**PONTIFÍCIA UNIVERSIDADE CATÓLICA DE MINAS GERAIS**
**NÚCLEO DE EDUCAÇÃO A DISTÂNCIA**
**Pós-graduação *Lato Sensu* em Ciência de Dados e Big Data**

**Gabriel Luis Rodrigues**

# PRÁTICA DE ATIVIDADE FÍSICA ENTRE ADULTOS BRASILEIROS NAS CAPITAIS

Belo Horizonte
2022

**Gabriel Luis Rodrigues**

# PRÁTICA DE ATIVIDADE FÍSICA ENTRE ADULTOS BRASILEIROS NAS CAPITAIS

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

## 1.1. Contextualização

A prática de atividade física é muito importante para o fortalecimento do corpo e também pode oferecer benefícios psicológicos, como promover a sensação de bem-estar. A redução dos níveis de atividade física e o aumento do tempo gasto em atividades sedentárias pode fazer com que a população desenvolva doenças cardiovasculares, respiratórias, metabólicas, musculoesqueléticas, câncer e depressão .

Por outro lado, a realização de atividade física é considerada como uma estratégia para a prevenção e gestão de doenças crônicas e ainda para a melhora da saúde mental, reduzindo o risco de depressão e deficiência cognitiva e melhorando e elevando a autoestima.

No Brasil, a inatividade física é responsável por 3% a 5% de todas as principais DCNT (Doenças crônicas não transmissíveis) e 5,31% de todas as causas de mortalidade, variando de 5,82% na região sudeste a 2,83% na região sul (Silva et al., 2017).

De acordo com a OMS, um em cada quatro adultos no mundo não atinge os níveis globais recomendados de atividade física, o que reflete nos cinco milhões de mortes por ano que poderiam ser evitadas se a população global fosse mais ativa (World Health Organization, 2018). O cumprimento ou não das recomendações sobre a prática da atividade física é uma variável investigada em diversos estudos e em diferentes países. Por exemplo, no Brasil, a taxa de sedentarismo aumentou desde 2002 em mais de 15%; e dados indicam que mais de 47% dos brasileiros em 2016 eram sedentários (Guthold, Stevens, Riley, & Bull, 2018).

A prática da atividade física é descrita como um dos principais fatores comportamentais de proteção das Doenças Crônicas, pela redução do risco de mortalidade por todas as causas, prevenção de doenças cardiovasculares (DCV) e diabetes,melhoria dos níveis de lipídios, redução da hipertensão e dos riscos de câncer de mama e cólon (Bull, Goenka, Lambert, & Pratt, 2017). De acordo com esses autores citados, ainda tem efeitos positivos na saúde mental, retarda o início da demência e pode ajudar na manutenção de um peso saudável.

## 1.2. O problema proposto

Neste trabalho, será utilizado Análise Exploratória e Modelagem Preditiva, para extração de informações importantes das práticas de atividade física, com o objetivo de realizar uma análise e encontrar padrões para classificá-lo “SIM” ou “Não” para prática de atividade fisicas. Para isso, todos os atributos serão classificados com um grau de importância. Desta maneira, conseguimos analisar os resultados e utilizá-los em predições futuras.

Serão analisados, os dados de prática de atividade física, disponibilizados (VIGITEL) e IBGE. Os principais objetivos dessa análise são:

- Realizar uma análise nos dados das práticas de atividade física por capitais. Desta forma, auxiliaremos para ver qual atividade mais praticada em determinada região e idade do praticante.

- Os dados que serão analisados, foram coletados do site da (VIGITEL) e IBGE. Foi necessário coletar algumas informações separadamente, são elas:

1. *Dataset das práticas de atividade física:* neste dataset são apresentadas informações sobre atividade física; contendo informações como: se o entrevistado pratica alguma atividade, tipo de atividade, altura e peso do entrevistado, tempo que se dedica a prática da atividade, região geográfica do entrevistado, bem como sua idade, gênero e escolaridade.

2. *Dataset dos valores Demográficos por Capital*: neste dataset é apresentado o valor demográfico da capital, como população estimada, sigla dos estados, renda per capita, escolaridade, IDH, região etcs...

- As análises realizadas, têm como objetivo encontrar padrões, métricas e tendências que auxiliarão no entendimento das bases trabalhadas. E assim, poderemos indicar quais características da atividade física praticada, gênero, tipo da prática e o tempo dedicado, para fins de entender quais são as atividades praticadas por região e o tempo da prática.

[1] http://svs.aids.gov.br/download/Vigitel/

[2] https://cidades.ibge.gov.br/brasil/

VIGITEL(Sistema de Vigilância de Fatores de Risco e Proteção para Doenças Crônicas por Inquérito Telefônico)

## 2. Coleta de Dados

Para o tratamento do problema proposto, foram utilizados cinco *datasets* (conjunto de dados). O processo amostral utilizado no VIGITEL foi do tipo probabilístico e foi constituído por sorteio de 5.000 residências com linha telefônica fixa por cidade, seguido de sorteio de um morador com idade >18 anos por domicílio até se obter o número mínimo de 2.000 entrevistas por cidade. O número de entrevistas completas realizadas pelo VIGITEL a cada ano foi: 53.210 em 2016, 53.034 em 2017, 52.395 em 2018 e 52.443 entrevistas em 2019, totalizando 211.082 no período, extraídos em 05/03/2022 do repositório de dados Secretaria de Vigilância em Saúde. E o dataset sobre informações demográficas foi extraido do repositório do IBGE na data de 06/03/2022.

Os quatros *dataset*, “Vigitel-2016-peso-rake.xls”, Vigitel-2017-peso-rake.xls”, Vigitel-2018-peso-rake.xls”,e Vigitel-2016-peso-rake.xls” disponível em http://svs.aids.gov.br/download/Vigitel/, traz as informações sobre a prática de atividade física. Os primeiros quatro datasets possuem os seguintes campos conforne tabela a seguir. Usaremos um dicionario de dados(Dicionario-de-dados-Vigitel.xls) para entender o que cada campo representa no dataset e assim renomeamos para os campos abaixo:

| Variável | Tipo | Descrição |
|---|---|---|
| Idade | Numérico | Idade do entrevistado |
| Sexo | Texto | Masculino ou Feminino |
| Grau_escolaridade | Texto | Grau escolaridade: curso primário<br>Admissão, curso ginasial ou ginásio,1º grau ou fundamental ou supletivo de 1º grau<br>2º grau ou colégio ou técnico ou normal ou científico científico ou ensino médio ou supletivo de 2º grau,3º grau ou curso superior<br>pós-graduação (especialização, mestrado, doutorado),nunca estudou, não sabe e não quis responder |
| Peso | Numérico | Peso do entrevistado |
| Altura | Numérico | Altura do entrevistado |
| Prática_exercício | Texto | Sim ou Não |
| Tipo_exercício | Texto | Tipo da Atividade:<br>caminhada (não vale deslocamento para trabalho),caminhada em esteira, corrida (cooper), corrida em esteira<br>musculação, ginástica aeróbica (spinning, step, jump), hidroginástica, ginástica em geral (alongamento, pilates, ioga), natação, artes marciais e luta (jiu-jitsu, karatê, judô, boxe, muay thai, capoeira), bicicleta (inclui ergométrica), futebol/futsal, basquetebol, voleibol/futevolei<br>tênis, dança (balé, dança de salão, dança do ventre) e outros. |
| Pratica_exercicio_1_vez_na_semana | Texto | Sim ou Não |
| Frequencia_exercicio | Texto | Quantidade de vezes praticadas durante a semana:1 a 2 dias por semana,3 a 4 dias por semana,5 a 6 dias por semana e<br>todos os dias ( inclusive sábado e domingo). |
| Duracao_exercicio | Texto | Tempo da pratica da atividade:<br>menos que 10 minutos, entre 10 e 19 minutos,<br>entre 20 e 29 minutos, entre 30 e 39 minutos,<br>entre 40 e 49 minutos, entre 50 e 59 minutos e<br>60 minutos ou mais |
| Cidade | Texto | Nome das capitais da pesquisa |

| | | |
|---|---|---|
| Ano | Inteiro | Ano da pesquisa |
| Ordem | Inteiro | Ordem que a pesquisa foi registrada posição dos entrevistaso inseridos na tabela. |
| Civil | Texto | Estado conjugal atual se a pessoa e: solteiro, casado legalmente, tem união estável há mais de seis meses, viúvo, separado ou divorciado e não quis informar |

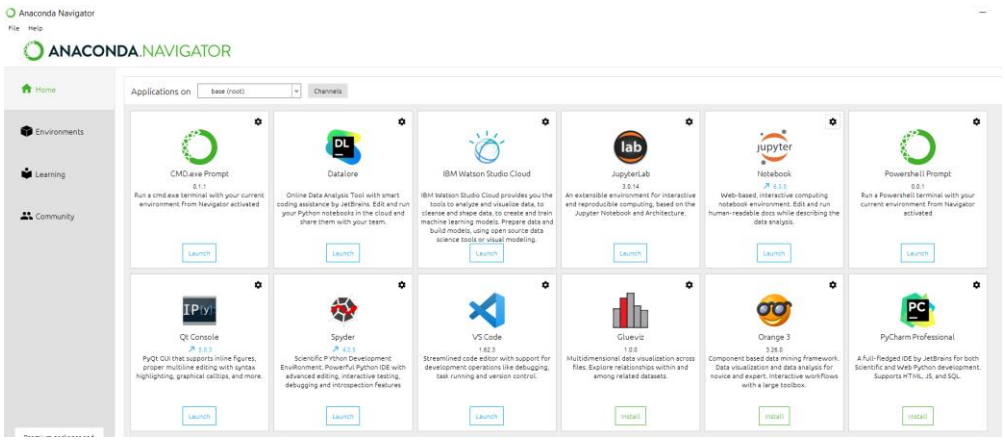
O segundo *dataset*, foi renomeado o arquivo para “Capitais_Senso.xls”, disponível em https://www.ibge.gov.br/exportacao/08ab48cf20f15702bf692cf118ed6c3f.xls?167, traz a informações das capitais como região, renda per capita, nome do estado sigla do estado, numero de habitantes estimada, ano pesquisa, IDH entre outras informações,vamos usar os seguintes campos:

| **Variável** | **Tipo** | **Descrição** |
|---|---|---|
| Cidade | Numérico | Codigo da Cidade |
| UF | Texto | AC,AL,AP,AM,BA,CE,DF,ES,GO,MA,MT,MS,MG,PA,PB,PR,PE,PI,RJ,RN,RS,RO,RR,SC,SP,SE,TO |
| Capital | Texto | Nome da Capital:Rio Branco, Maceió, Macapá, Manaus, Salvador, Fortaleza, Brasília, Vitória, Goiânia, São Luís,Cuiabá, Campo Grande, Belo Horizonte, Belém,João Pessoa,Curitiba, Recife,Teresina, Rio de Janeiro, Natal,Porto Alegre, Porto Velho, Boa Vista, Florianópolis, São Paulo, Aracaju, Palmas |
| Populacao estimada - pessoas [2021] | Numérico | Quantidade da população estimada |
| Pib per capita | Numérico | Produto Interno Bruto da capitais |
| Regiao | Texto | Regiões do Brasil :Norte, Sul, Sudeste, Centro-Oeste e Nordeste |
| Ano | Numérico | Ano da pesquisa |
| Salario_medio_mensal | Numérico | Salário médio mensal dos trabalhadores formais [2019] |
| PIB | Numérico | Produto Interno Bruto por capital valor |

## 3. Processamento/Tratamento de Dados

Nessa seção será apresentado todas as ferramentas e bibliotecas utilizadas para o processamento e o tratamento dos dados. Como ferramenta para desenvolvimento dos scripts em python, foi escolhida a distribuição Anaconda, versão 1.10.0 (figura 1), disponível em https://www.anaconda.com/distribution/.

**Figura 1**: Screenshot do Anaconda Navigator, da distribuição Anaconda

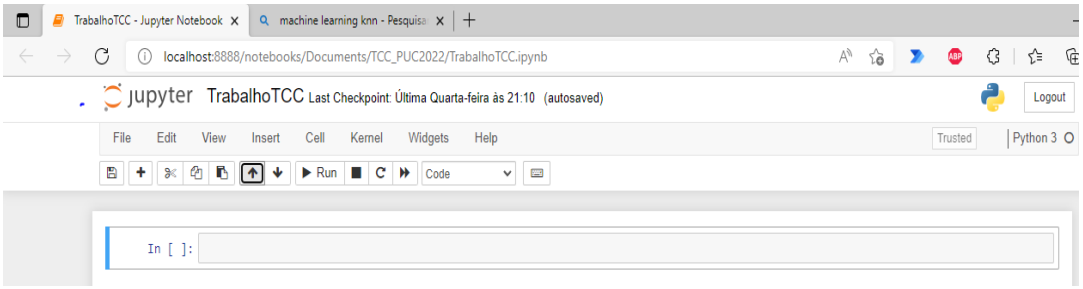



Fonte: Autor

Foi escolhida a distribuição *Anaconda,* pois possui as principais ferramentas e bibliotecas para realizarmos toda a codificação necessária para análise e tratamento dos dados. Desta forma não é necessário realizar a importação dos pacotes separadamente, pois o *Anaconda* já nos fornece tudo em uma só instalação.

Utilizamos o Jupyter Notebook (figura 2) como principal editor deste projeto.

**Figura 2**: Screenshot do editor Jupyter Notebook.




Fonte:Autor

O *Jupyter Notebook* possui o ambiente instalado e configurado com o python 3 (versão utilizada no projeto) incluso ao pacote do *Anaconda*.

Para realizar o processamento e o tratamento dos dados, foi necessário importar algumas bibliotecas conforme a Figura 3 abaixo.

**Figura 3**: Screenshot importação das bibliotecas.

```
In [1]: #Importação da biblioteca pandas
import pandas as pd
import numpy as np

import matplotlib.pyplot as plt
import seaborn as sns
sns.set_style('whitegrid')

from datetime import date
from collections import Counter

from sklearn.model_selection import train_test_split
from sklearn.metrics import mean_squared_error, mean_absolute_error

import seaborn as sb
from matplotlib import pyplot
import statsmodels.api as sm
import math

import operator

import warnings
warnings.filterwarnings('ignore')
```

Fonte: Autor

Em seguida, deve-se fazer a leitura e tratamento dos *Data Frame*, que, nesse caso, serão importados todos de uma vez conforme na Figura abaixo. Será importado por meio do comando “pd.read.csv”, seguindo os parâmetros estabelecidos no arquivo que traz as características do *Data Frame*.

**Figura 4**: Screenshot obtendo dados do arquivo xls.

```
In [2]: ##leitura dos 4 arquivos  XLS que contém os dados para os dataframes;
df_atividade_2016= pd.read_excel('Vigitel-2016-peso-rake.xls')
df_atividade_2017= pd.read_excel('Vigitel-2017-peso-rake.xls')
df_atividade_2018= pd.read_excel('Vigitel-2018-peso-rake.xls')
df_atividade_2019= pd.read_excel('Vigitel-2019-peso-rake.xls')
```

Fonte: Autor

**Figura 5**: Screenshot Concatenando os quatros Data Frame em um unico DF.

```
In [3]: # juntando os dataframes para um unico dataframe
df_atividade_junt = pd.merge(df_atividade_2016, df_atividade_2017, how = 'outer')

In [4]: # juntando os dataframes para um unico dataframe
df_atividade_junt2 = pd.merge(df_atividade_junt, df_atividade_2018, how = 'outer')

In [5]: # juntando os dataframes para um unico dataframe
df_atividade = pd.merge(df_atividade_junt2, df_atividade_2019, how = 'outer')
```

Fonte:Autor

O uso desse comando cria um *Data Frame*, que é uma estrutura bidimensional de dados similar a uma planilha. Para se obter informações do *dataframe*, pode-se utilizar a função “info()” que, no caso específico, mostrou que o *dataframe* possui 211.082 entradas, divididas nas 267 colunas apresentadas anteriormente.

**Figura 5**: Screenshot Informação do Data Frame.

```
9]: #informações do dataframe df_atividade
    df_atividade.info()

    <class 'pandas.core.frame.DataFrame'>
    Int64Index: 211082 entries, 0 to 211081
    Columns: 267 entries, ordem to score_upp_2cat
    dtypes: float64(194), int64(70), object(3)
    memory usage: 431.6+ MB
```

Fonte: Autor

Na Figura abaixo, é apresentado  o Data Frame que traz informações das cidades dos entrevistados; mas, como o objetivo do estudo é avaliar o resultado de atividade fisica para cada capital, foi necessário, antes de selecionar apenas as entradas de interesse, identificar quais eram as códigos de cidades que que constavam no arquivo, o que foi feito por meio da função “unique()”, que mostra quais são as opções existentes na coluna “cidade”, que traz a cidade onde a pesquisa foi realizada.

**Figura 6**: Screenshot Informação codigo da Cidade.

```
In [10]: #identificação das opções da coluna cidade do dataframe df_atividade
         df_atividade['cidade'].unique()

Out[10]: array([ 1,  3,  5,  4,  6,  7, 27,  8,  9, 10, 15, 12, 11, 13, 17, 16, 18,
                19, 22, 20, 21, 23, 26, 24, 25,  2, 14], dtype=int64)
```

Fonte: Autor

A seguir é apresentado o resultado em Data Frame dos dados coletados, selecionando apenas os 1001 primeiros registros encontrados através do

comando *df_atividade.head(1001)*.

**Figura 7**: Screenshot exibindo dados dataset do df_atividade.

```
In [11]: #Exibe as 1001 primeiras linhas
         df_atividade.head(1001)
```

Out[11]:

| | ordem | replica | ano | cidade | q6 | q7 | civil | q8a | q8b | q8_anos | ... | r175 | r176 | r901 | r901_ou | r902 | eletronico | score_sf | score_sf_2cat | score_upp | sco |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 0 | 1.0 | 1 | 2016 | 1 | 60 | 2 | 5 | 5 | 3.0 | 11 | ... | NaN | NaN | NaN | NaN | NaN | NaN | NaN | NaN | NaN | |
| 1 | 2.0 | 1 | 2016 | 3 | 25 | 2 | 1 | 6 | 3.0 | 14 | ... | NaN | NaN | NaN | NaN | NaN | NaN | NaN | NaN | NaN | |
| 2 | 3.0 | 1 | 2016 | 1 | 65 | 2 | 2 | 4 | 6.0 | 6 | ... | NaN | NaN | NaN | NaN | NaN | NaN | NaN | NaN | NaN | |
| 3 | 4.0 | 1 | 2016 | 1 | 42 | 2 | 2 | 5 | 3.0 | 11 | ... | NaN | NaN | NaN | NaN | NaN | NaN | NaN | NaN | NaN | |
| 4 | 5.0 | 1 | 2016 | 1 | 27 | 1 | 1 | 6 | 8.0 | 19 | ... | NaN | NaN | NaN | NaN | NaN | NaN | NaN | NaN | NaN | |
| ... | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... | |
| 996 | 997.0 | 1 | 2016 | 26 | 73 | 2 | 4 | 1 | 2.0 | 2 | ... | NaN | NaN | NaN | NaN | NaN | NaN | NaN | NaN | NaN | |
| 997 | 998.0 | 1 | 2016 | 23 | 59 | 2 | 3 | 5 | 3.0 | 11 | ... | NaN | NaN | NaN | NaN | NaN | NaN | NaN | NaN | NaN | |
| 998 | 999.0 | 1 | 2016 | 22 | 29 | 2 | 1 | 6 | 1.0 | 12 | ... | NaN | NaN | NaN | NaN | NaN | NaN | NaN | NaN | NaN | |
| 999 | 1000.0 | 1 | 2016 | 24 | 59 | 2 | 4 | 4 | 8.0 | 8 | ... | NaN | NaN | NaN | NaN | NaN | NaN | NaN | NaN | NaN | |
| 1000 | 1001.0 | 1 | 2016 | 24 | 60 | 1 | 2 | 6 | 6.0 | 17 | ... | NaN | NaN | NaN | NaN | NaN | NaN | NaN | NaN | NaN | |

1001 rows × 267 columns

Fonte: Autor

Na Figura abaixo vamos verificar no *dataframe* a presença de campos nulos:

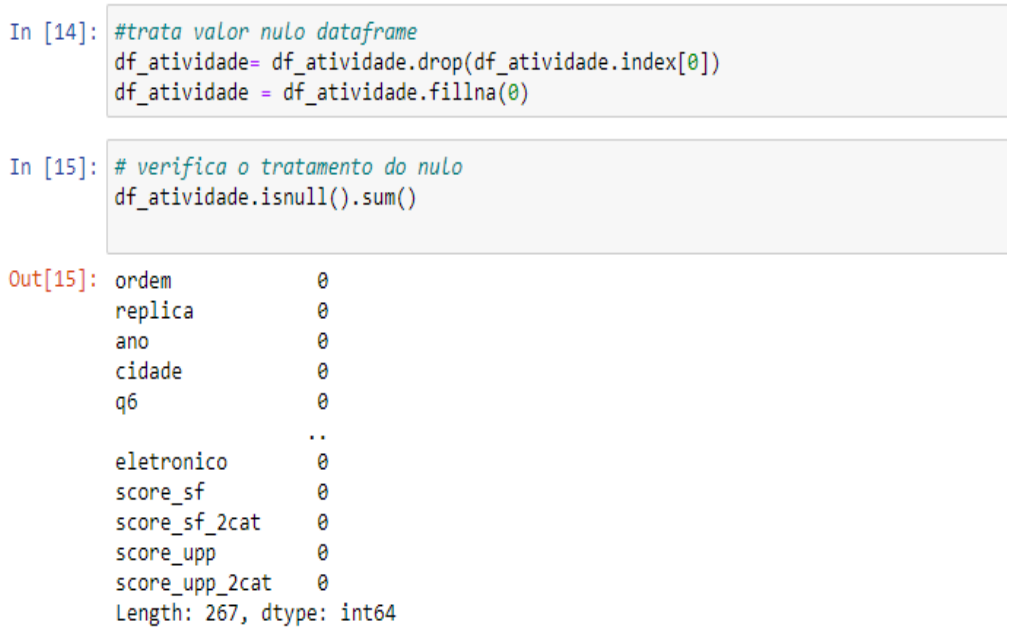
**Figura 8**: Screenshot exibindo campos nulos no Data Frame df_atividade.

```
In [13]: #Verificação dos valores nulos no dataframe df_atividade
         df_atividade.isnull().sum()

Out[13]: ordem             157872
         replica                0
         ano                    0
         cidade                 0
         q6                     0
                               ...
         eletronico        158639
         score_sf          158639
         score_sf_2cat     158639
         score_upp         158639
         score_upp_2cat    158639
         Length: 267, dtype: int64
```

Fonte: Autor

Para certificar que os dados estão padronizados e que não existem valores nulos no DataFrame, foi executado a função isnull() onde recupera todos os registros que estão com valores nulo e logo em seguida aplicado a função sum(), pois somamos a quantidade de registros nulos da coluna correspondente, conforme a Figura 9.

**Figura 9**: Screenshot Tratando campos nulos no Data Frame df_atividade.

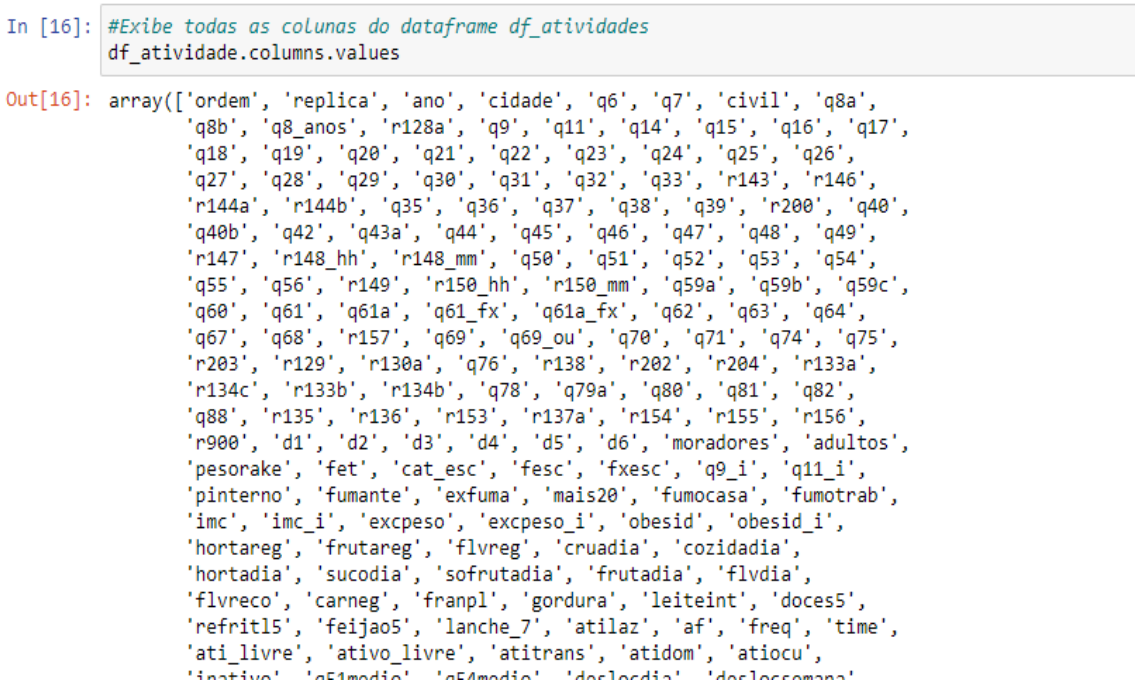


```
In [14]: #trata valor nulo dataframe
         df_atividade= df_atividade.drop(df_atividade.index[0])
         df_atividade = df_atividade.fillna(0)

In [15]: # verifica o tratamento do nulo
         df_atividade.isnull().sum()

Out[15]: ordem            0
         replica          0
         ano              0
         cidade           0
         q6               0
                         ..
         eletronico       0
         score_sf         0
         score_sf_2cat    0
         score_upp        0
         score_upp_2cat   0
         Length: 267, dtype: int64
```

Fonte: Autor

Na Figura abaixo vamos apresentar todas as colunas do Data Frame df_atividade.

**Figura 10**: Screenshot Tratando campos nulos no Data Frame df_atividade.



```
In [16]: #Exibe todas as colunas do dataframe df_atividades
         df_atividade.columns.values

Out[16]: array(['ordem', 'replica', 'ano', 'cidade', 'q6', 'q7', 'civil', 'q8a',
                'q8b', 'q8_anos', 'r128a', 'q9', 'q11', 'q14', 'q15', 'q16', 'q17',
                'q18', 'q19', 'q20', 'q21', 'q22', 'q23', 'q24', 'q25', 'q26',
                'q27', 'q28', 'q29', 'q30', 'q31', 'q32', 'q33', 'r143', 'r146',
                'r144a', 'r144b', 'q35', 'q36', 'q37', 'q38', 'q39', 'r200', 'q40',
                'q40b', 'q42', 'q43a', 'q44', 'q45', 'q46', 'q47', 'q48', 'q49',
                'r147', 'r148_hh', 'r148_mm', 'q50', 'q51', 'q52', 'q53', 'q54',
                'q55', 'q56', 'r149', 'r150_hh', 'r150_mm', 'q59a', 'q59b', 'q59c',
                'q60', 'q61', 'q61a', 'q61_fx', 'q61a_fx', 'q62', 'q63', 'q64',
                'q67', 'q68', 'r157', 'q69', 'q69_ou', 'q70', 'q71', 'q74', 'q75',
                'r203', 'r129', 'r130a', 'q76', 'r138', 'r202', 'r204', 'r133a',
                'r134c', 'r133b', 'r134b', 'q78', 'q79a', 'q80', 'q81', 'q82',
                'q88', 'r135', 'r136', 'r153', 'r137a', 'r154', 'r155', 'r156',
                'r900', 'd1', 'd2', 'd3', 'd4', 'd5', 'd6', 'moradores', 'adultos',
                'pesorake', 'fet', 'cat_esc', 'fesc', 'fxesc', 'q9_i', 'q11_i',
                'pinterno', 'fumante', 'exfuma', 'mais20', 'fumocasa', 'fumotrab',
                'imc', 'imc_i', 'excpeso', 'excpeso_i', 'obesid', 'obesid_i',
                'hortareg', 'frutareg', 'flvreg', 'cruadia', 'cozidadia',
                'hortadia', 'sucodia', 'sofrutadia', 'frutadia', 'flvdia',
                'flvreco', 'carneg', 'franpl', 'gordura', 'leiteint', 'doces5',
                'refritl5', 'feijao5', 'lanche_7', 'atilaz', 'af', 'freq', 'time',
                'ati_livre', 'ativo_livre', 'atitrans', 'atidom', 'atiocu',
                'inativo', 'q51medio', 'q54medio', 'deslocdia', 'deslocsemana'
```

Fonte: Autor

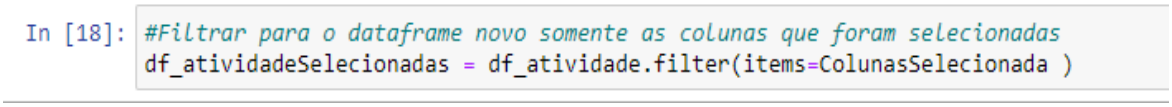
Na Figura abaixo vamos selecionar as colunas que serão usadas na analise.

**Figura 11**: Screenshot colunas que vão ser usadas na analise.


Fonte: Autor

Na Figura abaixo vamos filtrar as colunas para um novo Data Frame.

**Figura 12**: Screenshot Criação de um novo Data Frame com base na colunas filtradas.




Fonte: Autor

**Figura 13**: Screenshot exibe informações das colunas de um Data Frame criado.

In [19]:
```
#Exibe Dataframe df_atividadeSelecionadas
df_atividadeSelecionadas.head()
```

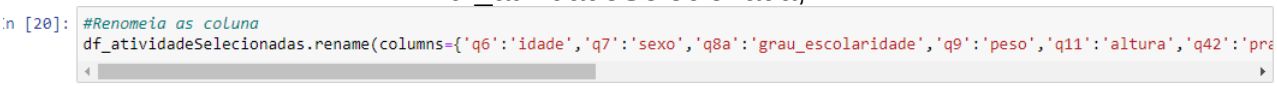
Out[19]:

| | ordem | ano | cidade | civil | q6 | q7 | q8a | q9 | q11 | q42 | q43a | q44 | q45 | q46 | q69 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 1 | 2.0 | 2016 | 3 | 1 | 25 | 2 | 6 | 60.0 | 157 | 1 | 3.0 | 1.0 | 2.0 | 7.0 | 1 |
| 2 | 3.0 | 2016 | 1 | 2 | 65 | 2 | 4 | 57.0 | 152 | 1 | 5.0 | 1.0 | 3.0 | 7.0 | 4 |
| 3 | 4.0 | 2016 | 1 | 2 | 42 | 2 | 5 | 52.0 | 156 | 2 | 0.0 | 0.0 | 0.0 | 0.0 | 1 |
| 4 | 5.0 | 2016 | 1 | 1 | 27 | 1 | 6 | 100.0 | 179 | 1 | 12.0 | 1.0 | 1.0 | 7.0 | 4 |
| 5 | 6.0 | 2016 | 3 | 2 | 50 | 2 | 5 | 62.0 | 153 | 2 | 0.0 | 0.0 | 0.0 | 0.0 | 1 |

Fonte: Autor

Para deixar nossos dados formatados e padronizados, algumas colunas foram renomeadas utilizando a função rename(), conforme a Figura 15 abaixo.

**Figura 14**: Screenshot renomeando colunas DataFrame: df_atividadeSelecionada;



Fonte: Autor

Na Figura abaixo criar coluna nova no Data Frame com calculo do IMC.

**Figura 15**: Screenshot criar coluna IMC no Data Frame

```
In [21]: #criar coluna com o calculo IMC dataframe df_atividadeSelecionadas
df_atividadeSelecionadas['imc'] = df_atividadeSelecionadas.apply(
    lambda row: round(row.peso / (((row.altura * row.altura)/1000)*0.1),0), axis=1)
```

Fonte: Autor

**Figura 16**: Screenshot criar novo Data Frame para ser usado analise do ano 2019

```
In [22]: df_censoSelecionadas2019Final = df_atividadeSelecionadas.loc[df_atividadeSelecionadas["ano"]== 2019]
```

Fonte: Autor

**Figura 17**: Screenshot substituindo valores da coluna com base no dicionario de dados.

```
In [24]: #formate a coluna pratica exercicio 1 para sim e 2 para não
df_atividadeSelecionadas['pratica_exercicio'].replace(1,'sim', inplace =True)
df_atividadeSelecionadas['pratica_exercicio'].replace(2,'nao', inplace =True)

#formate a coluna tipo exercicio
#1 caminhada (não vale deslocamento para trabalho)
#2 caminhada em esteira
#3 corrida (cooper)
#4 corrida em esteira
#5 musculação
#6 ginástica aeróbica (spinning, step, jump)
#7 hidroginástica
#8 ginástica em geral (alongamento, pilates, ioga)
#9 natação
#10 artes marciais e luta (jiu-jitsu, karatê, judô, boxe, muay thai, capoeira)
#11 bicicleta (inclui ergométrica)
#12 futebol/futsal
#13 basquetebol
#14 voleibol/futevolei
#15 tênis
#16 dança (balé, dança de salão, dança do ventre)
#17 outros
df_atividadeSelecionadas['tipo_exercicio'].replace(1,'caminhada (não vale deslocamento para trabalho)', inplace =True)
df_atividadeSelecionadas['tipo_exercicio'].replace(2,'caminhada em esteira', inplace =True)
df_atividadeSelecionadas['tipo_exercicio'].replace(3,'corrida (cooper)', inplace =True)
df_atividadeSelecionadas['tipo_exercicio'].replace(4,'corrida em esteira', inplace =True)
df_atividadeSelecionadas['tipo_exercicio'].replace(5,'musculação', inplace =True)
df_atividadeSelecionadas['tipo_exercicio'].replace(6,'ginástica aeróbica (spinning, step, jump)', inplace =True)
df_atividadeSelecionadas['tipo_exercicio'].replace(7,'hidroginástica', inplace =True)
df_atividadeSelecionadas['tipo_exercicio'].replace(8,'ginástica em geral (alongamento, pilates, ioga)', inplace =True)
df_atividadeSelecionadas['tipo_exercicio'].replace(9,'natação', inplace =True)
df_atividadeSelecionadas['tipo_exercicio'].replace(10,'artes marciais e luta (jiu-jitsu, karatê, judô, boxe, muay thai, capoeira
df_atividadeSelecionadas['tipo_exercicio'].replace(11,'bicicleta (inclui ergométrica)', inplace =True)
df_atividadeSelecionadas['tipo_exercicio'].replace(12,'futebol/futsal', inplace =True)
df_atividadeSelecionadas['tipo_exercicio'].replace(13,'basquetebol', inplace =True)
df_atividadeSelecionadas['tipo_exercicio'].replace(14,'voleibol/futevolei', inplace =True)
df_atividadeSelecionadas['tipo_exercicio'].replace(15,'tênis', inplace =True)
df_atividadeSelecionadas['tipo_exercicio'].replace(16,'dança (balé, dança de salão, dança do ventre)', inplace =True)
df_atividadeSelecionadas['tipo_exercicio'].replace(17,'outros', inplace =True)
```

Fonte: Autor

**Figura 18**: Screenshot substituindo valores da coluna com base no dicionario de dados.

```
In [25]: #formate a coluna grau escolaridade
#1 ---> curso primário
#2 ---> admissão
#3 ---> curso ginasial ou ginásio
#4 ---> 1º grau ou fundamental ou supletivo de 1º grau
#5 ---> 2º grau ou colégio ou técnico ou normal ou científico científico ou ensino médio ou supletivo de 2º grau
#6 ---> 3º grau ou curso superior
#7 ---> pós-graduação (especialização, mestrado, doutorado)
#8 ---> nunca estudou
#777 ---> não sabe
#888 ---> não quis responder

df_atividadeSelecionadas['grau_escolaridade'].replace(1,'curso primário', inplace =True)
df_atividadeSelecionadas['grau_escolaridade'].replace(2,'admissão', inplace =True)
df_atividadeSelecionadas['grau_escolaridade'].replace(3,'curso ginasial ou ginásio', inplace =True)
df_atividadeSelecionadas['grau_escolaridade'].replace(4,'1º grau ou fundamental ou supletivo de 1º grau', inplace =True)
df_atividadeSelecionadas['grau_escolaridade'].replace(5,'2º grau ou colégio ou técnico ou normal ou científico científico ou ens
df_atividadeSelecionadas['grau_escolaridade'].replace(6,'3º grau ou curso superior', inplace =True)
df_atividadeSelecionadas['grau_escolaridade'].replace(7,'pós-graduação (especialização, mestrado, doutorado)', inplace =True)
df_atividadeSelecionadas['grau_escolaridade'].replace(8,'nunca estudou', inplace =True)
df_atividadeSelecionadas['grau_escolaridade'].replace(777,'não sabe', inplace =True)
df_atividadeSelecionadas['grau_escolaridade'].replace(888,'não quis responder', inplace =True)
```

Fonte: Autor

**Figura 19**: Screenshot substituindo valores da coluna com base no dicionário de dados.

```
In [26]: #formate a coluna quantas vezes semana' 1 para sim e 2 para não
df_atividadeSelecionadas['pratica_exercicio_1_vez_na_semana'].replace(1,'sim', inplace =True)
df_atividadeSelecionadas['pratica_exercicio_1_vez_na_semana'].replace(2,'nao', inplace =True)

#formate a coluna Estado Civil

df_atividadeSelecionadas['civil'].replace(1,'solteiro', inplace =True)
df_atividadeSelecionadas['civil'].replace(2,'casado legalmente', inplace =True)
df_atividadeSelecionadas['civil'].replace(3,'tem união estável há mais de seis meses', inplace =True)
df_atividadeSelecionadas['civil'].replace(4,'viúvo', inplace =True)
df_atividadeSelecionadas['civil'].replace(5,'separado ou divorciado', inplace =True)
df_atividadeSelecionadas['civil'].replace(888,'não quis informar', inplace =True)


#formate a coluna frequencia exercicio
df_atividadeSelecionadas['frequencia_exercicio'].replace(1,'1 a 2 dias por semana', inplace =True)
df_atividadeSelecionadas['frequencia_exercicio'].replace(2,'3 a 4 dias por semana', inplace =True)
df_atividadeSelecionadas['frequencia_exercicio'].replace(3,'5 a 6 dias por semana', inplace =True)
df_atividadeSelecionadas['frequencia_exercicio'].replace(4,'todos os dias ( inclusive sábado e domingo)', inplace =True)


#formate a coluna duracao exercicio
df_atividadeSelecionadas['duracao_exercicio'].replace(1,'menos que 10 minutos', inplace =True)
df_atividadeSelecionadas['duracao_exercicio'].replace(2,'entre 10 e 19 minutos', inplace =True)
df_atividadeSelecionadas['duracao_exercicio'].replace(3,'entre 20 e 29 minutos', inplace =True)
df_atividadeSelecionadas['duracao_exercicio'].replace(4,'entre 30 e 39 minutos', inplace =True)
df_atividadeSelecionadas['duracao_exercicio'].replace(5,'entre 40 e 49 minutos', inplace =True)
df_atividadeSelecionadas['duracao_exercicio'].replace(6,'entre 50 e 59 minutos', inplace =True)
df_atividadeSelecionadas['duracao_exercicio'].replace(7,'60 minutos ou mais', inplace =True)


#formate a coluna cor
df_atividadeSelecionadas['cor'].replace(1,'branca', inplace =True)
df_atividadeSelecionadas['cor'].replace(2,'preta', inplace =True)
df_atividadeSelecionadas['cor'].replace(3,'amarela', inplace =True)
df_atividadeSelecionadas['cor'].replace(4,'parda', inplace =True)
df_atividadeSelecionadas['cor'].replace(5,'indígena', inplace =True)
df_atividadeSelecionadas['cor'].replace(80,'Morena', inplace =True)
df_atividadeSelecionadas['cor'].replace(777,'não sabe', inplace =True)
df_atividadeSelecionadas['cor'].replace(888,'não quis informar', inplace =True)
```

Fonte: Autor

**Figura 20**: Screenshot substituindo valores da coluna com base no dicionário de dados.

```
#formate a coluna quantas vezes semana' 1 para masculino e 2 para feminino
df_atividadeSelecionadas['sexo'].replace(1,'masculino', inplace =True)
df_atividadeSelecionadas['sexo'].replace(2,'feminino', inplace =True)
```

Fonte: Autor

Na Figura abaixo vamos obter o segundo DataFrame que vai ser usado na análise do ano de 2019. Esse Data frame vai conter informações demograficas das capitais Brasileiras.

**Figura 21**: Screenshot criação de um novo Data Frame contendo informações sobre as capitais Brasileiras.

```
[33]: #Tratar dados da Tabela Censo ano 2019 que vai ser usada para extrair informacões da Cidade como renda per capita, nome estado,
      df_censo= pd.read_excel('Capitais_Censo.xls')
```

Fonte: Autor

A seguir, é apresentado o resultado em Data Frame dos dados coletados, selecionando apenas os 5 primeiros registros encontrados através do comando df_censo().

**Figura 22**: Screenshot exibindo dados dataset do df_censo

```
In [36]: #exibe dados do segundo Dataframe df_censo demogradico
         df_censo.head()
```

Out[36]:

| | estado | cidade | uf | capital | prefeito | area territorial - km² [2021] | populacao estimada - pessoas [2021] | densidade demográfica - hab/km² [2010] | IDH | receitas realizadas - R$ (×1000) [2017] | despesas empenhadas - R$ (×1000) [2017] | pib per capita | regiao | ano | escolarizacao 6 a 14 anos |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 0 | Acre | 20 | AC | Rio Branco | SEBASTIÃO BOCALOM RODRIGUES | 8835.154 | 419452 | 38.03 | 0.727 | 884827.27 | 740733.11 | 22448.30 | Norte | 2019 | 95.1 |
| 1 | Alagoas | 13 | AL | Maceió | JOÃO HENRIQUE HOLANDA CALDAS | 509.321 | 1031597 | 1854.10 | 0.721 | 2341738.83 | 2223470.58 | 22976.51 | Nordeste | 2019 | 95.0 |
| 2 | Amapá | 12 | AP | Macapá | ANTONIO PAULO DE OLIVEIRA FURLAN | 6563.849 | 522357 | 62.14 | 0.733 | 836874.67 | 730153.48 | 22718.28 | Norte | 2019 | 94.8 |
| 3 | Amazonas | 14 | AM | Manaus | DAVID ANTONIO ABISAI PEREIRA DE ALMEIDA | 11401.092 | 2255903 | 158.06 | 0.737 | 4743520.97 | 4743520.97 | 38880.73 | Norte | 2019 | 94.2 |
| 4 | Bahia | 22 | BA | Salvador | BRUNO SOARES | 693.453 | 2900319 | 3859.44 | 0.759 | 8270102.75 | 6874146.32 | 22213.24 | Nordeste | 2019 | 95.0 |

Fonte: Autor

**Figura 23**: Screenshot exibe os nomes das colunas do Data Frame df_censo

```
In [37]: #Mostrar todas as colunas do dataframe df_censo demografico
         df_censo.columns.values

Out[37]: array(['estado', 'cidade', 'uf', 'capital', 'prefeito',
                'area territorial - km² [2021]',
                'populacao estimada - pessoas [2021]',
                'densidade demográfica - hab/km² [2010]', 'IDH ',
                'receitas realizadas - R$ (×1000) [2017]',
                'despesas empenhadas - R$ (×1000) [2017]', 'pib per capita',
                'regiao', 'ano', 'escolarizacao 6 a 14 anos '], dtype=object)
```

Fonte: Autor

Na Figura abaixo vamos selecionar as colunas que serão usadas na analise.

**Figura 23**: Screenshot colunas que vão ser usadas na analise.

```
[38]: #Criar lista das colunas que vão ser usadas Censo
      ColunasSelecionadaCenso = ['cidade', 'uf', 'capital', 'populacao estimada - pessoas [2021]', 'pib per capita', 'regiao', 'ano']
```

Fonte: Autor

Na Figura abaixo vamos filtrar as colunas para um novo Data Frame.

**Figura 24**: Screenshot Criação de um novo Data Frame com base na colunas filtradas.

```
:  #Filtra para o dataframe novo somente as colunas que foram selecionadas
   df_censoSelecionadas = df_censo.filter(items=ColunasSelecionadaCenso )
```

Fonte: Autor

**Figura 25**: Screenshot exibe informações das colunas de um Data Frame criado.

```
In [40]: # Exibe linhas do Dataset
         df_censoSelecionadas.head()
```

Out[40]:

| | cidade | uf | capital | populacao estimada - pessoas [2021] | pib per capita | regiao | ano |
|---|---|---|---|---|---|---|---|
| 0 | 20 | AC | Rio Branco | 419452 | 22448.30 | Norte | 2019 |
| 1 | 13 | AL | Maceió | 1031597 | 22976.51 | Nordeste | 2019 |
| 2 | 12 | AP | Macapá | 522357 | 22718.28 | Norte | 2019 |
| 3 | 14 | AM | Manaus | 2255903 | 38880.73 | Norte | 2019 |
| 4 | 22 | BA | Salvador | 2900319 | 22213.24 | Nordeste | 2019 |

Fonte: Autor

**Figura 26**: Screenshot renomeando colunas DataFrame: df_censoSelecionadas;

```
41]: #Renomeia as coluna
df_censoSelecionadas.rename(columns={'capital':'nome_capital', 'populacao estimada - pessoas [2021]': 'populacao_estimada_pess
```

Fonte: Autor

Na Figura abaixo, estamos tratando o Data Frame. Estamos filtrando o ano de 2019 para que possamos usar esse Data Frame para criar análise para o ano de 2019 e também criar o modelo de Machine Learning.

**Figura 27**: Screenshot Concatenando oData Frame em um unico DF.

```
##Unindo os datasets df_censoSelecionadas com df_atividadeSelecionada2019
###
###
df_atividadeSelecionada2019 = df_atividadeSelecionadas.loc[df_atividadeSelecionadas["ano"]== 2019]

df_censoSelecionadas2019 = pd.merge(df_atividadeSelecionada2019, df_censoSelecionadas, how = 'outer')

df_censoSelecionadas2019Final = pd.merge(df_censoSelecionadas2019Final, df_censoSelecionadas, how = 'outer')
```

Fonte:Autor

Foi utilizado a função *shape(),* onde podemos visualizar as dimensões doDataFrame.

**Figura 28**: Screenshot - Informações dimensões do DataFrame

```
5]: df_atividadeSelecionadas.shape

5]: (211081, 16)
```

Fonte: Autor

Neste caso obtemos como resultado 211.081 linhas e 16 colunas.

## 4. Análise e Exploração dos Dados

Para a realização da análise e exploração de dados, será utilizada a função “Counter” do módulo “Collections”, que conta quantas vezes uma determinada opção aparece em uma série de dados e armazena esses valores em um dicionário, a biblioteca “Matplotlib” para a criação de gráficos e, novamente, a biblioteca “pandas”.

Inicialmente, será feita a análise dos dados dos dataframes de maneira individual e, em seguida, as junções necessárias para continuação dos estudos. Portanto, as análises serão iniciadas pelo dataframe que traz as informações sobre a entrevistas realizadas para identificar o  número de praticante de atividade fisica no Data Frame df_atividadeSelecionadas.

Como um dos objetivos é avaliar a discrepância de característica entre os praticantes de atividade fisica com os não praticantes, usaremos um Data Frame contendo dados  para o ano de 2016, 2017, 2018, 2019  e depois  vamos fazer uma análise descritiva  mais profunda no ano de 2019  para tentar identificar alguns padrões.

A coluna Prática exercício(prática_exercicio), identifica-se o número de entrevistados que praticam exercicio ou não no intervalo de 2016, 2017, 2018 e 2019.

**Figura 29**: Screenshot -  Agrupamento se pratica ou nã atividade fisica.

```
[139]: #verifica o total se pratica ou não Atividades Fisicas
       pequisa = Counter(df_atividadeSelecionadas['pratica_exercicio'])
       pequisa

[139]: Counter({'sim': 117987, 'nao': 93094})
```

Fonte: Autor

Podemos verificar que mais da metade  dos entrevistados praticam atividades física no momento de folga,  sendo 117.987 praticante de algum tipo de atividade física e 93.094 que não praticam nenhum tipo de atividade. Para gerar os gráficos com seus títulos e porcentagens foram utilizados os comandos  na Figura abaixo:

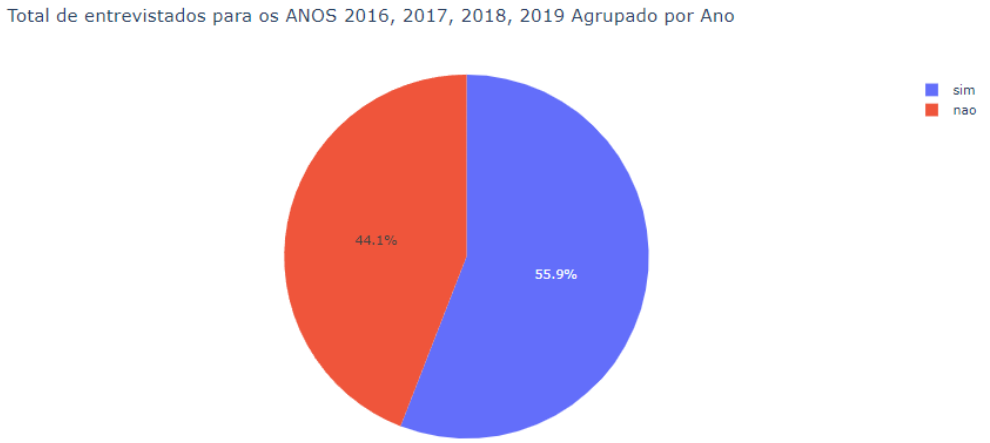
**Figura 30**: Screenshot - Codigo da plotagem do grafico agrupado por Pratica Atividade SIM ou NÃo.

```
In [144]: #Plotagem das informações de gênero dos Entrevistados

          import plotly.express as px
          pie = df_atividadeSelecionadas['pratica_exercicio'].value_counts()
          regions = pie.index
          population = pie.values
          fig = px.pie(df_atividadeSelecionadas, values=population, names=regions, title ="Total de entrevistados para os ANOS 2016,
          fig.show()
```

Fonte: Autor

Na figura abaixo podemos ver isso na forma de um grafico de pizza mostrando o resultado em porcentagem.

**Figura 31**: Screenshot - Plotagem do grafico Agrupado por se pratica ou não atividade fisica.




Fonte: Autor

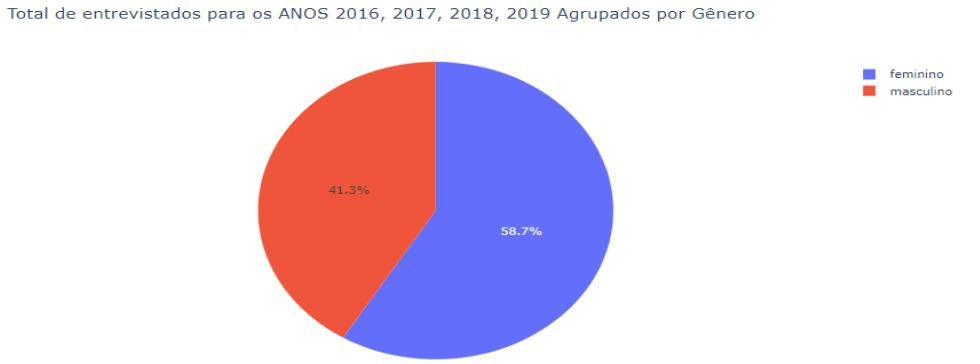
Analisando os gráficos pode-se perceber que o número de entrevistados de cada gênero já é proporcional a quantidade de praticante de atividade fisica. Apesar dos praticantes do sexo masculino serem a porcentagem de 41,3%, o percentual de entrevistados desse mesmo gênero é de 44,1% que são praticantes de atividade fisica, e 55,9% total de entrevistados do sexo feminino, sendo 58,7% praticantes de atividades fisica. Podemos observar na Figura abaixo.

**Figura 32**: Screenshot - Codigo da plotagem do grafico agrupado por Gênero que pratica atividade fisica.

```
]: #Plotagem das informações de gênero dos Entrevistados

import plotly.express as px
pie = df_atividadeSelecionadas.loc[df_atividadeSelecionadas["pratica_exercicio"]== "sim"]
pie = pie["sexo"].value_counts()
regions = pie.index
population = pie.values
fig = px.pie(df_atividadeSelecionadas, values=population, names=regions, title ="Total de entrevistados para
fig.show()
```

Fonte: Autor

**Figura 33**: Screenshot - Grafico agrupado por Gênero que pratica atividade fisica. Fonte




:

Fonte: Autor

O próximo item a ser analisado é a e evolucão da prática de atividade fisica ao logo dos anos 2016, 2017, 2018 e 2019. Podemos peceber que no ano de 2017 começou a haver um aumento de atividade fisica, ficando assim em aumento constante até o ano de 2018, onde houve uma queda no idicadores que persitiu até o último ano analisado 2019. Podemos observar nas Figuras abaixo.

**Figura 34**: Screenshot - Codigo que exibe a quantidade total por ano de entrevistados que praticam atividade fisica.

```
In [52]: # conta quantos registros tem o dataframe pequisaSim
         pequisaSim = Counter(dftotalpraticaSim['ano'])
         pequisaSim

Out[52]: Counter({2016: 29097, 2017: 29323, 2018: 30088, 2019: 29479})
```

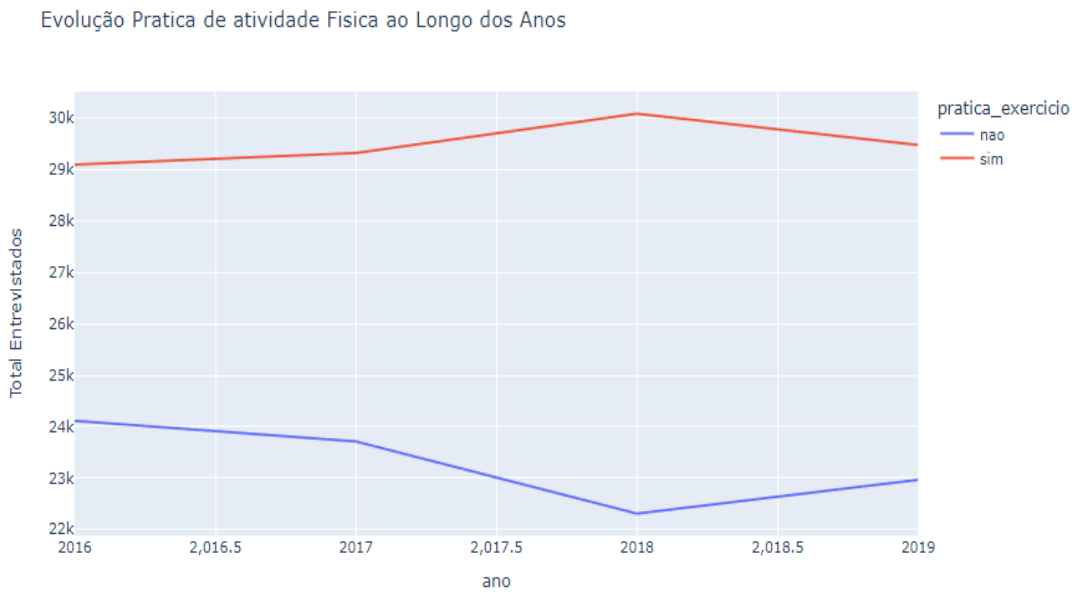
Fonte: Autor

**Figura 34**: Screenshot - Codigo da plotagem do grafico que mostra a evolução anual.

```
In [54]: #Plotagem das informações dos Entrevistados total evolução ano
df1=df_atividadeSelecionadas[['pratica_exercicio','ano']]
df2=df1.groupby(['ano','pratica_exercicio']).size().reset_index(name='Total Entrevistados')
fig3 = px.line(df2, x="ano", y="Total Entrevistados", color='pratica_exercicio',title='Evolução Pratica de atividade Fi
fig3.show()
```

Fonte: Autor

**Figura 35**: Screenshot - Plotagem do grafico que mostra a evolução ao longo dos anos.




Fonte: Autor

No próximo passo iremos contar o numéro de entrevistados que praticam atividades fisicas por modalidade e iremos mostrar quais são as atividade mais paraticadas na Figura abaixo.

**Figura 36**: Screenshot - Codigo da contagem dos numeros de praticantes por modalidade

```
#Contagem de modalidade por numero de praticantes
PraticaAtivida = Counter(dftotalpraticaSim['tipo_exercicio'])
PraticaAtivida
```

Fonte: Autor

**Figura 37**: Screenshot - Plotagem da contagem dos numeros de praticantes por modalidade.

```
Out[167]: Counter({'corrida (cooper)': 5834,
                   'musculação': 17364,
                   'futebol/futsal': 7304,
                   'caminhada em esteira': 3609,
                   'caminhada (não vale deslocamento para trabalho)': 48597,
                   'artes marciais e luta (jiu-jitsu, karatê, judô, boxe, muay thai, capoeira)': 1700,
                   'bicicleta (inclui ergométrica)': 5070,
                   'outros': 5548,
                   'ginástica aeróbica (spinning, step, jump)': 2452,
                   'ginástica em geral (alongamento, pilates, ioga)': 9186,
                   'hidroginástica': 4980,
                   'dança (balé, dança de salão, dança do ventre)': 2226,
                   'natação': 1737,
                   'corrida em esteira': 1083,
                   'voleibol/futevolei': 740,
                   'tênis': 293,
                   'basquetebol': 264})
```

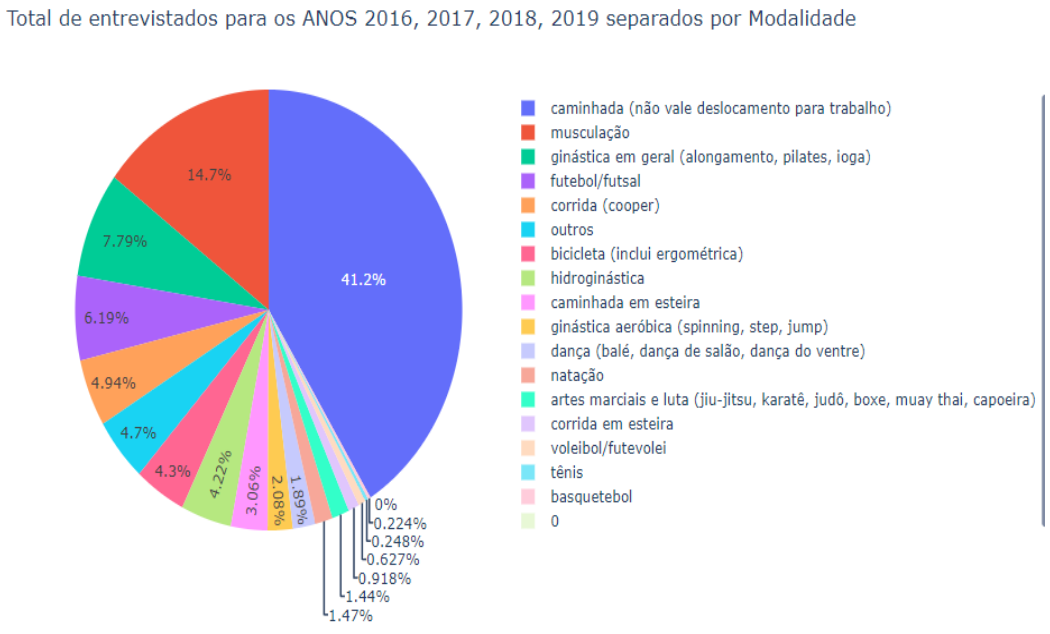
Fonte: Autor

**Figura 38**: Screenshot – Coódigo da Plotagem da contagem dos números de praticantes por modalidade percentual.

```
]:  #Plotagem das informações dos Entrevistados total evolução ano
df= dftotalpraticaSim
import plotly.express as px
pie = df["tipo_exercicio"].value_counts()
regions = pie.index
population = pie.values
fig = px.pie(df, values=population, names=regions, title ="Total de entrevistados para os ANOS 2016, 2017, 2018, 2
fig.show()
```

Fonte: Autor

**Figura 39**: Screenshot - Plotagem da contagem dos números de praticantes por modalidade percentual.




Fonte: Autor

Podemos observar no gráfico acima, que a modalidade mais praticada é a caminhada, sendo esta na porcentagem de 41,2% que representa um total de 48.597 praticantes dessa modalidade e a menos praticada e o Basquetebol com 0,224% sendo o total de praticantes de 264.

Analisaremos um periodo especifico sendo este o ano 2019 com o Data Frame criado na Figura 27, para que possamos entender melhor esse conjunto de dados de forma exploratória.

Como um dos objetivos é avaliar a discrepância de características entre os entrevistados que participaram das entrevistas no ano 2019, para saber se praticaram ou não atividade fisica; sempre que possível as informações serão separadas nesses critérios e, para isso, a partir do dataframe "df_censoSelecionadas" será criado dois dataframe apenas com os praticantes de atividade fisica "df_censoSelecionadas2019Sim",e os não praticantes de atividade fisica "df_censoSelecionadas2019Não".

Na Figura abaixo iremos apresentar o total de entrevistados de 2019.

**Figura 39**: Screenshot – Codigo da contagem dos numeros de entrevistados para ano 2019.

```
: # conta quantos registros tem o dataframe df_censoSelecionadas2019
  pequisa2019 = Counter(df_censoSelecionadas2019['ano'])
  pequisa2019

: Counter({2019: 52443})
```

Fonte: Autor

**Figura 40**: Screenshot – Codigo contagem dos numeros de entrevitados agrupado po Sim ou Não.

```
# conta quantos registros tem o dataframe df_censoSelecionadas2019
pequisa2019Pra = Counter(df_censoSelecionadas2019['pratica_exercicio'])
pequisa2019Pra

Counter({'nao': 22964, 'sim': 29479})
```

Fonte: Autor

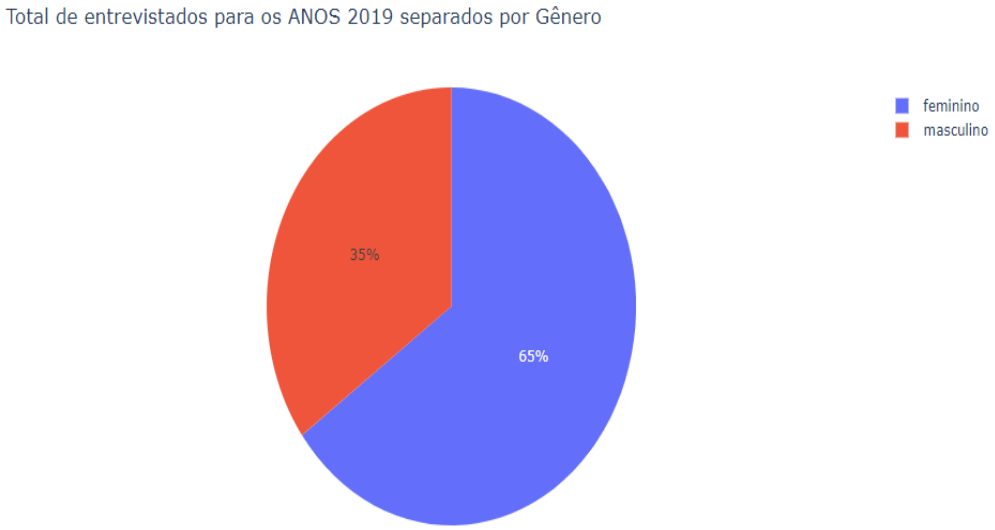
**Figura 41**: Screenshot – Codigo da Plotagem do grafico agrupado por gênero para o ano de 2019

```
#Vamos tratar um perriodo Especifico ANO 2019

#Plotagem das informações de gênero dos Entrevistados

#filtra atividade fisica e colocar o resultado no dataframe dftotalpraticaSim
df_censoSelecionadas2019Sim = df_censoSelecionadas2019.loc[df_censoSelecionadas2019['pratica_exercicio']== 'sim']

df_censoSelecionadas2019teste = df_censoSelecionadas2019
##df_censoSelecionadas2019.head()

import plotly.express as px
pie = df_censoSelecionadas2019["sexo"].value_counts()
regions = pie.index
population = pie.values
fig = px.pie(df_censoSelecionadas2019, values=population, names=regions, title ="Total de entrevistados para os ANOS 2019 separac
fig.show()
```

Fonte: Autor

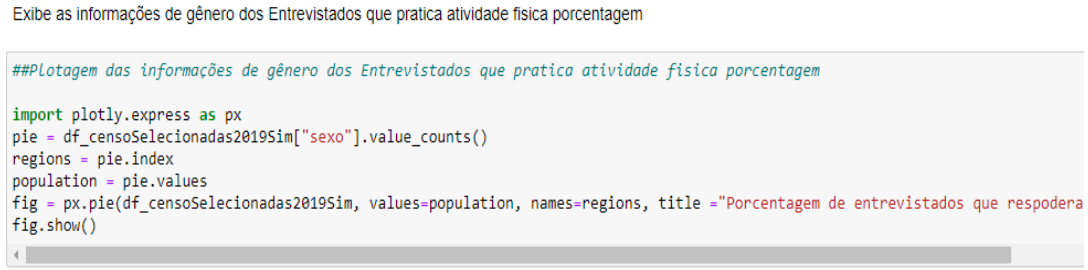
Na Figura 42 abaixo, podemos observar o agrupamento por gênero do número de entrevistados na pequisa. Observa-se que o número de entrevistados é maior para o gênero feminino, sendo a porcetagem de 65%.

**Figura 42**: Screenshot – Plotagem do grafico agrupado por gênero para o ano de 2019



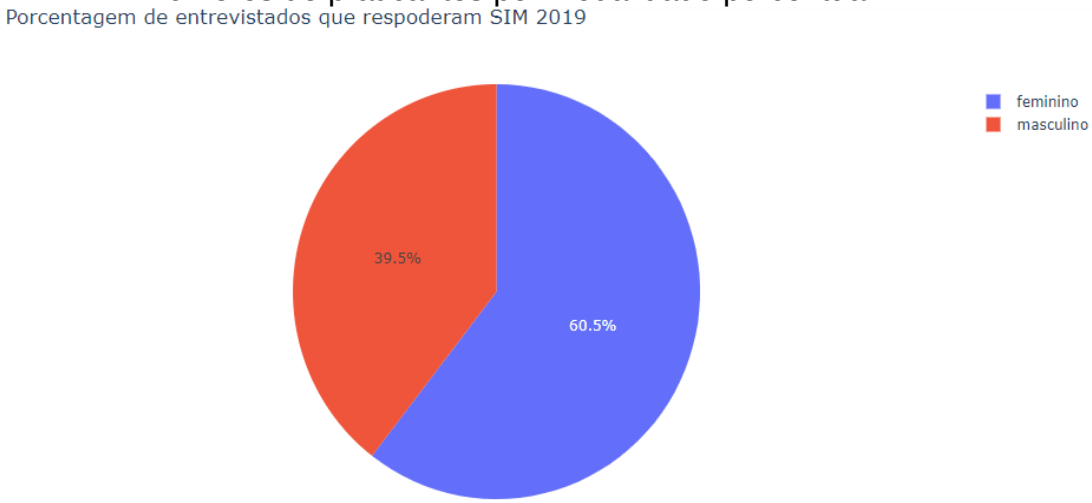

Fonte: Autor

**Figura 43**: Screenshot – Código da Plotagem da contagem dos numeros de praticantes por percentual.




Fonte: Autor

**Figura 45**: Screenshot – Código da Plotagem da contagem dos números de praticantes por modalidade percentual.




Fonte: Autor

Analisando os gráficos, pode-se perceber que o número de entrevistados de cada gênero não é proporcional. Visto que, o gênero feminio tem mais entrevistados que praticam atividade física , sendo  60,5% para 39,5% do gênero masculino.

O próximo item a ser analisado é a idade dos entrevistados. Para essa análise será utilizada a função “describe” que apresenta os principais indicadores estatísticos de uma série de dados. Na visualização da distribuição dessas idades serão utilizados histogramas e gráfico de linha.

**Figura 46**: Screenshot – Código com informações do campo idade do Data Frame.

```
In [179]: #Informações do campo idade
          df_censoSelecionadas2019['idade'].describe()

Out[179]: count    52443.000000
          mean        54.333314
          std         18.273770
          min         18.000000
          25%         40.000000
          50%         56.000000
          75%         68.000000
          max        106.000000
          Name: idade, dtype: float64
```

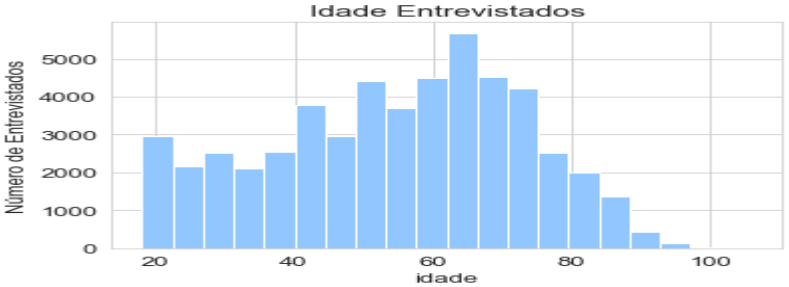
Fonte: Autor

**Figura 47:** Screenshot – Código da Plotagem da idade para os entrevistados total.

```
#Plotagem de histograma com as idade Entrevistados
df_censoSelecionadas2019.idade.hist(bins=20)
plt.style.use('seaborn-pastel')
plt.xlabel("idade")
plt.ylabel("Número de Entrevistados")
plt.title("Idade Entrevistados")
plt.show()
```

Fonte: Autor

**Figura 49**: Screenshot – Plotagem da idade para os entrevistados total.



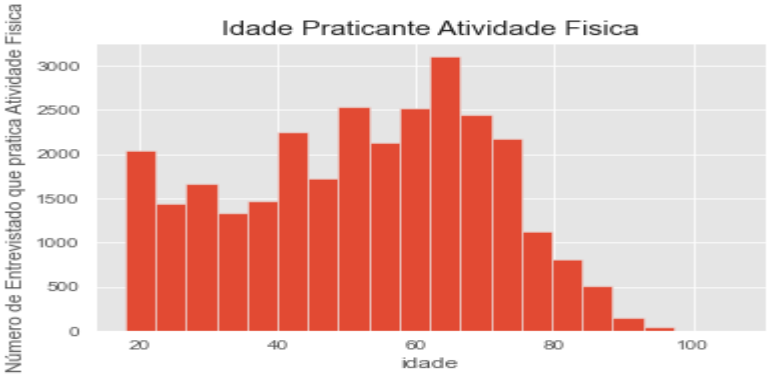

Fonte: Autor

**Figura 50**: Screenshot – Código da Plotagem da idade para os entrevistados praticantes de atividade fisica.

```
#Plotagem de histograma com as idade Praticante Atividade Fisica
df_censoSelecionadas2019Sim.idade.hist(bins=20)
plt.style.use('seaborn-pastel')
plt.xlabel("idade")
plt.ylabel("Número de Entrevistado que pratica Atividade Fisica")
plt.title("Idade Praticante Atividade Fisica")
plt.show()
```

Fonte: Autor

**Figura 51**: Screenshot – Plotagem da idade para os entrevistados praticantes de atividade fisica




Fonte: Autor

**Figura 52**: Screenshot – Código com informações do campo idade para Praticante de atividade fisica.

```
]: #Informações do campo idade Praticande de Atividade
   df_censoSelecionadas2019Sim['idade'].describe()

]: count    29479.000000
   mean        52.105906
   std         18.161793
   min         18.000000
   25%         38.000000
   50%         54.000000
   75%         66.000000
   max        106.000000
   Name: idade, dtype: float64
```

Fonte: Autor

**Figura 53**: Screenshot – Código com informações do campo idade para Não Praticante de atividade fisica.

```
[187]: #Informações do campo idade Não Praticande de Atividade
       df_censoSelecionadas2019Nao['idade'].describe()

[187]: count    22964.000000
       mean        57.192649
       std         18.017696
       min         18.000000
       25%         44.000000
       50%         60.000000
       75%         71.000000
       max        103.000000
       Name: idade, dtype: float64
```

Fonte: Autor

**Figura 53**: Screenshot – Código da Plotagem evolução por idade entrevistados.

```
#Plotagem das informações dos Entrevistados total evolução ano 2019
df1=df_censoSelecionadas2019[['pratica_exercicio','idade']]
df2=df1.groupby(['idade','pratica_exercicio']).size().reset_index(name='Total Entrevistados')
fig3 = px.line(df2, x="idade", y="Total Entrevistados", color='pratica_exercicio',title='Evolução Pratica de ativid
fig3.show()
```

Fonte: Autor

**Figura 54**: Screenshot – Plotagem evolução por idade entrevistados


Fonte: Autor

Analisando os histogramas, apesar dos entrevistados que praticam atividade fisica não terem um pico tão acentuado, não se verifica nenhuma mudança significativa. Essa constatação, pode ser confirmada por meio dos indicadores estatísticos das duas séries de dados que são muito similares, com variações mínimas na média de idade, por exemplo: 54,33 anos para todos os entrevistados e 52,10 anos para os praticantes. Percebe-se também que a distribuição de idade é pouco diferente à do entrevistados que não praticam nenhuma atividade 57,19 anos.

O próximo critério a ser analisado é a Cor de pele dos entrevidados que praticam atividade física; e, para a visualização desses dados, optou-se pelo gráfico de pizza na Figura abaixo

**Figura 55**: Screenshot – Código entrevistados que praticam atividade agrupado por cor.

```
In [88]: #exibe as cor dos entrevistados SIM
         df_censoSelecionadas2019Sim["cor"].value_counts()

Out[88]: branca              13270
         parda               11451
         preta                2235
         Morena               1752
         indígena              352
         amarela               239
         não sabe              156
         não quis informar      24
         Name: cor, dtype: int64
```

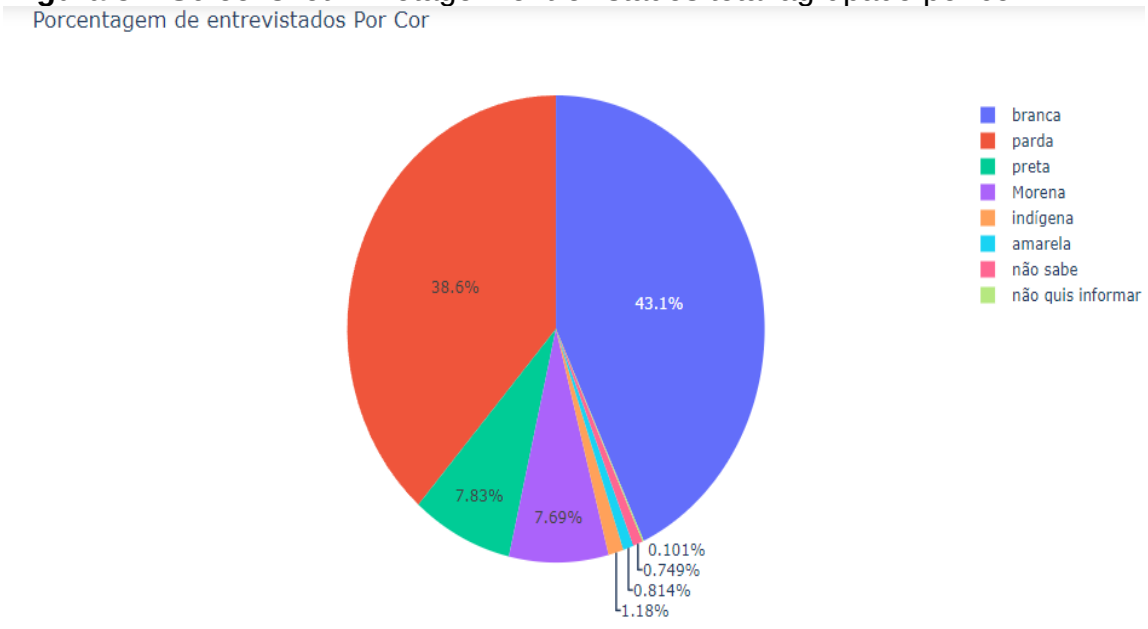
Fonte: Autor

**Figura 56**: Screenshot – Código da Plotagem entrevistados total agrupado por cor.

```
##Plotagem das informações Tipo Atividade Fisica dos Entrevistados por COR porcentagem
import plotly.express as px
pie =df_censoSelecionadas2019["cor"].value_counts()
regions = pie.index
population = pie.values
fig = px.pie(df_censoSelecionadas2019, values=population, names=regions, title ="Porcentagem de entrevistados Por Cor")
fig.show()
```

Fonte: Autor

**Figura 57**: Screenshot – Plotagem entrevistados total agrupado por cor.



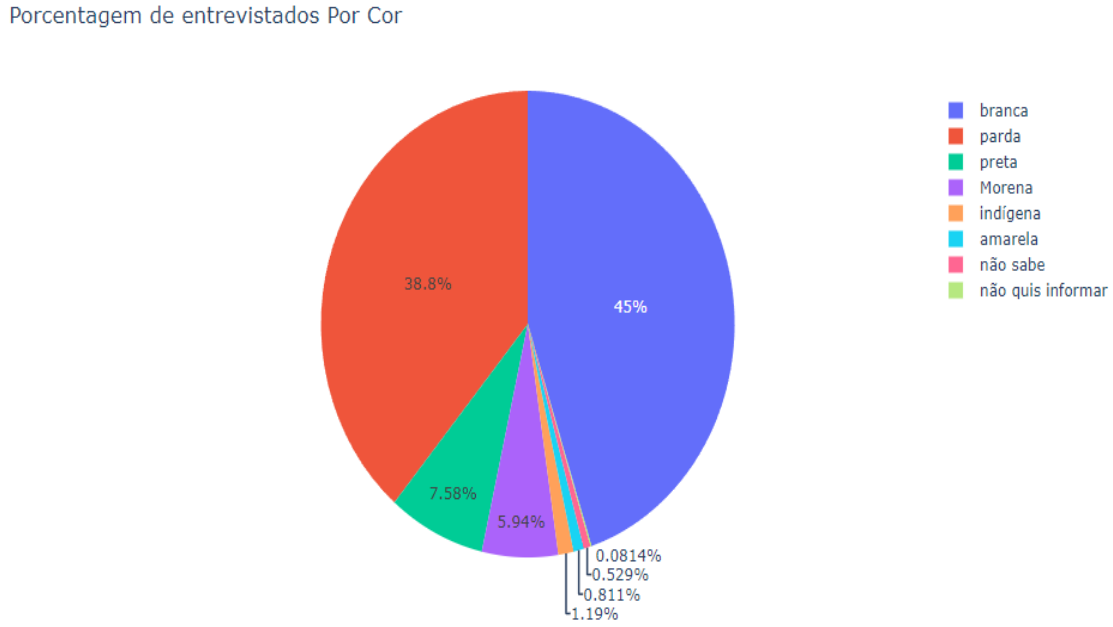

Fonte: Autor

**Figura 58**: Screenshot – Código da Plotagem entrevistados que praticam atividade agrupado por cor.

```
##Plotagem das informações Tipo Atividade Fisica dos Entrevistados por COR porcentagem
import plotly.express as px
pie =df_censoSelecionadas2019Sim["cor"].value_counts()
regions = pie.index
population = pie.values
fig = px.pie(df_censoSelecionadas2019Sim, values=population, names=regions, title ="Porcentagem de entrevistados Por Cor")
fig.show()
```

Fonte: Autor

**Figura 59**: Screenshot – Plotagem entrevistados que praticam atividade agrupado por cor.




Fonte: Autor

Na análise desse ponto, percebe-se uma semelhança com a análise de gênero, em que há diferenças significativas entre a população. De acordo com dados da Pesquisa Nacional por Amostra de Domicílios (PNAD) 2019, 42,7% da população se declara branca; e esse percentual sobe para 43,1% para os entrevistados e 45,0% quando se fala de praticante de atividade fisica.

O próximo critério a ser analisado é a Região dos entrevistados praticantes de atividade física. Para essa análise serão utilizadas as mesmas técnicas descritas anteriormente, apenas com a diferença na apresentação dos dados que será por meio do gráfico de pizza. Assim o primeiro passo é realizar a contagem dos valores da série desejada.

**Figura 60**: Screenshot – Código contagem entrevistados por região que pratica atividade.

```
In [196]: #Atividade Fisica dos Entrevistados que praticam atividade porcentagem por Região

          df_censoSelecionadas2019Sim["regiao"].value_counts()

Out[196]: Nordeste        10166
          Norte            6509
          Centro-Oeste     4828
          Sudeste          4394
          Sul              3582
          Name: regiao, dtype: int64
```

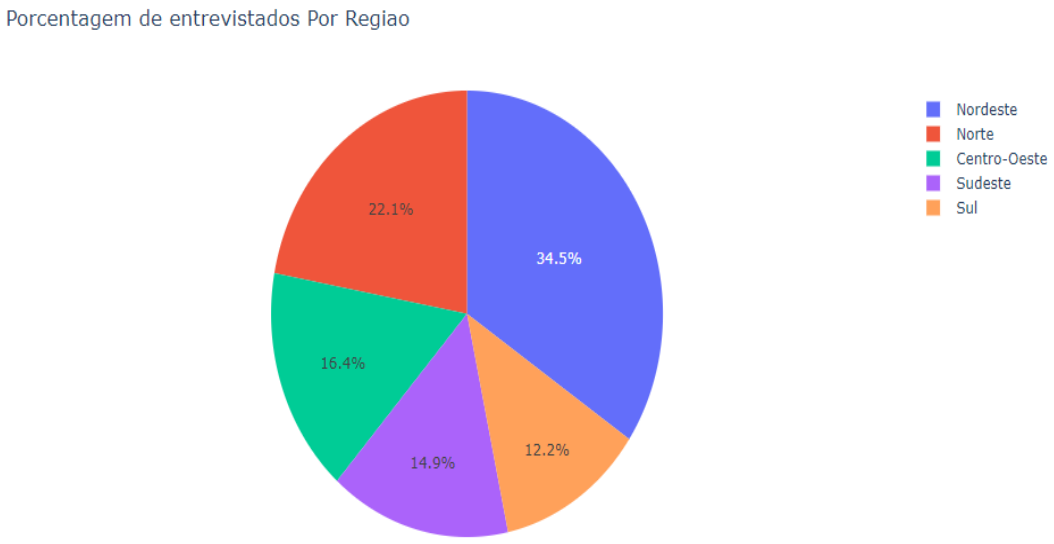
Fonte: Autor

**Figura 61**: Screenshot – Código da Plotagem entrevistados praticam atividade Região.

```
: ##Plotagem das informações Atividade Fisica dos Entrevistados que praticam atividade porcentagem por Região

import plotly.express as px
pie = df_censoSelecionadas2019Sim["regiao"].value_counts()
regions = pie.index
population = pie.values
fig = px.pie(df_censoSelecionadas2019Sim, values=population, names=regions, title ="Porcentagem de entrevistados Por Regiao")
fig.show()
```

Fonte: Autor

**Figura 62**: Screenshot – Plotagem entrevistados praticam atividade por Região.




Fonte: Autor

Nesse ponto, percebe-se mais uma vez uma grande discrepância entre a região do entrevistado praticante de atividade fisica e a região mais populosa do Brasil tendo em vista que, percebe-se, que a população nordestina pratica mais atividade física do que a região Sudeste, mesmo com o Sudeste contando com o maior número de habitantes.

Na próxima análise do dataframe em questão é referente a Capitais dos entrevistados que praticam atividades físicas. Serão utilizados comandos para obtenção do percentual dos valores de cada categoria, transformação dos dados do dicionário em listas e, por fim, apresentação dos resultados em gráficos pizza.

**Figura 63**: Screenshot – Código quantidade total entrevistados praticam atividade por Capital.

```
In [130]: # Conta quantos Respoderam sim por Capital
          df_censoSelecionadas2019Sim["nome_capital"].value_counts()

Out[130]: Brasília          1416
          Florianópolis     1281
          Teresina          1236
          Aracaju           1214
          São Luís          1193
          Vitória           1180
          Goiânia           1167
          Curitiba          1159
          Belo Horizonte    1158
          Natal             1153
          Belém             1148
          Porto Alegre      1142
          Cuiabá            1125
          João Pessoa       1124
          Campo Grande      1120
          Porto Velho       1108
          Fortaleza         1100
          Maceió            1096
          Salvador          1091
          Rio Branco        1063
          Rio de Janeiro    1047
          Manaus            1030
          Palmas            1020
          São Paulo         1009
          Recife             959
          Boa Vista          576
          Macapá             564
          Name: nome_capital, dtype: int64
```

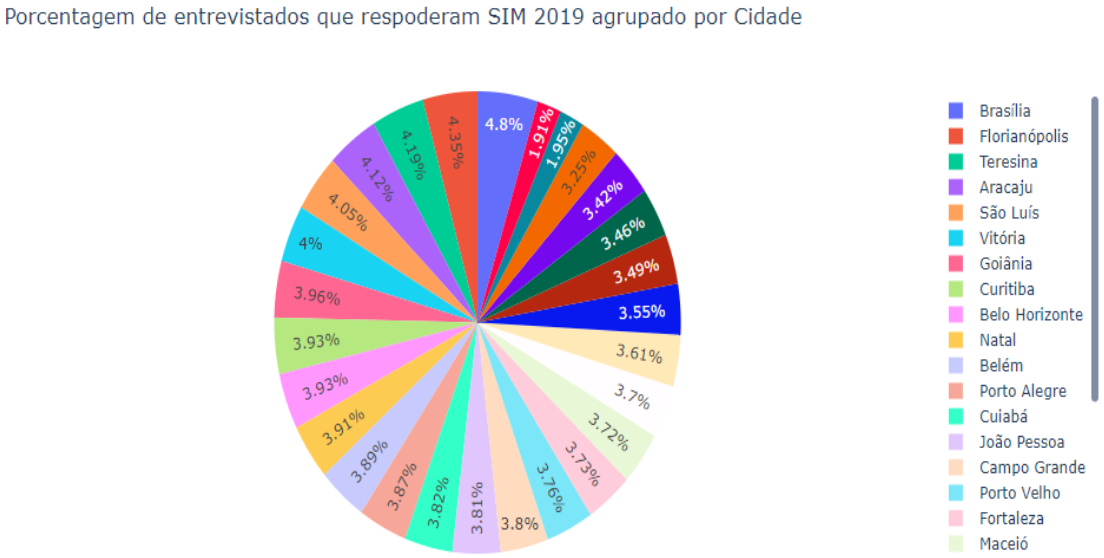
Fonte: Autor

**Figura 64**: Screenshot – Código da Plotagem quantidade total entrevistados praticam atividade por Capital.

```
##Plotagem das informações Tipo Atividade Fisica dos Entrevistados que praticam atividade porcentagem agrupado por Cidade

import plotly.express as px
pie = df_censoSelecionadas2019Sim["nome_capital"].value_counts()
regions = pie.index
population = pie.values
fig = px.pie(df_censoSelecionadas2019Sim, values=population, names=regions, title ="Porcentagem de entrevistados que respoderam S
fig.show()
```

Fonte: Autor

**Figura 65**: Screenshot – Plotagem quantidade total entrevistados praticam atividade por Capital.



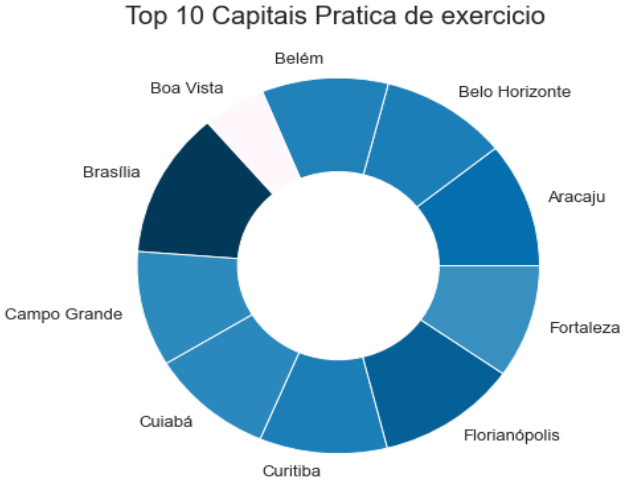

. Fonte: Autor

**Figura 66**: Screenshot – Código da Plotagem 10 Capitais que mais praticam atividades fisica.

```
8]: import matplotlib as mpl
    game = df_censoSelecionadas2019Sim.groupby("nome_capital")["ano"].count().head(10)
    custom_colors = mpl.colors.Normalize(vmin=min(game), vmax=max(game))
    colours = [mpl.cm.PuBu(custom_colors(i)) for i in game]
    plt.figure(figsize=(7,7))
    plt.pie(game, labels=game.index, colors=colours)
    central_circle = plt.Circle((0, 0), 0.5, color='white')
    fig = plt.gcf()
    fig.gca().add_artist(central_circle)
    plt.rc('font', size=13)
    plt.title("Top 10 Capitais Pratica de exercicio ", fontsize=20)
    plt.show()
```

Fonte: Autor

**Figura 67**: Screenshot –Plotagem 10 Capitais que mais praticam atividades fisica.




Fonte: Autor

Nesse ponto, percebe-se mais uma vez uma grande discrepância entre a cidade mais rica do pais com  a prática de atividade fisica. A capital onde mais se pratica atividade física é Brasilia, tendo um total 4,8% e a cidade que menos pratica atividade física e Macapá  com 1,91%.

Na proxima análise do dataframe em questão, é referente a qual prática de atividade física e praticada em cada capital, bem com o tempo em  que é destinado para cada prática da atividade e a frenquência que é executada durante a semana. Serão utilizados comandos para obtenção do percentual dos valores de cada categoria, transformação dos dados do dicionário em  listas e, por fim, apresentação dos resultados em gráficos pizza e barras.

**Figura 68**: Screenshot – Código da quantidade total agrupado por tipo atividade física

```
# Total agrupado por atividade fisica
df_censoSelecionadas2019Sim["tipo_exercicio"].value_counts()
```

```
caminhada (não vale deslocamento para trabalho)                    12352
musculação                                                          4430
ginástica em geral (alongamento, pilates, ioga)                     2373
futebol/futsal                                                      1578
outros                                                              1473
corrida (cooper)                                                    1355
bicicleta (inclui ergométrica)                                      1331
hidroginástica                                                      1319
caminhada em esteira                                                 846
ginástica aeróbica (spinning, step, jump)                            554
dança (balé, dança de salão, dança do ventre)                        532
natação                                                              449
artes marciais e luta (jiu-jitsu, karatê, judô, boxe, muay thai, capoeira)    367
corrida em esteira                                                   220
voleibol/futevolei                                                   166
basquetebol                                                           71
tênis                                                                 63
0.0                                                                    0
Name: tipo_exercicio, dtype: int64
```

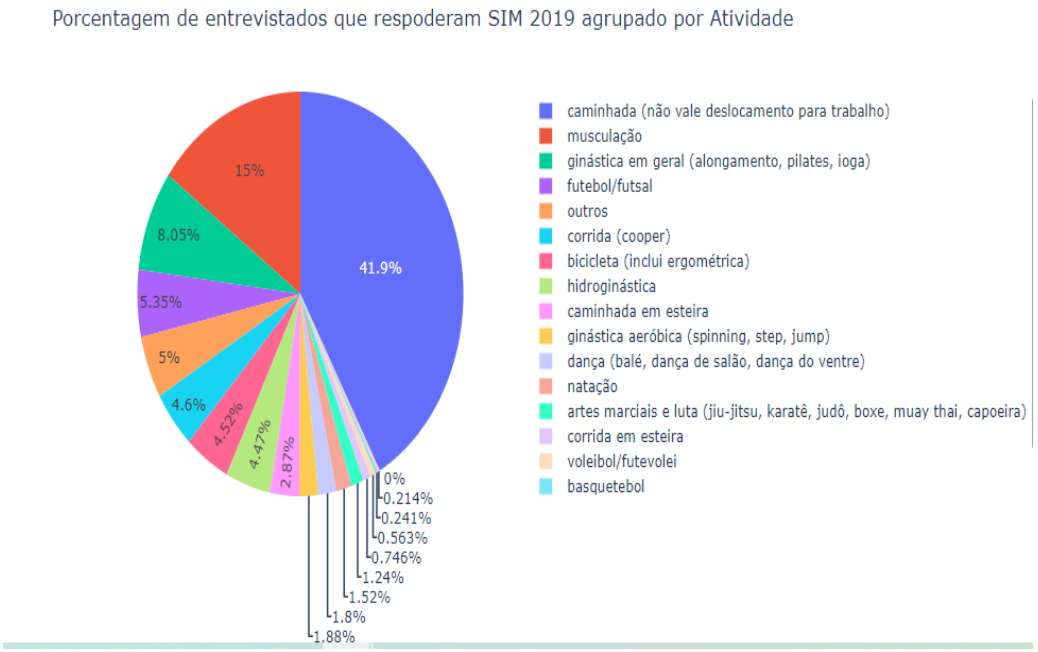
Fonte: Autor

**Figura 69**: Screenshot – Código da Plotagem Tipo exercicio praticados.

```
##Plotagem das informações Tipo Atividade Fisica dos Entrevistados que pratica atividade porcentagem agrupado por Atividade

import plotly.express as px
pie = df_censoSelecionadas2019Sim["tipo_exercicio"].value_counts()
regions = pie.index
population = pie.values
fig = px.pie(df_censoSelecionadas2019Sim, values=population, names=regions, title ="Porcentagem de entrevistados que respoderam
fig.show()
```

Fonte: Autor

**Figura 70**: Screenshot – Código da Plotagem Tipo exercício praticados.




Fonte: Autor

**Figura 71**: Screenshot – Código da Plotagem Top 10 exercicios mais praticados.

```
import matplotlib as mpl
game = df_censoSelecionadas2019Sim.groupby("tipo_exercicio")["ano"].count().head(10)
custom_colors = mpl.colors.Normalize(vmin=min(game), vmax=max(game))
colours = [mpl.cm.PuBu(custom_colors(i)) for i in game]
plt.figure(figsize=(7,7))
plt.pie(game, labels=game.index, colors=colours)
central_circle = plt.Circle((0, 0), 0.5, color='white')
fig = plt.gcf()
fig.gca().add_artist(central_circle)
plt.rc('font', size=12)
plt.title("Top 10 Tipo Exercico praticado ", fontsize=20)
plt.show()
```

Fonte: Autor

**Figura 61**: Screenshot – Código da Plotagem Top 10 exercicios mais praticados.

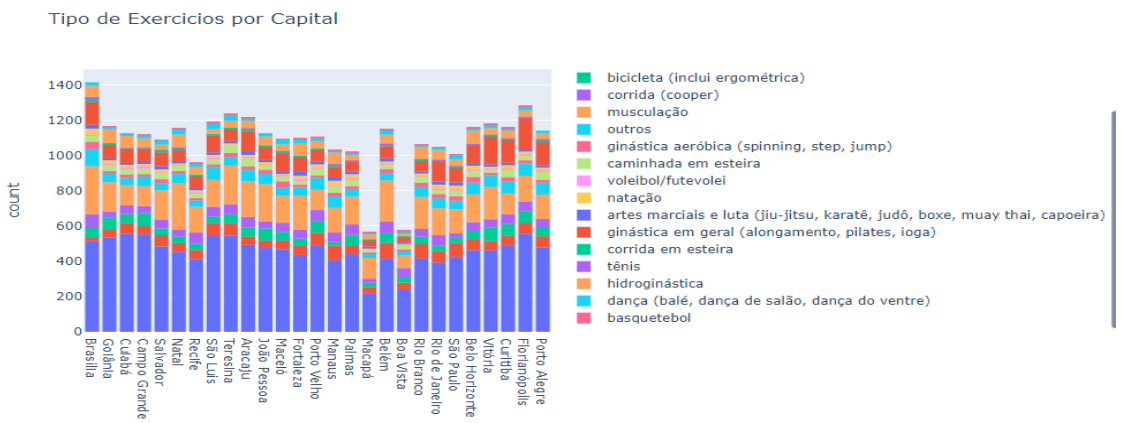

Fonte: Autor

**Figura 72**: Screenshot – Código da Plotagem Tipo exercicio praticados por  Capital.

```
#Plotagem das informações Duração dos Exercicios por Capital"
import plotly.express as px
df = df_censoSelecionadas2019Sim
figure = px.histogram(df, x = "nome_capital", color = "tipo_exercicio", title= "Tipo de Exercicios por Capital")
figure.show()
```

Fonte: Autor

**Figura 73**: Screenshot – Plotagem Tipos exercicios praticados por Capital.



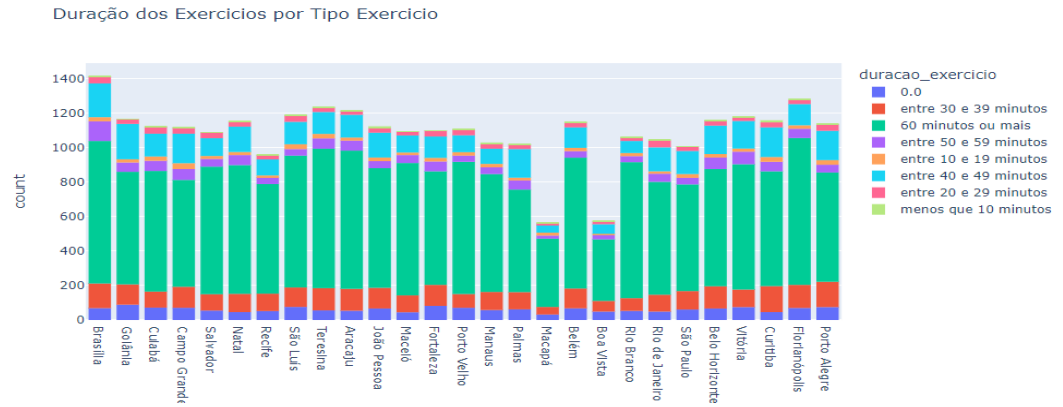

Fonte: Autor

**Figura 74**: Screenshot – Código da Plotagem quantidade total de entrevistados que praticam atividade por Capital e tempo da prática.

```
#Plotagem das informações Duração dos Exercicios por Tipo Exercicio"
import plotly.express as px
df = df_censoSelecionadas2019Sim
figure = px.histogram(df, x = "nome_capital", color = "duracao_exercicio", title= "Duração dos Exercicios por Tipo Exercicio")
figure.show()
```

Fonte: Autor

**Figura 75:** Screenshot – Plotagem quantidade total de entrevistados que praticam atividade por Capital e tempo da prática.



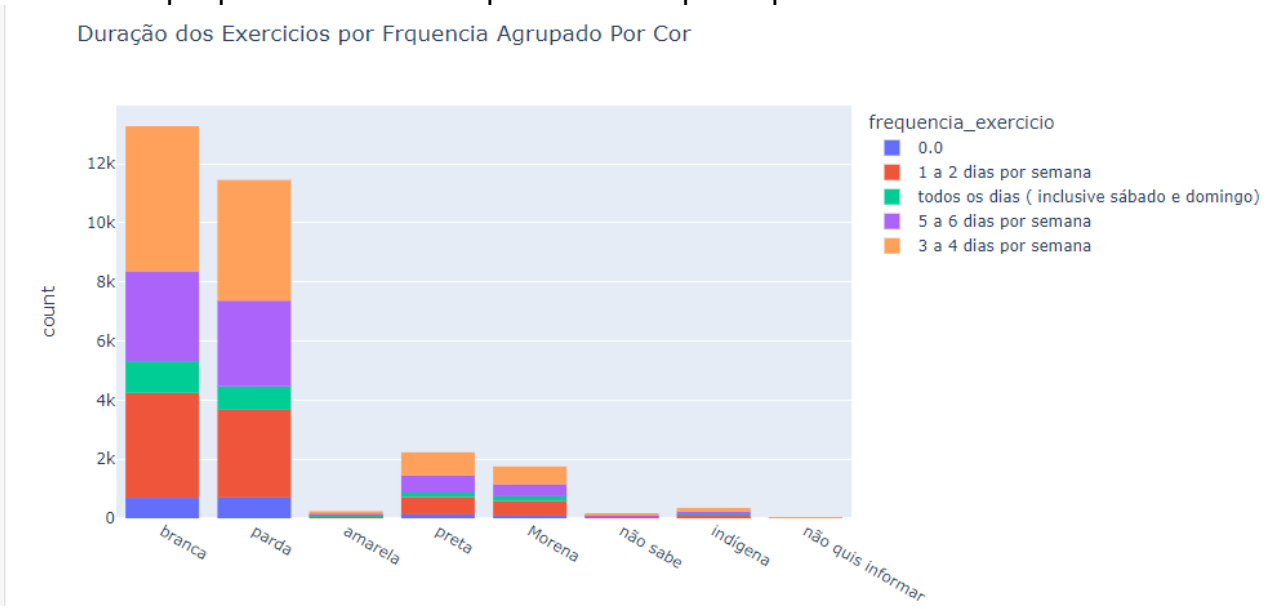

Fonte: Autor

**Figura 76**: Screenshot – Código da Plotagem quantidade total de entrevistados que praticam atividade por Cor e tempo da prática.

```
In [157]: #Plotagem das informações Duração dos Exercicios por Frequencia por Cor
          import plotly.express as px
          df = df_censoSelecionadas2019Sim
          figure = px.histogram(df, x = "cor", color = "frequencia_exercicio", title= "Duração dos Exercicios por Frquencia Agrupado Por Co
          figure.show()
```

Fonte: Autor

**Figura 73**: Screenshot – Plotagem quantidade total de entrevistados que praticam atividade por Cor e tempo da prática.



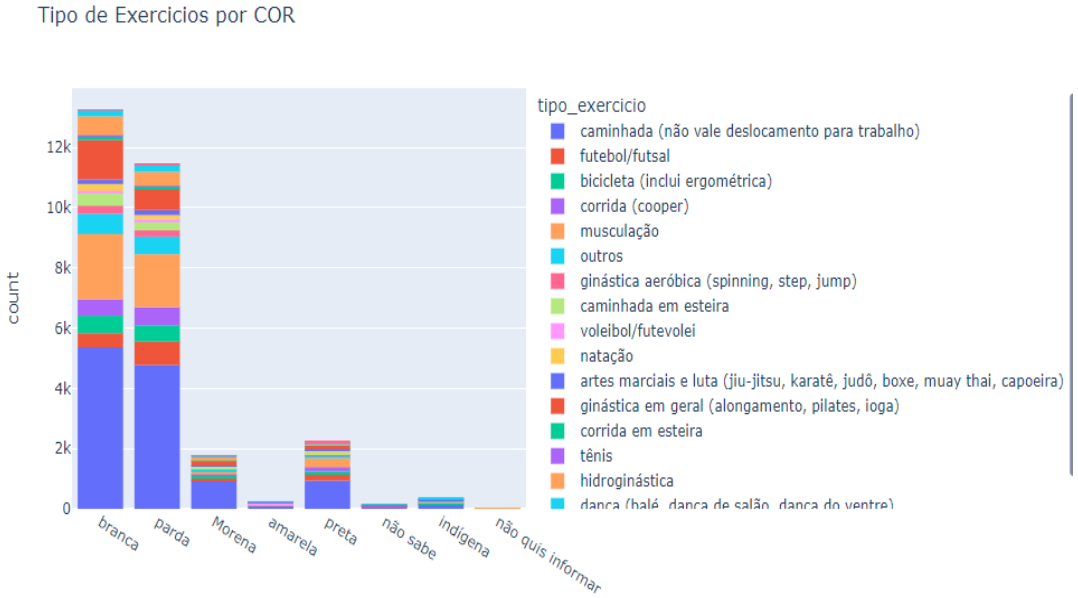

Fonte: Autor

**Figura 78**: Screenshot – Código da Plotagem Tipo de exercicio agrupado por Cor .

```
#Plotagem das informações Duração dos Exercicios por COR"
import plotly.express as px
df = df_censoSelecionadas2019Sim
figure = px.histogram(df, x = "cor", color = "tipo_exercicio", title= "Tipo de Exercicios por COR")
figure.show()
```

Fonte: Autor

**Figura 79**: Screenshot – Plotagem Tipo de exercicio agrupado por Cor .



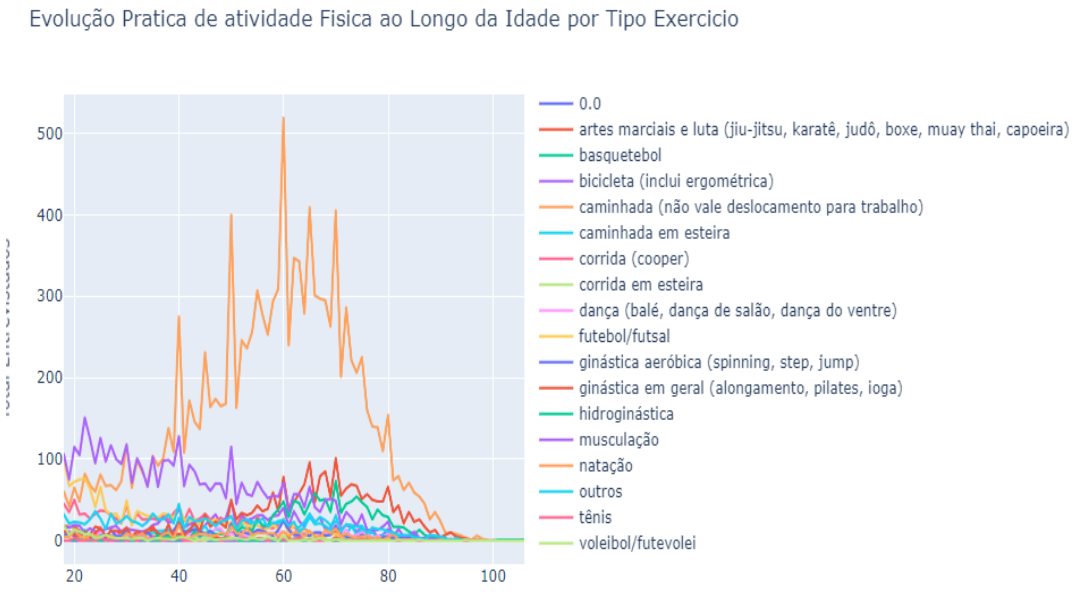

Fonte: Autor

**Figura 80**: Screenshot – Código da Plotagem Idade de praticantes de atividades fisicas agrupado por atividade.

```
#Plotagem das informações dos Entrevistados total evolução ano 2019
df1=df_censoSelecionadas2019Sim[['tipo_exercicio','idade',]]
df2=df1.groupby(['idade','tipo_exercicio']).size().reset_index(name='Total Entrevistados')
fig3 = px.line(df2, x="idade", y="Total Entrevistados", color='tipo_exercicio',title='Evolução Pratica de
fig3.show()
```

Fonte: Autor

**Figura 81**: Screenshot – Código da Plotagem Idade de praticantes de atividades fisicas agrupado por atividade.




Fonte: Autor

Da análise combinada de todas as informações desse dataframe, percebe-se que, dos dados dos entrevistados que praticam atividades fisicas, que a modalidade mais praticada é a caminhada com porcentagem de 41,9% e a modalidade menos praticada e o tênis com porcentagem de 0,2145%. Analisamos também, que a capital onde mais se pratica atividade fisica é Brasília com porcentagem de 4,8% e a capital que menos se pratica atividade física é Macapá com porcentagem de 1,914% e a modalidade menos praticada la e o Basquetebol. Podemos observar que, a modalidade mais praticada que é a Caminhada é mais realizada em Florianópolis(557 entrevistados) e lá também tem o maior tempo usado para prática de atividade 60 minutos ou mais(852 entrevistados). Podemos analisar que, a cor Branca e Parda é a que que tem a maior frenquência de prática de atividade fisica durante a semana. Na parte da análise da idade x prática de atividade fisica, podemos observar que a caminhada tem prevalência sobre as demais; tem um pico na idade de 60 anos onde a maior parte dos entrevistados dessa idade praticam essa modalidade. Observamos também que, o futebol é praticado de 1 a 2 dias por semana e na sua maior parte é praticado por Pardos.

Na proxima análise do dataframe em questão, é referente ao salário médio, o PIB e número da população estimada. Vamos analisar a relação desses valores na prática da atividade física para conseguirmos entender se tem alguma relação com o maior salário para a prática de atividade fisica.Serão utilizados comandos para obtenção do percentual dos valores de cada categoria, transformação dos dados do dicionário em listas e, por fim, apresentação dos resultados em gráficos pizza e barras.

**Figura 82**: Screenshot - Código do resultado Nome Capital com Salário médio.

```
#seleciona coluna Nome Capital e salario

ColunasSelecionadaCenso = ['nome_capital','Salario']
SalarioCapital = df_censoSelecionadas.filter(items=ColunasSelecionadaCenso)
SalarioCapital.
```

Fonte: Autor

**Figura 83**: Screenshot - Plotagem resultado Nome Capital Salário medio.

| | nome_capital | Salario |
|---|---|---|
| 0 | Rio Branco | 3194.00 |
| 1 | Maceió | 2695.00 |
| 2 | Macapá | 3992.00 |
| 3 | Manaus | 3094.00 |
| 4 | Salvador | 3393.00 |
| 5 | Fortaleza | 2695.00 |
| 6 | Brasília | 5289.00 |
| 7 | Vitória | 3892.00 |
| 8 | Goiânia | 3293.00 |
| 9 | São Luís | 3094.00 |
| 10 | Cuiabá | 3892.00 |
| 11 | Campo Grande | 3493.00 |
| 12 | Belo Horizonte | 3393.00 |
| 13 | Belém | 3493.00 |
| 14 | João Pessoa | 2695.00 |
| 15 | Curitiba | 3792.00 |
| 16 | Recife | 3293.00 |
| 17 | Teresina | 2695.00 |
| 18 | Rio de Janeiro | 4192.00 |
| 19 | Natal | 2994.00 |
| 20 | Porto Alegre | 4092.00 |
| 21 | Porto Velho | 3393.00 |
| 22 | Boa Vista | 3493.00 |
| 23 | Florianópolis | 4491.00 |
| 24 | São Paulo | 4092.00 |
| 25 | Aracaju | 2994.00 |
| 26 | Palmas | 3992.00 |

Fonte: Autor

**Figura 84**: Screenshot - Código da plotagem do gráfico Salário por Capital.

```
#Plotagem do gráfico Salario por capital
plt.style.use('seaborn-pastel')
plt.barh( df_censoSelecionadas.nome_capital, df_censoSelecionadas.Salario,)
plt.ylabel('Salario')
plt.xlabel('nome_capital')
plt.title('Candidatos por escolaridade')
plt.show()
```

Fonte: Autor

**Figura 85**: Screenshot - Plotagem do gráfico Salário por Capital.

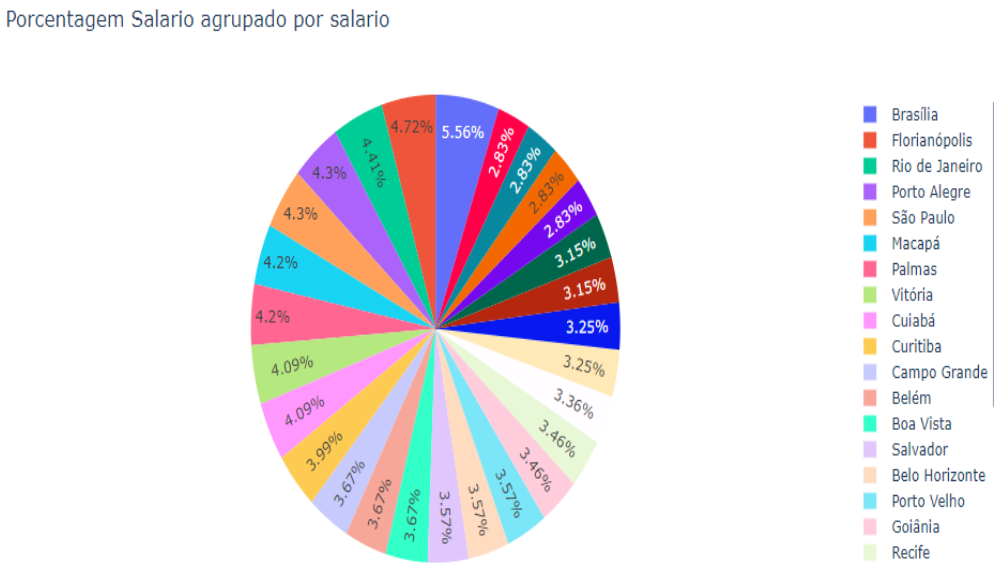

Fonte: Autor

**Figura 86**: Screenshot - Código da plotagem do grafico Salário por Capital porcentagem.

```
##Plotagem das informações Salario porcentagem agrupado por Cidade

import plotly.express as px
pie = df_censoSelecionadas["Salario"]
regions =df_censoSelecionadas["nome_capital"]
population = pie.values
fig = px.pie(df_censoSelecionadas, values=population, names=regions, title ="Porcentagem Salario agrupado por salario")
fig.show()
```

Fonte: Autor

**Figura 87**: Screenshot - Plotagem do gráfico Salário por Capital porcentagem.




Fonte: Autor

**Figura 88**: Screenshot - Código PIB agrupado por capital.

```
#seleciona coluna Nome Capital e PIB

ColunasSelecionadaCenso = ['nome_capital','PIB']
SalarioCapital = df_censoSelecionadas.filter(items=ColunasSelecionadaCenso)
SalarioCapital
```

Fonte: Autor

**Figura 32**: Screenshot - Plotagem do grafico PIB agrupado por capital.

| | nome_capital | PIB |
|---|---|---|
| 0 | Rio Branco | 9415984332.00 |
| 1 | Maceió | 23702498786.00 |
| 2 | Macapá | 11867052586.00 |
| 3 | Manaus | 87711155449.00 |
| 4 | Salvador | 64425482024.00 |
| 5 | Fortaleza | 68272625806.00 |
| 6 | Brasília | 280787559894.00 |
| 7 | Vitória | 22058836954.00 |
| 8 | Goiânia | 54293447495.00 |
| 9 | São Luís | 32513035918.00 |
| 10 | Cuiabá | 25068727784.00 |
| 11 | Campo Grande | 30910435425.00 |
| 12 | Belo Horizonte | 97926259712.00 |
| 13 | Belém | 32702193891.00 |
| 14 | João Pessoa | 21279185650.00 |
| 15 | Curitiba | 97610221341.00 |
| 16 | Recife | 55199348808.00 |
| 17 | Teresina | 22177561271.00 |
| 18 | Rio de Janeiro | 357974908203.00 |
| 19 | Natal | 25209340313.00 |
| 20 | Porto Alegre | 82918086237.00 |
| 21 | Porto Velho | 18568553918.00 |
| 22 | Boa Vista | 11561824692.00 |
| 23 | Florianópolis | 22645724131.00 |
| 24 | São Paulo | 772804830090.00 |
| 25 | Aracaju | 17950877133.00 |
| 26 | Palmas | 10946427427.00 |

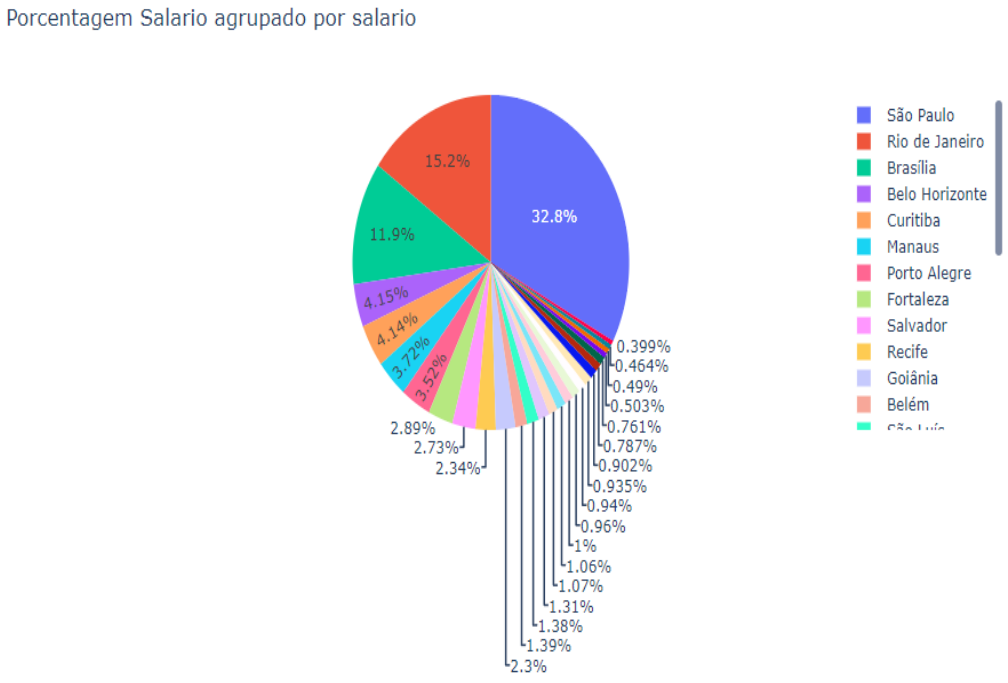
Fonte: Autor

**Figura 89**: Screenshot - Código da plotagem do grafico PIB agrupado por Capital.

```
##Plotagem das informações PIB porcentagem agrupado por Cidade

import plotly.express as px
pie = SalarioCapital["PIB"]
regions =SalarioCapital["nome_capital"]
population = pie1.values
fig = px.pie(df_censoSelecionadas, values=population, names=regions, title ="Porcentagem Salario agrupado por PIB")
fig.show()
```

Fonte: Autor

**Figura 90**: Screenshot - Plotagem do grafico PIB agrupado por Capital.




Fonte: Autor

**Figura 91**: Screenshot - Código da plotagem do gráfico População estimada agrupado por Capital.

```
##Plotagem das informações Total população porcentagem agrupado por Cidade

ColunasSelecionadaCenso = ['nome_capital','populacao_estimada_pessoas']
SalarioCapital = df_censoSelecionadas.filter(items=ColunasSelecionadaCenso)
SalarioCapital
```

Fonte: Autor

**Figura 92**: Screenshot - plotagem População estimada agrupado por Capital.

| | nome_capital | populacao_estimada_pessoas |
|---|---|---|
| 0 | Rio Branco | 419452 |
| 1 | Maceió | 1031597 |
| 2 | Macapá | 522357 |
| 3 | Manaus | 2255903 |
| 4 | Salvador | 2900319 |
| 5 | Fortaleza | 2703391 |
| 6 | Brasília | 3094325 |
| 7 | Vitória | 369534 |
| 8 | Goiânia | 1555626 |
| 9 | São Luís | 1115932 |
| 10 | Cuiabá | 623614 |
| 11 | Campo Grande | 916001 |
| 12 | Belo Horizonte | 2530701 |
| 13 | Belém | 1506420 |
| 14 | João Pessoa | 825796 |
| 15 | Curitiba | 1963726 |
| 16 | Recife | 1661017 |
| 17 | Teresina | 871126 |
| 18 | Rio de Janeiro | 6775561 |
| 19 | Natal | 896708 |
| 20 | Porto Alegre | 1492530 |
| 21 | Porto Velho | 548952 |
| 22 | Boa Vista | 436591 |
| 23 | Florianópolis | 516524 |
| 24 | São Paulo | 12396372 |
| 25 | Aracaju | 672614 |
| 26 | Palmas | 313349 |

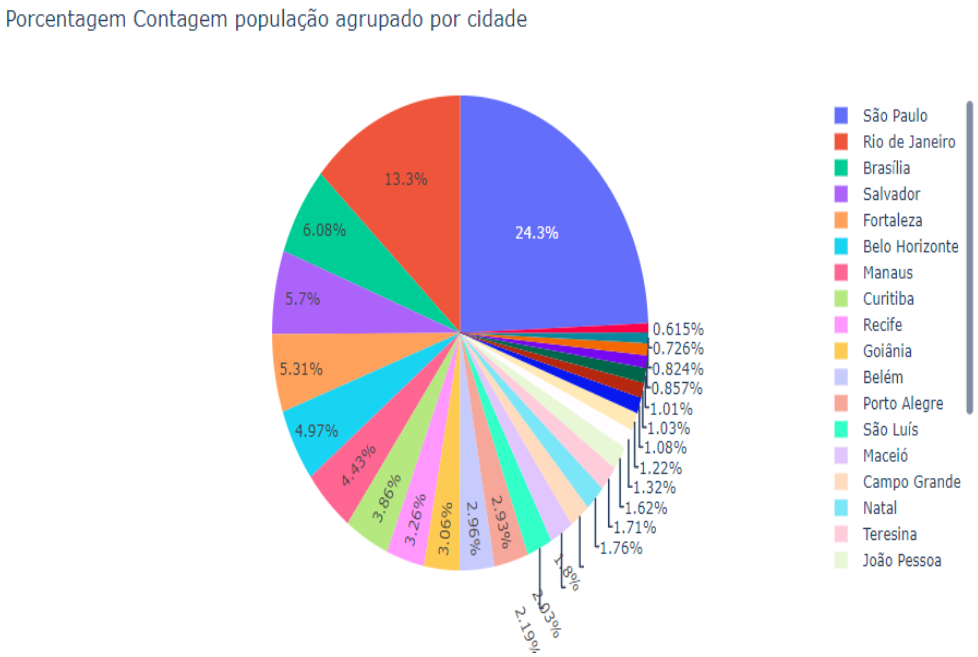
Fonte: Autor

**Figura 93**: Screenshot - Código da plotagem do gráfico População estimada agrupado por Capital.

```
##Plotagem das informações Total população porcentagem agrupado por Cidade

import plotly.express as px
pie = df_censoSelecionadas["populacao_estimada_pessoas"]
regions =df_censoSelecionadas["nome_capital"]
population = pie.values
fig = px.pie(df_censoSelecionadas, values=population, names=regions, title ="Porcentagem Contagem população agrupado por cidade")
fig.show()
```

Fonte: Autor

**Figura 94**: Screenshot - Plotagem do gráfico População estimada agrupado por Capital..




Fonte: Autor

Com base nos valores apresentados, percebe-se que, o salário médio tem muita relação com a prática de atividade fisica. Percebemos isso, quando olhamos para Brasilia onde o salário medio é de 5.289 reais e o número de praticantes é de 1.416 ou 4.8%; seguido de Florianópolis onde o salário médio é 4.491 reais e o número de praticante 1281 ou 4.35%.

Notamos que Rio de janeiro e São Paulo onde o salário médio está dentro do top 5 com maiores salários, que a prática de atividade fisica não acompanha, tendo em vista que São Paulo é o estado que tem o maior PIB e maior população. É um caso que tem que ser analisado mais a fundo pra entender por que existe essa diferença nessa pesquisa.

Na próxima análise do dataframe em questão, é referente ao estado civil dos entrevistados que praticam atividades físicas. Serão utilizados comandos para obtenção do percentual dos valores de cada categoria, transformação dos dados do dicionário em  listas e, por fim, apresentação dos resultados em gráficos de barras horizontais.

**Figura 95**: Screenshot – Código Lista Estado Civil Praticante de Atividade Fisica.

```
: #exibe lista estado Civil
  df_pequisa2019SIMEstadoCivil = Counter(df_censoSelecionadas2019Sim["civil"])
  df_pequisa2019SIMEstadoCivil
```

```
: Counter({'casado legalmente': 12109,
           'solteiro': 9325,
           'separado ou divorciado': 2496,
           'tem união estável há mais de seis meses': 2481,
           'viúvo': 2985,
           'não quis informar': 83})
```

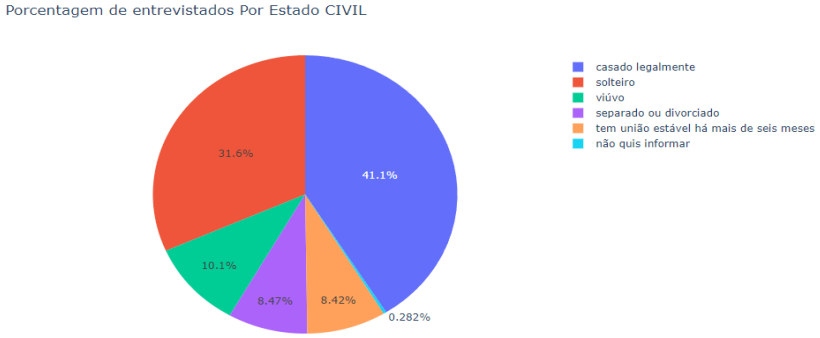
Fonte: Autor

**Figura 96**: Screenshot – Código da Plotagem Lista Estado Civil Praticante de Atividade Fisica.

```
: ##Plotagem das informações Atividade Fisica dos Entrevistados que praticam atividade porcentagem por Região

  import plotly.express as px
  pie = df_censoSelecionadas2019Sim["civil"].value_counts()
  regions = pie.index
  population = pie.values
  fig = px.pie(df_censoSelecionadas2019Sim, values=population, names=regions, title ="Porcentagem de entrevistados Por Regiao")
  fig.show()
```

Fonte: Autor

**Figura 97**: Screenshot – Plotagem Lista Estado Civil Praticante de Atividade Física.



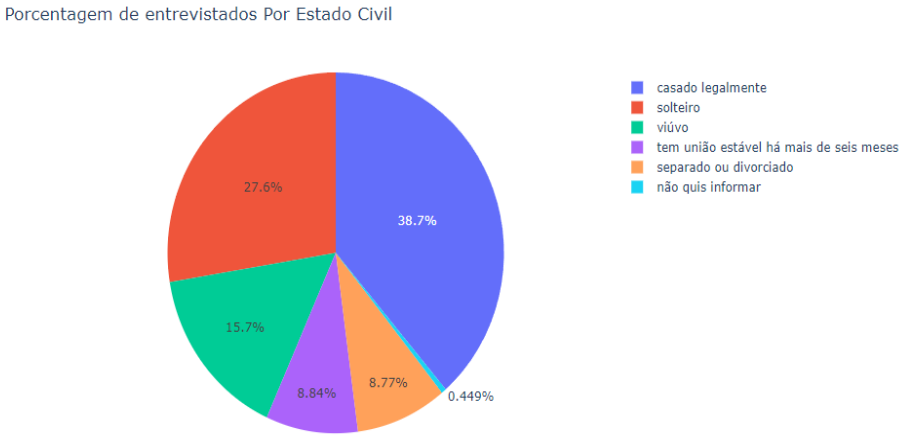

Fonte: Autor

**Figura 98**: Screenshot – Código da Plotagem Lista Estado Civil Não Praticante de Atividade Fisica.

```
##Plotagem das informações Atividade Fisica dos Entrevistados que Não praticam atividade porcentagem por Estado Civil

import plotly.express as px
pie = df_censoSelecionadas2019Nao["civil"].value_counts()
regions = pie.index
population = pie.values
fig = px.pie(df_censoSelecionadas2019Nao, values=population, names=regions, title ="Porcentagem de entrevistados Por Esta
fig.show()
```

Fonte: Autor

**Figura 99**: Screenshot – Plotagem Lista Estado Civil Não Praticante de Atividade Física.




Fonte: Autor

**Figura 100**: Screenshot – Código média idade  Estado civil

```
In [126]: df_casados =df_censoSelecionadas2019Sim.loc[df_censoSelecionadas2019Sim['civil']== 'casado legalmente']
          df_casados['idade'].describe()

Out[126]: count    12109.000000
          mean        56.967627
          std         13.927633
          min         18.000000
          25%         47.000000
          50%         58.000000
          75%         67.000000
          max         99.000000
          Name: idade, dtype: float64
```

Fonte: Autor

Percebe-se, por meio dos gráficos e percentuais que maioria dos entrevistados que praticam atividade fisica não são casados, com porcentagem de 58,9%, e que 41,15 para casados, número que quase mantém,  quando avaliamos os que não praticam atividade fisica, com porcentagem de  38,7%. Ao analisar essa informação comparada com a média de idade, é esperado que o número de entrevistados que praticam atividade casados seja a que se destaca, mas é interessante perceber que, mesmo a média de idade se mantendo entre os entrevistados que praticam atividades e os entrevistados que não praticam atividade , o percentual de casados apresenta uma alta de cerca de 8%.

Na última análise iremos analisar o critério escolaridade dos entrevistados que praticam atividade física e os entrevistados que não praticam nada. Para a visualização desses dados, optou-se pelo gráfico de barras horizontais. Para a construção desse tipo de gráfico foi necessário construir listas com os rótulos e os valores dos dados, criando a necessidade de alguns passos adicionais. O primeiro passo é a contagem dos valores de cada ocorrência conforme a figura abaixo.

**Figura 101**: Screenshot – Código Contagem escolaridade

```
#Contagem das opções da coluna grau_escolaridade dos entrevistados
df_escolaridade_praticanteAtividade = Counter(df_censoSelecionadas2019Sim['grau_escolaridade'])
df_escolaridade_praticanteAtividade
```

Fonte: Autor

**Figura 102**: Screenshot – Plotagem do resultado Escolaridade

```
Counter({'curso primário': 1837,
         'nunca estudou': 473,
         '1º grau ou fundamental ou supletivo de 1º grau': 3275,
         '3º grau ou curso superior': 10152,
         'não sabe': 182,
         'não quis responder': 112,
         'pós-graduação (especialização, mestrado, doutorado)': 3105,
         'curso ginasial ou ginásio': 529,
         'admissão': 68,
         '2º grau ou colégio ou técnico ou normal ou científico científico ou ensino médio ou supletivo de 2º grau': 9746})
```

Fonte: Autor

**Figura 103**: Screenshot – Código Identificação dos percentuais de escolaridade dos Entrevistados porcentagem

```
#Identificação dos percentuais de escolaridade dos Entrevistados
n_censoSelecionadas2019=sum(df_escolaridade_praticanteAtividade.values())
for x, y in df_escolaridade_praticanteAtividade.items():
    a = y/n_censoSelecionadas2019*100
    print(str(x) + ': ' +'\n'+ str(round(a,2)) + '%'+'\n')
```

Fonte: Autor

**Figura 104**: Screenshot – Plotagem Identificação dos percentuais de escolaridade dos Entrevistados porcentagem

```
curso primário:
6.23%

nunca estudou:
1.6%

1º grau ou fundamental ou supletivo de 1º grau:
11.11%

3º grau ou curso superior:
34.44%

não sabe:
0.62%

não quis responder:
0.38%

pós-graduação (especialização, mestrado, doutorado):
10.53%

curso ginasial ou ginásio:
1.79%

admissão:
0.23%

2º grau ou colégio ou técnico ou normal ou científico científico ou ensino médio ou supletivo de 2º grau:
33.06%
```

Fonte: Autor

**Figura 105**: Screenshot – Código Identificação dos percentuais de escolaridade dos Entrevistados porcentagem Não praticante de atividade Fisica

```
#Identificação dos percentuais de escolaridade dos Entrevistados
n_censoSelecionadas2019Nao=sum(df_escolaridade_Naopraticante.values())
for x, y in df_escolaridade_Naopraticante.items():
    a = y/n_censoSelecionadas2019Nao*100
    print(str(x) + ': ' +'\n'+ str(round(a,2)) + '%'+'\n')
```

Fonte: Autor

**Figura 106**: Screenshot – Plotagem Identificação dos percentuais de escolaridade dos entrevistados porcentagem Não praticante de atividade Fisica

```
curso primário:
13.04%

nunca estudou:
4.99%

1º grau ou fundamental ou supletivo de 1º grau:
16.59%

3º grau ou curso superior:
21.45%

não quis responder:
0.47%

não sabe:
1.48%

pós-graduação (especialização, mestrado, doutorado):
5.0%

curso ginasial ou ginásio:
3.24%

2º grau ou colégio ou técnico ou normal ou científico científico ou ensino médio ou supletivo de 2º grau:
33.24%

admissão:
0.49%
```

Fonte: Autor

**Figura 107**: Screenshot – Código construção da lista

```
: #Criação de listas para construção do gráfico de escolaridade
lista_df_escolaridade_praticanteAtividade_labels = []
lista_df_escolaridade_praticanteAtividade_valores = []
for x, y in df_escolaridade_praticanteAtividade.items():
    lista_df_escolaridade_praticanteAtividade_labels.append(x)
    lista_df_escolaridade_praticanteAtividade_valores.append(y)
```

Fonte: Autor

**Figura 108**: Screenshot – Código verificação da lista criada.

```
#Verificação da lista criada
lista_df_escolaridade_praticanteAtividade_valores
```

```
[1837, 473, 3275, 10152, 182, 112, 3105, 529, 68, 9746]
```

Fonte: Autor

**Figura 109**: Screenshot – Código e plotagem da lista criada

```
]: #Verificação da lista criada
lista_df_escolaridade_praticanteAtividade_labels
```

```
]: ['curso primário',
    'nunca estudou',
    '1º grau ou fundamental ou supletivo de 1º grau',
    '3º grau ou curso superior',
    'não sabe',
    'não quis responder',
    'pós-graduação (especialização, mestrado, doutorado)',
    'curso ginasial ou ginásio',
    'admissão',
    '2º grau ou colégio ou técnico ou normal ou científico científico ou ensino médio ou supletivo de 2º grau']
```

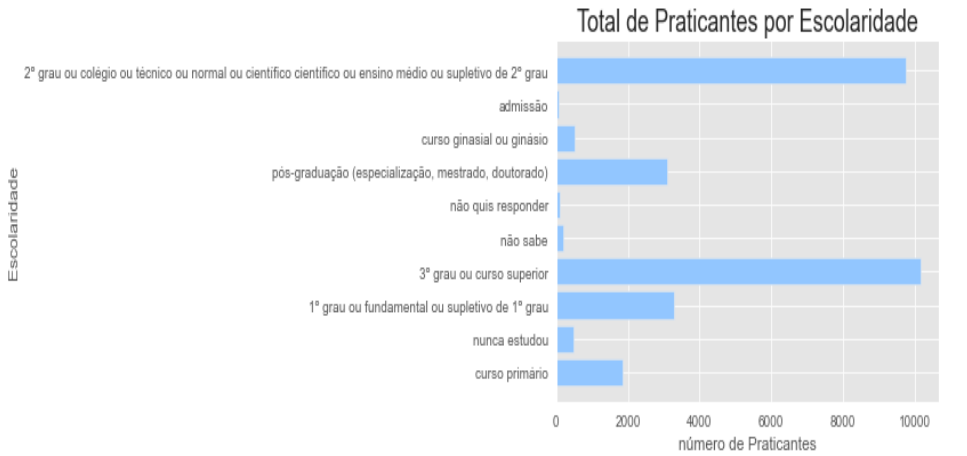
Fonte: Autor

**Figura 110**: Screenshot – Código da criação do gráfico escolaridade.

```
#Plotagem do gráfico de escolaridade dos Entrevistados
plt.style.use('seaborn-pastel')
plt.barh(lista_df_escolaridade_praticanteAtividade_labels, lista_df_escolaridade_praticanteAtividade_valores)
plt.ylabel('Escolaridade')
plt.xlabel('número de Praticantes')
plt.title('Total de Praticantes por Escolaridade', fontsize=18)
plt.show()
```

Fonte: Autor

**Figura 111**: Screenshot – Plotagem do gráfico escolaridade




Fonte: Autor

Nesse ponto, percebe-se mais uma vez uma grande discrepância entre o perfil dos entrevistados, que praticam e dos que não praticam atividade física. Enquanto apenas 5.0% do entrevistados não praticantes de atividade fisica possui pós- graduação (especialização, mestrado, doutorado) completo, esse percentual é de 10,53% para os entrevistados praticante de atividade fisica. Aqui é importante destacar que esse percentual de diferença escolaridade e muito importate para qualidade de vida, tendo em vista que, quanto maior a escolaridade maior o salário e assim melhorando a saúde e sobrando mais tempo para uma atividade fisica constante.

## 5. Criação de Modelos de Machine Learning

Após as análises realizadas nas seções anteriores, iremos aplicar modelos de Machine Learning, utilizando algoritmos de classificação sobre os dados. O principal objetivo, é identificar e classificar os atributos de maior importância para classificar se o entrevistado é um praticante de atividade fisica ou não praticante de atividade fisica.

Nesse estudo, serão utilizados 6 tipos de algoritmos de classificação, são eles: Árvore de Decisão, Regressão Logística, Naïve Bayes, Gradiente Descendente, KNN (K - Nearest Neighbors) e Randon Forest. A árvore de decisão é um algoritmo de aprendizado supervisionado adequado para problemas de classificação, pois é capaz de ordenar as classes em um nível preciso. Funciona

como um fluxograma, separando os pontos de dados em duas categorias semelhantes ao mesmo tempo, do “tronco da árvore” aos “galhos” e às “folhas”, onde as categorias se tornam mais finitamente semelhantes. Isso cria categorias dentro das categorias, permitindo a classificação orgânica com supervisão humana limitada. A vantagem de se utilizar a árvore de decisão é que ela é de simples entendimento e visualização, requer pouca preparação de dados e suporta tanto dados numéricos quando categóricos. Em contrapartida, podem ser criadas árvores complexas que não possuem uma boa generalização e podem ser instáveis devido a pequenas variações dos dados que podem criar uma árvore completamente diferente da anterior.

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

O algoritmo Naive Bayes é baseado no teorema de Bayes com a suposição de independência entre cada par de valores. A principal característica do algoritmo, e o motivo de receber “naive” (ingênuo) no nome, é que ele desconsidera completamente a correlação entre as variáveis. As principais vantagens deste algoritmo é que ele requer uma pequena quantidade de dados de treinamento para estimar os parâmetros necessários e é extremamente rápido em comparação com métodos mais sofisticados. A maior desvantagem é que ele é conhecido por não ser um bom avaliador.

O algoritmo gradiente descendente, é um dos algoritmos de maior sucesso em problemas de Machine Learning. O método consiste em encontrar, de forma interativa, os valores dos parâmetros que minimizam determinada função de interesse.

As vantagens desse algoritmo são sua eficiência e facilidade de implementação, mas ele requer vários hiperparâmetros e é sensível ao dimensionamento de recursos.

O algoritmo k-nearest neighbor, muitas vezes abreviado k-nn, é uma abordagem para classificação de dados que estima a probabilidade de um ponto de dados ser membro de um grupo ou de outro, dependendo de qual grupo os pontos de dados mais próximos a ele estão. Esse algoritmo é de fácil implementação, robusto a variações nos dados de treinamento e efetivo quando temos muitos dados de treinamento. O ponto negativo desse algoritmo é que exige muito recurso computacional.

Por fim, o algoritmo Random Forest ajusta árvores de decisão em várias subamostras de conjuntos de dados e usa a média para melhorar a precisão preditiva do modelo e controla o over-fitting. O tamanho da subamostra é sempre igual ao tamanho da amostra de entrada original, mas as amostras são retiradas com substituição. As dificuldades desse modelo são sua dificuldade de implementação e complexidade do algoritmo.

Para que se possa avaliar os algoritmos mencionados, optou-se por realizar alguns ajustes no dataframe “df_censoSelecionadas2019FinalTreinamento”. Não precisaremos tranformar as colunas em binários pois a mesma ja esta vindo do dataset ja tratado. A coluna pratica_exercicio, civil(estado civil), sexo, cor, grau_escolaridade e binária e tem como referencia se o entrevistado pratica ou não atividade fisica.

**Figura 112**: Screenshot – Colunas que vão ser usadas.

```
#Exibe os valores do Data Frame
df_censoSelecionadas2019FinalTreinamento.head()
```

| | ordem | ano | cidade | civil | idade | sexo | grau_escolaridade | peso | altura | pratica_exercicio | tipo_exercicio | pratica_exercicio_1_vez_na_sema |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 158639 | 0 | 227 | 27 | 5 | 88 | 1 | 1 | 74 | 168 | 0 | 0 | |
| 158640 | 0 | 227 | 27 | 2 | 65 | 1 | 8 | 57 | 159 | 0 | 0 | |
| 158641 | 0 | 227 | 10 | 2 | 68 | 1 | 1 | 9 | 180 | 0 | 0 | |
| 158642 | 0 | 227 | 10 | 2 | 79 | 1 | 1 | 62 | 170 | 1 | 1 | |
| 158643 | 0 | 227 | 10 | 2 | 70 | 1 | 1 | 66 | 162 | 1 | 1 | |

Fonte: Autor

**Figura 113**: Screenshot – Exibe a quantidade de cada entrevistado praticante e não praticante de atividade fisica

```
#total Pesquisa
pequisa2019Pra = Counter(df_censoSelecionadas2019FinalTreinamento['pratica_exercicio'])
pequisa2019Pra
```

```
Counter({0: 22964, 1: 29479})
```

Fonte: Autor

Na figura 113 acima temos:

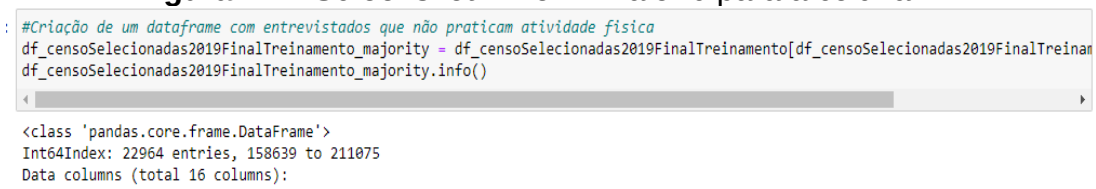
- Não é um Praticante de atividade física: 22.964
- É Praticante de atividade física: 29.479

Temos 22.964 entrevistados classificados como “Não praticante de atividade fisica ”, e 29.479 classificado como praticante de atividade.

Observamos que existe pequeno desbalanceamento entre os entrevistados.

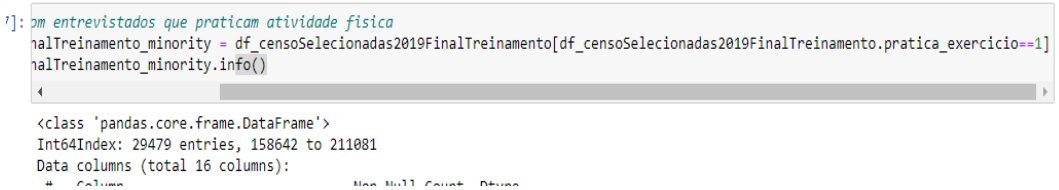
Em seguida o dataframe será dividido entre as entradas que possuem valor 0 e valor 1 na coluna pratica_exercicio:

**Figura 114**: Screenshot – Definir valor 0 para a coluna




Fonte: Autor

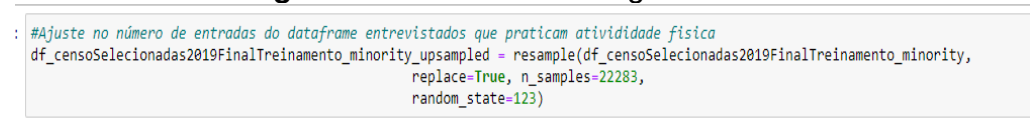
**Figura 115**: Screenshot – Definir valor 1 para a coluna




Fonte: Autor

O objetivo agora, é igualar a quantidade de entradas desses novos *dataframes*, aumentando a quantidade de registros do que possui menor número. Esse processo é feito pelo comando a seguir:

**Figura 116**: Screenshot – iguala dados




Fonte: Autor

**Figura 117**: Screenshot – Exibe informação do dados

```
#Informações do dataframe ajustado
df_censoSelecionadas2019FinalTreinamento_minority_upsampled.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 22283 entries, 194083 to 210332
Data columns (total 16 columns):
```

Fonte: Autor

O próximo passo para a aplicação dos algoritmos de classificação é dividir o dataframe em bases de treinamento e de teste.

Antes de utilizar a função mencionada, o dataframe será dividido em dois: um onde consta apenas os atributos que serão analisados e outra que mostra se o candidato foi ou não eleito. O dataframe que contém todos os atributos será chamado do de X_train e o que contém o resultado será chamado de y_train. Importante perceber que y_train agora é uma série de valores.

**Figura 118**: Screenshot – Divisão de treinamento da Bases

```
X_train = df_censoSelecionadas2019FinalTreinamento_upsampled.drop(['pratica_exercicio'], axis = 1)
y_train = df_censoSelecionadas2019FinalTreinamento_upsampled.pratica_exercicio
```

Fonte: Autor

**Figura 119**: Screenshot – Exibe informação do treinamento

```
#Informações do dataframe com os dados para treinamento
X_train.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 45247 entries, 158639 to 210332
Data columns (total 15 columns):
 #   Column                              Non-Null Count  Dtype
```

Fonte: Autor

Agora será feita a divisão entre bases de treinamento e de teste utilizando a função “train_test_split”. Para a divisão de percentual de dados para treinamento e para teste, será utilizado o padrão da função, que é de 75% para treinamento e 25% para teste.

**Figura 120**: Screenshot – Exibe tipo da serie

```
Em [68]: #Tipo da serie y_train
         type(y_train)

ora[68]: pandas.core.series.Series
```

Fonte: Autor

**Figura 121**: Screenshot – Treinamento

```
#Criação das bases de teste e treinamento
xtreinamento, xteste, ytreinamento, yteste = train_test_split(X_train, y_train, random_state =0)
```

Fonte: Autor

**Figura 122**: Screenshot – Exibe informação Data Frame Treinamento.

```
: #Informação do dataframe de treinamento
  xtreinamento.info()

<class 'pandas.core.frame.DataFrame'>
Int64Index: 33935 entries, 189441 to 165052
Data columns (total 15 columns):
```

Fonte: Autor

**Figura 123**: Screenshot – Exibe de Teste

```
#Informação do dataframe de teste
xteste.info()

<class 'pandas.core.frame.DataFrame'>
Int64Index: 11312 entries, 165890 to 180866
Data columns (total 15 columns):
 #   Column                                  Non-Null Cour
```

Fonte: Autor

**Figura 124**: Screenshot – Contagem resultados treinamento

```
: #Contagem dos resultados para treinamento
  ytreinamento.count()

: 33935
```

Fonte: Autor

**Figura 125**: Screenshot – Exibe informação Contagem teste

```
Em [56]: #Contagem dos resultados para teste
         yteste.count()

Fora[56]: 11312
```

Fonte: Autor

Antes de se aplicar os algoritmos de classificação, é necessário definir qual medida de avaliação será analisada para verificar a eficiência do modelo. Com esse objetivo, serão utilizadas duas funções da biblioteca Scikit-learn: “accuracy_score” e “classification_report”. Essas funções mostram as medidas de acurácia, precisão, revocação (recall) e f1-score.

A acurácia é a quantidade de acertos do modelo dividido pelo total da amostra e indica uma performance geral do modelo:

$$Acurácia = \frac{Verdadeiros\ Positivos + Verdadeiros\ Negativos}{Total\ de\ Amostras}$$

A precisão define os chamados positivos verdadeiros, ou seja, dentre os exemplos classificados como verdadeiros, quantos eram realmente verdadeiros.

$$Precisão = \frac{Verdadeiros\ Positivos}{Verdadeiros\ Positivos + Falsos\ Positivos}$$

A revocação (recall) indica qual a porcentagem de dados classificados como verdadeiros comparado com a quantidade real de resultados verdadeiros que existem na amostra.

$$Revocação = \frac{Verdadeiros\ Positivos}{Verdadeiros\ Positivos + Falsos\ Negativos}$$

A F1-score traz a média ponderada de precisão e revocação e traz um número único que determina a qualidade geral do modelo.

$$F1 - score = \frac{2 * Precisão\ * Revocação}{Precisão + Revocação}$$

A principal medida de avaliação que será utilizada nesse trabalho é a acurácia, porém todas as outras também serão consideradas na avaliação.

Definidos os dataframes de treinamento e de teste e escolhidas as medidas de avaliação, o próximo passo é a efetiva utilização dos algoritmos listados anteriormente. O processo para todos será o mesmo, começando pela importação do respectivo algoritmo, realizando o treinamento por meio da função “fit” nas duas

bases de treinamento e registrando sua acurácia por meio da função score. Em seguida, será utilizada a função “predict” na base teste xteste e sua saída será comparada com a série yteste, tendo suas medidas de avaliação sendo geradas por meio das funções “accuracy_score” e “classification_report”.

**Figura 126**: Screenshot – Modelo utilizado Árvore de decisão.

```
: #Criação do modelo utilizando a Árvore de decisão
from sklearn.tree import DecisionTreeClassifier
from sklearn import tree
df_censoSelecionadas2019FinalTreinamento_tree = DecisionTreeClassifier(random_state=0)
df_censoSelecionadas2019FinalTreinamento_tree = df_censoSelecionadas2019FinalTreinamento_tree.fit(xtreinamento, ytreinamento)
print("Acurácia: ", df_censoSelecionadas2019FinalTreinamento_tree.score(xtreinamento, ytreinamento))
Train_predict = df_censoSelecionadas2019FinalTreinamento_tree.predict(xteste)
print("Acurácia de previsão: ", accuracy_score(yteste, Train_predict))
print(classification_report(yteste, Train_predict))
```

```
Acurácia:  1.0
Acurácia de previsão:  1.0
              precision    recall  f1-score   support

           0       1.00      1.00      1.00      5708
           1       1.00      1.00      1.00      5604

    accuracy                           1.00     11312
   macro avg       1.00      1.00      1.00     11312
weighted avg       1.00      1.00      1.00     11312
```

Fonte: Autor

**Figura 127**: Screenshot – Modelo Regressão Logística

```
#Criação do modelo utilizando a Regressão Logística
from sklearn.linear_model import LogisticRegression
lr = LogisticRegression()
lr = lr.fit(xtreinamento, ytreinamento)
print("Acurácia: ", lr.score(xtreinamento, ytreinamento))
tp_lr = lr.predict(xteste)
print("Acurácia de previsão: ", accuracy_score(yteste, tp_lr))
print(classification_report(yteste, tp_lr))
```

```
Acurácia:  1.0
Acurácia de previsão:  1.0
              precision    recall  f1-score   support

           0       1.00      1.00      1.00      5708
           1       1.00      1.00      1.00      5604

    accuracy                           1.00     11312
   macro avg       1.00      1.00      1.00     11312
weighted avg       1.00      1.00      1.00     11312
```

Fonte: Autor

**Figura 128**: Screenshot –Validação Regressão Logística

```
: #Validação cruzada para o modelo Regressão Logística
from sklearn.model_selection import cross_val_score
validacao_arvore = cross_val_score(lr, X_train, y_train, scoring='accuracy', cv=5)
print(validacao_arvore.mean())
```

```
1.0
```

Fonte: Autor

**Figura 129**: Screenshot – Modelo Naive Bayes

```
1]: #Criação do modelo utilizando Naïve Bayes
    from sklearn.naive_bayes import GaussianNB
    nb = GaussianNB()
    nb = nb.fit(xtreinamento, ytreinamento)
    print("Acurácia: ", nb.score(xtreinamento, ytreinamento))
    tp_nb = nb.predict(xteste)
    print("Acurácia de previsão: ", accuracy_score(yteste, tp_nb))
    print(classification_report(yteste, tp_nb))
```

```
Acurácia:  1.0
Acurácia de previsão:  1.0
              precision    recall  f1-score   support

           0       1.00      1.00      1.00      5708
           1       1.00      1.00      1.00      5604

    accuracy                           1.00     11312
   macro avg       1.00      1.00      1.00     11312
weighted avg       1.00      1.00      1.00     11312
```

Fonte: Autor

**Figura 130**: Screenshot – Validação Cruzada Naive Bayes

```
: #Validação cruzada para o modelo Naive bayes
  from sklearn.model_selection import cross_val_score
  validacao_arvore = cross_val_score(nb, X_train, y_train, scoring='accuracy', cv=5)
  print(validacao_arvore.mean())
```

```
1.0
```

Fonte: Autor

**Figura 131**: Screenshot – Modelo Gradiente Descendente

```
Em [63]: #Criação do modelo utilizando Gradiente Descendente
         from sklearn.linear_model import SGDClassifier
         sgd = SGDClassifier()
         sgd = sgd.fit(xtreinamento, ytreinamento)
         print("Acurácia: ", sgd.score(xtreinamento, ytreinamento))
         tp_sgd = sgd.predict(xteste)
         print("Acurácia de previsão: ", accuracy_score(yteste, tp_sgd))
         print(classification_report(yteste, tp_sgd))
```

```
Acurácia:  1.0
Acurácia de previsão:  0.9999115983026874
              precision    recall  f1-score   support

           0       1.00      1.00      1.00      5708
           1       1.00      1.00      1.00      5604

    accuracy                           1.00     11312
   macro avg       1.00      1.00      1.00     11312
weighted avg       1.00      1.00      1.00     11312
```

Fonte: Autor

**Figura 132**: Screenshot – Modelo KNN

```
#Criação do modelo utilizando KNN (K - Nearest Neighbors)
from sklearn.neighbors import KNeighborsClassifier
knn = KNeighborsClassifier()
knn = knn.fit(xtreinamento, ytreinamento)
print("Acurácia: ", knn.score(xtreinamento, ytreinamento))
tp_knn = knn.predict(xteste)
print("Acurácia de previsão: ", accuracy_score(yteste, tp_knn))
print(classification_report(yteste, tp_knn))
```

```
Acurácia:  0.9689111536761456
Acurácia de previsão:  0.9489922206506365
              precision    recall  f1-score   support

           0       0.92      0.99      0.95      5708
           1       0.99      0.91      0.95      5604

    accuracy                           0.95     11312
   macro avg       0.95      0.95      0.95     11312
weighted avg       0.95      0.95      0.95     11312
```

Fonte: Autor

**Figura 133**: Screenshot – Validação cruzada Gradiente Descendente

```
5]: #Faz a validação cruzada com 5 folds e ao final exibe a média da acurácia
    validacao_knn = cross_val_score(knn, X_train, y_train, scoring='accuracy', cv=5)
    print(validacao_knn.mean())
```

```
0.9528808030434668
```

Fonte: Autor

**Figura 134**: Screenshot – Modelo Radon Forest

```
Em [66]: #Criação do modelo utilizando Randon Forest
         from sklearn.ensemble import RandomForestClassifier
         rfm = RandomForestClassifier()
         rfm = rfm.fit(xtreinamento, ytreinamento)
         print("Acurácia: ", rfm.score(xtreinamento, ytreinamento))
         tp_rfm = rfm.predict(xteste)
         print("Acurácia de previsão: ", accuracy_score(yteste, tp_rfm))
         print(classification_report(yteste, tp_rfm))
```

```
Acurácia:  1.0
Acurácia de previsão:  1.0
              precision    recall  f1-score   support

           0       1.00      1.00      1.00      5708
           1       1.00      1.00      1.00      5604

    accuracy                           1.00     11312
   macro avg       1.00      1.00      1.00     11312
weighted avg       1.00      1.00      1.00     11312
```

Fonte: Autor

**Figura 135**: Screenshot – Validação Cruzada Random Forest

```
: #Validação cruzada do classificador Random Forest
validacao_rf = cross_val_score(rfm, X_train, y_train, scoring='accuracy', cv=5)
print(validacao_rf.mean())
```

```
1.0
```

## 6. Apresentação dos Resultados

Nessa seção será apresentado os resultados obtidos. Para exemplificar foi desenvolvido o modelo Canvas proposto pelo Vasandani (clique aqui).

| **Título: PRATICA DE ATIVIDADE FISICA ENTRE ADULTOS BRASILEIROS NAS CAPITAIS** | | |
|---|---|---|
| **Problema**<br><br>Analisar o dataset de prática de atividade fisica nas capitais da base da VIGITEL para investigar atributos relacionados a pratica de atividade fisica , se pratica ou não e qual atividade física é praticada como outros atributos. | **Resultados e Previsões**<br><br>Avaliar os atributos relacionados a pratica ”Sim” ou “Não”de atividade fisica atributo, com a finalidade de tentar prever e classificar os atributos de maior importância para entender quais atividades fisicas são praticas em cada capital. | **Aquisição de Dados**<br><br>Os dados de ambos os datasets formato xls foram coletados do site da Vigitel e IBGE. |
| **Modelagem**<br><br>Realizado análises no dataset coletado, tanto de forma gráfica quanto análise descritiva dos dados utilizando a biblioteca *Pandas* em *Pyrthon.* Desta forma foi possível identificar um dataset adequado para aplicar modelo de classificação de ML. | **Avaliação do Modelo**<br><br>Para avaliação dos resultados obtidos no modelo de classificação, foram avaliados a Matriz de Confusão e o Relatório de Classificação conforme o notebook em Python no diretório deste projeto. | **Preparação dos Dados**<br><br>Após a união dos datasets, os dados foram tratados, as colunas foram renomeadas, os dados duplicados foram removidos e dados desnecessários para a análise também foram removidos. |

Os resultados médios dos algoritmos podem ser verificados na tabela e gráfico a seguir:

| Algoritmo | Acurácia de previsão | Precisão | Revocação | F1-Score |
|---|---|---|---|---|
| Árvore de Decisão | 1,0 | 1,0 | 1,0 | 1,0 |
| Regressão Logística | 1,0 | 1,0 | 1,0 | 1,0 |
| Naïve Bayes | 1,0 | 1,0 | 1,0 | 1,0 |
| Gradiente Descendente | 1,0 | 1,0 | 1,0 | 1,0 |
| KNN | 0,95 | 0,95 | 0,95 | 0,95 |
| Randon Forest | 1,0 | 1,0 | 1,0 | 1,0 |

Analisando as informações da tabela, percebe-se que cinco algoritmos se destacam: a árvore de decisão, Regressão Logistica, Naive Bayes Gradiente Descente o Random Forest. Eles apresentam resultados muito bons em todas as medidas de avaliação. com uma pequena desvantagem para o algoritmo KNN, que apresenta um resultado de 0,95 para todas elas.

A análise mais detalhada será feita nos resultados um dos algoritmos que apresentou melhores resultados, ou seja, Árvore de Decisão. Ao se analisar a precisão dos resultados individuais, o valor obtido para a previsão dos Entrevistados que não praticam atividade física chama muito a atenção, já que apresenta o valor 1,00, ou seja, o algoritmo não identificou nenhum falso positivo no momento de identificar entrevistados que não praticam atividade fisica. O valor apresentado para os valores para os entrevistados que praticam atividade física também chama a atenção, pois já traz o valor de 1,0 ou seja, a cada 100 previsões de que o entrevistado que praticariam atividade fisica, não identificou nenhum falso positivo.

A interpretação dos resultados de revocação são similares, com a diferença de que o item que apresenta resultado 1,00 é o de entrevistados praticantes de atividade fisica, ou seja, não se identificou nenhum falso negativo.

Por fim, analisando a acurácia de previsão, temos o valor de 1,0 ou seja, de todas as amostras, o algoritmo acerta em 100% das vezes, afirmando que um entrevistado será ou não praticante de atividade fisica.

Os resultados apontam um aumento na prática de atividade física no lazer entre adultos nas capitais de estados do país e no Distrito Federal, com maior prevalência nas cidades de Florianópolis, Distrito Federal e Teresina e em menor prevalência na cidade Macapá e São Paulo visto o PIB e Tamanho da população. As regiões Nordeste, Centro Oeste e Norte apresentaram os melhores índices de

praticante de atividade fisica e as regiões Sul e Sudeste os menores índices no período analisado. Esse aumento foi especialmente relevante entre os mulheres na faixa de 40 a 70 anos e aqueles indivíduos com maior nível de escolaridade.

Os resultados evidenciados neste estudo corroboram importantes discussões no que se refere à criação de estratégias e condições apropriadas, não só para aumentar a frequência da prática de atividade física no lazer na população brasileira de uma forma geral, mas especialmente para a população que apresenta menor frequência desta prática, como os homens, os jovens , as pessoas de baixa escolaridade e condições socioeconômicas precárias.

Esta reflexão deve perpassar pelos profissionais de saúde, gestores púbicos e sociedade civil organizada. Nesse sentido, programas populacionais e políticas de promoção da saúde voltados para incentivar a prática de atividade física na comunidade e nas escolas, assim como, políticas públicas na área do planejamento urbano e ambiente, são estratégias potencializadoras para aumentar os níveis de atividade física na população e para promoção de hábitos de vida mais saudáveis.

## 7. Links

**Link para o vídeo:**
https://www.youtube.com/watch?v=LQDJgICJ6Xs

**Link para o repositório Github (sem datasets):**
https://github.com/Gabriellr/TCC_CienciadeDados_PUCMINAS

## 8. Referencias

PYTHON. Disponível em: <https://www.python.org/>.
JUPYTER. Disponível em: <https://jupyter.org/>.
SEABORN. Disponível em: <https://seaborn.pydata.org/>.

World Health Organization (2018). **Global action plan on physical activity 2018–2030: more active people for a healthier world.** https://www.who.int/publications/i/item/9789241514187. ISBN 978-92-4-151418-7.

Bull, F., Goenka, S., Lambert, V. & Pratt, M. (2017). **Physical Activity for the**

**Prevention of Cardiometabolic Disease**. In: Prabhakaran D, Anand S, Gaziano TA, Mbanya JC, Wu Y, Nugent R, editors. Cardiovascular, Respiratory, and Related Disorders. (3a ed.) Washington (DC): The International Bank for Reconstruction and Development / The World Bank. 25. 10.1596/978-1-4648-0518-9_ch5

Guthold, R., Stevens, G. A., Riley, L. M., & Bull, F. C. (2018). **Worldwide trends in insufficient physical activity from 2001 to 2016:** a pooled analysis of 358 population-based surveys with 1·9 million participants. Lancet Global Health. 6(10),1077-1086. https://doi.10.1016/S2214-109X(18)30357-7

Silva, C. L., Souza, M. F., Rossi Filho, S., Silva, L. F., & Rigoni, A. C. C. (2017). **Atividade física de lazer e saúde: uma revisão sistemática. Mudanças – Psicologia da Saúde**. 25(1),57-65. https://doi.org/10.15603/2176-1019/mud.v25n1p57-65

**Documentação Scikit-learn, Machine Learning in Python, Random Forest Classifier**, disponível em <https://scikitlearn.org/stable/modules/generated/sklearn.ensemble.RandomForestClassifier.html> Acesso em: 29/03/2022

## APÊNDICE

### Programação/Scripts

```
#Importação da biblioteca pandas
import pandas as pd
import numpy as np

import matplotlib.pyplot as plt
import seaborn as sns
sns.set_style('whitegrid')

from datetime import date
from collections import Counter
```

```
from sklearn.model_selection import train_test_split
from sklearn.metrics import mean_squared_error, mean_absolute_error

import seaborn as sb
from matplotlib import pyplot
import statsmodels.api as sm
import math

import operator

#Importação das funções para as medidas de avaliação dos algoritmos
from sklearn.metrics import accuracy_score, classification_report
from sklearn.naive_bayes import GaussianNB

from sklearn.preprocessing import LabelEncoder
from sklearn.metrics import confusion_matrix, classification_report
from sklearn.metrics import ConfusionMatrixDisplay

from sklearn.tree import DecisionTreeClassifier
from sklearn.tree import plot_tree
import warnings
warnings.filterwarnings('ignore')

##leitura dos 4 arquivos  XLS que contém os dados para os dataframes;
df_atividade_2016= pd.read_excel('Vigitel-2016-peso-rake.xls')
df_atividade_2017= pd.read_excel('Vigitel-2017-peso-rake.xls')
df_atividade_2018= pd.read_excel('Vigitel-2018-peso-rake.xls')
df_atividade_2019= pd.read_excel('Vigitel-2019-peso-rake.xls')

# juntando os dataframes para um unico dataframe
df_atividade_junt = pd.merge(df_atividade_2016, df_atividade_2017, how = 'outer')

# juntando os dataframes para um unico dataframe
df_atividade_junt2 = pd.merge(df_atividade_junt, df_atividade_2018, how = 'outer')

# juntando os dataframes para um unico dataframe
df_atividade = pd.merge(df_atividade_junt2, df_atividade_2019, how = 'outer')

#informações do dataframe df_atividade
df_atividade.info()

#identificação das opções da coluna cidade do dataframe df_atividade
df_atividade['cidade'].unique()

#Exibe as 1001 primeiras linhas
df_atividade.head(1001)

#Informações do dataframe df_atividade
df_atividade.info()


#Verificação dos valores nulos no dataframe df_atividade
df_atividade.isnull().sum()
```

```
#trata valor nulo dataframe
df_atividade= df_atividade.drop(df_atividade.index[0])
df_atividade = df_atividade.fillna(0)

# verifica o tratamento do nulo
df_atividade.isnull().sum()

#Exibe todas as colunas do dataframe df_atividades
df_atividade.columns.values

# lista das colunas que vão ser usadas ColunasSelecionada
ColunasSelecionada = ['ordem',  'ano', 'cidade', 'civil','q6', 'q7', 'q8a', 'q9', 'q11', 'q42', 'q43a', 'q44', 'q45','q46',
'q69']

#Filtrar para o dataframe novo somente as colunas que foram selecionadas
df_atividadeSelecionadas = df_atividade.filter(items=ColunasSelecionada )

#Exibe Dataframe df_atividadeSelecionadas

df_atividadeSelecionadas.head()

#Renomeia as coluna
df_atividadeSelecionadas.rename(columns={'q6':'idade','q7':'sexo','q8a':'grau_escolaridade','q9':'peso','q11':'
altura','q42':'pratica_exercicio','q43a': 'tipo_exercicio','q44':'pratica_exercicio_1_vez_na_semana',
'q45':'frequencia_exercicio', 'q46':'duracao_exercicio', 'q69':'cor'}, inplace =True)
Cria coluna com o calculo IMC dataframe df_atividadeSelecionadas

#criar coluna com o calculo IMC dataframe df_atividadeSelecionadas
df_atividadeSelecionadas['imc'] = df_atividadeSelecionadas.apply(
    lambda row: round(row.peso / (((row.altura * row.altura)/1000)*0.1),0), axis=1)

Dataframe df_censoSelecionadas2019Final vai ser usado no Treinamento

df_censoSelecionadas2019FinalTreinamento =
df_atividadeSelecionadas.loc[df_atividadeSelecionadas["ano"]== 2019]
df_censoSelecionadas2019Final = df_atividadeSelecionadas.loc[df_atividadeSelecionadas["ano"]== 2019]

df_censoSelecionadas2019FinalTreinamento.head()

#formate a coluna pratica exercicio 1 para sim e 2 para não
df_atividadeSelecionadas['pratica_exercicio'].replace(1,'sim', inplace =True)
df_atividadeSelecionadas['pratica_exercicio'].replace(2,'nao', inplace =True)

#formate a coluna tipo exercicio
1 caminhada (não vale deslocamento para trabalho)
2 caminhada em esteira
3 corrida (cooper)
4 corrida em esteira
5 musculação
6 ginástica aeróbica (spinning, step, jump)
7 hidroginástica
8 ginástica em geral (alongamento, pilates, ioga)
9 natação
```

10 artes marciais e luta (jiu-jitsu, karatê, judô, boxe, muay thai, capoeira)
11 bicicleta (inclui ergométrica)
12 futebol/futsal
13 basquetebol
14 voleibol/futevolei
15 tênis
16 dança (balé, dança de salão, dança do ventre)
17 outros

```
df_atividadeSelecionadas['tipo_exercicio'].replace(1,'caminhada (não vale deslocamento para trabalho)', inplace =True)
df_atividadeSelecionadas['tipo_exercicio'].replace(2,'caminhada em esteira', inplace =True)
df_atividadeSelecionadas['tipo_exercicio'].replace(3,'corrida (cooper)', inplace =True)
df_atividadeSelecionadas['tipo_exercicio'].replace(4,'corrida em esteira', inplace =True)
df_atividadeSelecionadas['tipo_exercicio'].replace(5,'musculação', inplace =True)
df_atividadeSelecionadas['tipo_exercicio'].replace(6,'ginástica aeróbica (spinning, step, jump)', inplace =True)
df_atividadeSelecionadas['tipo_exercicio'].replace(7,'hidroginástica', inplace =True)
df_atividadeSelecionadas['tipo_exercicio'].replace(8,'ginástica em geral (alongamento, pilates, ioga)', inplace =True)
df_atividadeSelecionadas['tipo_exercicio'].replace(9,'natação', inplace =True)
df_atividadeSelecionadas['tipo_exercicio'].replace(10,'artes marciais e luta (jiu-jitsu, karatê, judô, boxe, muay thai, capoeira)', inplace =True)
df_atividadeSelecionadas['tipo_exercicio'].replace(11,'bicicleta (inclui ergométrica)', inplace =True)
df_atividadeSelecionadas['tipo_exercicio'].replace(12,'futebol/futsal', inplace =True)
df_atividadeSelecionadas['tipo_exercicio'].replace(13,'basquetebol', inplace =True)
df_atividadeSelecionadas['tipo_exercicio'].replace(14,'voleibol/futevolei', inplace =True)
df_atividadeSelecionadas['tipo_exercicio'].replace(15,'tênis', inplace =True)
df_atividadeSelecionadas['tipo_exercicio'].replace(16,'dança (balé, dança de salão, dança do ventre)', inplace =True)
df_atividadeSelecionadas['tipo_exercicio'].replace(17,'outros', inplace =True)
```

#formate a coluna grau escolaridade

1 curso primário
2 admissão
3 curso ginasial ou ginásio
4 1º grau ou fundamental ou supletivo de 1º grau
5 2º grau ou colégio ou técnico ou normal ou científico científico ou ensino médio ou supletivo de 2º au
6 3º grau ou curso superior
7 pós-graduação (especialização, mestrado, doutorado)
8 nunca estudou
777 não sabe
888 não quis responder

```
df_atividadeSelecionadas['grau_escolaridade'].replace(1,'curso primário', inplace =True)
df_atividadeSelecionadas['grau_escolaridade'].replace(2,'admissão', inplace =True)
df_atividadeSelecionadas['grau_escolaridade'].replace(3,'curso ginasial ou ginásio', inplace =True)
df_atividadeSelecionadas['grau_escolaridade'].replace(4,'1º grau ou fundamental ou supletivo de 1º grau', inplace =True)
df_atividadeSelecionadas['grau_escolaridade'].replace(5,'2º grau ou colégio ou técnico ou normal ou científico científico ou ensino médio ou supletivo de 2º grau', inplace =True)
df_atividadeSelecionadas['grau_escolaridade'].replace(6,'3º grau ou curso superior', inplace =True)
```

```
df_atividadeSelecionadas['grau_escolaridade'].replace(7,'pós-graduação (especialização, mestrado, doutorado)', inplace =True)
df_atividadeSelecionadas['grau_escolaridade'].replace(8,'nunca estudou', inplace =True)
df_atividadeSelecionadas['grau_escolaridade'].replace(777,'não sabe', inplace =True)
df_atividadeSelecionadas['grau_escolaridade'].replace(888,'não quis responder', inplace =True)

#formate a coluna quantas vezes semana' 1 para sim e 2 para não
df_atividadeSelecionadas['pratica_exercicio_1_vez_na_semana'].replace(1,'sim', inplace =True)
df_atividadeSelecionadas['pratica_exercicio_1_vez_na_semana'].replace(2,'nao', inplace =True)

#formate a coluna Estado Civil

df_atividadeSelecionadas['civil'].replace(1,'solteiro', inplace =True)
df_atividadeSelecionadas['civil'].replace(2,'casado legalmente', inplace =True)
df_atividadeSelecionadas['civil'].replace(3,'tem união estável há mais de seis meses', inplace =True)
df_atividadeSelecionadas['civil'].replace(4,'viúvo', inplace =True)
df_atividadeSelecionadas['civil'].replace(5,'separado ou divorciado', inplace =True)
df_atividadeSelecionadas['civil'].replace(888,'não quis informar', inplace =True)


#formate a coluna frequencia exercicio
df_atividadeSelecionadas['frequencia_exercicio'].replace(1,'1 a 2 dias por semana', inplace =True)
df_atividadeSelecionadas['frequencia_exercicio'].replace(2,'3 a 4 dias por semana', inplace =True)
df_atividadeSelecionadas['frequencia_exercicio'].replace(3,'5 a 6 dias por semana', inplace =True)
df_atividadeSelecionadas['frequencia_exercicio'].replace(4,'todos os dias ( inclusive sábado e domingo)', inplace =True)

#formate a coluna duracao exercicio
df_atividadeSelecionadas['duracao_exercicio'].replace(1,'menos que 10 minutos', inplace =True)
df_atividadeSelecionadas['duracao_exercicio'].replace(2,'entre 10 e 19 minutos', inplace =True)
df_atividadeSelecionadas['duracao_exercicio'].replace(3,'entre 20 e 29 minutos', inplace =True)
df_atividadeSelecionadas['duracao_exercicio'].replace(4,'entre 30 e 39 minutos', inplace =True)
df_atividadeSelecionadas['duracao_exercicio'].replace(5,'entre 40 e 49 minutos', inplace =True)
df_atividadeSelecionadas['duracao_exercicio'].replace(6,'entre 50 e 59 minutos', inplace =True)
df_atividadeSelecionadas['duracao_exercicio'].replace(7,'60 minutos ou mais', inplace =True)

#formate a coluna cor
df_atividadeSelecionadas['cor'].replace(1,'branca', inplace =True)
df_atividadeSelecionadas['cor'].replace(2,'preta', inplace =True)
df_atividadeSelecionadas['cor'].replace(3,'amarela', inplace =True)
df_atividadeSelecionadas['cor'].replace(4,'parda', inplace =True)
df_atividadeSelecionadas['cor'].replace(5,'indígena', inplace =True)
df_atividadeSelecionadas['cor'].replace(80,'Morena', inplace =True)
df_atividadeSelecionadas['cor'].replace(777,'não sabe', inplace =True)
df_atividadeSelecionadas['cor'].replace(888,'não quis informar', inplace =True)

#formate a coluna quantas vezes semana' 1 para masculino e 2 para feminino
df_atividadeSelecionadas['sexo'].replace(1,'masculino', inplace =True)
df_atividadeSelecionadas['sexo'].replace(2,'feminino', inplace =True)

#Verifica o tipo do dados no Dataframe
df_atividadeSelecionadas.info()
```

```
#Ajustando os Dados para os Formatos Corretos e Otimização dos Tipos de dados para economizar memoria

df_atividadeSelecionadas.ordem   = df_atividadeSelecionadas.ordem.astype('int32')
df_atividadeSelecionadas.ano     = df_atividadeSelecionadas.ano.astype('int32')
df_atividadeSelecionadas.cidade  = df_atividadeSelecionadas.cidade.astype('int32')
df_atividadeSelecionadas.civil  = df_atividadeSelecionadas.civil.astype('category')
df_atividadeSelecionadas.idade  = df_atividadeSelecionadas.idade.astype('int32')
df_atividadeSelecionadas.sexo   = df_atividadeSelecionadas.sexo.astype('category')
df_atividadeSelecionadas.grau_escolaridade  =
df_atividadeSelecionadas.grau_escolaridade.astype('category')
df_atividadeSelecionadas.peso   = df_atividadeSelecionadas.peso.astype('float64')
df_atividadeSelecionadas.altura = df_atividadeSelecionadas.altura.astype('int32')
df_atividadeSelecionadas.pratica_exercicio  =
df_atividadeSelecionadas.pratica_exercicio.astype('category')
df_atividadeSelecionadas.tipo_exercicio     = df_atividadeSelecionadas.tipo_exercicio.astype('category')
df_atividadeSelecionadas.pratica_exercicio_1_vez_na_semana =
df_atividadeSelecionadas.pratica_exercicio_1_vez_na_semana.astype('category')
df_atividadeSelecionadas.frequencia_exercicio =
df_atividadeSelecionadas.frequencia_exercicio.astype('category')
df_atividadeSelecionadas.duracao_exercicio =
df_atividadeSelecionadas.duracao_exercicio.astype('category')
df_atividadeSelecionadas.cor = df_atividadeSelecionadas.cor.astype('category')

# Exibe os campos tratados
df_atividadeSelecionadas.head()

#Visualização de Outliers
sns.boxplot(x=df_atividadeSelecionadas['altura'])


#Tratar dados da Tabela Censo ano 2019 que vai ser usada para extrair informacões da Cidade como renda per capita, nome estado, Regiao etc...
df_censo= pd.read_excel('Capitais_Censo.xls')

#exibe dados do segundo Dataframe df_censo demogradico
df_censo.head()

#Mostrar todas as colunas do dataframe df_censo demografico
df_censo.columns.values

#Criar lista das colunas que vão ser usadas Censo
ColunasSelecionadaCenso = ['cidade', 'uf', 'capital', 'populacao estimada - pessoas [2021]', 'pib per capita',
'regiao', 'ano','Salario_medio_mensal']

#Filtra para o dataframe novo somente as colunas que foram selecionadas
df_censoSelecionadas = df_censo.filter(items=ColunasSelecionadaCenso )

# Exibe linhas do Dataset
df_censoSelecionadas.head()

#Renomeia as coluna
```

```
df_censoSelecionadas.rename(columns={'capital':'nome_capital', 'populacao estimada - pessoas [2021]':
'populacao_estimada_pessoas','pib per capita': 'pib_per_capita'}, inplace =True)

df_censoSelecionadas.head()

#criar coluna com o calculo Salario dataframe df_censoSelecionadas
df_censoSelecionadas['Salario'] = df_censoSelecionadas.apply(
    lambda row: round ((row.Salario_medio_mensal * 998.00),0), axis=1)

#criar coluna com o calculo PIB dataframe df_censoSelecionadas
df_censoSelecionadas['PIB'] = df_censoSelecionadas.apply(
    lambda row: round ((row.pib_per_capita * row.populacao_estimada_pessoas),0), axis=1)

#formata campo para flutuante

pd.set_option('float_format', '{:.2f}'.format)

# Exibe linhas do Datase
df_censoSelecionadas.head()

df_censoSelecionadas.info()

##Unindo os datasets df_censoSelecionadas com df_atividadeSelecionada2019
df_atividadeSelecionada2019 = df_atividadeSelecionadas.loc[df_atividadeSelecionadas["ano"]== 2019]

df_censoSelecionadas2019 = pd.merge(df_atividadeSelecionada2019, df_censoSelecionadas, how =
'outer')

df_censoSelecionadas2019Final = pd.merge(df_censoSelecionadas2019Final, df_censoSelecionadas, how
= 'outer')

# Exibe linhas do Dataset
df_atividadeSelecionada2019.head()
.

df_atividadeSelecionadas.shape

#verifica o total se pratica ou não Atividades Fisicas
pequisa = Counter(df_atividadeSelecionadas['pratica_exercicio'])
pequisa
Exibe o valor total dos entrevistados por Ano

#Plotagem das informações numero Total de pesquisa ANO
plt.figure(figsize=(20,4))
plt.subplot(131)
sns.countplot(x= 'ano', data = df_atividadeSelecionadas, palette="GnBu_d",edgecolor="black")
plt.title("Total Entrevistados ANO", fontsize=18)

plt.show()

#Plotagem das informações de gênero dos Entrevistados

import plotly.express as px
pie = df_atividadeSelecionadas['pratica_exercicio'].value_counts()
```

```
regions = pie.index
population = pie.values
fig = px.pie(df_atividadeSelecionadas, values=population, names=regions, title ="Total de entrevistados para os ANOS 2016, 2017, 2018, 2019 Agrupado por Ano")
fig.show()
```

Plotagem das informações de gênero dos Entrevistados Total

```
#Plotagem das informações de gênero dos Entrevistados

import plotly.express as px
pie = df_atividadeSelecionadas["sexo"].value_counts()
regions = pie.index
population = pie.values
fig = px.pie(df_atividadeSelecionadas, values=population, names=regions, title ="Total de entrevistados para os ANOS 2016, 2017, 2018, 2019 Agrupado por Gênero")
fig.show()
```

Tratar dataframes para dois dataframes SIM e Não para a pratica de atividade fisica.

```
#Plotagem das informações de gênero dos Entrevistados

import plotly.express as px
pie = df_atividadeSelecionadas.loc[df_atividadeSelecionadas["pratica_exercicio"]== "sim"]
pie = pie["sexo"].value_counts()
regions = pie.index
population = pie.values
fig = px.pie(df_atividadeSelecionadas, values=population, names=regions, title ="Total de entrevistados para os ANOS 2016, 2017, 2018, 2019 Agrupados por Gênero")
fig.show()

#filtra atividade fisica e colocar o resultado no dataframe dftotalpraticaSim
dftotalpraticaSim= df_atividadeSelecionadas.loc[df_atividadeSelecionadas["pratica_exercicio"]== "sim"]

#filtra não pratica atividade fisica e colocar o resultado no dataframe dftotalpraticaNao
dftotalpraticaNao= df_atividadeSelecionadas.loc[df_atividadeSelecionadas["pratica_exercicio"]== "nao"]
#verifica o total de
pequisa = Counter(df_atividadeSelecionadas['ano'])
pequisa

# conta quantos registros tem o dataframe pequisaSim
pequisaSim = Counter(dftotalpraticaSim['ano'])
pequisaSim

#Plotagem das informaçõesdos Entrevistados que praticam atividade fisica porcentagem
plt.style.use('seaborn-pastel')
plt.pie(pequisaSim.values(),labels = pequisaSim.keys(),
      autopct = '%1.1f%%', textprops={'fontsize':16})
plt.axis("image")
plt.title("Porcentagem de entrevistados que respoderam SIM", fontsize=18)
plt.show

pequisaNao = Counter(dftotalpraticaNao['ano'])
pequisaNao
```

Exibe as informações dos Entrevistados total evolução ano gênero Pratica de Atividade Fisica

```
#Plotagem das informações dos Entrevistados que não praticam atividade fisica porcentagem
plt.style.use('seaborn-pastel')
plt.pie(pequisaNao.values(),labels = pequisaNao.keys(),
      autopct = '%1.1f%%', textprops={'fontsize':16})
plt.axis("image")
plt.title("Porcentagem de entrevistados que respoderam Não", fontsize=18)
plt.show

#Plotagem das informações dos Entrevistados total evolução ano
df1=df_atividadeSelecionadas[['pratica_exercicio','ano']]
df2=df1.groupby(['ano','pratica_exercicio']).size().reset_index(name='Total Entrevistados')
fig3 = px.line(df2, x="ano", y="Total Entrevistados", color='pratica_exercicio',title='Evolução Pratica de
atividade Fisica ao Longo dos Anos')
fig3.show()

#Plotagem das informações de gênero dos Entrevistados Praticam Atividades Fisicas

import plotly.express as px
pie = dftotalpraticaSim['sexo'].value_counts()
regions = pie.index
population = pie.values
fig = px.pie(dftotalpraticaSim, values=population, names=regions, title ="Total de entrevistados por gênero
que praticam Atividades Fisicas")
fig.show()
#verifica o total entrevitas agrupado por ano
pequisa1 = Counter(dftotalpraticaSim['sexo'])
pequisa1
```

Exibe as informações de gênero dos Entrevistados que não Praticam Atividades Fisicas

```
#Plotagem das informações de gênero dos Entrevistados que não Praticam Atividades Fisicas

import plotly.express as px
pie = dftotalpraticaNao["sexo"].value_counts()
regions = pie.index

population = pie.values
fig = px.pie(dftotalpraticaNao, values=population, names=regions, title ="Total de entrevistados por gênero
que não praticam Atividades Fisicas")
fig.show()

#Plotagem das informações de gênero dos Entrevistados Total
import plotly.express as px
df = df_atividadeSelecionadas
figure = px.histogram(df, x = "sexo", color = "pratica_exercicio", title= "Pesquisa Total ANO")
figure.show()

# conta quantos registros tem o dataframe pequisaNao
pequisaNao = Counter(dftotalpraticaNao['ano'])
pequisaNao

#Contagem de modalidade por numero de praticantes
```

```
PraticaAtivida = Counter(dftotalpraticaSim['tipo_exercicio'])
PraticaAtivida
```

Informações dos Entrevistados que não praticam atividade fisica porcentagem

Exibe as informações dos Tipos de Atividade Fisica total agrupado por modalidade

```
 #Plotagem das informações dos Entrevistados total evolução ano
df= dftotalpraticaSim
import plotly.express as px
pie = df["tipo_exercicio"].value_counts()
regions = pie.index
population = pie.values
fig = px.pie(df, values=population, names=regions, title ="Total de entrevistados para os ANOS 2016, 2017, 2018, 2019 separados por Modalidade")
fig.show()
```

Exibe o IMC maior que 15 e menor que 50 para todos os entrevistados

```
#Plotagem das informações numero Total de pesquisa ANO

df1=df_atividadeSelecionadas[['imc','ano']]
df1= df1.loc[(df1['imc']> 15 ) & (df1['imc']< 50)]
df2=df1.groupby(['ano','imc']).size().reset_index(name='Total Entrevistados')
fig3 = px.line(df2, x="ano", y="Total Entrevistados", color='imc',title='Evolução IMC ao Longo dos Anos')
fig3.show()
```

Exibe histograma com as idade dos Praticante de Atividade Fisica

```
#Plotagem de histograma com as idade Praticante Atividade Fisica
dftotalpraticaSim.idade.hist(bins=20)
plt.style.use('seaborn-pastel')
plt.xlabel("idade")
plt.ylabel("Número de Entrevistado que pratica Atividade Fisica")
plt.title("Idade Praticante Atividade Fisica")
plt.show()

#Plotagem de histograma com as idade Praticante Atividade Fisica
dftotalpraticaNao.idade.hist(bins=20)
plt.style.use('seaborn-pastel')
plt.xlabel("idade")
plt.ylabel("Número de Entrevistado que Não pratica Atividade Fisica")
plt.title("Idade Praticante Atividade Fisica")
plt.show()

# Media das idades dos praticantes de atividades fisica
dftotalpraticaSim['idade'].mean()
```

**Vamos analisar um período especifico ANO 2019**

Exibe total de entrevistados para os ANOS 2019 separados por Gênero

```
# conta quantos registros tem o dataframe df_censoSelecionadas2019
pequisa2019 = Counter(df_censoSelecionadas2019['ano'])
pequisa2019

# conta quantos registros tem o dataframe df_censoSelecionadas2019
```

```
pequisa2019Pra = Counter(df_censoSelecionadas2019['pratica_exercicio'])
pequisa2019Pra

#Vamos tratar um perriodo Especifico ANO 2019

#Plotagem das informações de gênero dos Entrevistados

#filtra atividade fisica e colocar o resultado no dataframe dftotalpraticaSim
df_censoSelecionadas2019Sim =
df_censoSelecionadas2019.loc[df_censoSelecionadas2019['pratica_exercicio']== 'sim']
df_censoSelecionadas2019Nao =
df_censoSelecionadas2019.loc[df_censoSelecionadas2019['pratica_exercicio']== 'nao']
df_censoSelecionadas2019teste = df_censoSelecionadas2019
##df_censoSelecionadas2019.head()

import plotly.express as px
pie = df_censoSelecionadas2019["pratica_exercicio"].value_counts()
regions = pie.index
population = pie.values
fig = px.pie(df_censoSelecionadas2019, values=population, names=regions, title ="Total de entrevistados
para os ANOS 2019 separados por Gênero")
fig.show()

#Vamos tratar um perriodo Especifico ANO 2019

#Plotagem das informações de gênero dos Entrevistados

#filtra atividade fisica e colocar o resultado no dataframe dftotalpraticaSim
df_censoSelecionadas2019Sim =
df_censoSelecionadas2019.loc[df_censoSelecionadas2019['pratica_exercicio']== 'sim']
df_censoSelecionadas2019Nao =
df_censoSelecionadas2019.loc[df_censoSelecionadas2019['pratica_exercicio']== 'nao']
df_censoSelecionadas2019teste = df_censoSelecionadas2019
##df_censoSelecionadas2019.head()

import plotly.express as px
pie = df_censoSelecionadas2019["sexo"].value_counts()
regions = pie.index
population = pie.values
fig = px.pie(df_censoSelecionadas2019, values=population, names=regions, title ="Total de entrevistados
para os ANOS 2019 separados por Gênero")
fig.show()

###Plotagem das informações Tipo Atividade Fisica dos Entrevistados ("Pratica(SIM,NÃO) e Sexo")
sns.countplot(df_censoSelecionadas2019.pratica_exercicio, hue=df_censoSelecionadas2019.sexo )
plt.title("Pratica(SIM,NÃO) e Sexo")
plt.show()
Exibe as informações de gênero dos Entrevistados que pratica atividade fisica porcentagem

##Plotagem das informações de gênero dos Entrevistados que pratica atividade fisica porcentagem

import plotly.express as px
```

```
pie = df_censoSelecionadas2019Sim["sexo"].value_counts()
regions = pie.index
population = pie.values
fig = px.pie(df_censoSelecionadas2019Sim, values=population, names=regions, title ="Porcentagem de entrevistados que respoderam SIM 2019")
fig.show()

#Informações do campo idade
df_censoSelecionadas2019['idade'].describe()

#Informações do campo IMC
df_censoSelecionadas2019['imc'].describe()
```

Exibe histograma com todas as campos do dataframe df_censoSelecionadas2019Sim

```
#Plotagem de histograma com as idade Entrevistados
df_censoSelecionadas2019.idade.hist(bins=20)
plt.style.use('seaborn-pastel')
plt.xlabel("idade")
plt.ylabel("Número de Entrevistados")
plt.title("Idade Entrevistados")
plt.show()

#Informações do campo idade Praticande de Atividade
df_censoSelecionadas2019Sim['idade'].describe()

#Informações do campo idade Não Praticande de Atividade
df_censoSelecionadas2019Nao['idade'].describe()

#Plotagem de histograma com as idade Não Praticante Atividade Fisica
df_censoSelecionadas2019Nao.idade.hist(bins=20)
plt.style.use('seaborn-pastel')
plt.xlabel("idade")
plt.ylabel("Número de Entrevistado que pratica Atividade Fisica")
plt.title("Idade Praticante Atividade Fisica")
plt.show()

#Plotagem de histograma com as idade Praticante Atividade Fisica
df_censoSelecionadas2019Sim.idade.hist(bins=20)
plt.style.use('seaborn-pastel')
plt.xlabel("idade")
plt.ylabel("Número de Entrevistado que pratica Atividade Fisica")
plt.title("Idade Praticante Atividade Fisica")
plt.show()

#Plotagem das informações dos Entrevistados total evolução ano 2019
df1=df_censoSelecionadas2019[['pratica_exercicio','idade']]
df2=df1.groupby(['idade','pratica_exercicio']).size().reset_index(name='Total Entrevistados')
fig3 = px.line(df2, x="idade", y="Total Entrevistados", color='pratica_exercicio',title='Evolução Pratica de atividade Fisica ou não ao Longo da Idade')
fig3.show()

#filtra atividade fisica e colocar o resultado no dataframe dftotalpraticaSim
```

```
df_censoSelecionadas2019Sim=
df_censoSelecionadas2019.loc[df_censoSelecionadas2019["pratica_exercicio"]== "sim"]


#filtra não pratica atividade fisica e colocar o resultado no dataframe dftotalpraticaNao
df_censoSelecionadas2019Nao=
df_censoSelecionadas2019.loc[df_censoSelecionadas2019["pratica_exercicio"]== "nao"]
#verifica o total de
df_pequisa2019 = Counter(df_censoSelecionadas2019['ano'])
df_pequisa2019
Total Pequisa Sim 2019

df_pequisa2019SIM = Counter(df_censoSelecionadas2019Sim['ano'])
df_pequisa2019SIM
Total Pequisa Não 2019

df_pequisa2019NAO = Counter(df_censoSelecionadas2019Nao['ano'])
df_pequisa2019NAO

#Descrição estatística da idade dos Entrevistados Sim 2019
df_censoSelecionadas2019Sim['idade'].describe()

#Nome colunas daaframe df_censoSelecionadas2019Sim
df_censoSelecionadas2019Sim.columns.values

#exibe as cor dos entrevistados SIM
df_censoSelecionadas2019Sim["cor"].value_counts()

#exibe as cor dos entrevistados SIM
df_censoSelecionadas2019Nao["cor"].value_counts()

##Plotagem das informações Tipo Atividade Fisica dos Entrevistados por COR porcentagem
import plotly.express as px
pie =df_censoSelecionadas2019["cor"].value_counts()
regions = pie.index
population = pie.values
fig = px.pie(df_censoSelecionadas2019, values=population, names=regions, title ="Porcentagem de
entrevistados Por Cor")
fig.show()

##Plotagem das informações Tipo Atividade Fisica dos Entrevistados por COR porcentagem
import plotly.express as px
pie =df_censoSelecionadas2019Sim["cor"].value_counts()
regions = pie.index
population = pie.values
fig = px.pie(df_censoSelecionadas2019Sim, values=population, names=regions, title ="Porcentagem de
entrevistados Por Cor")
fig.show()
Exibe as informações Tipo Atividade Fisica dos Entrevistados por COR e Gênero

###Plotagem das informações Tipo Atividade Fisica dos Entrevistados cor da pele e sexo
sns.countplot(df_censoSelecionadas2019Sim.cor, hue=df_censoSelecionadas2019Sim.sexo )
plt.title("Cor e Sexo")
```

```
plt.show()
```
Exibe as informações Atividade Fisica dos Entrevistados por Região

```
##Plotagem das informações Atividade Fisica dos Entrevistados que praticam atividade porcentagem por Região

import plotly.express as px
pie = df_censoSelecionadas2019Sim["regiao"].value_counts()
regions = pie.index
population = pie.values
fig = px.pie(df_censoSelecionadas2019Sim, values=population, names=regions, title ="Porcentagem de entrevistados Por Regiao")
fig.show()

#Atividade Fisica dos Entrevistados que praticam atividade porcentagem por Região

df_censoSelecionadas2019Sim["regiao"].value_counts()

##Plotagem das informações Atividade Fisica dos Entrevistados que Não praticam atividade porcentagem por Estado Civil

import plotly.express as px
pie = df_censoSelecionadas2019Nao["civil"].value_counts()
regions = pie.index
population = pie.values
fig = px.pie(df_censoSelecionadas2019Nao, values=population, names=regions, title ="Porcentagem de entrevistados Por Estado Civil")
fig.show()

df_casados =df_censoSelecionadas2019Sim.loc[df_censoSelecionadas2019Sim['civil']== 'casado legalmente']
df_casados['idade'].describe()

df_censoSelecionadas2019Sim['idade'].describe()

#exibe lista estado Civil
df_pequisa2019SIMEstadoCivil = Counter(df_censoSelecionadas2019Sim["civil"])
df_pequisa2019SIMEstadoCivil

##Plotagem das informações Atividade Fisica dos Entrevistados que praticam atividade porcentagem por Estado Civil

import plotly.express as px
pie = df_censoSelecionadas2019Sim["civil"].value_counts()
regions = pie.index
population = pie.values
fig = px.pie(df_censoSelecionadas2019Sim, values=population, names=regions, title ="Porcentagem de entrevistados Por Estado CIVIL")
fig.show()
```
Exibe as informações Atividade Fisica dos Entrevistados que praticam atividade porcentagem agrupado por Cidade

```
##Plotagem das informações Tipo Atividade Fisica dos Entrevistados que praticam atividade porcentagem agrupado por Cidade

import plotly.express as px
pie = df_censoSelecionadas2019Sim["nome_capital"].value_counts()
regions = pie.index
population = pie.values
fig = px.pie(df_censoSelecionadas2019Sim, values=population, names=regions, title ="Porcentagem de entrevistados que respoderam SIM 2019 agrupado por Cidade")
fig.show()

# Conta quantos Respoderam sim por Capital
df_censoSelecionadas2019Sim["nome_capital"].value_counts()

df_censoSelecionadas2019Sim.head()

#Plotagem das informações dos Entrevistados total evolução ano 2019
df1=df_censoSelecionadas2019Sim[['nome_capital','imc',]]
df1=df1.loc[(df1['imc']> 15 ) & (df1['imc']< 50)]
df2=df1.groupby(['imc','nome_capital']).size().reset_index(name='Total Entrevistados')
fig3 = px.line(df2, x="imc", y="Total Entrevistados", color='nome_capital',title='Evolução Pratica de atividade Fisica ao Longo do IMC')
fig3.show()
Exibe top 10 Capitais na Pratica de exercicios

import matplotlib as mpl
game = df_censoSelecionadas2019Sim.groupby("nome_capital")["ano"].count().head(10)
custom_colors = mpl.colors.Normalize(vmin=min(game), vmax=max(game))
colours = [mpl.cm.PuBu(custom_colors(i)) for i in game]
plt.figure(figsize=(7,7))
plt.pie(game, labels=game.index, colors=colours)
central_circle = plt.Circle((0, 0), 0.5, color='white')
fig = plt.gcf()
fig.gca().add_artist(central_circle)
plt.rc('font', size=13)
plt.title("Top 10 Capitais Pratica de exercicio ", fontsize=20)
plt.show()

#seleciona coluna Nome Capital e salario

ColunasSelecionadaCenso = ['nome_capital','Salario']
SalarioCapital = df_censoSelecionadas.filter(items=ColunasSelecionadaCenso)
SalarioCapital

#Plotagem do gráfico Salario por capital
plt.style.use('seaborn-pastel')
plt.barh( df_censoSelecionadas.nome_capital, df_censoSelecionadas.Salario,)
plt.ylabel('Salario')
plt.xlabel('nome_capital')
plt.title('Salario Medio Por capital')
plt.show()

##Plotagem das informações Salario porcentagem agrupado por Cidade
```

```
import plotly.express as px
pie = df_censoSelecionadas["Salario"]
regions =df_censoSelecionadas["nome_capital"]
population = pie.values
fig = px.pie(df_censoSelecionadas, values=population, names=regions, title ="Porcentagem Salario
agrupado por salario")
fig.show()

#seleciona coluna Nome Capital e PIB

ColunasSelecionadaCenso = ['nome_capital','PIB']
SalarioCapital = df_censoSelecionadas.filter(items=ColunasSelecionadaCenso)
SalarioCapital

##Plotagem das informações PIB porcentagem agrupado por Cidade

import plotly.express as px
pie = SalarioCapital["PIB"]
regions =SalarioCapital["nome_capital"]
population = pie.values
fig = px.pie(df_censoSelecionadas, values=population, names=regions, title ="Porcentagem Salario
agrupado por PIB")
fig.show()

 ##Plotagem das informações Total população porcentagem agrupado por Cidade

ColunasSelecionadaCenso = ['nome_capital','populacao_estimada_pessoas']
SalarioCapital = df_censoSelecionadas.filter(items=ColunasSelecionadaCenso)
SalarioCapital

#Total População
dfCont= sum(SalarioCapital.populacao_estimada_pessoas)
dfCont

 ##Plotagem das informações Total população porcentagem agrupado por Cidade

import plotly.express as px
pie = df_censoSelecionadas["populacao_estimada_pessoas"]
regions =df_censoSelecionadas["nome_capital"]
population = pie.values
fig = px.pie(df_censoSelecionadas, values=population, names=regions, title ="Porcentagem Contagem
população agrupado por cidade")
fig.show()

#Plotagem do gráfico Salario por capital

plt.style.use('seaborn-pastel')
plt.barh( df_censoSelecionadas.nome_capital, df_censoSelecionadas.Salario,)
plt.ylabel('PIB')
plt.xlabel('nome_capital')
plt.title('Valor PIB')
plt.show()
```

```
#Plotagem das informações Duração dos Exercicios por Frequencia Exercicio
import plotly.express as px
df = df_censoSelecionadas2019Sim
figure = px.histogram(df, x = "tipo_exercicio", color = "frequencia_exercicio", title= "Duração dos Exercicios por Frquencia Agrupado Exercicio")
figure.show()

#Plotagem das informações Duração dos Exercicios por Capital"
import plotly.express as px
df = df_censoSelecionadas2019Sim
figure = px.histogram(df, x = "nome_capital", color = "tipo_exercicio", title= "Tipo de Exercicios por Capital")
figure.show()

#Plotagem das informações Duração dos Exercicios por Frequencia por Cor
import plotly.express as px
df = df_censoSelecionadas2019Sim
figure = px.histogram(df, x = "cor", color = "frequencia_exercicio", title= "Duração dos Exercicios por Frquencia Agrupado Por Cor")
figure.show()

#Plotagem das informações Duração dos Exercicios por Tipo Exercicio"
import plotly.express as px
df = df_censoSelecionadas2019Sim
figure = px.histogram(df, x = "nome_capital", color = "duracao_exercicio", title= "Duração dos Exercicios por Tipo Exercicio")
figure.show()

# Total agrupado por atividade fisica
df_censoSelecionadas2019Sim["tipo_exercicio"].value_counts()

##Plotagem das informações Tipo Atividade Fisica dos Entrevistados que pratica atividade porcentagem agrupado por Atividade

import plotly.express as px
pie = df_censoSelecionadas2019Sim["tipo_exercicio"].value_counts()
regions = pie.index
population = pie.values
fig = px.pie(df_censoSelecionadas2019Sim, values=population, names=regions, title ="Porcentagem de entrevistados que respoderam SIM 2019 agrupado por Atividade")
fig.show()

df_censoSelecionadas.head()

#Plotagem das informações dos Entrevistados total evolução ano 2019
df1=df_censoSelecionadas2019Sim[['tipo_exercicio','idade',]]
df2=df1.groupby(['idade','tipo_exercicio']).size().reset_index(name='Total Entrevistados')
fig3 = px.line(df2, x="idade", y="Total Entrevistados", color='tipo_exercicio',title='Evolução Pratica de atividade Fisica ao Longo da Idade por Tipo Exercicio')
fig3.show()

#Plotagem das informações Duração dos Exercicios por COR"
import plotly.express as px
df = df_censoSelecionadas2019Sim
```

```
figure = px.histogram(df, x = "cor", color = "tipo_exercicio", title= "Tipo de Exercicios por COR")
figure.show()

#Plotagem das informações Duração dos Exercicios por COR"
import plotly.express as px
df = df_censoSelecionadas2019Sim
figure = px.histogram(df, x = "cor", color = "duracao_exercicio", barmode ='group', title= "Duração dos
Exercicios por COR")
figure.show()

#Plotagem das informações Duração dos Exercicios por Tipo Exercicio"
import plotly.express as px
df = df_censoSelecionadas2019Sim
figure = px.histogram(df, x = "tipo_exercicio", color = "duracao_exercicio", title= "Duração dos Exercicios por
Tipo Exercicio")
figure.show()
```

Exibe Top 10 Tipo Exercico praticados

```
import matplotlib as mpl
game = df_censoSelecionadas2019Sim.groupby("tipo_exercicio")["ano"].count().head(10)
custom_colors = mpl.colors.Normalize(vmin=min(game), vmax=max(game))
colours = [mpl.cm.PuBu(custom_colors(i)) for i in game]
plt.figure(figsize=(7,7))
plt.pie(game, labels=game.index, colors=colours)
central_circle = plt.Circle((0, 0), 0.5, color='white')
fig = plt.gcf()
fig.gca().add_artist(central_circle)
plt.rc('font', size=12)
plt.title("Top 10 Tipo Exercico praticado ", fontsize=20)
plt.show()

#Conta lista exercicos
df_censoSelecionadas2019Sim["tipo_exercicio"].value_counts()
```

Evolução Pratica de atividade Fisica ao Longo da Idade por Cidade

```
#Plotagem das informações dos Entrevistados total evolução ano 2019
df1=df_censoSelecionadas2019[['nome_capital','idade']]
df2=df1.groupby(['idade','nome_capital']).size().reset_index(name='Total Entrevistados')
fig3 = px.line(df2, x="idade", y="Total Entrevistados", color='nome_capital',title='Evolução Pratica de
atividade Fisica ao Longo da Idade por Cidade')
fig3.show()

#Plotagem das informações dos Entrevistados total evolução Região
df1=df_censoSelecionadas2019Sim[['regiao','idade']]
df2=df1.groupby(['idade','regiao']).size().reset_index(name='Total Entrevistados')
fig3 = px.line(df2, x="idade", y="Total Entrevistados", color='regiao',title='Evolução Pratica de atividade
Fisica ao Longo da Idade por Região')
fig3.show()
```

Evolução Pratica de atividade Fisica ao Longo da Idade por Cor

```
#Plotagem das informações dos Entrevistados total evolução ano 2019
df1=df_censoSelecionadas2019Sim[['cor','idade',]]
```

```
df2=df1.groupby(['idade','cor']).size().reset_index(name='Total Entrevistados')
fig3 = px.line(df2, x="idade", y="Total Entrevistados", color='cor',title='Evolução Pratica de atividade Fisica ao Longo da Idade por Cor')
fig3.show()

#Plotagem das informações dos Entrevistados total evolução ano 2019
df_cidade = df_censoSelecionadas2019Sim.loc[(df_censoSelecionadas2019Sim["regiao"]== 'Sudeste')]
df1=df_cidade[['nome_capital','idade']]
df2=df1.groupby(['idade','nome_capital']).size().reset_index(name='Total Entrevistados')
fig3 = px.line(df2, x="idade", y="Total Entrevistados", color='nome_capital',title='Evolução Pratica de atividade Fisica ao Longo da Idade Capital Sudeste')
fig3.show()
# Agrupa e exibe tipço exercico
df_censoSelecionadas2019SimGroup  = df_censoSelecionadas2019Sim.groupby(['tipo_exercicio'], as_index=False)
pequisaNao1 = Counter(df_censoSelecionadas2019Sim['tipo_exercicio'])
pequisaNao1

df_censoSelecionadas2019Sim.head()

#Contagem das opções da coluna grau_escolaridade dos entrevistados
df_escolaridade_praticanteAtividade = Counter(df_censoSelecionadas2019Sim['grau_escolaridade'])
df_escolaridade_praticanteAtividade

#Contagem das opções da coluna grau_escolaridade dos entrevistados
df_escolaridade_Naopraticante= Counter(df_censoSelecionadas2019Nao['grau_escolaridade'])
df_escolaridade_Naopraticante

#Identificação dos percentuais de escolaridade dos Entrevistados Não praticam
n_censoSelecionadas2019Nao=sum(df_escolaridade_Naopraticante.values())
for x, y in df_escolaridade_Naopraticante.items():
    a = y/n_censoSelecionadas2019Nao*100
    print(str(x) + ': ' +'\n'+ str(round(a,2)) + '%'+'\n')

#Identificação dos percentuais de escolaridade dos Entrevistados
n_censoSelecionadas2019=sum(df_escolaridade_praticanteAtividade.values())
for x, y in df_escolaridade_praticanteAtividade.items():
    a = y/n_censoSelecionadas2019*100
    print(str(x) + ': ' +'\n'+ str(round(a,2)) + '%'+'\n')

#Criação de listas para construção do gráfico de escolaridade
lista_df_escolaridade_praticanteAtividade_labels = []
lista_df_escolaridade_praticanteAtividade_valores = []
for x, y in df_escolaridade_praticanteAtividade.items():
    lista_df_escolaridade_praticanteAtividade_labels.append(x)
    lista_df_escolaridade_praticanteAtividade_valores.append(y)

#Verificação da lista criada
lista_df_escolaridade_praticanteAtividade_valores

#Verificação da lista criada
lista_df_escolaridade_praticanteAtividade_labels

#Plotagem do gráfico de escolaridade dos Entrevistados
```

```
plt.style.use('seaborn-pastel')
plt.barh(lista_df_escolaridade_praticanteAtividade_labels,
lista_df_escolaridade_praticanteAtividade_valores)
plt.ylabel('Escolaridade')
plt.xlabel('número de Praticantes')
plt.title('Total de Praticantes por Escolaridade', fontsize=18)
plt.show()
```

**Analise de Escolaridade**

```
#Plotagem Escolaridade"
import plotly.express as px
df =df_atividadeSelecionadas.loc[df_atividadeSelecionadas['ano']== 2019]
figure = px.histogram(df, x = "grau_escolaridade", color = "pratica_exercicio", title= "Atividade Fisica
Escolaridade")
figure.show()


#Verifica se existe valores nulo no Data Frame
df_censoSelecionadas2019FinalTreinamento.isnull().sum()

df_censoSelecionadas2019FinalTreinamento['pratica_exercicio'].replace(2,0, inplace =True)

#Exibe os valores do Data Frame
df_censoSelecionadas2019FinalTreinamento.head()

#Exibe informação do Data Frame
df_censoSelecionadas2019FinalTreinamento.info()

#trata tipo campo
df_censoSelecionadas2019FinalTreinamento.ordem    =
df_censoSelecionadas2019FinalTreinamento.ordem.astype('int64')
df_censoSelecionadas2019FinalTreinamento.ano  =
df_censoSelecionadas2019FinalTreinamento.ano.astype('uint8')
df_censoSelecionadas2019FinalTreinamento.civil  =
df_censoSelecionadas2019FinalTreinamento.civil.astype('uint8')
df_censoSelecionadas2019FinalTreinamento.idade  =
df_censoSelecionadas2019FinalTreinamento.idade.astype('uint8')
df_censoSelecionadas2019FinalTreinamento.sexo   =
df_censoSelecionadas2019FinalTreinamento.sexo.astype('uint8')
df_censoSelecionadas2019FinalTreinamento.grau_escolaridade  =
df_censoSelecionadas2019FinalTreinamento.grau_escolaridade.astype('uint8')
df_censoSelecionadas2019FinalTreinamento.peso   =
df_censoSelecionadas2019FinalTreinamento.peso.astype('uint8')
df_censoSelecionadas2019FinalTreinamento.altura =
df_censoSelecionadas2019FinalTreinamento.altura.astype('uint8')
df_censoSelecionadas2019FinalTreinamento.pratica_exercicio  =
df_censoSelecionadas2019FinalTreinamento.pratica_exercicio.astype('uint8')
df_censoSelecionadas2019FinalTreinamento.tipo_exercicio     =
df_censoSelecionadas2019FinalTreinamento.tipo_exercicio.astype('uint8')
df_censoSelecionadas2019FinalTreinamento.pratica_exercicio_1_vez_na_semana =
df_censoSelecionadas2019FinalTreinamento.pratica_exercicio_1_vez_na_semana.astype('uint8')
df_censoSelecionadas2019FinalTreinamento.frequencia_exercicio =
df_censoSelecionadas2019FinalTreinamento.frequencia_exercicio.astype('uint8')
```

```
df_censoSelecionadas2019FinalTreinamento.duracao_exercicio =
df_censoSelecionadas2019FinalTreinamento.duracao_exercicio.astype('uint8')
df_censoSelecionadas2019FinalTreinamento.cor =
df_censoSelecionadas2019FinalTreinamento.cor.astype('uint8')

#Verificação dos valores da coluna Sexo
df_censoSelecionadas2019FinalTreinamento['sexo'].unique()

#total Pesquisa
pequisa2019Pra = Counter(df_censoSelecionadas2019FinalTreinamento['pratica_exercicio'])
pequisa2019Pra

#Verificação dos valores da coluna Grau Escolaridade
df_censoSelecionadas2019FinalTreinamento['grau_escolaridade'].unique()

#Verificação dos valores da coluna Estado Civil
df_censoSelecionadas2019FinalTreinamento['civil'].unique()

#Verificação dos valores da coluna Pratica de Exercicio
df_censoSelecionadas2019FinalTreinamento['pratica_exercicio'].unique()

#Primeiras linhas do dataframe df_censoSelecionadas2019Final
df_censoSelecionadas2019FinalTreinamento.head()

# Exibe total data frame
df_censoSelecionadas2019FinalTreinamento['pratica_exercicio'].value_counts()

#Importação da função resample da biblioteca sklearn
from sklearn.utils import resample

#Criação de um dataframe com entrevistados que não praticam atividade fisica
df_censoSelecionadas2019FinalTreinamento_majority =
df_censoSelecionadas2019FinalTreinamento[df_censoSelecionadas2019FinalTreinamento.pratica_exercici
o==0]
df_censoSelecionadas2019FinalTreinamento_majority.info()

#Criação de um dataframe com entrevistados que praticam atividade fisica
df_censoSelecionadas2019FinalTreinamento_minority =
df_censoSelecionadas2019FinalTreinamento[df_censoSelecionadas2019FinalTreinamento.pratica_exercici
o==1]
df_censoSelecionadas2019FinalTreinamento_minority.info()

#Ajuste no número de entradas do dataframe entrevistados que praticam atividade fisica
df_censoSelecionadas2019FinalTreinamento_minority_upsampled =
resample(df_censoSelecionadas2019FinalTreinamento_minority,
                                    replace=True, n_samples=22283,
                                    random_state=123)

#Informações do dataframe ajustado
df_censoSelecionadas2019FinalTreinamento_minority_upsampled.info()

#Concatenação dos dataframes de entrevistados que praticam atividade fisica e que não praticam
df_censoSelecionadas2019FinalTreinamento_upsampled =
pd.concat([df_censoSelecionadas2019FinalTreinamento_majority,
```

```
                                df_censoSelecionadas2019FinalTreinamento_minority_upsampled])

#Informações do dataframe ajustado
df_censoSelecionadas2019FinalTreinamento_upsampled.info()

#Contagem dos valores de entrevistados que praticam atividade fisica e que não praticam
df_censoSelecionadas2019FinalTreinamento_upsampled['pratica_exercicio'].value_counts()

#Importação da função train_test_split
from sklearn.model_selection import train_test_split

#Divisão para as bases de treinamento
X_train = df_censoSelecionadas2019FinalTreinamento_upsampled.drop(['pratica_exercicio'], axis = 1)
y_train = df_censoSelecionadas2019FinalTreinamento_upsampled.pratica_exercicio

#Informações do dataframe com os dados para treinamento
X_train.info()

#Tipo da serie y_train
type(y_train)

#Criação das bases de teste e treinamento
xtreinamento, xteste, ytreinamento, yteste = train_test_split(X_train, y_train, random_state =0)

#Informação do dataframe de treinamento
xtreinamento.info()

#Informação do dataframe de teste
xteste.info()

#Contagem dos resultados para treinamento
ytreinamento.count()

#Contagem dos resultados para teste
yteste.count()

#Criação do modelo utilizando a Árvore de decisão
from sklearn.tree import DecisionTreeClassifier
from sklearn import tree
df_censoSelecionadas2019FinalTreinamento_tree = DecisionTreeClassifier(random_state=0)
df_censoSelecionadas2019FinalTreinamento_tree =
df_censoSelecionadas2019FinalTreinamento_tree.fit(xtreinamento, ytreinamento)
print("Acurácia: ", df_censoSelecionadas2019FinalTreinamento_tree.score(xtreinamento, ytreinamento))
Train_predict = df_censoSelecionadas2019FinalTreinamento_tree.predict(xteste)
print("Acurácia de previsão: ", accuracy_score(yteste, Train_predict))
print(classification_report(yteste, Train_predict))

# criando modelo arvore
classificador_arvore = DecisionTreeClassifier(random_state = 1)
classificador_arvore.fit(xtreinamento,ytreinamento)

predicao_arvore = classificador_arvore.predict(xteste)
print(predicao_arvore)

#Gerando matrix de confusão
```

```
confusao_nb = confusion_matrix(yteste, Train_predict)
print(confusao_nb)
cmd_arvore = ConfusionMatrixDisplay(confusao_nb, display_labels=classificador_arvore.classes_).plot()

#Criação do modelo utilizando a Regressão Logística
from sklearn.linear_model import LogisticRegression
lr = LogisticRegression()
lr = lr.fit(xtreinamento, ytreinamento)
print("Acurácia: ", lr.score(xtreinamento, ytreinamento))
tp_lr = lr.predict(xteste)
print("Acurácia de previsão: ", accuracy_score(yteste, tp_lr))
print(classification_report(yteste, tp_lr))

#Validação cruzada para o modelo Regressão Logística
from sklearn.model_selection import cross_val_score
validacao_arvore = cross_val_score(lr, X_train, y_train, scoring='accuracy', cv=5)
print(validacao_arvore.mean())

#Criação do modelo utilizando Naïve Bayes
from sklearn.naive_bayes import GaussianNB
nb = GaussianNB()
nb = nb.fit(xtreinamento, ytreinamento)
print("Acurácia: ", nb.score(xtreinamento, ytreinamento))
tp_nb = nb.predict(xteste)
print("Acurácia de previsão: ", accuracy_score(yteste, tp_nb))
print(classification_report(yteste, tp_nb))

#Validação cruzada para o modelo Naive bayes
from sklearn.model_selection import cross_val_score
validacao_arvore = cross_val_score(nb, X_train, y_train, scoring='accuracy', cv=5)
print(validacao_arvore.mean())

#Criação do modelo utilizando Gradiente Descendente
from sklearn.linear_model import SGDClassifier
sgd = SGDClassifier()
sgd = sgd.fit(xtreinamento, ytreinamento)
print("Acurácia: ", sgd.score(xtreinamento, ytreinamento))
tp_sgd = sgd.predict(xteste)
print("Acurácia de previsão: ", accuracy_score(yteste, tp_sgd))
print(classification_report(yteste, tp_sgd))

#Criação do modelo utilizando KNN (K - Nearest Neighbors)
from sklearn.neighbors import KNeighborsClassifier
knn = KNeighborsClassifier()
knn = knn.fit(xtreinamento, ytreinamento)
print("Acurácia: ", knn.score(xtreinamento, ytreinamento))
tp_knn = knn.predict(xteste)
print("Acurácia de previsão: ", accuracy_score(yteste, tp_knn))
print(classification_report(yteste, tp_knn))

#Faz a validação cruzada com 5 folds e ao final exibe a média da acurácia
validacao_knn = cross_val_score(knn, X_train, y_train, scoring='accuracy', cv=5)
print(validacao_knn.mean())
```

```
#Criação do modelo utilizando Randon Forest
from sklearn.ensemble import RandomForestClassifier
rfm = RandomForestClassifier()
rfm = rfm.fit(xtreinamento, ytreinamento)
print("Acurácia: ", rfm.score(xtreinamento, ytreinamento))
tp_rfm = rfm.predict(xteste)
print("Acurácia de previsão: ", accuracy_score(yteste, tp_rfm))
print(classification_report(yteste, tp_rfm))

#Validação cruzada do classificador Random Forest
validacao_rf = cross_val_score(rfm, X_train, y_train, scoring='accuracy', cv=5)
print(validacao_rf.mean())
```