# Uma análise de dados no Python sobre a participação feminina nos estados de São Paulo e Alagoas

*Antonio Eduardo de Oliveira Carmo*[1], *Fabricio de Figueredo Oliveira*[2]

**Resumo:** A evolução da tecnologia vem possibilitando avanços em diversas áreas, na área da ciência de dados conseguimos analisar muito mais os dados fornecidos devido a esse avanço, portanto o trabalho irá apresentar uma análise sobre as três últimas eleições que ocorreram para prefeito e vice-prefeito nos estados de São Paulo e Alagoas, visando a participação das mulheres no meio político, essa análise se deu através do uso da linguagem de programação Python e a sua biblioteca Pandas, feita a análise é possível ver que durantes essas eleições houve o aumento da participação feminina no meio político.
**Palavras-chave:** Python; Pandas; Análise de dados.

## 1. INTRODUÇÃO

O aumento da necessidade da tecnologia está cada vez mais crescente, hoje por exemplo, dificilmente encontramos pessoas que não possuem contato com a tecnologia. Computadores, celulares, caixas de bancos, televisões, etc, estão presentes no nosso dia a dia e por trás dos seus sistemas está a programação, hoje sem dúvida uma das áreas mais importantes para o desenvolvimento humano.

Devido a isso, a proximidade dos alunos com esse aprendizado deve ser elevada, até para que entendam e possam trabalhar e atuar com maior qualidade na sua respectiva área. O Python é uma linguagem de programação que traz uma facilidade para o usuário e possui diversas ferramentas como por exemplo as suas vastas bibliotecas, principalmente as que englobam a ciência de dados, que será usado neste trabalho e também os seus códigos mais limpos e simplificados, facilitando ao usuário uma melhor qualidade na aprendizagem, além de ser uma linguagem leve e gratuita.

A ciência de dados no presente momento está acendendo no meio tecnológico, devido que grande parte da informação para otimizar o trabalho, sistemas e lucro de empresas, se encontram nesse estudo, junto a isso o trabalho irá abordar uma análise de dados sobre as últimas eleições para verificar se houve aumento na participação feminina no meio político, destacando os estados de São Paulo e Alagoas, sendo esses estados um de IDH muito elevado e o outro de baixo IDH respectivamente e se por acaso as cidades com maior IDH desses estados, influenciam ou não nessa participação.

## 2. DESENVOLVIMENTO

### 2.1 PYTHON

#### 2.1.1 Dados

Nesta seção abordaremos a linguagem Python. De forma básica os dados são conhecidos como valores, que podem ser tanto valores de saída ou de entrada, sendo o valor de saída a resposta que o programa lhe dá e o valor de entrada os dados que o operador fornece. Como mostra a figura 1, um exemplo de um programa que calcula uma soma de números no Python.

```
i = int(input('Digite um numero: '))
f = int(input('Digite outro numero: '))
g = i + f

print('A soma dos dois numeros é: ',g)
```

Figura 1. Soma de números no Python. (Autoria própria)

No Python os dados possuem seus tipos primitivos que podem ser do tipo string(conjunto de caracteres), int(valores inteiros), float(valores reais), bool(valores de verdadeiro ou falso), no programa foram utilizados o do tipo int(inteiro), mas nada impede que seja trocado para um valor float(real) ou algum outro tipo de dado, exceto

caso não haja a concordância com o programa, por exemplo, ao digitar um valor string quando é pedido um valor inteiro, pois assim o programa poderá falhar caso o programador tenha especificado o tipo do dado.

### 2.1.2 Comandos Básicos

No Python para entender como funciona a entrada de dados, as funções e como utilizá-las podemos utilizar variáveis de tal modo que alocamos a essas variáveis valores e assim podemos trabalhar com elas, por exemplo, queremos fazer uma soma, como o programa irá saber quais números queremos somar? para isso a seguir teremos um exemplo de alocação de valor em uma variável, como mostra a figura 2.

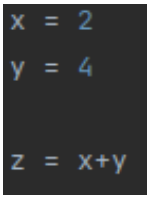



Figura 2. Alocação de valores e soma. (Autoria própria)

Podemos ver que para alocarmos valores em variáveis basta dizer que essa variável receberá um valor, na foto vemos que x receberá 2 e que y receberá 4 e por último que z receberá a soma de x mais y.

Além das variáveis temos a função **print** que nada mais é a impressão de algo na tela, podendo ser desde um dado fornecido até uma mensagem escolhida por você, deixando claro que se queremos fornecer uma informação para o usuário é necessário utilizar desse comando.

O **print** possui uma estrutura definida para que possamos utilizá-lo, de modo que essa função deve ser expressada assim: **prin**t(), tudo que for posto dentro dos parênteses irá sair para o usuário, caso queira escrever algo e não apenas mostrar, terá que utilizar as aspas de modo que a estrutura passa a ser: **print**('O que você gostaria de escrever'), a seguir teremos um exemplo para demonstrar melhor a utilização dessa função no Python, mostrado na figura 3, um exemplo do comando **print**.

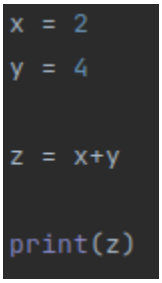



Figura 3. Comando print. (Autoria própria)

Com esse **print** o programa irá retornar o valor da soma para o usuário, de modo que mostrará o valor 6.

Mas gostaria de mandar uma mensagem para o usuário, como que faço? Para isso temos o exemplo do **print** com aspas, como mostrado na figura 4 um exemplo do comando **print** com texto.


Figura 4. Hello World. (Autoria própria)

Com esse **print** iremos mostrar a frase Hello World para o usuário.

Sabendo o que é a função **print** e como podemos alocar valores em variáveis, como podemos pedir para um usuário valores? Existe essa possibilidade? A resposta é sim e para isso usaremos a função **input**, a função **input** fará quase a mesma coisa que a função **print**, de modo que irá mostrar uma mensagem para o usuário, só que agora a função irá aguardar uma resposta do usuário e salvará esse valor em uma variável, como mostrado na figura 5 a seguir.

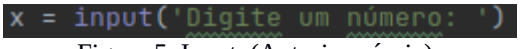



Figura 5. Input. (Autoria própria)

Como podemos ver temos a necessidade de declararmos uma variável que irá receber a informação da função **input**, analisando então o código a nossa variável x irá receber um número escolhido pelo usuário e assim armazenando o valor.

### 2.1.3 Operações

Para manipularmos os dados de forma que possamos somar, subtrair, dividir, multiplicar, etc. Temos que entender como funcionam os operadores no Python. Nesta tabela podemos ver ao lado esquerdo os operadores e do lado direito um exemplo do seu resultado:

Tabela 1. Tabela operações (Autoria própria)

| Operação | Símbolo | Exemplo |
|---|---|---|
| Adição | + | 5+2 == 7 |
| Subtração | - | 5-2 == 2 |
| Multiplicação | * | 5*2 == 10 |
| Divisão | / | 5/2 == 2,5 |
| Potência | ** | 5**2 == 25 |
| Divisão Inteira | // | 5//2 == 2 |
| Resto da Divisão | % | 5%2 == 1 |

Os operadores também possuem ordem de procedência, para que o seu resultado seja preciso, então deve-se destacar qual operação ocorreu primeiro.

Tabela 2. Ordem dos operadores (Autoria própria)

| Ordem | Operador |
|---|---|
| 1 | ( ) |
| 2 | ** |
| 3 | *, /, //, % |
| 4 | +, - |

Na tabela 2 vemos que o parêntese possui maior ordem de procedência, vindo na sequência a potência, depois a multiplicação, divisão, divisão inteira e o resto da divisão, esses possuem uma regra, na equação quem vier primeiro é a operação que irá ocorrer e por último com a menor ordem de procedência é a soma e subtração.

### 2.1.4 Condições

No Python condições são expressões que podem ser definidas como verdadeiras ou falsas, por exemplo 42 > 12, essa expressão trará um valor verdadeiro quando a expressão for testada, outro exemplo é 60 < 20, essa expressão trará um valor falso quando a expressão for testada.

```
x = 30
y = 20

print(x>y)
```

Figura 6. Condição. (Autoria própria)

As condições possuem expressões relacionais e expressões lógicas, os operadores das expressões relacionais são chamados de operadores relacionais e cada um deles terá uma função diferente quando for testar a condição, como pode ser visto:

```
== --> Igual a
!= --> Diferente de
>= --> Maior ou igual a
<= --> Menor ou igual a
>  --> Maior que
<  --> Menor que
```

Figura 7. Operadores relacionais. (Autoria própria)

Já os operadores das expressões lógicas, são chamados de operadores lógicos e cada deles tem a função de combinar duas expressões relacionais, de diferentes formas, como pode ser visto:

```
and --> E
or  --> Ou
not --> Negação
```

Figura 8. Operadores lógicos. (Autoria própria)

O operador **and** terá a função de conectar duas expressões relacionais de modo que a condição só seja válida quando as duas expressões relacionais sejam cumpridas.

O operador **or** terá a função de conectar também duas expressões relacionais de modo que a condição só seja válida quando uma das duas expressões relacionais seja cumprida.

O operador **not** terá a função de reverter a expressão, no caso se ela for true vira false e se for false vira true. Existe a tabela que indicará como os operadores se comportam quando são testadas as condições em cada um deles, como pode ser visto:

Tabela 3. Tabela verdade operador and (Autoria própria)

| Tabela Verdade Operador And | | |
|---|---|---|
| Condição A | Condição B | ( A and B ) |
| True | True | True |
| True | False | False |
| False | True | False |
| False | False | False |

Tabela 4. Tabela verdade operador or (Autoria própria)

| Tabela Verdade Operador Or | | |
|---|---|---|
| Condição A | Condição B | ( A or B ) |
| True | True | True |
| True | False | True |
| False | True | True |
| False | False | False |

Tabela 5. Tabela verdade operador not (Autoria própria)

| Tabela Verdade Operador Not | |
|---|---|
| Condição A | ( Not a ) |
| True | False |
| False | True |

### 2.1.5 If

No Python nem todas as linhas do seu código precisam ser lidas, pois podemos ter condições no programa que restringe essas linhas que seriam lidas, mas então por que eu escreveria essas linhas?

Por que você pode atribuir uma condição e se ela for cumprida o seu programa lerá esse código escrito e se não for cumprida ele não fará a leitura, possibilitando assim escolhas que podem ser feitas ou não no seu programa. Exemplo: Supondo que você possua um programa que representa um caixa eletrônico, teríamos opções como:
1- Sacar, 2- Depositar, 3- Transferir, 4- Sair

Se o usuário digita o número 1, e o seu programa, não possui condições que verifiquem essa escolha, ele irá sacar o dinheiro, mas também irá pedir para depositar, depois transferir e por último sair. Mas se o seu programa possui condições e o usuário digitar 1, o programa verá qual foi a escolha e encaminhará o código até as linhas que atendem aquela condição, fazendo com que a única ação do usuário seja o "Sacar".

O **if** no Python é a nossa estrutura de decisão, com ela poderemos escolher se uma ação será tomada ou não, deste modo o **if** nada mais é do que o "se", "se" tal condição for cumprida faça isso, "se" outra condição for cumprida faça aquilo.

```
x = 2
y = 4
z = x+y

if z == 6:
    print(f'A soma de {x} mais {y} é {x+y}')
```

Figura 9. Estrutura if. (Autoria própria)

Nesse exemplo, se a soma de x e y for igual a 6 o programa mostrará ao usuário que a soma será 6, se não for, o programa não mostrará nada e irá finalizar. É importante salientar que podemos utilizar o **if** várias vezes em sequência para colocarmos várias condições.

O comando **else** é usado para caso o resultado da condição seja falso, assim o programa executará uma outra decisão e não irá apenas finalizar.

```
x = 2
y = 2
z = x+y

if z == 6:
    print(f'A soma de {x} mais {y} é {z}')
else:
    print(f'A sua soma não deu igual a 6, mas deu igual a {z}')
```

Figura 10. Estrutura if-else. (Autoria própria)

Nesse exemplo vemos que a soma de x e y irá dá igual a 4, logo não irá cumprir a condição, mas com o **else** o programa não vai finalizar e sim executar a linha **else**, respondendo ao usuário que a soma não deu igual a 6 e sim igual a 4.

Com **else** podemos aninhar vários **if** para que possamos colocar mais condições e assim ampliar mais ainda os programas. Também é possível trocar o **else** aninhado pelo comando **elif**, perceba que o **elif** é a junção do **else** mais o **if**, descomplicando e encurtando bastante o código.

### 2.1.6 Repetições

No Python possuímos funções que repetem uma ação, o comando **for** para repetições que já sabemos quantas vezes queremos repetir e o comando **while** que serve para repetições que não sabemos quantas vezes iremos repetir, essas funções ficarão mais fáceis de serem entendidas abaixo com os exemplos.

**For** é a estrutura que muitas vezes usamos em somatórios no curso, por exemplo, caso precisamos calcular esse somatório de i=1, n=36 e t=i^2, a mão daria muito trabalho e a programação está disponível para facilitar a sua vida, veja como é fácil responder esse somatório em Python.

```
n=int(input('digite o valor inicial de n: '))

soma=0

for i in range(0,n,1):

    soma = soma + (i+1)**2

print(soma)
```

Figura 11. Estrutura for. (Autoria própria)

Olhando para o programa linha por linha, vemos que temos a solicitação da variável n ao usuário, temos também uma variável chamada soma que está recebendo um valor 0, isso por que a necessidade de declaramos essa variável logo, para que ela fique armazenando valores, em seguida temos o comando for, veja a estrutura, em uma tradução livre teríamos o comando **for** assim: para i em uma variação de 0 a n em passo 1, isso é nossa linha for i in range(0,n,1), vendo agora a tradução conseguimos entender que o programa irá repetir até que a variável i chegue ao valor da variável n, e isso tem que ser no passo de 1 em 1, pois foi escolhido na estrutura.
Continuando vemos que a variável soma irá receber a soma mais a variável i mais 1 ao quadrado, esse mais 1 é apenas para que a variável comece do valor 1, pois nosso i deve começar com valor 1 no somatório, como foi especificado.

O **while** é um pouco diferente, nele não sabemos ao certo quantas vezes iremos repetir a ação, diferente do **for** e isso dará a oportunidade de termos infinitas repetições até que a condição de parada da repetição seja cumprida. Exemplo estrutura **while** em Python:

```
n = 's'
while n == 's':
    x = int(input('Digite um número: '))
    n = input('Deseja continuar? s/n ')

print('Até mais')
```

Figura 12. Estrutura while. (Autoria própria)

Observando a estrutura **while**, vemos que a condição para que o **while** continue é que a variável 'n' sempre receba valor igual a 's' e para que ela seja encerrada basta digitar qualquer outro valor, de modo que sempre que você digitar 's' para variável 'n', a estrutura irá repetir infinitamente. Lembrando que existem várias maneiras de condicionar a estrutura **while**, na secção de condições deste artigo existem vários exemplos que podem ser implementados no **while**.

### 2.1.7 Listas

No Python existe uma variável que permite armazenar vários valores, esse tipo de variável é a lista, ela por sua vez pode conter zero ou até mais elementos de um mesmo tipo ou de tipos diversos, assim, acaba por poder conter até mesmo outras listas. Os valores da lista são acessados por índice que seria basicamente a colocação de valor dentro da lista, lembrando que o tamanho da lista é igual à quantidade de elementos que ela possui.

```
l=[]

k=[25,36,12]

d=["a","b","c"]

i=[2.2,5.3,1.7]
```

Figura 13. Listas. (Autoria própria)

Esses são exemplos de listas que seguem seu tipo na ordem de cima para baixo de forma vazia, inteira, string e float. Para pegarmos um dos valores da lista é necessário indicar o índice onde esse valor está alocado, os índices começam a partir do valor 0 até o valor do tamanho da lista -1, pois o índice começa no valor 0 e essa diferença é retirada quando diminuímos 1 do tamanho total da lista. Para lermos quantos itens estão alocados em nossa lista, utilizamos o comando **len**, ele irá contar a quantidade de itens na lista e te dirá o valor total, para adicionarmos mais um elemento a lista utilizamos o comando **append**, de forma que o elemento adicionado ficará no último índice quando adicionado na lista e por fim para retirarmos um elemento da lista utilizamos o comando **del**, ele retira da lista o elemento indicado pelo índice que você escolher.

```
k=[25,36,12]

len(k)

k.append(18)

del(k[2])
```

Figura 14. Manipulação de listas. (Autoria própria)

Utilizando o comando print para verificarmos as alterações na lista veremos que se fizemos primeiro o comando **len**, ele irá dizer que o tamanho da lista é 3, em seguida o comando **append** irá adicionar no último índice da lista o valor de 18 e por fim o comando **del**, irá deletar o índice 2 da lista, logo o valor 12, lembrando que os índices da lista começam no valor 0.

## 2.2 ANÁLISE DE DADOS

### 2.2.1 Biblioteca

A biblioteca do Python é uma coleção de módulos de script, ou seja, um arquivo Python com instruções e definições em Python, portanto poderemos escolher na biblioteca módulos que irão simplificar o processo de programação e remover a necessidade de escrevermos comandos que já estão nesses módulos.
Para utilizarmos uma biblioteca é necessário importá-la. No Python, usamos o comando **import**, além disso podemos também atribuir outros nomes a biblioteca utilizando o comando as para facilitar quando a chamarmos no programa.

```
import pandas


import pandas as pd
```

Figura 15. Importando biblioteca. (Autoria própria)

A principal biblioteca que utilizamos para análise de dados no Python é a biblioteca Pandas, sem dúvidas uma das mais simples para manipulação e análise de dados, sendo possível através dela ler, manipular, agregar e plotar os dados em poucos passos.

### 2.2.2 Pandas

Para iniciarmos a análise de dados é necessário saber quais comandos utilizamos no Pandas para primeiro, alocar esses dados e em seguida tratá-los a partir do seu objetivo, dito isso, o comando de alocação de dados no Pandas é o comando **read**, ele irá ler nosso arquivo onde estão os dados e irá colocá-los em uma variável, o **read** pode ler diversos tipos de arquivos, desde arquivos excel até arquivos csv, a única coisa que irá mudar será uma parte do comando para cada tipo de arquivo.

```
dados = pd.read_xlsx("Diretório.xlsx")

dados2 = pd.read_csv("Diretório.csv")
```

Figura 16. Comando read. (Autoria própria)

Como é possível ver para o comando **read** ler o arquivo é necessário especificar dentro da sua linha de código qual arquivo quer utilizar, além disso é necessário também descrever o diretório de onde o arquivo se encontra, assim o arquivo será alocado na variável que você escolheu, criando assim um dataframe.
Com os dados alocados, agora podemos visualizá-lo com o comando **display** que irá mostrar a quantidade de linhas e colunas, em formato de tabela. Além do **display** também temos o comando **head** que irá mostrar apenas as primeiras 5 linhas dessa tabela formada.
Utilizando esses dois comandos teremos duas apresentações diferentes dessa tabela, veja:

```
display(dados_covid)
```

| | epidemiological_week | date | order_for_place | state | city | city_ibge_code | place_type |
|---|---|---|---|---|---|---|---|
| 0 | 202136 | 2021-09-05 | 543 | RN | NaN | 24.0 | state |
| 1 | 202134 | 2021-08-26 | 501 | RN | Acari | 2400109.0 | city |
| 2 | 202134 | 2021-08-26 | 517 | RN | Açu | 2400208.0 | city |
| 3 | 202134 | 2021-08-26 | 479 | RN | Afonso Bezerra | 2400307.0 | city |
| 4 | 202134 | 2021-08-26 | 425 | RN | Água Nova | 2400406.0 | city |
| ... | ... | ... | ... | ... | ... | ... | ... |
| 164 | 202134 | 2021-08-26 | 468 | RN | Várzea | 2414704.0 | city |
| 165 | 202134 | 2021-08-26 | 455 | RN | Venha-Ver | 2414753.0 | city |
| 166 | 202134 | 2021-08-26 | 456 | RN | Vera Cruz | 2414803.0 | city |
| 167 | 202134 | 2021-08-26 | 463 | RN | Viçosa | 2414902.0 | city |
| 168 | 202134 | 2021-08-26 | 451 | RN | Vila Flor | 2415008.0 | city |

169 rows × 16 columns

Figura 17. Comando display. (Autoria própria)

```
dados_covid.head()
```

| | date | state | city | place_type | confirmed | deaths | is_last | estimated_population |
|---|---|---|---|---|---|---|---|---|
| 0 | 2021-09-05 | AM | Alvarães | city | 2908 | 46 | True | 16220.0 |
| 1 | 2021-09-05 | AM | Amaturá | city | 2389 | 18 | True | 11736.0 |
| 2 | 2021-09-05 | AM | Anamã | city | 1997 | 10 | True | 13956.0 |
| 3 | 2021-09-05 | AM | Anori | city | 1914 | 33 | True | 21477.0 |
| 4 | 2021-09-05 | AM | Apuí | city | 1478 | 31 | True | 22359.0 |

Figura 18. Comando head. (Autoria própria)

Como podemos ver, o comando **display** mostra o número de linhas e colunas da tabela, além de mostrar as 5 primeiras linhas e as 5 últimas, já o comando **head** não mostra a quantidade de linhas em colunas e faz apenas a demonstração das 5 primeiras linhas.

Esses 2 comandos de visualização podem ser muito úteis, pois podemos ter uma ideia do que o database se trata sem exigir a abertura total do arquivo.

### 2.2.3 Tratamento de Dados

Bem, antes de tentarmos extrair informações dos nossos dados temos que tratá-los, pois inúmeras vezes a base de dados pode ter dados errados, incompletos ou distorcidos, além de, na maioria das vezes, não estarem tabelados e contados.

Por exemplo, utilizando os dados da eleição de 2020 dos candidatos do estado de São Paulo retirados do site repositório de dados eleitorais do TSE (TRIBUNAL SUPERIOR ELEITORAL. **Repositório de dados**, 2021. Disponível em: <https://www.tse.jus.br/eleicoes/estatisticas/repositorio-de-dados-eleitorais-1>. Acesso em: 11, out e 2021).

Podemos tabelar os candidatos em várias categorias. uma delas, por exemplo, são os candidatos inaptos, de modo que apenas alocamos todos os candidatos nessa situação em uma variável, veja:
Exemplo do comando para catalogar os candidatos inaptos das eleições de 2020 no estado de São Paulo

```
inaptos_sp = pref_vicepref_sp[(pref_vicepref_sp['DS_SITUACAO_CANDIDATURA']=='INAPTO')]
```

Figura 19. Catálogo de inaptos. (Autoria própria)

Além disso com esse novo catálogo já podemos retirar algumas informações, como por exemplo, o número de prefeitos e vice-prefeitos que estão inaptos e até o gênero majoritário de inaptos, observe:

```
inaptos_sp['DS_CARGO'].value_counts()
```

```
VICE-PREFEITO    221
PREFEITO         179
Name: DS_CARGO, dtype: int64
```

```
inaptos_sp['DS_GENERO'].value_counts()
```

```
MASCULINO    334
FEMININO      66
Name: DS_GENERO, dtype: int64
```

Figura 20. Inaptos. (Autoria própria)

Outro exemplo de tratamento é a seleção, como dito anteriormente os dados podem ser encontrados errados, incompletos ou distorcidos. Neste exemplo dos dados do TSE, ao tratar os dados observa-se que alguns dados se encontram duplicados, pois candidatos estavam contados mais de uma vez nessa base de dados, isso provavelmente é um erro quando o candidato fez o registro, enviando duas vezes o seu pedido de candidatura. Para a retirada desses dados duplicados, primeiramente teremos que verificar se existem esses dados duplicados, fez-se o uso do comando **len** junto ao comando **duplicated**, assim o comando **len** contará quantos dados estão duplicados, e o comando **duplicated** irá verificar a partir de parâmetros, quais dados estão duplicados, veja:

```
len(candidatos_sp[candidatos_sp.duplicated(['CD_CARGO','NR_CANDIDATO','SG_UE'])])
```

```
96
```

```
final_sp = candidatos_sp.drop_duplicates(subset = ['CD_CARGO','NR_CANDIDATO','SG_UE'], keep = 'last', ignore_index=True)
```

Figura 21. Retirando os duplicados. (Autoria própria)

Com o comando **duplicated** e **len**, foi descoberto que 96 dos candidatos estavam com suas informações duplicadas, então em seguida aplicando o comando **drop.duplicates** que através do comando **drop** irá remover todas as últimas informações duplicadas que possuem os mesmos parâmetros, assim eliminando os dados duplicados.

Existem também comandos como o **count** que mostra a quantidade de vezes que um mesmo elemento está contido no dataframe, destaca-se ainda o comando **iloc** que irá selecionar linhas e colunas através dos números das linhas e colunas escolhido.

Lembrando que todos esses exemplos são voltados para a análise desse arquivo escolhido, provavelmente para a sua análise caso escolha outro arquivo com outras informações, você terá que fazer uma adaptação utilizando estes e outros comandos.

### 2.2.4 Análise de dados

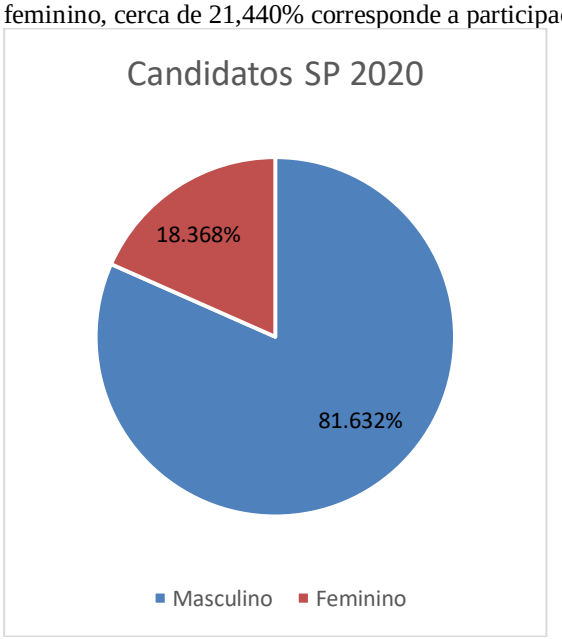
Bem com todos os passos explicados até aqui, a análise de dados pode ser feita e iremos então chegar ao objetivo de obter as informações se existe ou não um aumento da participação feminina em cargos da política nos estados de São Paulo e Alagoas durantes os anos e se por acaso algumas cidades com IDH alto do estado de São Paulo possuem maior participação feminina nesse cenário estabelecido com relação às cidades com IDH alto do estado de Alagoas.

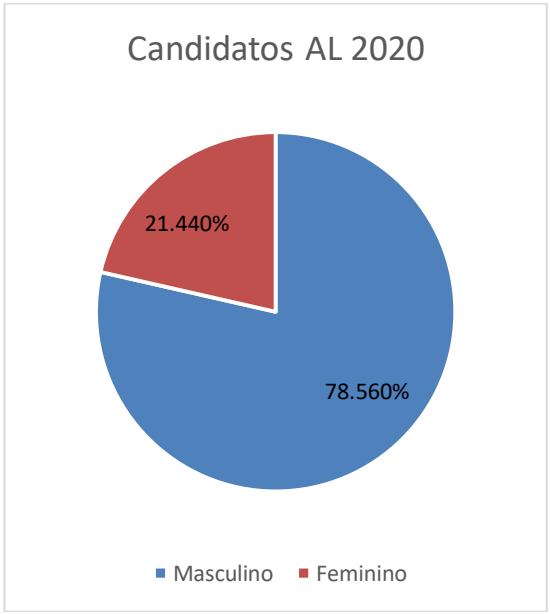
Para que possamos conseguir essas informações iremos ao site do repositório do TSE e coletar os dados dos candidatos de 2020, 2016 e 2012.

Com isso faremos todo o tratamento de dados já explicado e iremos catalogar os candidatos das 3 cidades que estão entre os maiores IDHs do estado de São Paulo, sendo essas cidades Águas de São Pedro, Santos e Jundiaí, iremos também catalogar os candidatos das 3 cidades que estão entre os maiores IDHs do estado de Alagoas, sendo essas cidades Maceió, Satuba e Arapiraca.

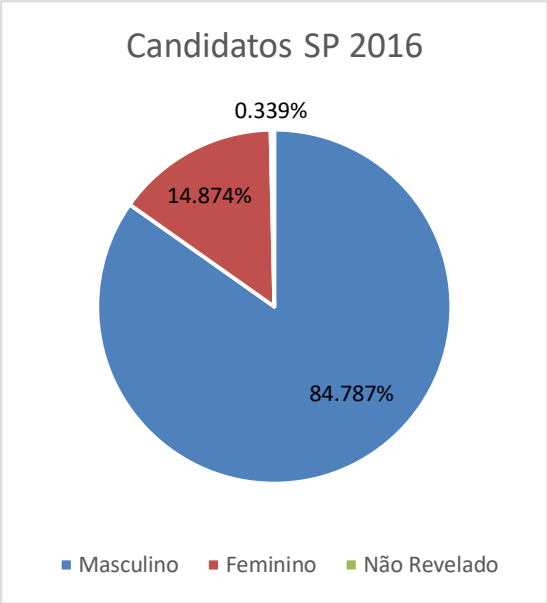
## 2.3. Resultados e Discussões

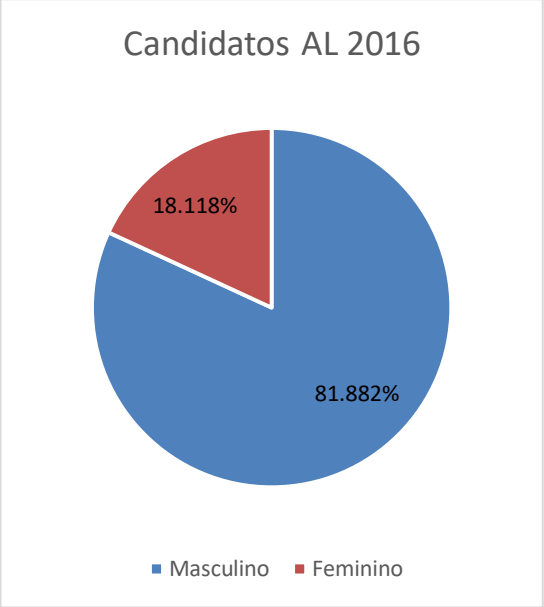
Começando com a hipótese se houve aumento de número de participantes feminino durante os anos de 2020, 2016 e 2012, nos estados de São Paulo e Alagoas, devemos utilizar o comando count, assim saberemos que entre os candidatos a prefeito e vice-prefeito temos, que em 2020 ouve 4976 candidaturas em São Paulo, sendo 2466 candidatos a prefeito e 2510 candidatos a vice-prefeito, desse total 4062 são do sexo masculino e 914 são do sexo feminino, cerca de 18,368% corresponde a participação feminina. Já no estado de Alagoas temos 611 candidatos no total sendo 301 a prefeito e 310 a vice-prefeito, desse total 480 são do sexo masculino e 131 são do sexo feminino, cerca de 21,440% corresponde a participação feminina.



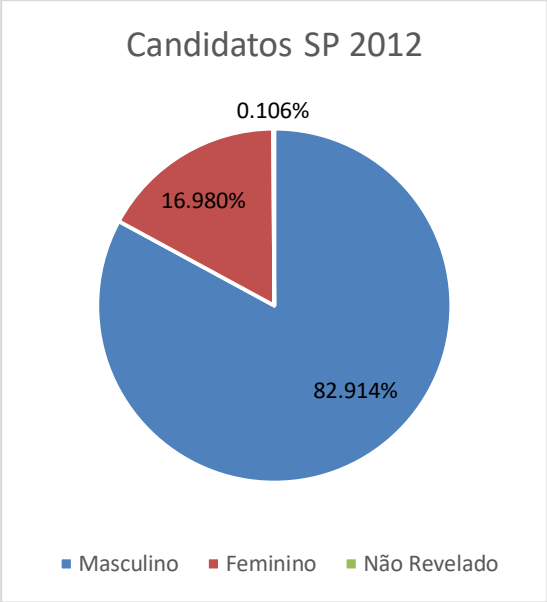



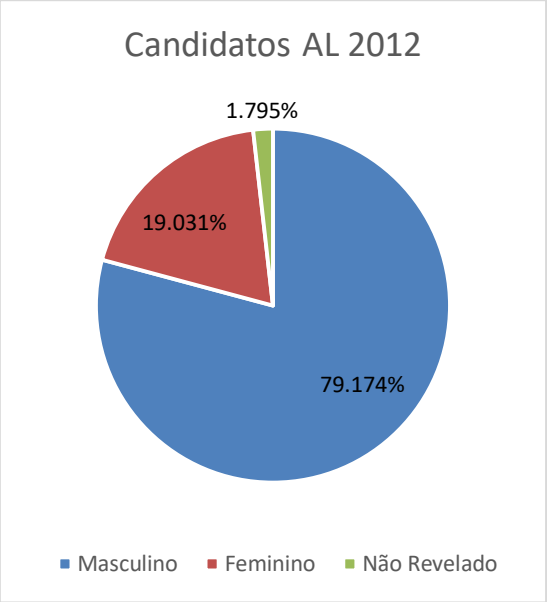
Passando para 2016 temos que em São Paulo ouve 4128 candidaturas, sendo 2070 para prefeitos e 2058 para vice-prefeitos, desse total 3500 são do sexo masculino e 614 são do sexo feminino, houve também 14 candidaturas que não foi revelado o sexo, temos que 14,874% corresponde a participação feminina. Em Alagoas houve 574 candidaturas 281 para prefeito e 293 para vice-prefeito, sendo 470 do sexo masculino e 104 do sexo feminino, temos que 18,118% corresponde a participação feminina.




Agora para 2012 temos que em São Paulo ouve 3781 candidaturas, sendo 1849 para prefeitos e 1932 para vice-prefeitos, desse total 3135 são do sexo masculino e 642 do sexo feminino, houve também 4 candidaturas que não foi revelado o sexo, temos que 16,980% corresponde a participação feminina. Já no estado de Alagoas temos 557 candidatos no total sendo 273 prefeitos e 284 vice-prefeitos, desse total 441 são do sexo masculino e 106 do sexo feminino, houve também 10 candidaturas que não foi revelado o sexo, temos que 19,031% corresponde a participação feminina.



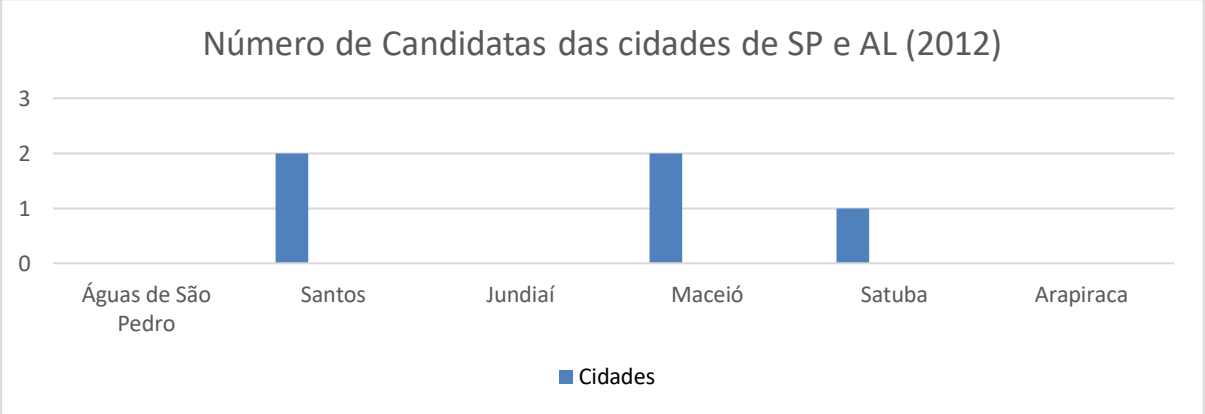

Analisando os dados percebemos que no estado de São Paulo em 2012 tivemos 16,980% de participação feminina nas candidaturas, um total de 642 candidatas a prefeito e vice-prefeito, enquanto em 2016 houve 14,874% de participação feminina dando um total de 614 candidatas, portanto uma queda de 4,361% ou 28 candidatas da contagem total de candidatos, já comparando 2016 com 2020, temos que em 2016 houve 14,874% de participação feminina que representa 614 candidatas e em 2020 houve 18,368% de participação feminina representando 914 candidatas, um aumento de 48,860% e um total de 300 candidatas, sendo superior a queda que houve no período de 2012 para 2016, demonstrando assim durante essas 3 últimas eleições um aumento de candidatas a concorrência do cargo de prefeita e vice-prefeita no estado de São Paulo.

Partindo para o estado de Alagoas vemos que em 2012 teve 19,031% de participação feminina nas candidaturas sendo um total de 106 candidatas a prefeito e vice-prefeito, enquanto em 2016 houve 18,118% de participação

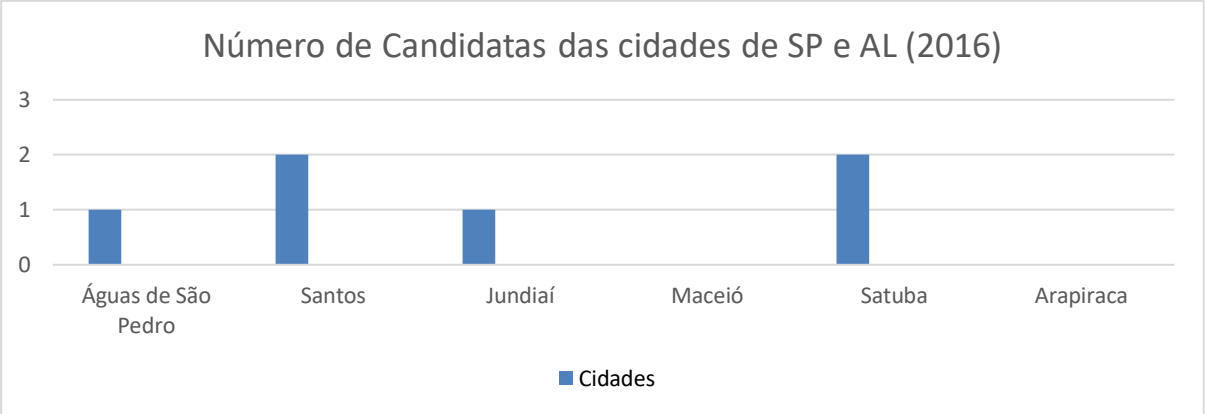
feminina dando um total de 104 candidatas, portanto uma queda de 1,887% ou 2 candidatas, já comparando 2016 com 2020, temos que em 2016 houve 18,118% de participação feminina, um total de 104 candidatas e em 2020 ouve 21,440% de participação feminina, sendo 131 candidatas, um aumento de 25,961% e um total de 27 candidatas, sendo superior a queda que houve no período de 2012 para 2016, demonstrando também que nestas 3 últimas eleições houve um aumento de candidatas a concorrência do cargo de prefeita e vice-prefeita no estado de Alagoas.

Partindo para a segunda hipótese se entre os estados que possui maior IDH a presença maior de mulheres concorrendo aos cargos de prefeito ou vice-prefeito em comparação ao estado de menor IDH, para isso iremos catalogar os candidatos a prefeito e vice-prefeito das cidades de Águas de São Pedro, Santos e Jundiaí que pertencem ao estado de São Paulo, em seguida catalogar também os candidatos a prefeito e vice-prefeito das cidades de Maceió, Satuba e Arapiraca que pertencem ao estado de Alagoas.

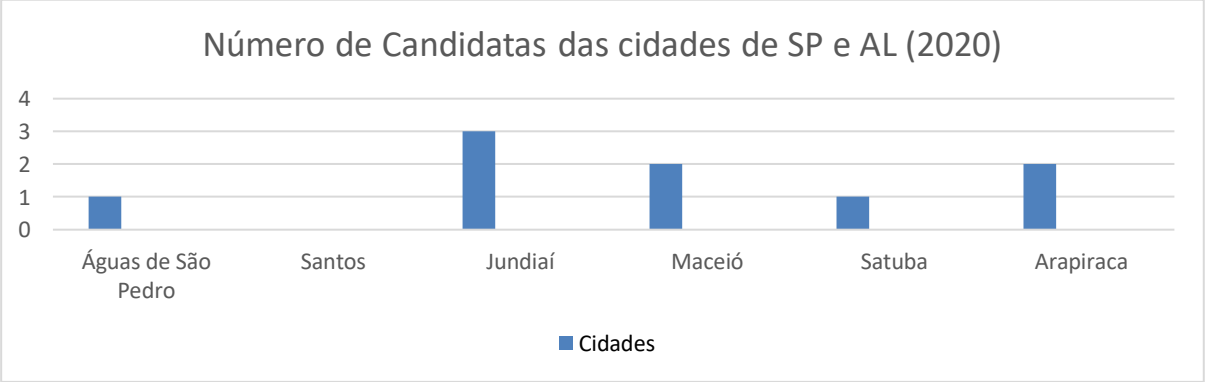
No total a soma de candidatas nas cidades do estado de São Paulo em 2012 foram, uma candidata em Águas de São Pedro e duas candidatas em Santos, em Jundiaí não houve candidatas. Já nas cidades de Alagoas houve duas candidatas em Maceió, uma candidata em Satuba e uma candidata em Arapiraca.




Em 2016 nas cidades do estado de São Paulo houve uma candidata em Águas de São Pedro, duas candidatas em Santos e uma candidata em Jundiaí. Nas cidades do estado de Alagoas houve uma candidata em Maceió, duas candidatas em Satuba e uma candidata em Arapiraca.




Por fim, em 2020 temos nas cidades do estado de São Paulo, uma candidata em Águas de São Pedro e três candidatas em Jundiaí, em Santos não houve candidatas. Partindo para o estado de Alagoas, em Maceió houve duas candidatas, em Satuba uma candidata e por fim em Arapiraca duas candidatas.



Analisando os dados e nesse contexto apresentado, podemos concluir que houve eleições em que as cidades de maior IDH, possuíram maior números de candidatas e em outras eleições houve maior número de candidatas nas cidades de baixo IDH, mas somando todas as candidatas das cidades de IDH alto e somando também todas as candidatas das cidades de IDH baixo durante as três últimas eleições temos um empate de dez candidatas para ambas, deixando a conclusão que o IDH não interfere nesse cenário apresentado.

## 3. CONCLUSÕES

A busca pela informação e a extração de dados dessa informação vem crescendo durante os anos e os cientistas de dados estão aprimorando a cada dia mais uma melhor maneira de como chegar a esses dados, o trabalho aqui apresentado reflete como podemos trabalhar com essas informações e extrair seus resultados, partindo de como podemos utilizar a linguagem Python e chegando na análise final sobre as eleições.

Partindo das análises, podemos ver que a primeira hipótese foi confirmada, o número de mulheres presente no cenário político de prefeito e vice-prefeito nos estados de São Paulo e Alagoas, cresceu nessas três últimas eleições, já a segunda hipótese não deixa claro se as cidades do estado de São Paulo possuem mais concorrentes femininas do que as cidades do estado de Alagoas, devido que em algumas eleições as cidades do estado de São Paulo tinham maior número e em outras eleições as cidades de Alagoas tinham maior número, chegando em um resultado que a soma total de candidatas nas três últimas eleições empatou sendo 10 candidatas para ambos os lados.

Por fim deixando algumas sugestões, caso o leitor queira analisar mais algumas cidades dos estados de São Paulo e Alagoas para aprofundar na questão da participação das mulheres e o IDH e elevar o trabalho para as candidaturas de vereadores, tendo uma base maior pois existem mais candidaturas a vereadores, por fim analisando no total a presença feminina na política nesses estados.

## 4. REFERÊNCIAS BIBLIOGRÁFICAS


[1] MENEZES, N. N. C. **Introdução à programação com Python:** Algoritmos e Lógica de Programação Para Iniciantes. Novatec, 2019.

[2] Como abrir(importar) um dataset em # Python? programação dinâmica. [*S.l.:s.n*], 2019. 1 vídeo(14min). Publicado pelo canal Programação Dinâmica. Disponível em: https://www.youtube.com/watch?v=3k_j5SKy8k0. Acesso em: 28 out 2021.

[3] HASHTAG(Brasil). Pandas. *In*: HASHTAG(Brasil). **Introdução ao Pandas.** [Rio de Janeiro/RJ]: Hashtag, 2021. Disponível em: https://www.hashtagtreinamentos.com/introducao-pandas-python. Acesso em: 29 out 2021.

[4] Introdução a Análise de Dados com Python hashtag programação. [*S.l.:s.n*],2021. 1 vídeo(24min). Publicado pelo canal Hashtag Programação. Disponível em: https://www.youtube.com/watch?v=kCMaqla6Grs. Acesso em: 29 out 2021.

[5] Tratamento para análise de dados eleições 2020 python fun. [*S.l.:s.n*],2019. 1 vídeo(24min). Publicado pelo canal Python Fun. Disponível em: https://www.youtube.com/watch?v=TCAX4prFHpI. Acesso em: 29 out 2021.

[6] Introdução ao Pandas no Python – [saia do zero em 1 aula]. [*S.l.: s.n.*], 2021. 1 vídeo (51 min). Publicado pelo canal Hashtag Programação. Disponível em: https://www.youtube.com/watch?v=C0aj3FjN5e0. Acesso em: 8 nov 2021.

[7] MCKINNEY, W. **Python Para Análise de Dados:** Tratamento de Dados com Pandas, NumPy e IPython. Novatec, 2018.

MINISTÉRIO DA EDUCAÇÃO
Universidade Federal Rural do Semi-Árido – UFERSA
*Centro de Ciências Exatas e Naturais – CCEN*

# ATA DE DEFESA DE TRABALHO DE CONCLUSÃO DE CURSO

Às dez horas e trinta minutos do dia dezesseis de novembro, via Google Meet da Universidade Federal Rural do Semi-Árido, reuniu-se a Banca Examinadora de defesa de trabalho de conclusão de curso de autoria do aluno **Antônio Eduardo de Oliveira Carmo**, aluno do curso de Bacharelado em Ciência e Tecnologia desta Universidade, N° de matrícula **2017020624**, com o título " **UMA ANÁLISE DE DADOS NO PYTHON SOBRE A PARTICIPAÇÃO FEMININA NOS ESTADOS DE SÃO PAULO E ALAGOAS**". A Banca Examinadora ficou assim constituída por três membros: Prof. Dr. Fabricio de Figueredo Oliveira, presidente da banca e orientador do Trabalho de Conclusão de Curso; Prof. Dr. Walter Martins Rodrigues e Prof. Dr. Davi Ribeiro dos Santos, como membros. Concluída a defesa, procedeu-se o julgamento pelos membros da banca examinadora, em reunião fechada, tendo o aluno sido considerado APROVADO. E para constar, eu, Fabricio de Figueredo Oliveira, lavrei a presente ata que, após lida e aprovada pelos membros da banca examinadora, será assinada por todos.

Mossoró, 16 de novembro de 2021.

Assinatura dos membros da Banca Examinadora.

FABRICIO DE FIGUEREDO OLIVEIRA:64905896304
Assinado de forma digital por FABRICIO DE FIGUEREDO OLIVEIRA:64905896304
Dados: 2021.11.16 07:52:26 -03'00'

Prof. Dr. FABRICIO DE FIGUEREDO OLIVEIRA – UFERSA-RN
Presidente e orientador

gov.br
Documento assinado digitalmente
WALTER MARTINS RODRIGUES
Data: 16/11/2021 14:06:08-0300
Verifique em https://verificador.iti.br

Prof. Dr. WALTER MARTINS RODRIGUES – UFERSA-RN
Primeiro Membro

Davi Ribeiro dos Santos

Prof. Dr. DAVI RIBEIRO DOS SANTOS – UVA-CE
Segundo Membro