Documentação Sprint 5

O arquivo utilizado foi o “fatec_pessoa_fisica”, foram resgatadas as colunas CPF e uma coluna nula. O sistema de filtragem foi exibir os CPF’s válidos e separar por estado. Alguns CPF’s tem um identificador que é comum para mais de um estado. De um total de 3002 dados inseridos 192 eram CPF’s válidos, o que se refere a 6,4% de dados válidos para 93,6% de dados inválidos.

Ao chamar os dados no *jupyter notebook* os CPF’s, embora números inteiros, foram exibidos em notação científica com ponto flutuante (Fig. 1). Perdeu-se a precisão com os valores em notação científica, primeiro eliminou-se os dados nulos com o método *dropna(subset=[‘’])* o tratamento nesse estágio reduziu a quantidade de linhas de 3011 para 3002, em seguida fez-se uma função para conversão dos dados para números inteiros com sua aplicação na coluna CPF (Fig. 2).

Figura 1: CPF’s em notação científica. Tem um total de 3011 linhas (3010 mais a linha zero).

	nulo	cpf
0	NaN	NaN
1	NaN	NaN
2	NaN	3.731117e+10
3	NaN	3.411527e+10
4	NaN	3.562266e+10
...	...	...
3006	NaN	2.238859e+11
3007	NaN	2.658650e+11
3008	NaN	2.622086e+11
3009	NaN	2.554871e+11
3010	marcador	NaN

Figura 2: função converte criada e sua aplicação na tabela. A sintaxe da função é mostrada em In [166] e a tabela é exibida logo abaixo. Note que as linhas restantes são 3002. Do inglês: 3002 *rows*.

```
In [166]: def converte(c):
              c = int(c)
              return c
```

	nulo	cpf
2	NaN	37311167211
3	NaN	34115266395
4	NaN	35622655491
5	NaN	33222324480
6	NaN	1505546115
...	...	...
3005	NaN	255051104931
3006	NaN	223885940544
3007	NaN	265864987976
3008	NaN	262208637843
3009	NaN	255487112307

3002 rows × 2 columns

Para a verificação da validade dos CPF's pegou-se cada número que o constitui e somou-se entre si. Se o primeiro algarismo do resultado da soma fosse igual ao segundo, então o CPF é válido. Por exemplo: 44, 4 = 4 (Verdadeiro), então o CPF é válido; 43, 4 = 3 (Falso), então o CPF é inválido. Para a verificação dos estados analisou-se o último número do CPF antes do dígito. Exemplo: XXX.XXX.XX(**NUM**)-XX (CPF-Dígito), em **NUM** lê-se o algarismo de 0 a 9 que dirá qual é o estado (COMO SABER SE UM CPF É VERDADEIRO?, 2017).

Para somar os algarismos do CPF entre si e verificar a validade deles, usou-se a função da figura 3 In[167] e, para a identificação dos estados, criou-se a função da figura 3 In[168]. O resultado é exibido na tabela 1. Pode-se verificar que há identificadores de estado que correspondem a mais de uma compatibilidade de estado como visto na linha 1, coluna 1: Rio de Janeiro-Espírito Santo. Depois de tratados os dados, montou-se um gráfico de barras horizontais com o total de CPF's inseridos com o total de CPF's válidos e um gráfico de barras horizontais mostrando a quantidade de CPF's por estado (Fig. 4).

Figura 3: funções "valido" e "uf" para a validação do CPF e verificação dos estados respectivamente. **In[167]:** soma e verificação dos CPF's com o retorno somente dos válidos. **In[168]:** identificação dos estados dos CPF's válidos.

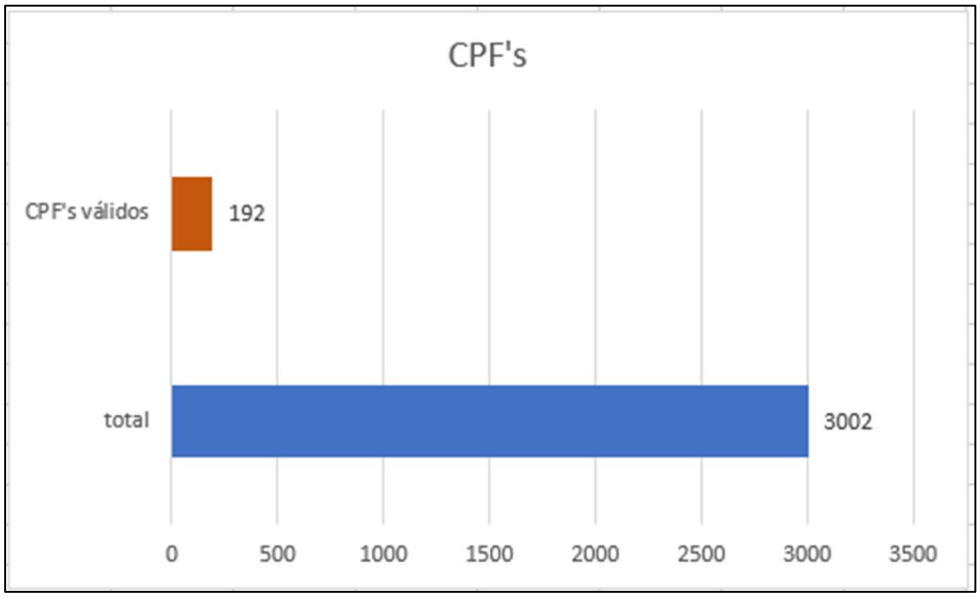
```
In [167]: def valido(g):
              b = str(g)

              cont = 0
              for i in range(len(b)):
                  cont += int(b[i])

              cont = str(cont)
              if len(cont) == 2:
                  if cont[0] == cont[1]:
                      return g
```

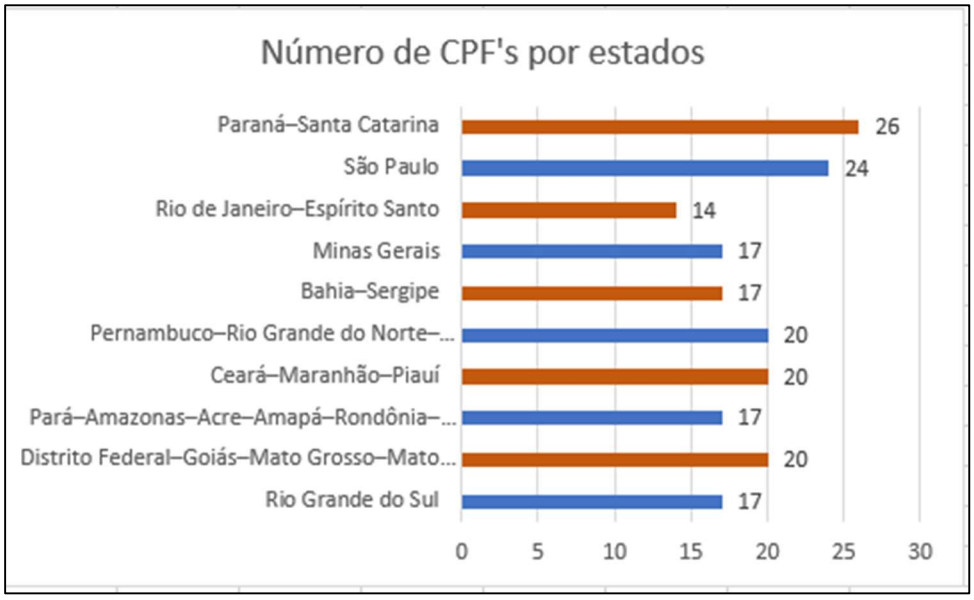
```
In [168]: def uf(h):
              h= str(h)

              if h[-3] == '0':
                  return 'Rio Grande do Sul'
              if h[-3] == '1':
                  return 'Distrito Federal-Goiás-Mato Grosso-Mato Grosso do Sul-Tocantins'
              if h[-3] == '2':
                  return 'Pará-Amazonas-Acre-Amapá-Rondônia-Roraima'
              if h[-3] == '3':
                  return 'Ceará-Maranhão-Piauí'
              if h[-3] == '4':
                  return 'Pernambuco-Rio Grande do Norte-Paraíba-Alagoas'
              if h[-3] == '5':
                  return 'Bahia-Sergipe'
              if h[-3] == '6':
                  return 'Minas Gerais'
              if h[-3] == '7':
                  return 'Rio de Janeiro-Espírito Santo'
              if h[-3] == '8':
                  return 'São Paulo'
              if h[-3] == '9':
                  return 'Paraná-Santa Catarina'
```

Tabela 1: Os estados foram distribuídos em dez linhas com suas respectivas ocorrências, organizados em ordem crescente. Observe que para determinados identificadores de estado há a possibilidade de vários estados.

```
In [183]: a['nulo'].value_counts().sort_values()

Out[183]: Rio de Janeiro-Espírito Santo                                  14
          Pará-Amazonas-Acre-Amapá-Rondônia-Roraima                      17
          Bahia-Sergipe                                                  17
          Minas Gerais                                                   17
          Rio Grande do Sul                                              17
          Ceará-Maranhão-Piauí                                           20
          Pernambuco-Rio Grande do Norte-Paraíba-Alagoas                 20
          Distrito Federal-Goiás-Mato Grosso-Mato Grosso do Sul-Tocantins 20
          São Paulo                                                      24
          Paraná-Santa Catarina                                          26
```

Figura 4: Gráficos mostrando a quantidade de CPF's válidos e o Total de CPF's inseridos (**A**) e o número de CPF's por estado(**B**). **A:** o número de CPF's válido é 192, equivalendo a 63,4% dos dados inseridos para 93,6% dos dados inválidos.





REFERÊNCIA BIBLIOGRÁFICA

COMO SABER SE UM CPF É VERDADEIRO?. **Guarulhos Empresarial**, 2017. Disponível em: <https://www.aceguarulhos.com.br/blog/como-saber-se-um-cpf-e-verdadeiro/#gsc.tab=0>. Acesso em: 28/06/2020.