|----------------|---------|
| 0 | 2022-03-27 | AP | NaN | state | 160328 | 2122 | 734 | True | 845731.0 | 861773.0 | 16.0 | |
| 1 | 2022-03-26 | AP | NaN | state | 160321 | 2122 | 733 | False | 845731.0 | 861773.0 | 16.0 | |
| 2 | 2022-03-25 | AP | NaN | state | 160314 | 2122 | 732 | False | 845731.0 | 861773.0 | 16.0 | |
| 3 | 2022-03-24 | AP | NaN | state | 160301 | 2122 | 731 | False | 845731.0 | 861773.0 | 16.0 | |
| 4 | 2022-03-23 | AP | NaN | state | 160288 | 2122 | 730 | False | 845731.0 | 861773.0 | 16.0 | |
| ... | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... |
| 2837998 | 2020-06-23 | SP | Óleo | city | 1 | 0 | 5 | False | 2496.0 | 2471.0 | 3533809.0 | |
| 2837999 | 2020-06-23 | SP | Óleo | city | 1 | 0 | 5 | False | 2496.0 | 2471.0 | 3533809.0 | |

```
In [23]: 1 #Entendendo tipo dos dados
2 casos_e_mortes_confirmadas.info()

<class 'pandas.core.frame.DataFrame'>
RangeIndex: 2838003 entries, 0 to 2838002
Data columns (total 13 columns):
#   Column                                Dtype
---  ---
0   date                                  object
1   state                                 object
2   city                                  object
3   place_type                           object
4   confirmed                             int64
5   deaths                               int64
6   order_for_place                       int64
7   is_last                               bool
8   estimated_population_2019             float64
9   estimated_population                 float64
10  city_ibge_code                        float64
11  confirmed_per_100k_inhabitants         float64
12  death_rate                            float64
dtypes: bool(1), float64(5), int64(3), object(4)
memory usage: 262.5+ MB
```

```
In [24]: 1 #Filtrando coluna DATE entre 04/03/2020 e 12/08/2020
2 casos_e_mortes_confirmadas = casos_e_mortes_confirmadas[(casos_e_mortes_confirmadas['date'] >= '2020-04-03')
3               & (casos_e_mortes_confirmadas['date'] <= '2020-08-12')].sort_values(by=['date'], ascending= True )
```

```
In [25]: 1 #filtrando apenas por cidades
2 casos_e_mortes_confirmadas = casos_e_mortes_confirmadas[casos_e_mortes_confirmadas['place_type'] == 'city']
3 #filtrando apenas estado de interesse (MG)
4 casos_e_mortes_confirmadas = casos_e_mortes_confirmadas[casos_e_mortes_confirmadas['state'] == 'MG']
5 #filtrando apenas colunas de interesse
6 casos_e_mortes_confirmadas = casos_e_mortes_confirmadas.loc[:, ['city', 'death_rate']]
```

```
In [26]: 1 #Tabela final com taxa de mortalidade arredondada pelas cidades de MG, no período especificado
2 casos_e_mortes_confirmadas = (casos_e_mortes_confirmadas.groupby(by=['city'], as_index=False).mean())\
3 .sort_values('death_rate', ascending = False)
4
5 casos_e_mortes_confirmadas['death_rate'] = (casos_e_mortes_confirmadas['death_rate']*100).round(2)
```

```
In [27]: 1 #verificando existência de valores nulos
2 casos_e_mortes_confirmadas.isnull().sum()
```

```
Out[27]: city      0
death_rate  0
dtype: int64
```

```
In [28]: 1 #Normalizando a coluna city, deixando maiuscula e sem acentuação
2
3 casos_e_mortes_confirmadas['city'] = casos_e_mortes_confirmadas['city'].str.normalize('NFKD')\
4 .str.encode('ascii', errors='ignore')\
5 .str.decode('utf-8').astype(str).str.upper()
```

```
In [29]: 1 casos_e_mortes_confirmadas.sort_values(by='city',ascending=True)
```

```
Out[29]:
```

| | city | death_rate |
|-----|------------------------|------------|
| 0 | ABADIA DOS DOURADOS | 0.47 |
| 1 | ABAETE | 0.29 |
| 2 | ABRE CAMPO | 0.00 |
| 3 | ACAIACA | 10.55 |
| 48 | ACUCENA | 10.76 |
| ... | ... | ... |
| 842 | VIRGINOPOLIS | 0.00 |
| 843 | VIRGOLANDIA | 0.00 |
| 845 | VISCONDE DO RIO BRANCO | 3.46 |
| 847 | VOLTA GRANDE | 4.91 |
| 849 | WENCESLAU BRAZ | 0.00 |

854 rows × 2 columns

```
In [30]: 1 #Verificando e Removendo duplicatas
2 print('Linhas antes de remover duplicatas:',casos_e_mortes_confirmadas.shape[0])
3 casos_e_mortes_confirmadas=casos_e_mortes_confirmadas.drop_duplicates()
4 print('Linhas depois de remover duplicatas:',casos_e_mortes_confirmadas.shape[0])
```

Linhas antes de remover duplicatas: 854
Linhas depois de remover duplicatas: 854

Unindo Dataframes

```
In [31]: 1 #join dos dfs
2
3 df = casos_confirmados.merge(obitos,
4                               how = 'left',
5                               on = 'MUNICIPIO_RESIDENCIA',
6                               suffixes=('_casos', '_obitos'))
```

```
In [32]: 1 #verificação de linhas extras no df devido ao join
2 print(casos_confirmados.merge(obitos, how = 'left', on = 'MUNICIPIO_RESIDENCIA', suffixes=('_casos', '_obitos')).shape)

(34701, 12)
```

```
In [33]: 1 #Substituindo valores Nulos
2 df = df.fillna(0)
3 df.isnull().sum()
```

```
Out[33]: MACRO                                0
DATA_NOTIFICACAO                             0
CLASSIFICACAO_CASO                           0
SEXO                                           0
IDADE                                          0
MUNICIPIO_RESIDENCIA                         0
COMORBIDADE                                  0
EVOLUCAO                                      0
INTERNACAO                                    0
UTI                                            0
RACA                                           0
MEDIA_IDADE_POR_MUNICIPIO                    0
dtype: int64
```

```
In [34]: 1 print(df.groupby('MUNICIPIO_RESIDENCIA')['MUNICIPIO_RESIDENCIA'].nunique().sum())
2 print(obitos.groupby('MUNICIPIO_RESIDENCIA')['MUNICIPIO_RESIDENCIA'].nunique().sum())
3 print(casos_confirmados.groupby('MUNICIPIO_RESIDENCIA')['MUNICIPIO_RESIDENCIA'].nunique().sum())
```

```
664
398
664
```

```
In [35]: 1 print(df.groupby('MUNICIPIO_RESIDENCIA')['MUNICIPIO_RESIDENCIA'].count().sum())
2 print(obitos.groupby('MUNICIPIO_RESIDENCIA')['MUNICIPIO_RESIDENCIA'].count().sum())
3 print(casos_confirmados.groupby('MUNICIPIO_RESIDENCIA')['MUNICIPIO_RESIDENCIA'].count().sum())
4
5 #28430
```

```
34701
398
34701
```

```
In [37]: 1 #join dos casos de morte
2
3 df = df.merge(casos_e_mortes_confirmadas,
4               how = 'left',
5               left_on='MUNICIPIO_RESIDENCIA',
6               right_on='city',
7               suffixes=('_df', '_mortes'))
8 df
```

Out[37]:

| | MACRO | DATA_NOTIFICACAO | CLASSIFICACAO_CASO | SEXO | IDADE | MUNICIPIO_RESIDENCIA | COMORBIDADE | EVOLUCAO | INTERNA |
|---|----------|---------------------|--------------------|-----------|-------|----------------------|---------------|-------------------|---------|
| 0 | SUDESTE | 2020-05-08T00:00:00 | CASO CONFIRMADO | MASCULINO | 33.0 | JUIZ DE FORA | NAO INFORMADO | RECUPERADO | |
| 1 | LESTE | 2020-06-19T00:00:00 | CASO CONFIRMADO | FEMININO | 25.0 | GOVERNADOR VALADARES | NAO | EM ACOMPANHAMENTO | |
| 2 | NORDESTE | 2020-06-14T00:00:00 | CASO CONFIRMADO | MASCULINO | 51.0 | TEOFILO OTONI | NAO INFORMADO | INVESTIGACAO | |

```
In [38]: 1 df.info()

<class 'pandas.core.frame.DataFrame'>
Int64Index: 34701 entries, 0 to 34700
Data columns (total 14 columns):
#   Column                Non-Null Count  Dtype
---  -
0   MACRO                  34701 non-null  object
1   DATA_NOTIFICACAO      34701 non-null  object
2   CLASSIFICACAO_CASO     34701 non-null  object
3   SEXO                   34701 non-null  object
4   IDADE                  34701 non-null  object
5   MUNICIPIO_RESIDENCIA   34701 non-null  object
6   COMORBIDADE            34701 non-null  object
7   EVOLUCAO               34701 non-null  object
8   INTERNACAO             34701 non-null  object
9   UTI                    34701 non-null  object
10  RACA                   34701 non-null  object
11  MEDIA_IDADE_POR_MUNICIPIO 34701 non-null  float64
12  city                   34685 non-null  object
13  death_rate             34685 non-null  float64
dtypes: float64(2), object(12)
memory usage: 4.0+ MB
```

Aplicação Machine Learning

Igualando amostras da coluna 'UTI' ¶

```
In [40]: 1 #verificando a maior quantidade de itens na coluna
2 from sklearn.utils import resample
3 df_maior = df[df['UTI'] == 'NAO']
4 print('linhas:', df_maior.shape[0])
```

linhas: 22771

```
In [41]: 1 #verificando a menor quantidade de itens na coluna
2 df_menor = df[df['UTI'] == 'SIM']
3 print('linhas:', df_menor.shape[0])
```

linhas: 881

```
In [42]: 1 #Igualando a quantidade de entradas de dos dfs
2 df_unsampled = resample(df_menor, replace = True, n_samples=19372, random_state=123)
3 df_unsampled.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 19372 entries, 17110 to 610
Data columns (total 14 columns):
#   Column                Non-Null Count  Dtype
---  -
0   MACRO                  19372 non-null  object
1   DATA_NOTIFICACAO      19372 non-null  object
2   CLASSIFICACAO_CASO     19372 non-null  object
3   SEXO                   19372 non-null  object
4   IDADE                  19372 non-null  object
5   MUNICIPIO_RESIDENCIA   19372 non-null  object
6   COMORBIDADE            19372 non-null  object
7   EVOLUCAO               19372 non-null  object
8   INTERNACAO             19372 non-null  object
9   UTI                    19372 non-null  object
10  RACA                   19372 non-null  object
11  MEDIA_IDADE_POR_MUNICIPIO 19372 non-null  float64
12  city                   19332 non-null  object
13  death_rate             19332 non-null  float64
```

```
In [43]: 1 #Concatenando resultados
2 df = pd.concat([df_unsampled, df_maior])
3 df.info()

<class 'pandas.core.frame.DataFrame'>
Int64Index: 42143 entries, 17110 to 34700
Data columns (total 14 columns):
#   Column                Non-Null Count  Dtype
---  -
0   MACRO                  42143 non-null  object
1   DATA_NOTIFICACAO      42143 non-null  object
2   CLASSIFICACAO_CASO     42143 non-null  object
3   SEXO                   42143 non-null  object
4   IDADE                  42143 non-null  object
5   MUNICIPIO_RESIDENCIA   42143 non-null  object
6   COMORBIDADE            42143 non-null  object
7   EVOLUCAO               42143 non-null  object
8   INTERNACAO             42143 non-null  object
9   UTI                    42143 non-null  object
10  RACA                   42143 non-null  object
11  MEDIA_IDADE_POR_MUNICIPIO 42143 non-null  float64
12  city                   42089 non-null  object
13  death_rate             42089 non-null  float64
dtypes: float64(2), object(12)
memory usage: 4.8+ MB
```

Transformação de dados categóricos em números inteiros

```
In [44]: 1 #Retirando colunas desnecessárias
2 df = df.drop(['MACRO', 'DATA_NOTIFICACAO', 'CLASSIFICACAO_CASO', 'city'], axis = 1)
```

```
In [45]: 1 #Verificando dados das colunas
2 for column in df:
3     print(df[column].value_counts())
4     print('-----')
5
6 #percebe-se que nas colunas 'COMORBIDADE' e 'RACA' temos um grande valor de linhas classificadas como "NAO INFORMADO" que
7 #se equivale a null... como eles representam mais de 33% do total de linhas, vamos retirar essas colunas do df
```

```
MASCULINO      23575
FEMININO       18567
NAO INFORMADO    1
Name: SEXO, dtype: int64
-----
39.0           955
38.0           948
50.0           935
42.0           923
36.0           916
...
97.0            3
101.0            2
103.0            1
100.0            1
108.0            1
Name: IDADE, Length: 103, dtype: int64
-----
BELO HORIZONTE      4399
UFES
```

```
In [46]: 1 #Retirando colunas que possuem muitos dados inválidos
2 df = df.drop(['COMORBIDADE', 'RACA'], axis = 1)
```

```
In [47]: 1 #retirando dados inválidos
2 df = df[df['INTERNACAO'] != 'NAO INFORMADO']
3 df['INTERNACAO'].value_counts()
```

```
Out[47]: SIM      21401
NAO      20721
Name: INTERNACAO, dtype: int64
```

```
In [51]: 1 #Verificando dados das colunas
2 for column in df:
3     print(df[column].value_counts())
4     print('-----')
```

```
MASCULINO    23510
FEMININO     18557
Name: SEXO, dtype: int64
-----
39.0         954
38.0         947
50.0         934
42.0         921
36.0         916
...
97.0          3
101.0         2
103.0         1
100.0         1
108.0         1
Name: IDADE, Length: 103, dtype: int64
-----
BELO HORIZONTE    4398
UBERLANDIA        2786
GOVERNADOR VALADARES    2786
```

```
In [52]: 1 #Convertendo categorias em 0 e 1
2 df["SEXO"] = np.where(df["SEXO"] == "FEMININO", 1, 0)
3 df["INTERNACAO"] = np.where(df["INTERNACAO"] == "SIM", 1, 0)
4 df["UTI"] = np.where(df["UTI"] == "SIM", 1, 0)
5 df
```

```
Out[52]:
```

| | SEXO | IDADE | MUNICIPIO_RESIDENCIA | EVOLUCAO | INTERNACAO | UTI | MEDIA_IDADE_POR_MUNICIPIO | death_rate |
|-------|------|-------|----------------------|-------------------|------------|-----|---------------------------|------------|
| 17110 | 0 | 55.0 | PATOS DE MINAS | EM ACOMPANHAMENTO | 1 | 1 | 69.0 | 3.45 |
| 10790 | 1 | 42.0 | SAO JOAO DEL REI | RECUPERADO | 1 | 1 | 73.0 | 0.99 |
| 11329 | 1 | 53.0 | BELO HORIZONTE | EM ACOMPANHAMENTO | 1 | 1 | 72.0 | 2.38 |
| 8434 | 1 | 65.0 | EXTREMA | EM ACOMPANHAMENTO | 1 | 1 | 63.0 | 2.39 |
| 1327 | 1 | 73.0 | UBERLANDIA | RECUPERADO | 1 | 1 | 70.0 | 3.11 |
| ... | ... | ... | ... | ... | ... | ... | ... | ... |

```
In [53]: 1 #Convertendo em números as categorias da coluna 'MUNICIPIO_RESIDENCIA' e 'EVOLUCAO'
2 df = pd.get_dummies(df, columns=['MUNICIPIO_RESIDENCIA', 'EVOLUCAO'])
3 df
```

```
Out[53]:
```

| | SEXO | IDADE | INTERNACAO | UTI | MEDIA_IDADE_POR_MUNICIPIO | death_rate | MUNICIPIO_RESIDENCIA_ABADIA DOS DOURADOS | MUNICIPIO_RESIDENCIA_ABAETE | MUNICIPIO_RESIDENCIA_PATOS DE MINAS | MUNICIPIO_RESIDENCIA_SA O JOAO DEL REI | MUNICIPIO_RESIDENCIA_UBERLANDIA |
|-------|------|-------|------------|-----|---------------------------|------------|--|-----------------------------|-------------------------------------|--|---------------------------------|
| 17110 | 0 | 55.0 | 1 | 1 | 69.0 | 3.45 | 0 | 0 | 0 | 0 | 0 |
| 10790 | 1 | 42.0 | 1 | 1 | 73.0 | 0.99 | 0 | 0 | 0 | 0 | 0 |
| 11329 | 1 | 53.0 | 1 | 1 | 72.0 | 2.38 | 0 | 0 | 0 | 0 | 0 |
| 8434 | 1 | 65.0 | 1 | 1 | 63.0 | 2.39 | 0 | 0 | 0 | 0 | 0 |
| 1327 | 1 | 73.0 | 1 | 1 | 70.0 | 3.11 | 0 | 0 | 0 | 0 | 0 |
| ... | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... |
| 34696 | 0 | 25.0 | 0 | 0 | 69.0 | 6.71 | 0 | 0 | 0 | 0 | 0 |
| 34697 | 0 | 20.0 | 0 | 0 | 0.0 | 0.16 | 0 | 0 | 0 | 0 | 0 |
| 34698 | 1 | 39.0 | 0 | 0 | 85.0 | 0.32 | 0 | 0 | 0 | 0 | 0 |
| 34699 | 0 | 27.0 | 0 | 0 | 70.0 | 2.99 | 0 | 0 | 0 | 0 | 0 |

Aplicando Modelos de Machine Learning

```
In [55]: 1 from sklearn.model_selection import train_test_split
2 X_train = df.drop('UTI', axis = 1)
3 Y_train = df['UTI']
```

```
In [56]: 1 #Criando conjuntos de dados de teste e treino
2 x_treino, x_teste, y_treino, y_teste = train_test_split(X_train, Y_train, test_size = 0.3)
```

```
In [57]: 1 print(x_treino.shape)
2 print(x_teste.shape)
3
4 print(y_treino.shape)
5 print(y_teste.shape)
```

```
(29446, 645)
(12621, 645)
(29446,)
(12621,)
```

```
In [58]: 1 #ExtraTreesClassifier
2 from sklearn.ensemble import ExtraTreesClassifier
3 #Criação do modelo
4 modelo = ExtraTreesClassifier(random_state= 42)
5 modelo.fit(x_treino, y_treino)
6
7 #imprimindo resultados
8 resultado = modelo.score(x_teste, y_teste)
9 print('Acurácia:', resultado)
10
11 #Outras métricas de eficiência do modelo
12 Train_predict = modelo.predict(x_teste)
13 print(classification_report(y_teste, Train_predict ))
```

Acurácia: 0.9806671420648126

| | precision | recall | f1-score | support |
|--------------|-----------|--------|----------|---------|
| 0 | 1.00 | 0.96 | 0.98 | 6783 |
| 1 | 0.96 | 1.00 | 0.98 | 5838 |
| accuracy | | | 0.98 | 12621 |
| macro avg | 0.98 | 0.98 | 0.98 | 12621 |
| weighted avg | 0.98 | 0.98 | 0.98 | 12621 |

```
In [59]: 1 #Random Forest
2 from sklearn.ensemble import RandomForestClassifier
3 #Criação do modelo
4 random_forest = RandomForestClassifier(random_state= 42)
5 random_forest.fit(x_treino, y_treino)
6
7 #imprimindo resultados
8 resultado = random_forest.score(x_teste, y_teste)
9 print('Acurácia:', resultado)
10
11 #Outras métricas de eficiência do modelo
12 Train_predict = random_forest.predict(x_teste)
13 print(classification_report(y_teste, Train_predict))
```

Acurácia: 0.9809840741621108

| | precision | recall | f1-score | support |
|--------------|-----------|--------|----------|---------|
| 0 | 1.00 | 0.96 | 0.98 | 6783 |
| 1 | 0.96 | 1.00 | 0.98 | 5838 |
| accuracy | | | 0.98 | 12621 |
| macro avg | 0.98 | 0.98 | 0.98 | 12621 |
| weighted avg | 0.98 | 0.98 | 0.98 | 12621 |

```
In [60]: 1 #Regressão Logística
2 from sklearn.linear_model import LogisticRegression
3 #Criação do modelo
4 logistic_regression = LogisticRegression(random_state= 42, max_iter=2000)
5 logistic_regression.fit(x_treino, y_treino)
6
7 #imprimindo resultados
8 resultado = logistic_regression.score(x_teste, y_teste)
9 print('Acurácia:', resultado)
10
11 Train_predict = logistic_regression.predict(x_teste)
12 print(classification_report(y_teste, Train_predict ))
```

Acurácia: 0.9519055542350051

| | precision | recall | f1-score | support |
|--------------|-----------|--------|----------|---------|
| 0 | 1.00 | 0.91 | 0.95 | 6783 |
| 1 | 0.91 | 1.00 | 0.95 | 5838 |
| accuracy | | | 0.95 | 12621 |
| macro avg | 0.95 | 0.96 | 0.95 | 12621 |
| weighted avg | 0.96 | 0.95 | 0.95 | 12621 |


```
In [61]: 1 #Naive Bayes
2 from sklearn.naive_bayes import GaussianNB
3 #Criação do modelo
4 naive_bayes = GaussianNB()
5 naive_bayes.fit(x_treino, y_treino)
6
7 #imprimindo resultados
8 resultado = naive_bayes.score(x_teste, y_teste)
9 print('Acurácia:', resultado)
10
11 Train_predict = naive_bayes.predict(x_teste)
12 print(classification_report(y_teste, Train_predict ))
```

```
Acurácia: 0.5763410189366928
      precision    recall  f1-score   support

     0         1.00      0.21      0.35      6783
     1         0.52      1.00      0.69      5838

 accuracy          0.76
 macro avg          0.61
 weighted avg       0.58
```

```
In [62]: 1 #KNeighborsClassifier
2 from sklearn.neighbors import KNeighborsClassifier
3 #Criação do modelo
4 neighbors = KNeighborsClassifier()
5 neighbors.fit(x_treino, y_treino)
6
7 #imprimindo resultados
8 resultado = neighbors.score(x_teste, y_teste)
9 print('Acurácia:', resultado)
10
11 Train_predict = neighbors.predict(x_teste)
12 print(classification_report(y_teste, Train_predict ))
```

```
Acurácia: 0.9534109816971714
      precision    recall  f1-score   support

     0         1.00      0.91      0.95      6783
     1         0.91      1.00      0.95      5838

 accuracy          0.95
 macro avg          0.96
 weighted avg       0.95
```

```
In [63]: 1 #SGDClassifier
2 from sklearn.linear_model import SGDClassifier
3 #Criação do modelo
4 sgdc = SGDClassifier()
5 sgdc.fit(x_treino, y_treino)
6
7 #imprimindo resultados
8 resultado = sgdc.score(x_teste, y_teste)
9 print('Acurácia:', resultado)
10
11 Train_predict = sgdc.predict(x_teste)
12 print(classification_report(y_teste, Train_predict ))
```

```
Acurácia: 0.9413675619998415
      precision    recall  f1-score   support

     0         0.97      0.92      0.94      6783
     1         0.91      0.97      0.94      5838

 accuracy          0.94
 macro avg          0.94
 weighted avg       0.94
```

```
In [64]: 1 #LinearDiscriminantAnalysis
2 from sklearn.discriminant_analysis import LinearDiscriminantAnalysis
3 #Criação do modelo
4 disc = LinearDiscriminantAnalysis()
5 disc.fit(x_treino, y_treino)
6
7 #imprimindo resultados
8 resultado = disc.score(x_teste, y_teste)
9 print('Acurácia:', resultado)
10
11 Train_predict = disc.predict(x_teste)
12 print(classification_report(y_teste, Train_predict ))
```

```
Acurácia: 0.9515093891133825
      precision    recall  f1-score   support

    0         1.00      0.91      0.95         6783
    1         0.91      1.00      0.95         5838

 accuracy
macro avg      0.95      0.95      0.95         12621
weighted avg    0.96      0.95      0.95         12621
```

Análise exploratória

Análise: Óbitos por covid

```
In [221]: 1 from collections import Counter
2 import json
```

```
In [222]: 1 obitos = pd.read_csv('obitos.csv', index_col=False)
```

```
In [223]: 1 obitos.head()
```

```
Out[223]:
```

| | _id | PACIENTE | SEXO | IDADE | MUNICIPIO_RESIDENCIA | DATA_OBITO | COMORBIDADE |
|---|-----|----------|------|-------|----------------------|---------------------|-------------|
| 0 | 1 | 1 | M | 79 | Patos de Minas | 2020-03-28T00:00:00 | SIM |
| 1 | 2 | 2 | F | 82 | Belo Horizonte | 2020-03-29T00:00:00 | SIM |
| 2 | 3 | 3 | M | 66 | Belo Horizonte | 2020-03-30T00:00:00 | SIM |
| 3 | 4 | 4 | M | 44 | Mariana | 2020-03-30T00:00:00 | NÃO |
| 4 | 5 | 5 | M | 80 | Uberlândia | 2020-03-30T00:00:00 | SIM |

```
In [224]: 1 obitos.describe().round(2)
```

```
Out[224]:
```

| | _id | PACIENTE | IDADE |
|-------|---------|----------|---------|
| count | 2894.00 | 2894.00 | 2894.00 |
| mean | 1447.50 | 1447.50 | 69.86 |
| std | 835.57 | 835.57 | 15.12 |
| min | 1.00 | 1.00 | 0.00 |
| 25% | 724.25 | 724.25 | 61.00 |
| 50% | 1447.50 | 1447.50 | 72.00 |
| 75% | 2170.75 | 2170.75 | 81.00 |
| max | 2894.00 | 2894.00 | 107.00 |

```
In [225]: 1 #Convetendo coluna DATA_OBITO para datetime
2 obitos["DATA_OBITO"] = pd.to_datetime(obitos["DATA_OBITO"], format="%Y/%m/%d")
3 obitos['DATA_OBITO'].info()
4 obitos.head()
```

```
<class 'pandas.core.series.Series'>
RangeIndex: 2894 entries, 0 to 2893
Series name: DATA_OBITO
Non-Null Count  Dtype
-----
2823 non-null   datetime64[ns]
dtypes: datetime64[ns](1)
memory usage: 22.7 KB
```

```
Out[225]:
```

| | _id | PACIENTE | SEXO | IDADE | MUNICIPIO_RESIDENCIA | DATA_OBITO | COMORBIDADE |
|---|-----|----------|------|-------|----------------------|------------|-------------|
| 0 | 1 | 1 | M | 79 | Patos de Minas | 2020-03-28 | SIM |
| 1 | 2 | 2 | F | 82 | Belo Horizonte | 2020-03-29 | SIM |
| 2 | 3 | 3 | M | 66 | Belo Horizonte | 2020-03-30 | SIM |
| 3 | 4 | 4 | M | 44 | Mariana | 2020-03-30 | NÃO |
| 4 | 5 | 5 | M | 80 | Uberlândia | 2020-03-30 | SIM |

```
In [226]: 1 #Maior e menor data da coluna DATA_OBITO
          2 print(obitos['DATA_OBITO'].max())
          3 print(obitos['DATA_OBITO'].min())
```

```
2020-12-07 00:00:00
2020-01-04 00:00:00
```

```
In [227]: 1 #Quantidade de Municipios do estado de MG
          2 obitos.groupby('MUNICIPIO_RESIDENCIA').MUNICIPIO_RESIDENCIA.nunique().sum()
```

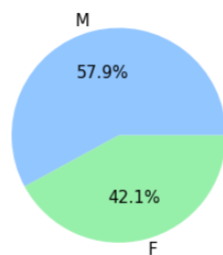
```
Out[227]: 405
```

```
In [228]: 1 #Quantidade de pessoas em cada genero
          2
          3 from collections import Counter
          4 genero_obitos = Counter(obitos['SEXO'])
          5 genero_obitos
```

```
Out[228]: Counter({'M': 1676, 'F': 1218})
```

```
In [229]: 1 #Quantidade de pessoas em cada genero
          2
          3 from matplotlib import pyplot as plt
          4 import numpy as np
          5
          6 plt.style.use('seaborn-pastel')
          7 plt.pie(genero_obitos.values(), labels = genero_obitos.keys(),
          8       autopct = '%1.1f%%', textprops = {'fontsize':15})
          9 plt.title('Gênero dos pacientes que foram a óbito', fontsize=18, pad = 40)
         10 plt.axis('image')
         11 plt.show()
```

Gênero dos pacientes que foram a óbito

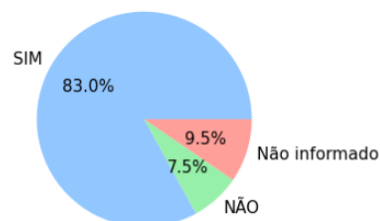


```
In [230]: 1 #Quantidade de pessoas que apresentavam comorbidades
          2
          3 from collections import Counter
          4 genero_obitos = Counter(obitos['COMORBIDADE'])
          5 genero_obitos
```

```
Out[230]: Counter({'SIM': 2402, 'NÃO': 216, 'Não informado': 276})
```

```
In [231]: 1 #Quantidade de pessoas que apresentavam comorbidades
          2
          3 from matplotlib import pyplot as plt
          4 import numpy as np
          5
          6 plt.style.use('seaborn-pastel')
          7 plt.pie(genero_obitos.values(), labels = genero_obitos.keys(),
          8       autopct = '%1.1f%%', textprops = {'fontsize':15})
          9 plt.title('Quantidade de pessoas que apresentavam comorbidades', fontsize=18, pad = 40)
         10 plt.axis('image')
         11 plt.show()
```

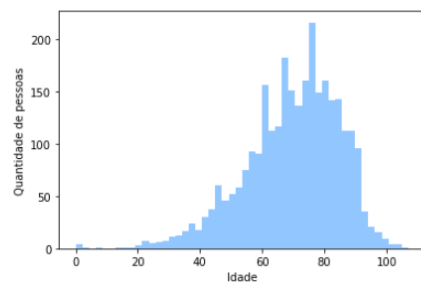
Quantidade de pessoas que apresentavam comorbidades



In [242]: 1 obitos.IDADE.value_counts()

```
Out[242]: 74    95
68    93
67    89
70    82
84    82
..|
2      1
17     1
107    1
24     1
14     1
Name: IDADE, Length: 92, dtype: int64
```

```
In [238]: 1 #Histograma de idade a óbito
2
3 from matplotlib import pyplot as plt
4 x = obitos.IDADE
5 # plt.style.use('ggplot')
6 plt.xlabel('Idade')
7 plt.ylabel('Quantidade de pessoas')
8 plt.hist(x, bins=50)
9
10 plt.show()
```



```
In [235]: 1 #Convertendo categorias em 0 e 1 para análise de correlação
2 obitos = pd.read_csv('obitos.csv', index_col=False)
3 obitos["COMORBIDADE"] = np.where(obitos["COMORBIDADE"] == "SIM", 1, 0)
4 obitos["SEXO"] = np.where(obitos["SEXO"] == "M", 1, 0)
5 obitos.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 2894 entries, 0 to 2893
Data columns (total 7 columns):
#   Column                Non-Null Count  Dtype
---  -
0   _id                   2894 non-null   int64
1   PACIENTE              2894 non-null   int64
2   SEXO                  2894 non-null   int32
3   IDADE                 2894 non-null   int64
4   MUNICIPIO_RESIDENCIA  2894 non-null   object
5   DATA_OBITO           2823 non-null   object
6   COMORBIDADE           2894 non-null   int32
dtypes: int32(2), int64(3), object(2)
memory usage: 135.8+ KB
```

```
In [236]: 1 #Plotando correlação entre
2 import pandas as pd
3 import numpy as np
4
5 obitos = obitos.drop(labels=['_id', 'PACIENTE'], axis=1)
6
7 rs = np.random.RandomState(0)
8 corr = obitos.corr()
9 corr.style.background_gradient(cmap='coolwarm').format(precision=2)
10
```

```
Out[236]:
```

| | SEXO | IDADE | COMORBIDADE |
|-------------|-------|-------|-------------|
| SEXO | 1.00 | -0.09 | -0.06 |
| IDADE | -0.09 | 1.00 | 0.14 |
| COMORBIDADE | -0.06 | 0.14 | 1.00 |

Análise: Casos confirmados Covid

```
In [272]: 1 casos_confirmados = pd.read_csv('casos_confirmados.csv')
```

```
In [273]: 1 casos_confirmados
```

```
Out[273]:
```

| | _id | URS | MICRO | MACRO | ID | DATA_NOTIFICACAO | CLASSIFICACAO_CASO | SEXO | IDADE | FAIXA_ETARIA | M |
|-----|-------|-------|----------------------|---------------------------|----------|------------------|---------------------|-----------------|-----------|--------------|--------------|
| | 0 | 1 | NaN | NaN | NaN | 1 | 2020-05-15T00:00:00 | CASO CONFIRMADO | MASCULINO | 0.0 | <1ANO |
| | 1 | 2 | JUIZ DE FORA | JUIZ DE FORA | SUDESTE | 2 | 2020-05-08T00:00:00 | CASO CONFIRMADO | MASCULINO | 33.0 | 30 A 39 ANOS |
| | 2 | 3 | GOVERNADOR VALADARES | GOVERNADOR VALADARES | LESTE | 3 | 2020-06-19T00:00:00 | CASO CONFIRMADO | FEMININO | 25.0 | 20 A 29 ANOS |
| | 3 | 4 | NaN | NaN | NaN | 4 | 2020-06-14T00:00:00 | CASO CONFIRMADO | MASCULINO | 36.0 | 30 A 39 ANOS |
| | 4 | 5 | TEOFILO OTONI | TEOFILO OTONI/MALACACHETA | NORDESTE | 5 | 2020-06-14T00:00:00 | CASO CONFIRMADO | MASCULINO | 51.0 | 50 A 59 ANOS |
| ... | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... |
| | 46441 | 46442 | UBA | UBA | SUDESTE | 46442 | 2020-07-24T00:00:00 | CASO CONFIRMADO | MASCULINO | 20.0 | 20 A 29 ANOS |
| | 46442 | 46443 | NaN | NaN | NaN | 46443 | 2020-07-28T00:00:00 | CASO CONFIRMADO | FEMININO | 39.0 | 30 A 39 ANOS |
| | 46443 | 46444 | ALFENAS | GUAXUPE | SUL | 46444 | 2020-08-03T00:00:00 | CASO CONFIRMADO | FEMININO | 39.0 | 30 A 39 ANOS |
| | 46444 | 46445 | DIVINOPOLIS | BOM DESPACHO | OESTE | 46445 | 2020-08-06T00:00:00 | CASO CONFIRMADO | MASCULINO | 27.0 | 20 A 29 ANOS |
| | 46445 | 46446 | DIVINOPOLIS | DIVINOPOLIS | OESTE | 46446 | 2020-08-08T00:00:00 | CASO CONFIRMADO | MASCULINO | 50.0 | 50 A 59 ANOS |

46446 rows × 19 columns

```
In [245]: 1 casos_confirmados.describe().round(2)
```

```
Out[245]:
```

| | _id | ID | IDADE | CODIGO |
|-------|----------|----------|----------|-----------|
| count | 46446.00 | 46446.00 | 45581.00 | 45238.00 |
| mean | 23223.50 | 23223.50 | 43.58 | 313516.00 |
| std | 13407.95 | 13407.95 | 18.48 | 2294.36 |
| min | 1.00 | 1.00 | 0.00 | 310010.00 |
| 25% | 11612.25 | 11612.25 | 31.00 | 311330.00 |
| 50% | 23223.50 | 23223.50 | 41.00 | 313190.00 |
| 75% | 34834.75 | 34834.75 | 55.00 | 315460.00 |
| max | 46446.00 | 46446.00 | 220.00 | 317220.00 |

```
In [262]: 1 #Convetendo coluna DATA_NOTIFICACAO para datetime
2 casos_confirmados["DATA_NOTIFICACAO"] = pd.to_datetime(\
3 casos_confirmados["DATA_NOTIFICACAO"], format="%Y/%m/%d")
4
5 casos_confirmados["DATA_NOTIFICACAO"].info()
```

```
<class 'pandas.core.series.Series'>
RangeIndex: 46446 entries, 0 to 46445
Series name: DATA_NOTIFICACAO
Non-Null Count  Dtype
-----
39302 non-null  datetime64[ns]
dtypes: datetime64[ns](1)
memory usage: 363.0 KB
```

```
In [247]: 1 #Maior e menor data da coluna DATA_NOTIFICACAO
2 print(casos_confirmados["DATA_NOTIFICACAO"].max())
3 print(casos_confirmados["DATA_NOTIFICACAO"].min())
```

```
2020-08-12 00:00:00
2020-03-04 00:00:00
```

```
In [279]: 1 #Quantidade de Municipios únicos
2 ## 400 a mais que o obitos por covid, o que faz sentido
3 casos_confirmados.groupby('MUNICIPIO_RESIDENCIA')['MUNICIPIO_RESIDENCIA'].nunique().sum()
```

Out[279]: 868

```
In [249]: 1 # a = [numero for numero in genero_obitos][1:]
2 # json_string = casos_confirmados['MACRO'].value_counts().to_json(orient='columns')
3 # json_object = json.loads(json_string)
4 # # genero_obitos
5 # b = [json_object[i] for i in json_object]
```

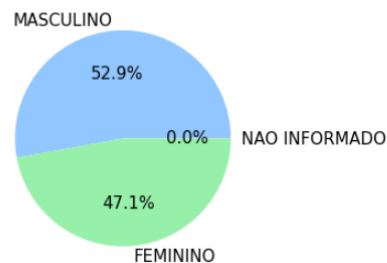
```
In [250]: 1 # #Regiões na coluna MACRO
2
3 #não vou usar pq tem muitas regiões
4 # from collections import Counter
5 # genero_obitos = Counter(casos_confirmados['MACRO'])
6 # genero_obitos
7
8 # import matplotlib.pyplot as plt
9 # fig = plt.figure()
10 # ax = fig.add_axes([0,0,8,5])
11 # langs = a
12 # students = b
13 # ax.bar(langs,students)
14 # plt.xticks(fontsize=35, rotation='vertical')
15 # plt.yticks(fontsize=35)
16 # plt.show()
```

```
In [251]: 1 #Quantidade de pessoas em cada genero
2
3 from collections import Counter
4 genero_obitos = Counter(casos_confirmados['SEXO'])
5 genero_obitos
```

Out[251]: Counter({'MASCULINO': 24547, 'FEMININO': 21889, 'NAO INFORMADO': 10})

```
In [252]: 1 #Quantidade de pessoas em cada genero
2
3 from matplotlib import pyplot as plt
4 import numpy as np
5
6 plt.style.use('seaborn-pastel')
7 plt.pie(genero_obitos.values(), labels = genero_obitos.keys(),
8 autopct = '%1.1f%%', textprops = {'fontsize':15})
9 plt.title('Gênero dos pacientes com casos Confirmados Covid-19', fontsize=18, pad = 40)
10 plt.axis('image')
11 plt.show()
```

Gênero dos pacientes com casos Confirmados Covid-19

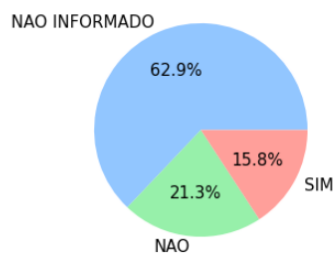


```

In [253]: 1 #Quantidade de pessoas que apresentavam comorbidades
2
3 from collections import Counter
4 comorbidade = Counter(casos_confirmados['COMORBIDADE'])
5 comorbidade
6
7 from matplotlib import pyplot as plt
8 import numpy as np
9
10 plt.style.use('seaborn-pastel')
11 plt.pie(comorbidade.values(), labels = comorbidade.keys(),
12 autopct = '%1.1f%%', textprops= {'fontsize':15})
13 plt.title('Pacientes possuíam comorbidade?', fontsize=18, pad = 40)
14 plt.axis('image')
15 plt.show()

```

Pacientes possuíam comorbidade?

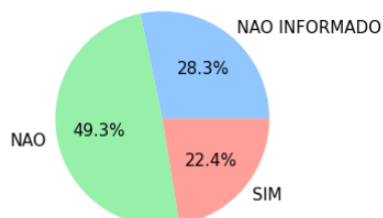


```

In [254]: 1 #Interação SIM/NÃO
2
3 from collections import Counter
4 internacao = Counter(casos_confirmados['INTERNAÇÃO'])
5 internacao
6
7 from matplotlib import pyplot as plt
8 import numpy as np
9
10 plt.style.use('seaborn-pastel')
11 plt.pie(internacao.values(), labels = internacao.keys(),
12 autopct = '%1.1f%%', textprops= {'fontsize':15})
13 plt.title('Interação dos casos registrados', fontsize=18, pad = 40)
14 plt.axis('image')
15 plt.show()

```

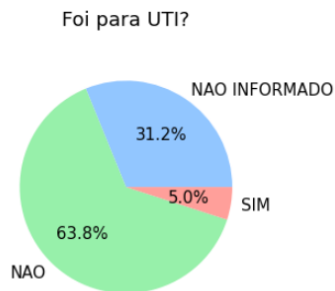
Interação dos casos registrados



```

In [255]: 1 #Foi para UTI?
2
3 from collections import Counter
4 uti = Counter(casos_confirmados['UTI'])
5
6 from matplotlib import pyplot as plt
7 import numpy as np
8
9 plt.style.use('seaborn-pastel')
10 plt.pie(uti.values(), labels = uti.keys(),
11 autopct = '%1.1f%%', textprops= {'fontsize':15})
12 plt.title('Foi para UTI?', fontsize=18, pad = 40)
13 plt.axis('image')
14 plt.show()

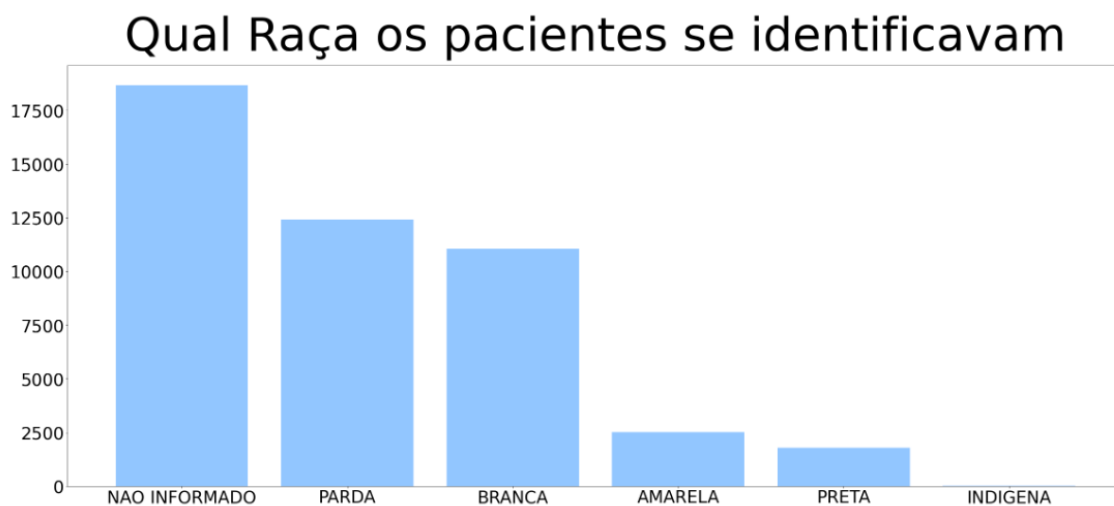
```



```

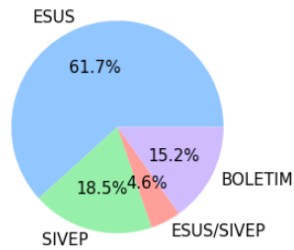
In [283]: 1 #Qual a Raça os pacientes se identificavam
2
3
4 import json
5 #Convertendo em Json
6 json = json.loads(casos_confirmados['RACA'].value_counts().to_json())
7
8 a = [i for i in json] #Pegando Keys em Lista
9 b = [json[i] for i in json] #Pegando Values em Lista
10
11 import numpy as np
12 import pandas as pd
13 import matplotlib.pyplot as plt
14 cmap = plt.cm.tab10
15 import matplotlib.pyplot as plt
16 fig = plt.figure()
17 ax = fig.add_axes([0,0,5,3])
18 langs = a
19 students = b
20 ax.bar(langs,students)
21 plt.xticks(fontsize=35)
22 plt.yticks(fontsize=35)
23 plt.title('Qual Raça os pacientes se identificavam', fontsize=95, pad = 30)
24 plt.show()

```




```
In [257]: 1 casos_confirmados['ORIGEM_DA_INFORMACAO'].value_counts()
2
3 #Origem da informação
4
5 from collections import Counter
6 origem_info = Counter(casos_confirmados['ORIGEM_DA_INFORMACAO'])
7 origem_info
8
9 from matplotlib import pyplot as plt
10 import numpy as np
11
12 plt.style.use('seaborn-pastel')
13 plt.pie(origem_info.values(), labels = origem_info.keys(),
14 autopct = '%1.1f%%', textprops = {'fontsize':15})
15 plt.title('Origem da informação', fontsize=18, pad = 40)
16 plt.axis('image')
17 plt.show()
```

Origem da informação



Análise: Mortes por Covid-19

```
In [306]: 1 casos_e_mortes_confirmadas.to_excel('C:\\Users\\Gustavo Nassau\\Desktop\\files tcc\\df3.xlsx')
```

```
In [301]: 1 casos_e_mortes_confirmadas.describe().round(2)
```

```
Out[301]:
```

| | death_rate |
|-------|------------|
| count | 854.00 |
| mean | 3.53 |
| std | 9.21 |
| min | 0.00 |
| 25% | 0.00 |
| 50% | 0.14 |
| 75% | 2.14 |
| max | 76.12 |

```
In [287]: 1 casos_e_mortes_confirmadas['city'].value_counts()
```

```
Out[287]: Importados/Indefinidos    10385
Bom Jesus                          2830
São Domingos                       2508
Planalto                           2078
Santa Helena                       2048
...
Grupiara                           317
Guidoval                           317
Gurinhata                           317
Ibertioga                           317
Dona Euzébia                       317
Name: city, Length: 5298, dtype: int64
```