

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

William Sanches Lima

OBTENÇÃO DE INSIGHTS DOS DADOS DO PORTAL DA TRANSPARÊNCIA
ATRAVÉS DE ANÁLISE EXPLORATÓRIA DOS DADOS

Belo Horizonte
2020

William Sanches Lima

**OBTENÇÃO DE INSIGHTS DOS DADOS DO PORTAL DA TRANSPARÊNCIA
ATRAVÉS DE ANÁLISE EXPLORATÓRIA DOS DADOS**

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

2020

AGRADECIMENTO

Agradeço a minha esposa Erika e aos meus filhos João e Helena pela compreensão e paciência nos momentos de ausência.

Agradeço ao meu amigo Ludgero que me incentivou e me ajudou na organização do meu tempo e estruturação do plano do projeto.

Agradeço aos professores Danilo Costa e Cristiano Carvalho pela mentoria na estruturação das ideias e objetivos do trabalho.

Agradeço aos demais professores e integrantes da equipe PUC Minas pela oportunidade de aprendizado e pela jornada de conhecimento.

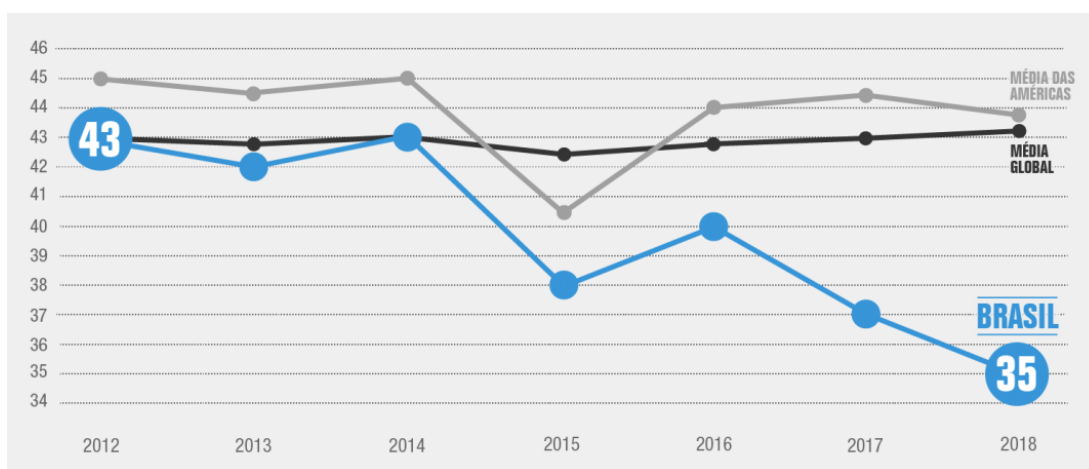
SUMÁRIO

1. Introdução.....	5
1.1. Contextualização	5
1.2. O problema proposto	6
2. Coleta de Dados	7
3. Processamento/Tratamento de Dados	13
4. Análise e Exploração dos Dados	21
5. Apresentação dos Resultados	38
6. Links.....	38
REFERÊNCIAS.....	45

1. Introdução

1.1. Contextualização

Desde 2012 o Brasil vem caindo no ranking do Índice de Percepção da Corrupção (IPC), indicador gerado pela organização Transparência Internacional desde 1995 que avalia a percepção de corrupção em 180 países no mundo. Esse índice é baseado em uma escala onde 0 é um país percebido como altamente corrupto e 100 um país percebido como muito íntegro.



Uma manchete no próprio site da Transparência Internacional afirma o seguinte: “*Brasil mantém a pior pontuação da série histórica, com apenas 35 pontos*”. Nesse mesmo site existe uma lista de recomendações para combate a corrupção entre as quais destaca-se a seguinte: “*Congresso Nacional deve deliberar e aprovar reformas estruturais anticorrupção baseadas no pacote de medidas elaboradas por especialistas brasileiros, as Novas Medidas contra a Corrupção*”. Esse pacote de medidas é um documento elaborado em conjunto com a Escola de Direito do Rio de Janeiro, da Fundação Getúlio Vargas, publicado em 2018 (ISBN: 9788595970205). Nesse documento, dentre uma quantidade grande de sugestões de medidas contra corrupção, ressalta-se a encontrada na seção VII (Melhorias dos Controles Internos e Externos), no tópico 35, que diz o seguinte: “*O controle interno na Administração Pública, exercido a âmbito federal no Brasil pela Controladoria-Geral da União, é uma forma de controle da Administração sobre si mesma. Entre suas finalidades estão a detecção e o combate à corrupção na gestão de recursos públicos*”. Mais adiante,

ainda nesse mesmo tópico, onde está descrito um anteprojeto de lei, lê-se o seguinte no artigo 6º, inciso II: “*Administrar portal da transparência na internet, zelando por fácil acesso, abertura dos dados e completude das informações*”. Ora, é público e notório que a Controladoria Geral da União (CGU) ratifica, através de sua página institucional “Transparência Pública”, que acredita na transparência como melhor forma de combate à corrupção. Nessa mesma página encontra-se os seguintes dizeres: “*A gestão pública transparente permite à sociedade, com informações, colaborar no controle das ações de seus governantes, com intuito de checar se os recursos públicos estão sendo usados como deveriam*”. Ainda nesse contexto a CGU criou o programa Brasil Transparente para auxiliar Estados e Municípios na implementação das medidas de governo transparente previstas na Lei de Acesso à Informação (LAI) com o objetivo de reforçar o avanço da transparência pública e a adoção de medidas de governo aberto. Para atender esses requisitos o Ministério da Transparência e Controladoria-Geral da União lançaram em 2004 o Portal da Transparência do Governo Federal, que é um site de acesso livre, onde pode-se encontrar informações sobre como o dinheiro público é utilizado, além de informar sobre assuntos relacionados à gestão pública do Brasil. Considerando a piora no cenário da corrupção, que, aparentemente, está na contramão de iniciativas como a do Ministério da Transparência e CGU, entende-se que é necessário avaliar se as medidas adotadas para garantir a transparência das informações tem sido, de fato, eficazes para aquilo que foi proposto.

1.2. O problema proposto

O problema proposto foi analisar os dados disponíveis no Portal da Transparência da Controladoria Geral da União para obtenção de “insights”, aplicando técnicas de Análise Exploratória de Dados e Visualização de Dados, com o objetivo de comprovar a eficácia dessas técnicas para esse propósito.

Desde 2012 o Brasil vem caindo no ranking do Índice de Percepção da Corrupção (IPC), elaborado pela organização Transparência Internacional que avalia a percepção da corrupção no setor público em 180 países. Em 2016, o Brasil ficou em 79º; em 2017 o país estava na 96ª colocação; em 2019 ocupou o 105º e agora em

2020 está em 106º lugar. Dessa forma entende-se que o problema em questão é de extrema relevância pois o “custo” da corrupção tem impacto direto na diminuição dos investimentos na saúde, na educação, em infraestrutura, segurança, habitação, entre outros direitos essenciais à vida. Nesse contexto, a avaliação desses dados é importante porque podem ser decisivos no apoio ao processo de seleção e contratação de fornecedores no âmbito da administração pública e também para direcionar os órgãos sancionadores e legisladores no entendimento do cenário afim de possibilitar melhorias nas normas e legislações que tratam desse assunto.

Os dados são do governo, do Portal da Transparência, do Ministério da Transparência e Controladoria-Geral da União, fruto do programa Brasil Transparente para auxiliar Estados e Municípios na implementação das medidas de governo transparente previstas na Lei de Acesso à Informação (LAI).

O objetivo é recuperar os dados do Portal da Transparência para realização de Análise Exploratória dos Dados e Visualização dos Dados afim de se obter “insights”, para comprovar a eficácia da aplicação dessas técnicas. Os dados analisados serão do Cadastro de Empresas Inidôneas e Suspensas (CEIS), do Portal da Transparência da Controladoria Geral da União.

Os dados tem abrangência nacional, mas possuem detalhes relacionados aos estados e municípios, possibilitando a classificação dos dados com essas informações.

2. Coleta de Dados

Os dados foram obtidos do portal da transparência, do Ministério da Transparência e Controladoria-Geral da União. Segue trecho extraído da página “O que é e como funciona” que explica de onde e como os dados são obtidos e com funciona o acesso a esse “dataset”:

“Os dados divulgados no Portal são provenientes de diversas fontes de informação, entre as quais estão os grandes sistemas estruturadores do Governo Federal – como o Sistema Integrado de Administração Financeira do Governo Federal (Siafi) e o Sistema Integrado de Administração de Recursos Humanos (Siape) –, as

bases de benefícios sociais, as faturas de Cartão de Pagamentos do Governo Federal, as bases de imóveis funcionais, entre diversas outras.

Os órgãos responsáveis por cada fonte de informação encaminham seus dados para a CGU, que recebe, reúne e disponibiliza as informações na ferramenta. A periodicidade de envio dos dados depende do assunto tratado, assim como a periodicidade de atualização das informações no Portal.

Uma vez carregadas no Portal, as informações são disponibilizadas para conhecimento do cidadão de diversas formas, como: painéis, consultas detalhadas, gráficos, dados abertos.

O acesso ao Portal não requer usuário nem senhas, sendo permitido a qualquer cidadão navegar pelas páginas de forma livre, bem como visualizar e utilizar os dados disponíveis da forma que melhor lhe convier.”

Analisando as documentações e tutoriais do portal observa-se que os dados são disponibilizados das seguintes formas: acesso web pelo próprio portal para visualização, download de arquivo “.csv” e API de dados. Para o desenvolvimento desse trabalho foi escolhido o consumo dos dados via API e persistência dos mesmo em um banco de dados MongoDB local. Nesse sentido encontra-se abaixo um trecho extraído da página “API de dados” que fornece detalhes sobre essa forma de obtenção dos dados:

“O Portal da Transparência está atento aos princípios de Governo Eletrônico e sabe que os dados devem ser disponibilizados de formas diferentes a fim de atender aos diversos perfis de usuários. Para isso, além de consultas online e com visualizações que buscam transmitir, de forma simples, como o governo usa os recursos públicos, formas de acesso aos dados para desenvolvedores e engajados com a tecnologia da informação também estão disponíveis.

O acesso para desenvolvedores e engajados ocorre através de uma Interface de Programa de Aplicativos (do inglês, “Application Programming Interface”), ou simplesmente “API”. Com ela, é possível ter um serviço de consulta direta aos dados do Portal da Transparência sem precisar navegar pelo site ou utilizar robôs para a obtenção das informações de forma automática. Os dados disponíveis são os mesmos apresentados em tela, com a flexibilidade característica das APIs.

No Portal da Transparência, o serviço de consulta via API foi implementado em REST (Representational State Transfer). Entendemos que o REST é mais simples de ser utilizado, além de poder ser testado aqui mesmo no Portal. Além do mais, o REST costuma ser a escolha preferida dos desenvolvedores.

As consultas via API têm restrição quanto ao número de requisições por minuto, de modo a não sobrecarregar o Portal da Transparência e impactar no tempo de resposta das consultas realizadas por meio dos navegadores. No período de 6:00 às 23:59, o Portal aceita 90 requisições por minuto. Já no período das 00:00 às 5:59, são aceitas 300 requisições por minuto.

O acesso via API fornece toda uma flexibilidade para consultas pontuais. Para acesso ao conjunto completo de dados, no entanto, sugerimos utilizar as planilhas de dados abertos. O Portal da Transparência agrega uma enorme quantidade de dados e baixar as planilhas para uso local certamente fornecerá melhores resultados para grandes volumes de dados. O download de planilhas pode ser feito em seção específica.

A documentação do uso das APIs pode ser acessada em <http://www.transparencia.gov.br/swagger-ui.html>. Estão disponíveis as seguintes consultas: Bolsa Família, Programa de Erradicação do Trabalho Infantil (Peti), Garantia-Safra, Seguro Defeso, Cadastro de Expulsões da Administração Federal (CEAF), Cadastro Nacional de Empresas Inidôneas e Suspensas (CEIS), Cadastro Nacional de Empresas Punidas (CNEP), Contratos do Poder Executivo Federal, Convênios do Poder Executivo Federal, Despesas Públicas, Entidades Privadas sem Fins Lucrativos Impedidas (CEPIM), Licitações do Poder Executivo Federal, Servidores do Poder Executivo Federal e Viagens a Serviço.”

Foi escolhido o Cadastro Nacional de Empresas Inidôneas e Suspensa (CEIS), que apresenta um “dataset” maior e com mais atributos para as análises. Os dados foram obtidos através da API de Consulta dos registros do CEIS. Abaixo a documentação da API:

GET /api-de-dados/ceis

Notas de Implementação

Filtros mínimos: Página (padrão = 1);

Classe de resposta (Status 200)

OK

Example Value:

```
[
  {
    "abrangenciaDefinidaDecisaoJudicial": "string",
    "dataFimSancao": "string",
    "dataInicioSancao": "string",
    "dataOrigemInformacao": "string",
    "dataPublicacaoSancao": "string",
    "dataReferencia": "string",
    "dataTransitadoJulgado": "string",
    "detalhamentoPublicacao": "string",
    "fonteSancao": {
      "enderecoContato": "string",
      "nomeExibicao": "string",
      "telefoneContato": "string"
    },
    "id": 0,
    "informacoesAdicionaisDoOrgaoSancionador": "string",
    "legislacao": {
      "descricaoFundamentacaoLegal": "string",
      "fundamentacaoLegal": "string"
    },
    "linkPublicacao": "string",
    "numeroProcesso": "string",
    "orgaoSancionador": {
      "nome": "string",
      "poder": "string",
      "siglaUf": "string"
    },
    "pessoa": {
      "cnae": {
        "classe": "string",
        "codigoClasse": "string",
        "codigoDivisao": "string",
        "codigoGrupo": "string",
        "codigoSecao": "string",
        "codigoSubclasse": "string",
        "divisao": "string",
        "grupo": "string",
        "secao": "string",
        "subclasse": "string"
      },
      "codigoFormatado": "string",
      "complementoEndereco": "string",
      "dataAbertura": "string",
      "descricaoLogradouro": "string",
      "enderecoEletronico": "string",
      "localidadePessoa": "string",
      "municipio": {
        "codigoIBGE": "string",
        "nomeIBGE": "string",
        "pais": "string",
        "uf": {
          "nome": "string",
          "sigla": "string"
        }
      }
    },
    "naturezaJuridica": {
      "codigo": "string",

```

```

"codigoTipo": "string",
"descricao": "string",
"descricaoTipo": "string"
},
"nome": "string",
"nomeBairro": "string",
"nomeFantasiaReceita": "string",
"numeroCEP": "string",
"numeroEndereco": "string",
"numeroInscricaoSocial": "string",
"numeroTelefone": "string",
"razaoSocialReceita": "string",
"tipoCodigo": "string",
"tipoPessoa": "string"
},
"sancionado": {
"codigoFormatado": "string",
"nome": "string"
},
"textoPublicacao": "string",
"tipoSancao": {
"descricaoPortal": "string",
"descricaoResumida": "string"
}
}
]

```

Parâmetros

Parâmetro	Descrição	Tipo de parâmetro	Tipo de dados
cnpjSancionado	CNPJ ou CPF do Sancionado	query	string
nomeSancionado	Nome, nome fantasia ou razão social do Sancionado	query	string
orgaoSancionador	Órgão Sancionador	query	string
dataInicialSancao	Data Inicial da Sanção (DD/MM/AAAA)	query	string
dataFinalSancao	Data Final da Sanção (DD/MM/AAAA)	query	string
pagina (required)	Página consultada	query	integer

Mensagens de resposta

Código de status HTTP	Razão
401	Unauthorized
403	Forbidden
404	Not Found

Tabela 1 - Documentação API CEIS

Para obtenção dos dados foi utilizada função “get” da biblioteca “requests” (<https://requests.readthedocs.io/en/master/>) - “requests.get(url, parameters)” - passando a URL da API no primeiro parâmetro e o número da página no segundo parâmetro. Para que não fosse necessário calcular o número total de páginas que seriam retornadas pela API foi implementado um laço que executa a função “get” e

incrementa o número da página enquanto o status HTTP de retorno for igual 200 (<https://developer.mozilla.org/pt-BR/docs/Web/HTTP/Status/200>). A cada iteração os dados eram persistidos no Banco MongoDB e, para tanto, foi utilizada a biblioteca “pymongo”. Através da classe “MongoClient”, da biblioteca “pymongo”, obtém-se o objeto de conexão com o MongoDB local passando como parâmetro a coleção que será manipulada. Para essas regras foi implementada a classe “getApiData.py”. Essa classe importa, além da biblioteca “requests”, já mencionada anteriormente, a classe utils.py, importada com pseudônimo “utl”, onde está implementada a função de conexão com o MongoDB, chamada “connect_mongo”.

```

getApiData.py > ...
1  import requests
2  import utils as utl
3
4  db = 'ceis'
5  col = 'register'
6  url = 'http://www.transparencia.gov.br/api-de-dados/ceis'
7  pg_parameter = 'pagina'
8  offset = 1
9
10 col = utl.connect_mongo(db, col)
11 parameters = {pg_parameter: offset}
12 status = 200
13 while(status == 200):
14     response = requests.get(url, parameters)
15     status = response.status_code
16     print(pg_parameter+' '+str(parameters[pg_parameter])+' => status = '+str(status))
17     if(response.text == '[]' or status != 200):
18         print('response empty')
19         break
20     if(status != 200):
21         print('response status => '+status)
22         break
23     col.insert_many(response.json())
24     parameters[pg_parameter] += 1
25
6
7  def connect_mongo(db, col, host='localhost', port=27017):
8      client = pymongo.MongoClient(host, port)
9      db = client[db]
10     return db[col]
11

```

3. Processamento/Tratamento de Dados

Após a persistência de todo conteúdo obtido através da API do Portal da Transparência no MongoDB, recuperou-se o “dataset” para o início da fase de validação e tratamento dos dados. Para recuperar o “dataset” foi criada a função “getListFromMongoCol”, implementada na classe “utils.py”.

```
eda.py > ...  
1  import utils as utl  
2  
3  data = utl.getListFromMongoCol('ceis', 'register')  
4
```

```
17 def getListFromMongoCol(db, col):  
18     col = connect_mongo(db, col)  
19     data = list(col.find())  
20     return data
```

O primeiro ponto tratado é referente à estrutura dos dados. Conforme pode ser observado na Tabela 1, os dados possuem uma estrutura aninhada por categorias (ex: “pessoa.municipio.uf.sigla”). Como os dados foram persistidos “in natura” o MongoDB preservou essa estrutura. Na recuperação do “dataset” e posterior armazenamento em um “pandas DataFrame”, notou-se a dificuldade em obtenção de “subsets” e consultas agrupadas, porque os dados aninhados eram do tipo “dict” (dicionário), o que impossibilitava utilização de funções do “pandas DataFrame” e, para tanto, seriam necessárias diversas implementações em estruturas de repetição para efetuar as análises. Para contornar esse problema o “dataset” foi “achatado” e todo conteúdo convertido em “pandas DataFrame”. Para isso foi utilizada a biblioteca “flatten_json” (<https://pypi.org/project/flatten-json/>). Foi criada uma função para executar essa instrução, chamada “flattenListAsDF” (implementada na classe utils.py).

```
eda.py > ...  
1  import pandas as pd  
2  import utils as utl  
3  
4  data = utl.getListFromMongoCol('ceis', 'register')  
5  df = utl.flattenListAsDF(data)  
6
```

```

21
22 def flattenListAsDF(data):
23     data_row_flattened = []
24     for data_row in data:
25         data_row_flattened.append(flatten(data_row))
26     return pd.DataFrame(data_row_flattened)
27

```

Após o armazenamento dos dados foram executadas análises preliminares no “dataframe”.

```

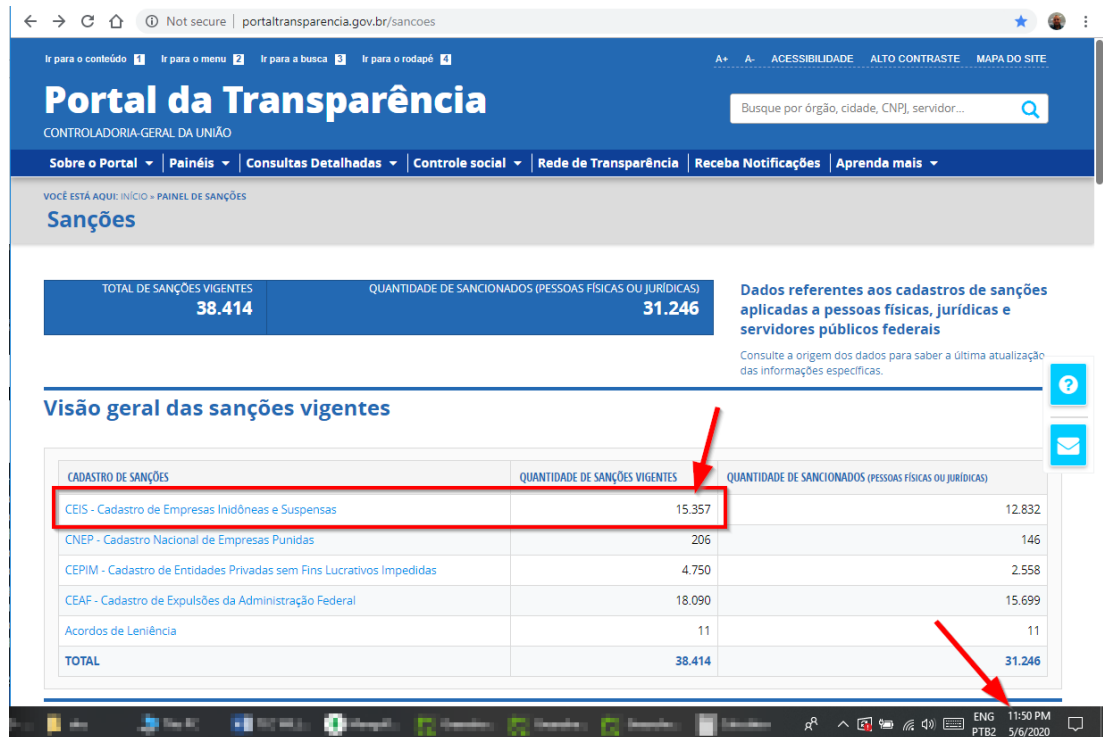
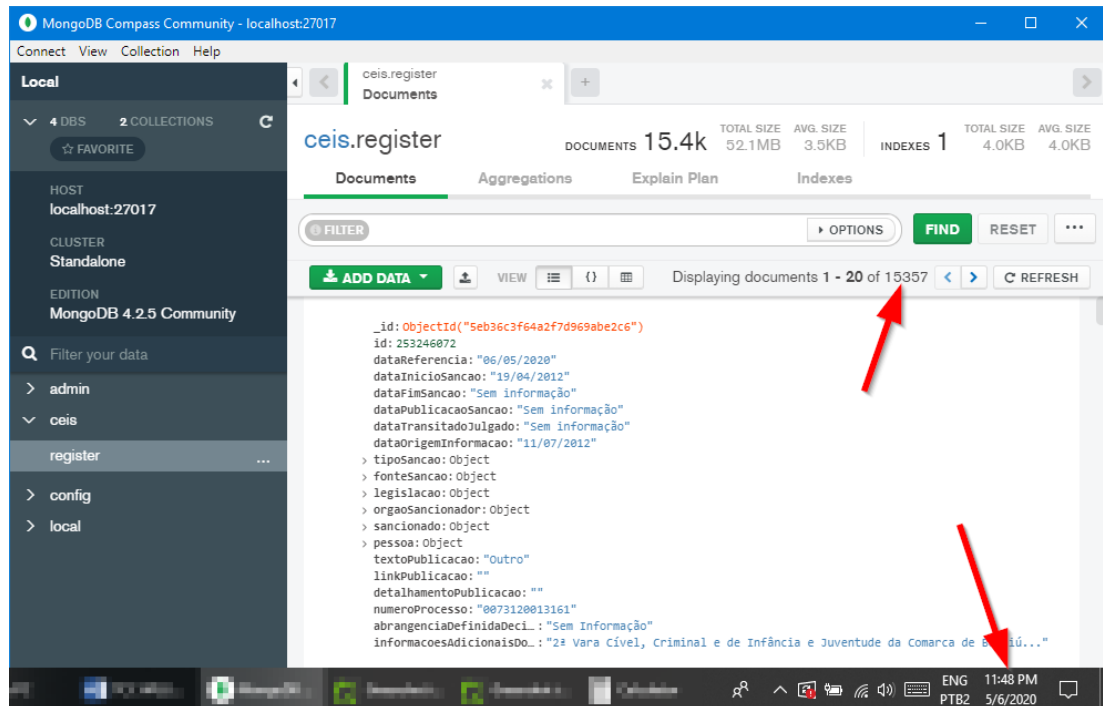
6
7 data = utl.getListFromMongoCol('ceis', 'register')
8 df = utl.flattenListAsDF(data)
9
10 # analises preliminares no dataframe
11 print(df.shape)
12 print(df.head())
13 print(df.columns)
14 print(df.describe(include='all'))
15 print(df.pessoa_tipoCodigo.value_counts())
16 print(df.pessoa_tipoPessoa.value_counts())
17 print(df.pessoa_municipio_uf_sigla.value_counts())
18 print(df.pessoa_cnae_secao.value_counts())
19 print(df.orgaoSancionador_nome.value_counts())
20 print(df.orgaoSancionador_siglaUf.value_counts())
21 print(df.orgaoSancionador_poder.value_counts())
22

```

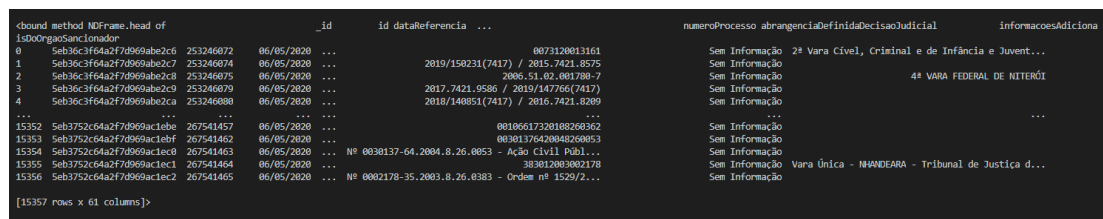
- Propriedade “df.shape”

```
(15357, 61)
```

Informa que o frame tem 15.357 linhas e 61 colunas. A quantidade de linhas corrobora com o quantidade total de linhas informada pelo MongoDB e também com o quantidade total de sanções informadas no Portal da Transparência na data em que as análises foram publicadas (06/05/2020). Isso demonstra que os dados analisados possuem um certo grau de confiabilidade em relação à sua origem.



- Função “df.head()”



Exibe os 5 primeiros e 5 últimos itens do “dataframe”.

- Propriedade “df.columns”

```
Index(['_id', 'id', 'dataReferencia', 'dataInicioSancao', 'dataFimSancao',
      'dataPublicacaoSancao', 'dataTransitadoJulgado', 'dataOrigemInformacao',
      'tipoSancao_descricaoResumida', 'tipoSancao_descricaoPortal',
      'fonteSancao_nomeExibicao', 'fonteSancao_telefoneContato',
      'fonteSancao_enderecoContato', 'legislacao_fundamentacaoLegal',
      'legislacao_descricaoFundamentacaoLegal', 'orgaoSancionador_nome',
      'orgaoSancionador_siglaUf', 'orgaoSancionador_poder', 'sancionado_nome',
      'sancionado_codigoFormatado', 'pessoa_numeroInscricaoSocial',
      'pessoa_nome', 'pessoa_razaoSocialReceita',
      'pessoa_nomeFantasiaReceita', 'pessoa_cnae_codigoSecao',
      'pessoa_cnae_secao', 'pessoa_cnae_codigoSubclasse',
      'pessoa_cnae_subclasse', 'pessoa_cnae_codigoDivisao',
      'pessoa_cnae_divisao', 'pessoa_cnae_codigoGrupo', 'pessoa_cnae_grupo',
      'pessoa_cnae_codigoClasse', 'pessoa_cnae_classe',
      'pessoa_municipio_codigoIBGE', 'pessoa_municipio_nomeIBGE',
      'pessoa_municipio_pais', 'pessoa_municipio_uf_sigla',
      'pessoa_municipio_uf_nome', 'pessoa_localidadePessoa',
      'pessoa_naturezaJuridica_codigo', 'pessoa_naturezaJuridica_descricao',
      'pessoa_naturezaJuridica_codigoTipo',
      'pessoa_naturezaJuridica_descricaoTipo', 'pessoa_dataAbertura',
      'pessoa_enderecoEletronico', 'pessoa_numeroTelefone',
      'pessoa_descricaoLogradouro', 'pessoa_numeroEndereco',
      'pessoa_complementoEndereco', 'pessoa_numeroCEP', 'pessoa_nomeBairro',
      'pessoa_codigoFormatado', 'pessoa_tipoCodigo', 'pessoa_tipoPessoa',
      'textoPublicacao', 'linkPublicacao', 'detalhamentoPublicacao',
      'numeroProcesso', 'abrangenciaDefinidaDecisaoJudicial',
      'informacoesAdicionaisDoOrgaoSancionador'],
      dtype='object')
```

Exibe os nomes das colunas do “dataframe”.

- Função “df.describe(include='all')”

```
count      _id      id dataReferencia dataInicioSancao ... detalhamentoPublicacao  numeroProcesso  abrangenciaDefinidaDecisaoJudicial  informacoesAdicionaisDoOrgaoSancionador
unique      15357  1.535700e+04      15357      15357 ...      15357      15357      15357      15357
top      5eb3741364a2f7d969ac17e4      NaN      06/05/2020      03/10/2017 ...      Não Identificado      Sem Informaçã      12887
freq      1      NaN      15357      69 ...      13988      184      12467
mean      NaN      2.604279e+08      NaN      NaN ...      NaN      NaN      NaN
std      NaN      4.135448e+06      NaN      NaN ...      NaN      NaN      NaN
min      NaN      2.532461e+08      NaN      NaN ...      NaN      NaN      NaN
25%      NaN      2.568363e+08      NaN      NaN ...      NaN      NaN      NaN
50%      NaN      2.605389e+08      NaN      NaN ...      NaN      NaN      NaN
75%      NaN      2.630407e+08      NaN      NaN ...      NaN      NaN      NaN
max      NaN      2.675415e+08      NaN      NaN ...      NaN      NaN      NaN
[11 rows x 61 columns]
```

Exibe alguns dados estatísticos sobre o “dataframe”.

- Função “df.pessoa_tipoCodigo.value_counts()”

```
CPF      8473
CNPJ     6874
      10
Name: pessoa_tipoCodigo, dtype: int64
```

Contagem de valores da série “pessoa_tipoPessoa”. Nota-se uma quantidade irrelevante de “missing value” nessa série (cerca de 0,06% do total). De qualquer forma esses dados serão tratados como valores não numéricos para que não sejam considerados em eventuais análises (https://pandas.pydata.org/pandas-docs/stable/user_guide/missing_data.html).

```
# identificados valores vazios => substituindo por valores nao numericos
df['pessoa_tipoCodigo'].replace([''], np.nan, inplace=True)
print(df.pessoa_tipoCodigo.value_counts())
```



```
CPF      8473
CNPJ     6874
Name: pessoa_tipoCodigo, dtype: int64
```

- Função “df.pessoa_tipoPessoa.value_counts()”

```
Pessoa Física      8476
Entidades Empresariais Privadas  6671
Entidades Sem Fins Lucrativos    191
Sem Informação      11
Administração Pública Municipal    5
Administração Pública Estadual ou do Distrito Federal    2
Administração Pública Federal      1
Name: pessoa_tipoPessoa, dtype: int64
```

Contagem de valores da série “pessoa_tipoPessoa”.

- Função “df.pessoa_municipio_uf_sigla.value_counts()”

```
SP      4513
PR      1159
RS      1114
BA      1097
MG       927
RJ       844
SC       595
GO       518
MA       489
DF       422
RN       367
PB       347
ES       337
PE       337
MT       334
CE       324
RO       284
PA       232
AM       207
MS       185
TO       132
PI       123
SE       117
AL        86
AP        81
AC        72
-1        67
RR         47
Name: pessoa_municipio_uf_sigla, dtype: int64
```

Contagem de valores da série “pessoa_municipio_uf_sigla”. Aqui é possível observar que existem 67 registros onde esse dado está incorreto. Esses dados serão tratados como valores não numéricos para que não sejam considerados em eventuais análises.

```
# identificado valor invalido => substituindo por valor nao numerico
df['pessoa_municipio_uf_sigla'].replace(['-1'], np.nan, inplace=True)
print(df.pessoa_municipio_uf_sigla.value_counts())
```

```

SP      4513
PR      1159
RS      1114
BA      1097
MG      927
RJ      844
SC      595
GO      518
MA      489
DF      422
RN      367
PB      347
PE      337
ES      337
MT      334
CE      324
RO      284
PA      232
AM      207
MS      185
TO      132
PI      123
SE      117
AL      86
AP      81
AC      72
RR      47
Name: pessoa_municipio_uf_sigla, dtype: int64

```

- Função “df.pessoa_cnae_secao.value_counts()”

```

Sem informação      10129
COMÉRCIO; REPARAÇÃO DE VEÍCULOS AUTOMOTORES E MOTOCICLETAS  2250
ATIVIDADES ADMINISTRATIVAS E SERVIÇOS COMPLEMENTARES      1007
ATIVIDADES PROFISSIONAIS, CIENTÍFICAS E TÉCNICAS           530
TRANSPORTE, ARMAZENAGEM E CORREIO                          468
INDÚSTRIAS DE TRANSFORMAÇÃO                                402
INFORMAÇÃO E COMUNICAÇÃO                                    154
ALOJAMENTO E ALIMENTAÇÃO                                    123
SAÚDE HUMANA E SERVIÇOS SOCIAIS                             86
EDUCAÇÃO                                                     43
ARTES, CULTURA, ESPORTE E RECREAÇÃO                       40
ATIVIDADES IMOBILIÁRIAS                                     33
ATIVIDADES FINANCEIRAS, DE SEGUROS E SERVIÇOS RELACIONADOS  28
INDÚSTRIAS EXTRATIVAS                                       23
OUTRAS ATIVIDADES DE SERVIÇOS                               16
ADMINISTRAÇÃO PÚBLICA, DEFESA E SEGURIDADE SOCIAL          12
AGRICULTURA, PECUÁRIA, PRODUÇÃO FLORESTAL, PESCA E AQUICULTURA 10
SERVIÇOS DOMÉSTICOS                                         3
Name: pessoa_cnae_secao, dtype: int64

```

Contagem dos valores da série “pessoa_cnae_secao”.

- Função “df.orgaoSancionador_nome.value_counts()”

```

Governo do Estado da Bahia (BA) 478
Governo do Estado do Rio Grande do Sul 348
PROCURADORIA GERAL DO ESTADO 323
Tribunais de Justiça Estaduais / Tribunal de Justiça do Estado de São Paulo / 2º Grau - TJSP / 1º GRUPO DE CÂMARAS DE DIREITO PÚBLICO 297
Governo do Estado da Bahia 293
...
PREFEITURA MUNICIPAL DE SANTA MARIA - RS 1
Tribunal de Justiça do Estado do Amazonas / 1º Grau - TJAM / URUCARA / VARA DA COMARCA DE URUCARÁ 1
1º Grau - TRF5 / Seção Judiciária de Pernambuco / Subseção Judiciária de Pernambuco / 23ª Vara 1
Ministério Público do Estado de Goiás (MPGO) 1
Tribunal Regional Federal da 5ª Região 1
Name: orgaoSancionador_nome, Length: 2572, dtype: int64

```

Contagem dos valores da série “orgaoSancionador_nome”. Aqui é possível observar que os nomes dos órgãos sancionadores não possuem uma padronização, ou seja, caso essa série venha a ser considerada no roteiro de EDA, será necessário aplicar alguma técnica de classificação do texto por similaridade (ex: “cosine similarity” ou “levenshtein distance”).

- Função “df.orgaoSancionador_siglaUf.value_counts()”

```

SP 4001
DF 1241
RS 1088
BA 1059
MG 959
PR 930
RJ 818
796
SC 563
MA 458
RN 343
PB 333
GO 313
RO 309
PE 287
ES 278
CE 256
PA 208
AM 173
MT 161
MS 159
TO 132
PI 121
SE 117
AL 88
AC 69
AP 47
RR 46
ba 2
rj 2
Name: orgaoSancionador_siglaUf, dtype: int64

```

Contagem dos valores da série “orgaoSancionador_siglaUf”. Aqui identificou-se dois problemas nos dados: 1) algumas siglas estão em letra minúscula (“lowercase”), o que pode interferir nas análises comparativas e agrupamentos; 2) existe um volume considerável de “missing value” nessa série, onde 728 registros, cerca 5% do total, não possuem esse dado. O primeiro problema será tratado transformando todas as siglas minúsculas em maiúsculas (“uppercase”). O segundo

problema será tratado inferindo-se valores não numéricos para que esses registros não sejam considerados em eventuais análises.

```
# identificados estados com siglas minusculas => padronizando todas para maiusculas
df['orgaoSancionador_siglaUf'] = df['orgaoSancionador_siglaUf'].str.upper()
# identificados estados vazios => substituindo para valores nao numericos
df['orgaoSancionador_siglaUf'].replace([''], np.nan, inplace=True)
print(df.orgaoSancionador_siglaUf.value_counts())
```

```
SP      4001
DF      1241
RS      1088
BA      1061
MG       959
PR       930
RJ       820
SC       563
MA       458
RN       343
PB       333
GO       313
RO       309
PE       287
ES       278
CE       256
PA       208
AM       173
MT       161
MS       159
TO       132
PI       121
SE       117
AL        88
AC        69
AP         47
RR         46
Name: orgaoSancionador_siglaUf, dtype: int64
```

- Função “df.orgaoSancionador_poder.value_counts()”

```
Judiciário      9336
Executivo       5521
Legislativo      386
Tribunal de Contas    71
Ministério Público   28
Entidade Paraestatal  15
Name: orgaoSancionador_poder, dtype: int64
```

Contagem dos valores da série “orgaoSancionador_poder”.

Posteriormente às análises preliminares também foi identificada a necessidade de transformação das séries “dataInicioSancao” e “dataFimSancao”, de “string” para “dateTime”, pois serão alvo do roteiro de análise (https://pandas.pydata.org/pandas-docs/stable/reference/api/pandas.to_datetime.html).

```
# convertendo series dataInicioSancao de string para dateTime
dataInicioSancao = pd.to_datetime(df['dataInicioSancao'])
df['dataInicioSancao'] = dataInicioSancao

# convertendo serie dataFimSancao de string para dateTime utilizando "coerce"
dataFimSancao = pd.to_datetime(df['dataFimSancao'], errors='coerce')
df['dataFimSancao'] = dataFimSancao
```

4. Análise e Exploração dos Dados

Com o intuito de obter insights dos dados algumas séries (“pandas.Series”) foram eleitas para serem alvo da EDA (“Exploratory Data Analysis”). As séries “dataInicioSancao” e “dataFimSancao”, primordialmente, trazem informações da distribuição das sanções ao longo do tempo e seus respectivos prazos de duração. A série “pessoa_tipoCodigo” informa a quantidade total de sanções entre pessoas físicas e jurídicas. A série “pessoa_tipoPessoa” pode informar qual a categorização das entidades que mais incorrem nas sanções. A série “pessoa_municipio_uf_sigla” pode informar em quais estados estão os maiores números de entidades sancionadas. A série “pessoa_cnae_secao” pode informar quais os tipos de atividade mais comuns das pessoas jurídicas sancionadas. A série “orgaoSancionador_nome” pode informar quais os órgãos aplicam mais sanções. A série “orgaoSancionador_siglaUf” pode informar em quais estados os órgãos aplicam mais sanções. A série “orgaoSancionador_poder” pode informar quais são os poderes que mais aplicam sanções. Outras séries podem ser incluídas no roteiro das análises conforme os insights obtidos.

- Série “pessoa_tipoCodigo”

```
#inicio do eda

# distribuicao PF/PJ
print(df.pessoa_tipoCodigo.value_counts(normalize=True)*100)
```

```
CPF      55.209487
CNPJ     44.790513
Name: pessoa_tipoCodigo, dtype: float64
```

Distribuição de sanções em função dos valores da série “pessoa_tipoPessoa”. Nota-se que a distribuição é balanceada entre pessoas físicas e jurídicas.

- Série “pessoa_tipoPessoa”

```
# distribuicao categoria pessoa
print(df.pessoa_tipoPessoa.value_counts(normalize=True)*100)
```

```
Pessoa Física                55.193072
Entidades Empresariais Privadas  43.439474
Entidades Sem Fins Lucrativos    1.243732
Sem Informação                0.071629
Administração Pública Municipal  0.032558
Administração Pública Estadual ou do Distrito Federal  0.013023
Administração Pública Federal    0.006512
Name: pessoa_tipoPessoa, dtype: float64
```

Aqui é possível verificar que o universo de análise é preenchido, quase que em sua totalidade, por Pessoas Físicas e Entidades Empresariais Privadas. Também observa-se que a quantidade de pessoas físicas é praticamente igual ao valor obtido na série anterior, o que leva a crer que, praticamente, todo valor “CPF”, da série “pessoa_tipoCodigo”, deve estar relacionado ao valor “Pessoa Física”, da série “pessoa_tipoPessoa”. Para confirmar essa hipótese essas séries foram pivotadas, utilizando a função “pivot_table”, do Pandas:

```
# pivot para verificar a relacao entre pessoa_tipoCodigo e pessoa_tipoPessoa
pivot = pd.pivot_table(df, index=["pessoa_tipoCodigo", "pessoa_tipoPessoa"], values=["id"], aggfunc={"id": len})
result = pivot.sort_values('id', ascending=False)
print(result)
```

```

pessoa_tipoCodigo pessoa_tipoPessoa      id
CPF               Pessoa Física      8473
CNPJ              Entidades Empresariais Privadas  6671
                  Entidades Sem Fins Lucrativos    191
                  Administração Pública Municipal     5
                  Pessoa Física                   3
                  Administração Pública Estadual ou do Distrito F...  2
                  Administração Pública Federal       1
                  Sem Informação                   1
```

Com a pivotação dos dados ficou claro que todo CPF está, de fato, relacionado à Pessoa Física, e que as Entidades Empresarias Privadas representam, praticamente, a totalidade dos códigos CNPJ. A diferença entre a porcentagem de CPF e Pessoas Físicas ($55.209487 - 55.193072 = 0.016415$) pode ser devido à regras de arredondamento e/ou aproximação.

- Série “pessoa_municipio_uf_sigla”

```
# distribuicao sancoes por estado
print(df.pessoa_municipio_uf_sigla.value_counts(normalize=True)*100)
```

```
SP      29.516024
PR       7.580118
RS       7.285808
BA       7.174624
MG       6.062786
RJ       5.519948
SC       3.891432
GO       3.387835
MA       3.198169
DF       2.759974
RN       2.400262
PB       2.269457
PE       2.204055
ES       2.204055
MT       2.184434
CE       2.119032
RO       1.857423
PA       1.517332
AM       1.353826
MS       1.209941
TO       0.863309
PI       0.804447
SE       0.765206
AL       0.562459
AP       0.529758
AC       0.470896
RR       0.307390
Name: pessoa_municipio_uf_sigla, dtype: float64
```

Aqui é possível observar que, somente o estado de São Paulo, concentra cerca de 29% do total de sanções aplicadas. Também observa-se que o “top 5” concentra cerca de 57% do total de sanções, o que é um valor muito expressivo.

- Série “pessoa_cnae_secao”

```
# distribuicao sancoes por cnae
print(df.pessoa_cnae_secao.value_counts(normalize=True)*100)
```

```

Sem informação                65.956893
COMÉRCIO; REPARAÇÃO DE VEÍCULOS AUTOMOTORES E MOTOCICLETAS  14.651299
ATIVIDADES ADMINISTRATIVAS E SERVIÇOS COMPLEMENTARES        6.557270
ATIVIDADES PROFISSIONAIS, CIENTÍFICAS E TÉCNICAS             3.451195
TRANSPORTE, ARMAZENAGEM E CORREIO                             3.047470
INDÚSTRIAS DE TRANSFORMAÇÃO                                   2.617699
INFORMAÇÃO E COMUNICAÇÃO                                       1.002800
ALOJAMENTO E ALIMENTAÇÃO                                       0.800938
SAÚDE HUMANA E SERVIÇOS SOCIAIS                                0.560005
EDUCAÇÃO                                                        0.280003
ARTES, CULTURA, ESPORTE E RECREAÇÃO                          0.260468
ATIVIDADES IMOBILIÁRIAS                                       0.214886
ATIVIDADES FINANCEIRAS, DE SEGUROS E SERVIÇOS RELACIONADOS   0.182327
INDÚSTRIAS EXTRATIVAS                                         0.149769
OUTRAS ATIVIDADES DE SERVIÇOS                                  0.104187
ADMINISTRAÇÃO PÚBLICA, DEFESA E SEGURIDADE SOCIAL            0.078140
AGRICULTURA, PECUÁRIA, PRODUÇÃO FLORESTAL, PESCA E AQUICULTURA 0.065117
SERVIÇOS DOMÉSTICOS                                           0.019535
Name: pessoa_cnae_secao, dtype: float64

```

Aqui é possível verificar que, boa parte dos registros não possuem essa informação (quase 66%). Provavelmente isso pode estar relacionado ao fato de que cerca de 55% dos registros são de pessoas físicas, que, realmente, não possuem esse atributo. Mesmo assim, além dos 55% de pessoas físicas, cerca de 11% dos registros também não possuem esse atributo. Aqui, mais uma vez, foi utilizada a função “pivot_table”, do pandas, para verificar a relação entre tipo de pessoa (física ou jurídica) e o cadastro nacional de atividades econômicas (CNAE) e entender onde estão localizados os registros de CNAE com valor “Sem informação”.

```

pessoa_tipoCodigo pessoa_cnae_secao    id
CPF               Sem informação        8473
CNPJ              COMÉRCIO; REPARAÇÃO DE VEÍCULOS AUTOMOTORES E M... 2250
                  Sem informação        1646
                  ATIVIDADES ADMINISTRATIVAS E SERVIÇOS COMPLEMEN... 1007
                  ATIVIDADES PROFISSIONAIS, CIENTÍFICAS E TÉCNICAS   530
                  TRANSPORTE, ARMAZENAGEM E CORREIO                   468
                  INDÚSTRIAS DE TRANSFORMAÇÃO                         402
                  INFORMAÇÃO E COMUNICAÇÃO                            154
                  ALOJAMENTO E ALIMENTAÇÃO                            123
                  SAÚDE HUMANA E SERVIÇOS SOCIAIS                     86
                  EDUCAÇÃO                                              43
                  ARTES, CULTURA, ESPORTE E RECREAÇÃO                 40
                  ATIVIDADES IMOBILIÁRIAS                             33
                  ATIVIDADES FINANCEIRAS, DE SEGUROS E SERVIÇOS R... 28
                  INDÚSTRIAS EXTRATIVAS                               23
                  OUTRAS ATIVIDADES DE SERVIÇOS                        16
                  ADMINISTRAÇÃO PÚBLICA, DEFESA E SEGURIDADE SOCIAL   12
                  AGRICULTURA, PECUÁRIA, PRODUÇÃO FLORESTAL, PESCA... 10
                  SERVIÇOS DOMÉSTICOS                                  3

```

Ratificando o que já havia sido observado anteriormente, é possível verificar que, no escopo de pessoas jurídicas (“pessoa_tipoCodigo”), também existe um alto

volume de entidades sem CNAE (cerca de 24%), o que pode diminuir a assertividade quando os dados são analisados por esse prisma.

- Série “orgaoSancionador_siglaUf”

```
# distribuicao sancoes por estado do orgao sancionador
print(df.orgaoSancionador_siglaUf.value_counts(normalize=True)*100)
```

```
SP      27.477508
DF       8.522766
RS       7.472014
BA       7.286587
MG       6.586086
PR       6.386924
RJ       5.631481
SC       3.866493
MA       3.145388
RN       2.355607
PB       2.286931
GO       2.149578
RO       2.122107
PE       1.971018
ES       1.909210
CE       1.758121
PA       1.428473
AM       1.188105
MT       1.105693
MS       1.091958
TO       0.906531
PI       0.830987
SE       0.803516
AL       0.604354
AC       0.473869
AP       0.322780
RR       0.315912
Name: orgaoSancionador_siglaUf, dtype: float64
```

Como já vinha sendo visto nas análises anteriores aqui também o estado de São Paulo lidera a quantidade órgãos sancionadores (órgão responsáveis pelas aplicações das sanções), o que, aparentemente, demonstra a existência de correlação entre o domicílio da entidade sancionada e o domicílio do órgão sancionador. Para confirmar essa possibilidade mais uma vez foi utilizada a pivotação afim de verificar a relação entre estado do órgão sancionador e estado da entidade sancionada.

```
<bound method NDFrame.head of                                     id
orgaoSancionador_siglaUf pessoa_municipio_uf_sigla
AC                        AC                        54
                        AM                        1
                        AP                        3
                        BA                        1
                        DF                        1
...                      ..
TO                        GO                        8
                        ES                        2
                        DF                        5
                        MA                        2
                        TO                        99

[383 rows x 1 columns]>
```

Aqui foi utilizada a função “df.head()” para que fosse analisada, visualmente, apenas uma amostragem do “dataframe”. De fato observa-se que órgãos sancionadores aplicam, predominantemente, sanções à entidades do mesmo estado.

- Série “orgaoSancionador_poder”

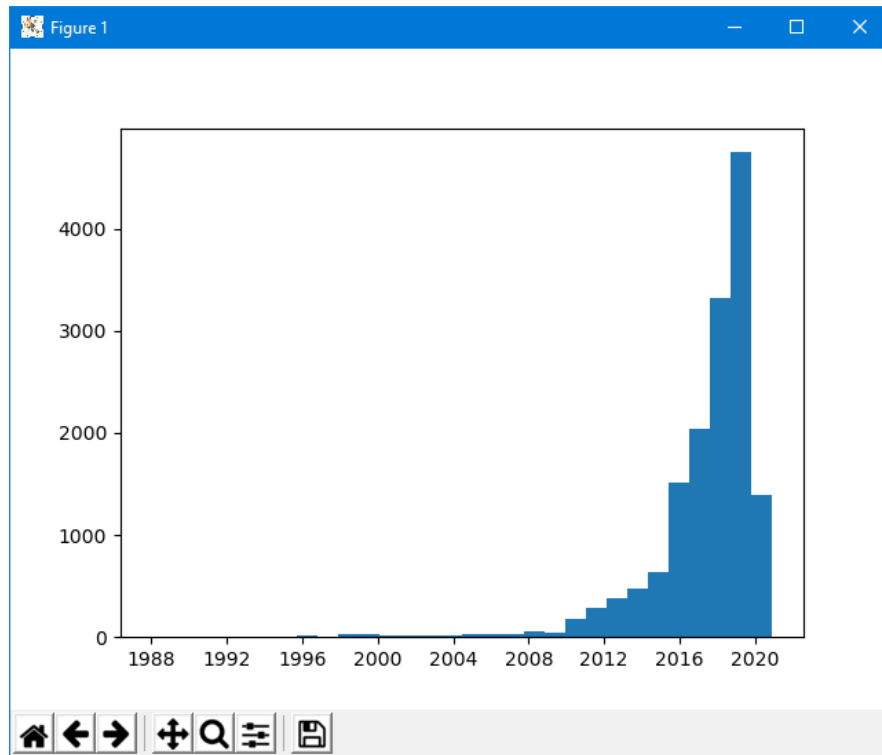
```
# distribuicao sancoes pela esfera de poder do orgao sancionador
print(df.orgaoSancionador_poder.value_counts(normalize=True)*100)
```

```
Judiciário          60.793124
Executivo           35.951032
Legislativo          2.513512
Tribunal de Contas  0.462330
Ministério Público  0.182327
Entidade Paraestatal 0.097675
Name: orgaoSancionador_poder, dtype: float64
```

Aqui é possível notar que os poderes judiciários e executivos, juntamente, são predominantes na aplicação das sanções, representando quase 97% do total de sanções aplicadas.

- Série “dataInicioSancao”

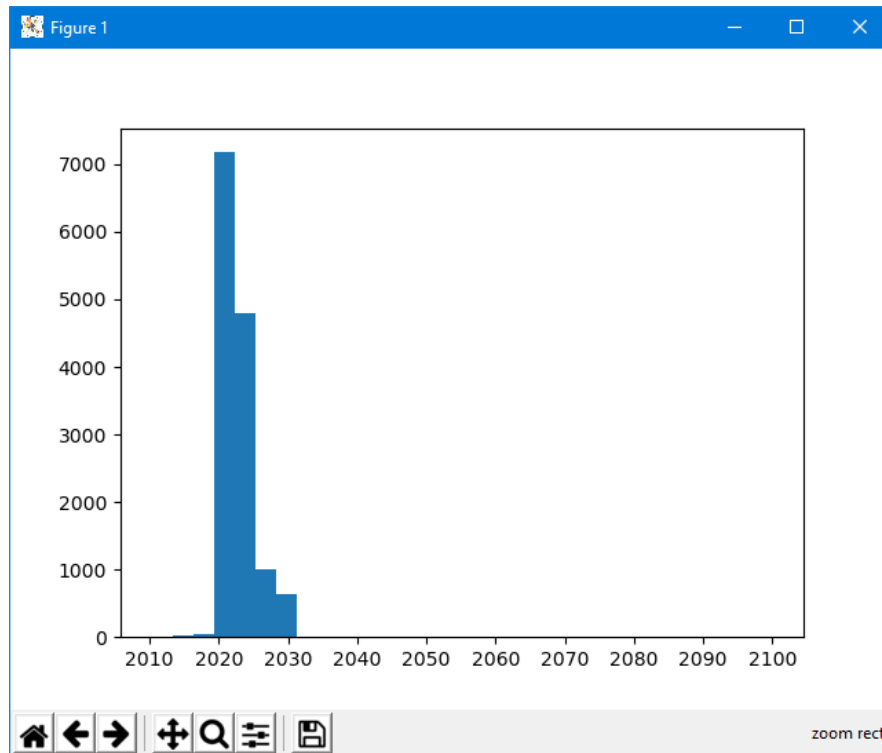
```
# verificando a distribuicao de sancoes por "dataInicioSancao"
plt.hist(df['dataInicioSancao'].dropna(), bins=30)
plt.show()
```



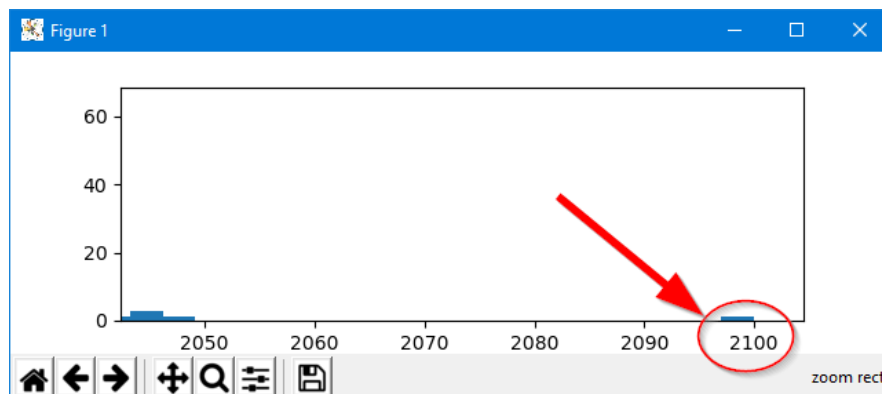
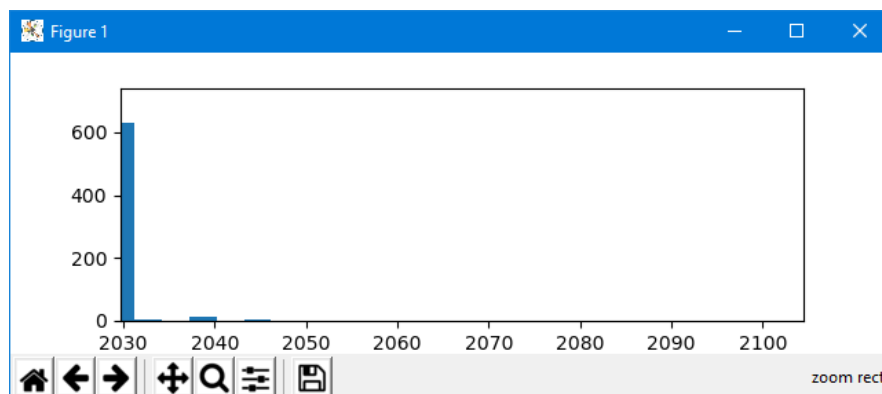
Na análise da distribuição da série “dataInicioSancao” foi utilizado um histograma (https://matplotlib.org/3.2.1/api/as_gen/matplotlib.pyplot.hist.html). Aqui pode-se notar que, de 1988 até, aproximadamente, 2009, a quantidade de sanções aplicadas é inexpressiva. A partir de 2010 inicia-se uma crescente vertiginosa.

- Série “dataFimSancao”

```
# verificando a distribuicao de sancoes por "dataFimSancao"
plt.hist(df['dataFimSancao'].dropna(), bins=30)
plt.show()
```




Aqui pode-se observar que os prazos das sanções concentram-se, predominantemente, entre 2020 e 2030. Também nota-se que o eixo x do gráfico foi “plotado” até 2100, o que indica que existem valores nesses intervalos.



Com o aumento do “zoom” no histograma foi possível verificar que, de fato, existe algum valor (ou alguns valores) com características de “outlier”.

```
# no grafico anterior foram identificados valores extremos na serie dataFimSancao
# filtrando apenas datas posteriores a 2030 na serie dataFimSancao para analise
print(dataFimSancao[df['dataFimSancao'] > '31/12/2030'].value_counts().sort_index())
```

```
2032-06-09      1
2032-07-02      2
2032-09-18      1
2032-09-23      1
2039-10-07     14
2040-09-24      1
2044-01-30      1
2044-09-24      2
2048-12-05      1
2099-12-31      1
Name: dataFimSancao, dtype: int64
```

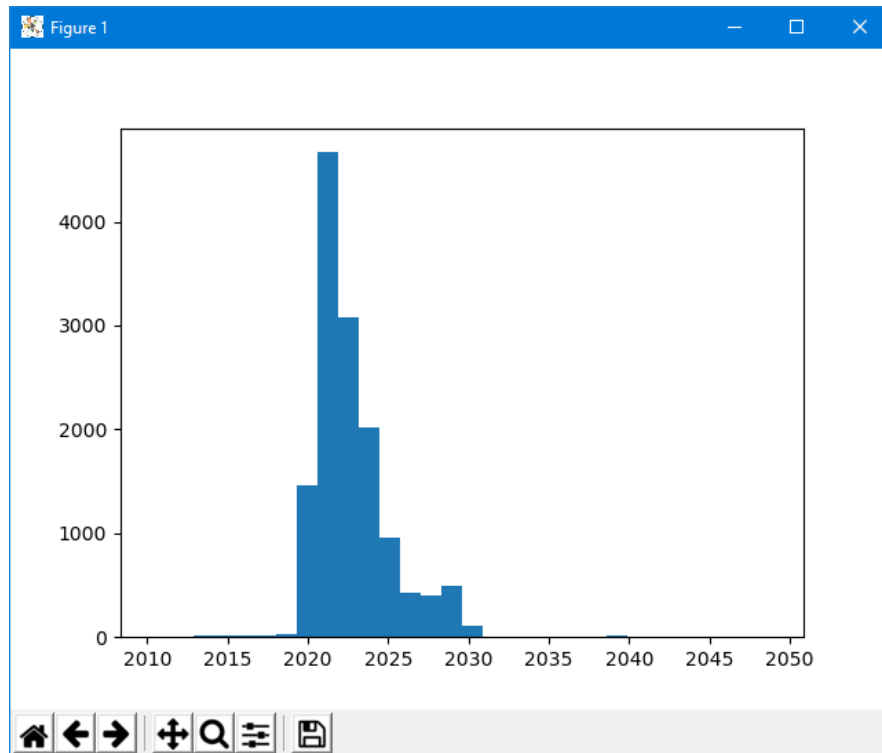


Aqui foi possível identificar o valor “outlier”. Embora seja apenas um registro foi atribuído um valor “not a time” (NaT) para que o mesmo não interfira nas análises posteriores.

```
# atribuindo valor NaT para data outlier
df.loc[df['dataFimSancao'] == '31/12/2099', 'dataFimSancao'] = pd.Timedelta('nat')
# atribuindo a correcao a variavel dataFimSancao
dataFimSancao = df['dataFimSancao']
# vizualizando datas acima de 2030 para verificar a exclusao da data outlier
print(dataFimSancao[df['dataFimSancao'] > '31/12/2030'].value_counts().sort_index())
```

```
2032-06-09      1
2032-07-02      2
2032-09-18      1
2032-09-23      1
2039-10-07     14
2040-09-24      1
2044-01-30      1
2044-09-24      2
2048-12-05      1
Name: dataFimSancao, dtype: int64
```

Após essa correção o histograma foi gerado novamente.



É possível verificar que, após a correção do valor “outlier”, a escala do eixo x diminuiu.

As análises prosseguiram concentrada nas datas, de onde poderiam ser obtidos mais “insights”. Uma das análises propostas é sobre os prazos das sanções.

```
# obtendo os prazos das sancoes para analises
prazoSancao = dataFimSancao - dataInicioSancao
# describe para analisar os dados ref prazo das sancoes
print(prazoSancao.describe(include='all'))
```

```
count          13724
mean    1786 days 06:05:14.777032
std      1076 days 02:47:41.264456
min          -156 days +00:00:00
25%         1096 days 00:00:00
50%         1826 days 00:00:00
75%         1827 days 00:00:00
max         11894 days 00:00:00
dtype: object
```

Através da função “describe” da série “prazoSancao” – resultado em dias da subtração entre as séries “dataFimSancao” e “dataInicioSancao” – observou-se que o mínimo (min) apresentou valor negativo, o que não faz sentido, pois não existe prazo negativo. Considerando que essa informação é inconsistente foi necessário corrigi-la no “dataframe”.

```
# no describe acima foi identificado que o min e negativo, o que nao faz sentido
# essas datas serao removidas do dataframe
# filtrando os prazos negativos (onde dataFimSancao < dataInicioSancao)
dataFimSancao_negative_boolean = dataFimSancao < dataInicioSancao
# substituindo essas datas por NaT no dataframe
df.at[dataFimSancao_negative_boolean, 'dataFimSancao'] = pd.Timedelta('nat')
# atribuindo os valores atualizados a serie dataFimSancao
dataFimSancao = df['dataFimSancao']
# recalculando o prazo
prazoSancao = dataFimSancao - dataInicioSancao
# visualizando o describe para validar a correcao
print(prazoSancao.describe(include='all'))
```

```
count          13681
mean    1791 days 23:40:06.403040
std      1072 days 22:16:35.928344
min          0 days 00:00:00
25%        1096 days 00:00:00
50%        1826 days 00:00:00
75%        1827 days 00:00:00
max        11894 days 00:00:00
dtype: object
```

Dando continuidade na exploração após a correção obteve-se o prazo médio das sanções em meses.

```
# obtendo o valor medio em meses
prazoSancao_mean = prazoSancao.describe().loc['mean']
prazoSancao_meses = prazoSancao_mean / np.timedelta64(1, 'M')
print(round(prazoSancao_meses))
```

59

O valor obtido é de 59 meses, ou seja, o prazo médio de uma sanção é de, praticamente, 5 anos, em média. Porém, na função “describe” anterior, foi possível observar que o valor do desvio padrão é muito alto, pois o coeficiente de variação é próximo a 60%. Considerando isso é interessante explorar mais esses dados afim de obter algum “insight”. A primeira análise foi sobre as estatísticas dos prazos em função das sanções. Para isso utilizou-se um gráfico de barras agrupados. Para essa implementação, primeiramente, obteve-se os valores únicos das descrições resumidas de cada sanção e armazenou-os em um “array”. Esse “array” foi iterado obtendo-se as respectivas descrições resumidas, prazo médio e desvio padrão de cada sanção.

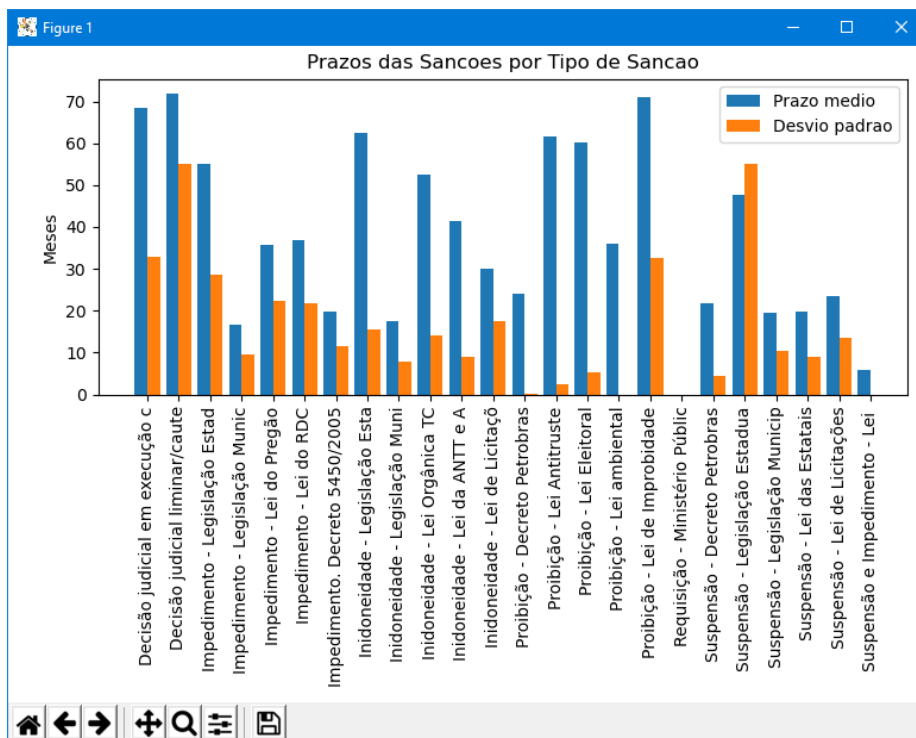
```
# analisando prazos das sancoes por tipo de sancao
df_tipoSancao = df['tipoSancao_descricaoResumida'].unique()
df_tipoSancao = list(df_tipoSancao)
df_tipoSancao = np.sort(df_tipoSancao)

labels = []
prazo_mean = []
prazo_std = []
for tipoSancao in df_tipoSancao:
    df_tipoSancao_it = df[df['tipoSancao_descricaoResumida'] == tipoSancao]
    prazo_tipoSancao = (df_tipoSancao_it["dataFimSancao"] - df_tipoSancao_it["dataInicioSancao"])
    labels.append(tipoSancao[:30])
    prazo_mean.append(prazo_tipoSancao.describe().loc['mean'] / np.timedelta64(1, 'M'))
    prazo_std.append(prazo_tipoSancao.describe().loc['std'] / np.timedelta64(1, 'M'))

utils.groupedBarWithLabels(prazo_mean, prazo_std, labels,
    'Prazo medio', 'Desvio padrao', 'Meses', 'Prazos das Sancoes por Tipo de Sancao')
```

A função de impressão do gráfico (“groupedBarWithLabels”) foi implementada na classe “utils.py”.

```
def groupedBarWithLabels(bar1, bar2, labels,
    labelBar1='labelBar1', labelBar2='labelBar2', ylabel='ylabel', title='title'):
    x = np.arange(len(labels)) # the label locations
    width = 0.4 # the width of the bars
    fig, ax = plt.subplots()
    ax.bar(x - width/2, bar1, width, label=labelBar1)
    ax.bar(x + width/2, bar2, width, label=labelBar2)
    fig.tight_layout()
    ax.set_ylabel(ylabel)
    ax.set_title(title)
    ax.set_xticks(x)
    ax.set_xticklabels(labels, rotation=90)
    ax.legend()
    plt.show()
```



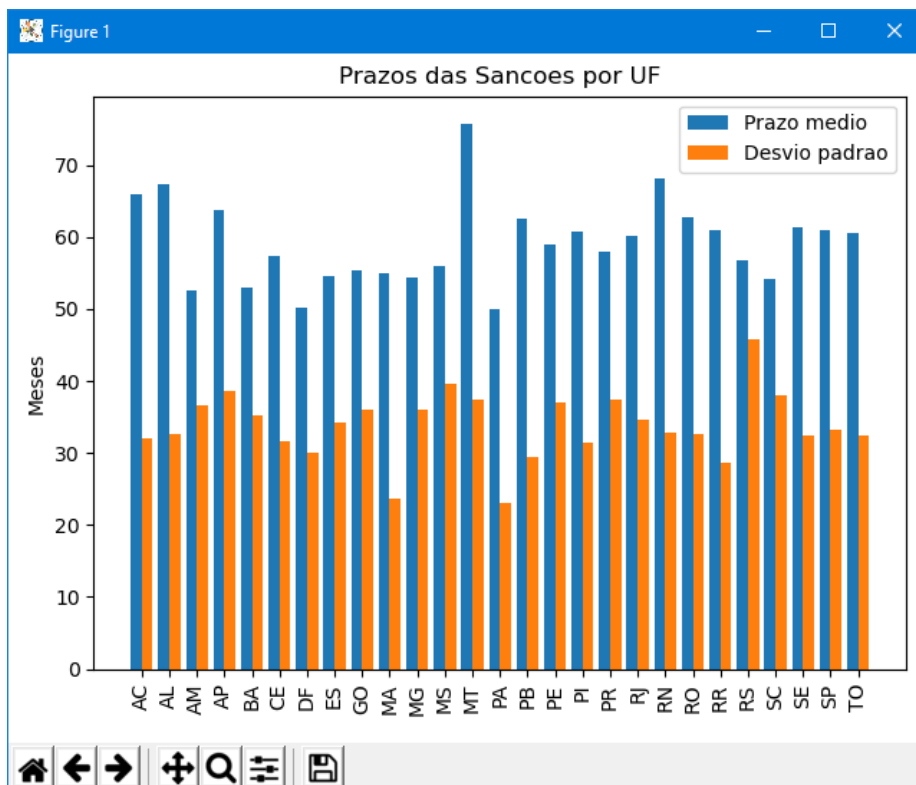
A segunda análise foi sobre as estatísticas dos prazos das sanções em função dos estados das entidades sancionadas. Para isso também utilizou-se um gráfico de

barras agrupados. Assim como anteriormente, obteve-se os valores únicos dos estados e armazenando-os em um “array”. Esse “array” foi iterado para se obter os estados e respectivos prazo médio e desvio padrão de cada sanção por estado.

```
# analisando prazos das sancoes por estado
df_uf = df['pessoa_municipio_uf_sigla'].unique()
df_uf = list(df_uf)
df_uf.remove(np.nan)
df_uf = np.sort(df_uf)

labels = []
prazo_mean = []
prazo_std = []
prazoSancao_uf = []
for uf in df_uf:
    df_uf_it = df[df['pessoa_municipio_uf_sigla'] == uf]
    prazo_uf = (df_uf_it["dataFimSancao"] - df_uf_it["dataInicioSancao"])
    prazoSancao_uf.append(prazo_uf.describe())
    labels.append(uf)
    prazo_mean.append(prazo_uf.describe().loc['mean'] / np.timedelta64(1, 'M'))
    prazo_std.append(prazo_uf.describe().loc['std'] / np.timedelta64(1, 'M'))

utl.groupedBarWithLabels(prazo_mean, prazo_std, labels,
    'Prazo medio', 'Desvio padrao', 'Meses', 'Prazos das Sancoes por UF')
```



Após essas análises foram realizadas análises exploratórias pela perspectiva das entidades sancionadas. A primeira análise verificou o índice de reincidência de das entidades, ou seja, se uma mesma entidade já sofreu mais de uma sanção.

```
# efetuando analises nas entidades sancionadas
# verificando o indice de reincidencia de sancoes por entidades
pessoaCodigo = df['pessoa_codigoFormatado']
pessoaCodigo_value_counts = pessoaCodigo.value_counts()
print(pessoaCodigo_value_counts[pessoaCodigo_value_counts > 2].sum() / pessoaCodigo_value_counts.sum())
```

```
0.16357361463827572
```

Pode-se verificar que o índice de reincidência é de cerca de 16,3 %, uma taxa razoável que vale a pena ser analisada. O “dataframe” foi filtrado para obter apenas pessoas reincidentes, para focar a análise nesse escopo.

```
# filtrando o dataframe para obter apenas pessoas com quantidade de sancoes > 2
pessoaReincidente = pessoaCodigo_value_counts[pessoaCodigo_value_counts > 2]
df_pessoaReincidente = df[df.pessoa_codigoFormatado.isin(pessoaReincidente.index)]
print(df_pessoaReincidente.describe(include='all'))
```

count	2512	2.512000e+03	2512	2512	...	2512	2512	2512	2512	2512
unique	2512	NaN	1	1018	...	124	2027	5	247	247
top	5eb3732a64a2f7d969ac11cc	NaN	06/05/2020	2019-08-27 00:00:00	Não Identificado	Sem Informação
freq	1	NaN	2512	28	...	2162	41	2052	2004	2004
first	NaN	NaN	NaN	1988-01-09 00:00:00	...	NaN	NaN	NaN	NaN	NaN
last	NaN	NaN	NaN	2020-12-02 00:00:00	...	NaN	NaN	NaN	NaN	NaN
mean	NaN	2.603636e+08	NaN	NaN	...	NaN	NaN	NaN	NaN	NaN
std	NaN	4.877080e+06	NaN	NaN	...	NaN	NaN	NaN	NaN	NaN
min	NaN	2.532461e+08	NaN	NaN	...	NaN	NaN	NaN	NaN	NaN
25%	NaN	2.567405e+08	NaN	NaN	...	NaN	NaN	NaN	NaN	NaN
50%	NaN	2.604481e+08	NaN	NaN	...	NaN	NaN	NaN	NaN	NaN
75%	NaN	2.635495e+08	NaN	NaN	...	NaN	NaN	NaN	NaN	NaN
max	NaN	2.675414e+08	NaN	NaN	...	NaN	NaN	NaN	NaN	NaN

[13 rows x 61 columns]

Verificou-se a distribuição de reincidentes entre pessoa física e jurídica

```
# verificando os tipos de pessoa (PF ou PJ?)
print(df_pessoaReincidente.pessoa_tipoCodigo.value_counts(normalize=True)*100)
```

```
CPF      54.956035
CNPJ     45.043965
Name: pessoa_tipoCodigo, dtype: float64
```

Nota-se que a distribuição de reincidentes é muito parecida com a distribuição geral entre pessoas físicas e jurídicas.

Também foi analisado qual é o estado de origem das entidades reincidentes.

```
# verificando onde se encontram as entidades com maior indice de reincidencia
print(df_pessoaReincidente.pessoa_municipio_uf_sigla.value_counts(normalize=True)*100)
```

```

SP      36.530775
MA       7.434053
RS       5.995204
PR       5.595524
MT       5.275779
RJ       5.195843
MG       4.756195
BA       3.677058
ES       3.477218
SC       2.917666
DF       2.757794
RN       2.557954
PA       2.078337
GO       1.878497
MS       1.718625
PB       1.718625
PE       1.558753
AM       1.438849
TO       0.679456
PI       0.559552
CE       0.559552
AC       0.479616
AP       0.439648
RO       0.359712
SE       0.359712
Name: pessoa_municipio_uf_sigla, dtype: float64

```

Aqui é possível verificar que o estado de São Paulo extrapolou, que lidera o ranking de sanções, com cerca de 29% do total de sanções aplicadas, supera esse índice no quesito reincidência, sendo responsável por um pouco mais de 36% do quantidade total de reincidências

Outro dado verificado é a distribuição de sanções em função das entidades reincidentes.

```

# verificando a distribuicao de sancoes das entidades reincidentes
print(df_pessoaReincidente.tipoSancao_descricaoResumida.value_counts(normalize=True)*100)

```

```

Proibição - Lei de Improbidade                57.563694
Impedimento - Lei do Pregão                    17.237261
Suspensão - Lei de Licitações                  9.394904
Inidoneidade - Lei de Licitações               4.219745
Inidoneidade - Legislação Estadual              2.547771
Decisão judicial liminar/cautelar que impeça contratação  2.070064
Suspensão - Lei das Estatais                   2.030255
Inidoneidade - Lei da ANTT e ANTAQ             1.671975
Suspensão - Legislação Estadual                0.915605
Inidoneidade - Lei Orgânica TCU                0.915605
Impedimento - Legislação Municipal             0.398089
Impedimento. Decreto 5450/2005                 0.358280
Decisão judicial em execução cível que impeça a contratação  0.238854
Impedimento - Lei do RDC                      0.199045
Impedimento - Legislação Estadual              0.199045
Suspensão - Decreto Petrobras                  0.039809
Name: tipoSancao_descricaoResumida, dtype: float64

```

Outro dado importante que foi analisado é o tempo de vida das empresas reincidentes. Obviamente que, nesse caso, esse atributo é exclusivo de pessoas jurídicas.

```

# tempo de vida das empresas reincidentes
df_pessoaReincidente_pj = df_pessoaReincidente[df_pessoaReincidente.pessoa_tipoCodigo == 'CNPJ']
pessoaReincidentePJ_dataAbertura = pd.to_datetime(df_pessoaReincidente_pj['pessoa_dataAbertura'], errors='coerce')
pessoaReincidentePJ_tempoVida = pd.Timestamp.today() - pessoaReincidentePJ_dataAbertura
print(pessoaReincidentePJ_tempoVida.describe())

```

```

count                1127
mean      5940 days 21:43:51.110345
std       3824 days 12:10:05.388946
min        552 days 05:12:20.072191
25%       2953 days 05:12:20.072191
50%       5063 days 05:12:20.072190
75%       8480 days 05:12:20.072190
max       19787 days 05:12:20.072191
Name: pessoa_dataAbertura, dtype: object

```

Pode-se observar que as empresas reincidente possuem, em média, cerca de 16 anos de vida mas, como em outros dados que já foram analisados, aqui também o coeficiente de variação é alto (cerca de 64%), o que faz com o que a média não seja um parâmetro de comparação assertivo. De qualquer forma é interessante que essas

informações sejam comparadas com as estatísticas das empresas não reincidentes, para que seja verificado se o padrão é o mesmo ou diferente.

```
# comparando com o tempo de vida das empresas nao reincidentes
pessoaNaoReincidente = pessoaCodigo_value_counts[pessoaCodigo_value_counts == 1]
df_pessoaNaoReincidente = df[df.pessoa_codigoFormatado.isin(pessoaNaoReincidente.index)]
df_pessoaNaoReincidente_pj = df_pessoaNaoReincidente[df_pessoaNaoReincidente.pessoa_tipoCodigo == 'CNPJ']
pessoaNaoReincidentePJ_dataAbertura = pd.to_datetime(df_pessoaNaoReincidente_pj['pessoa_dataAbertura'], errors='coerce')
pessoaNaoReincidentePJ_tempoVida = pd.Timestamp.today() - pessoaNaoReincidentePJ_dataAbertura
print(pessoaNaoReincidentePJ_tempoVida.describe())
```

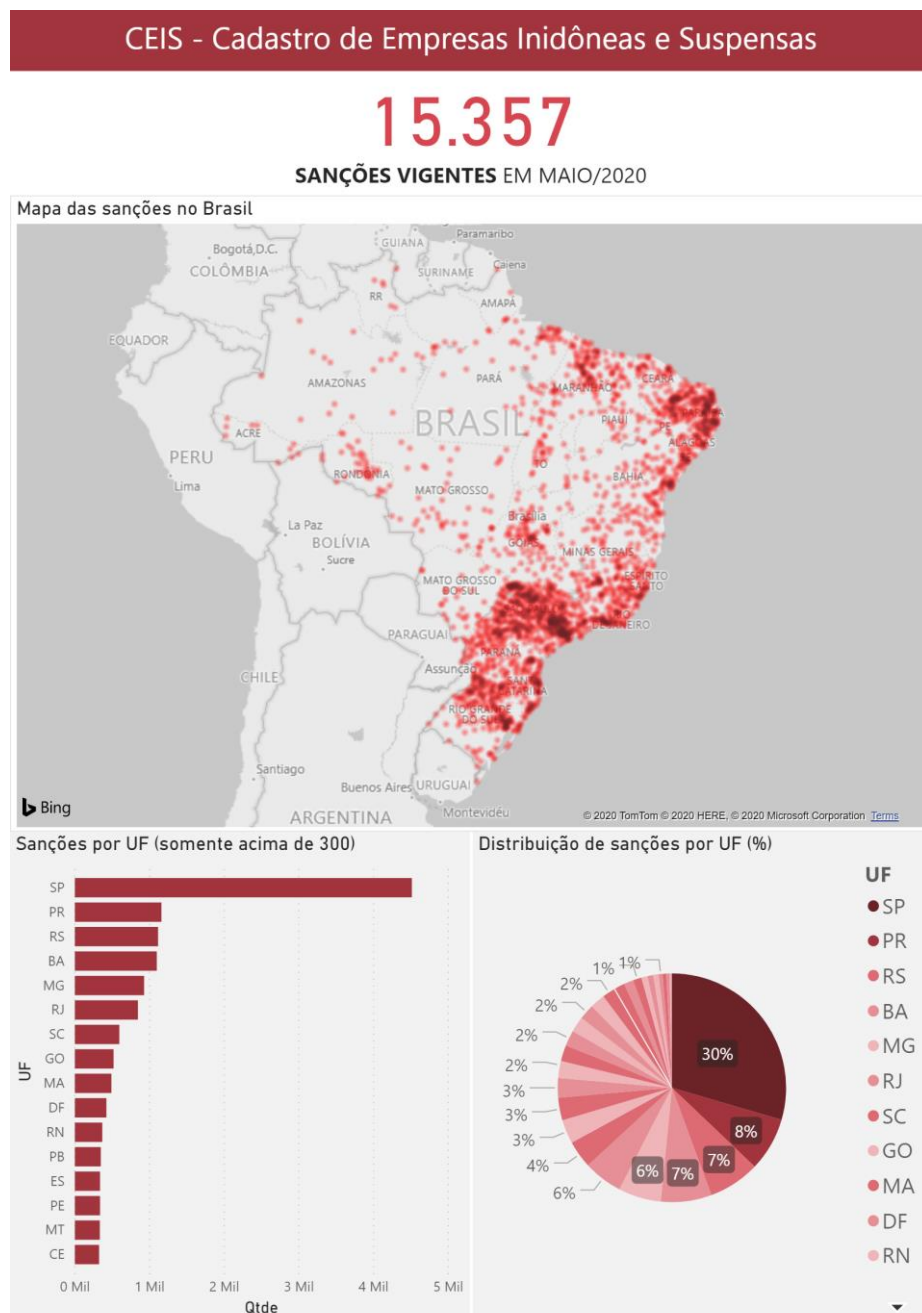
```
count          4505
mean      6120 days 08:33:42.880306
std       3607 days 12:54:31.315755
min        189 days 05:13:56.198841
25%       3384 days 05:13:56.198841
50%       5562 days 05:13:56.198841
75%       8133 days 05:13:56.198840
max       26507 days 05:13:56.198841
Name: pessoa_dataAbertura, dtype: object
```

Verificou-se que a média é, praticamente, a mesma, sendo que as empresas não reincidentes possuem, em média, cerca de 6 meses a mais de vida que as empresas reincidentes. Também foi possível verificar que o coeficiente baixou um pouco, para cerca de 59%, mas, mesmo assim, é um coeficiente alto.

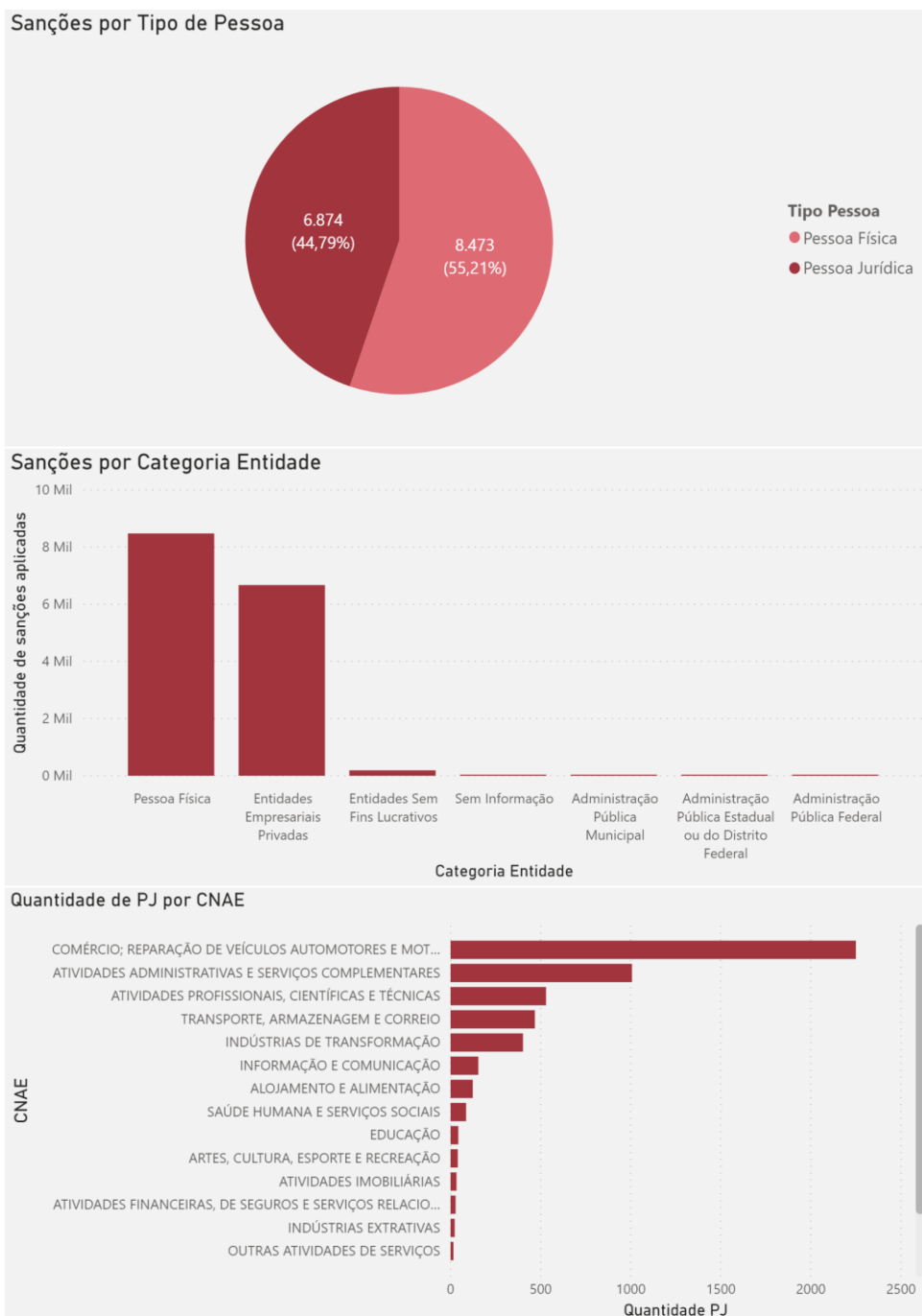
5. Apresentação dos Resultados

O intuito dessa seção é a demonstração da técnica de visualização de dados conhecida como “Data Story Telling”, para contar a história dos dados na perspectiva das análises realizadas.

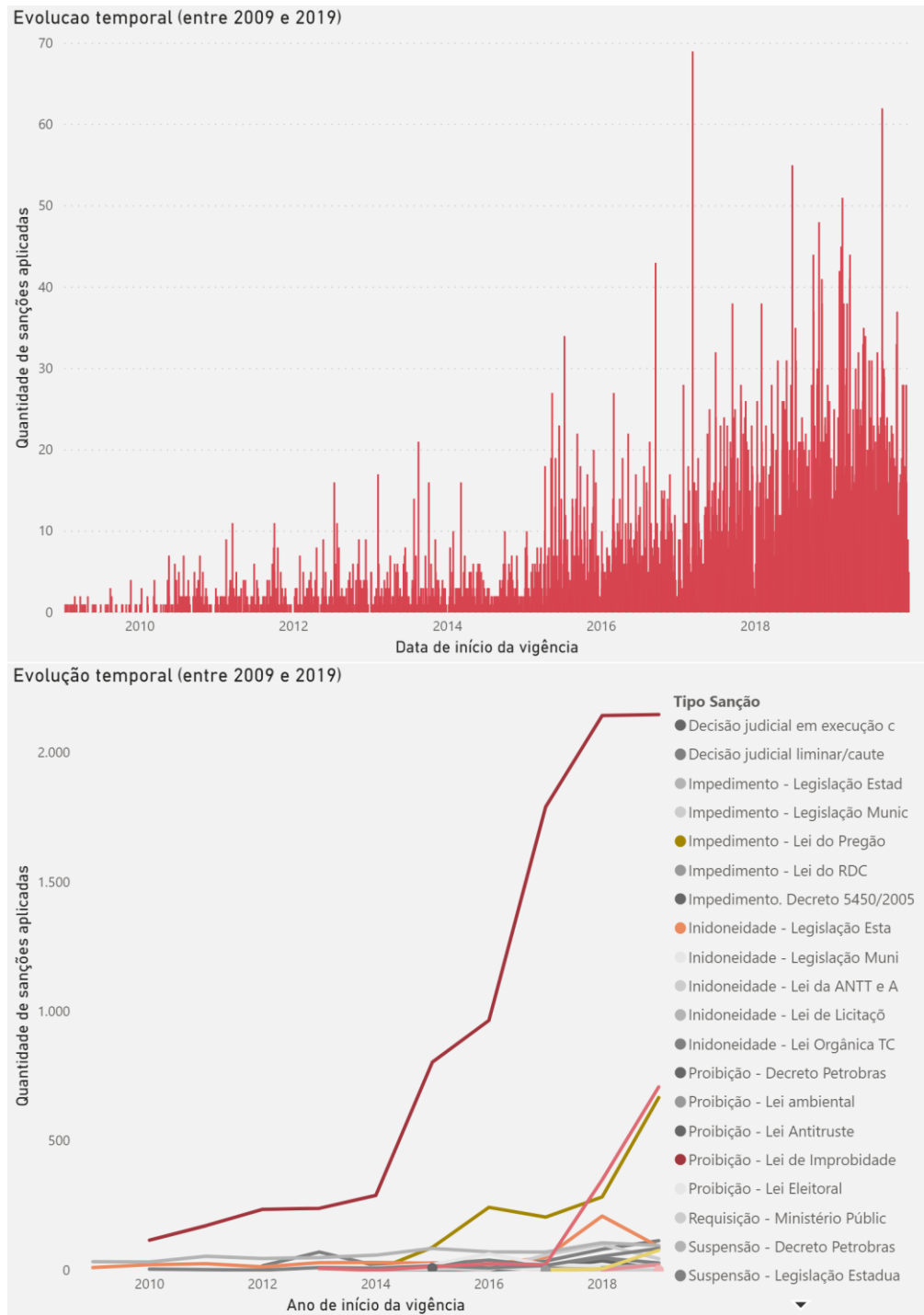
Como pode ser observado abaixo, na primeira página do “Data Story Telling” é possível verificar o número total de sanções vigentes no Brasil no mês de maio de 2020. Nos gráficos seguintes também é possível observar que o estado de São Paulo predomina no cenário das sanções, detendo cerca de 30% do volume total de sanções vigentes.



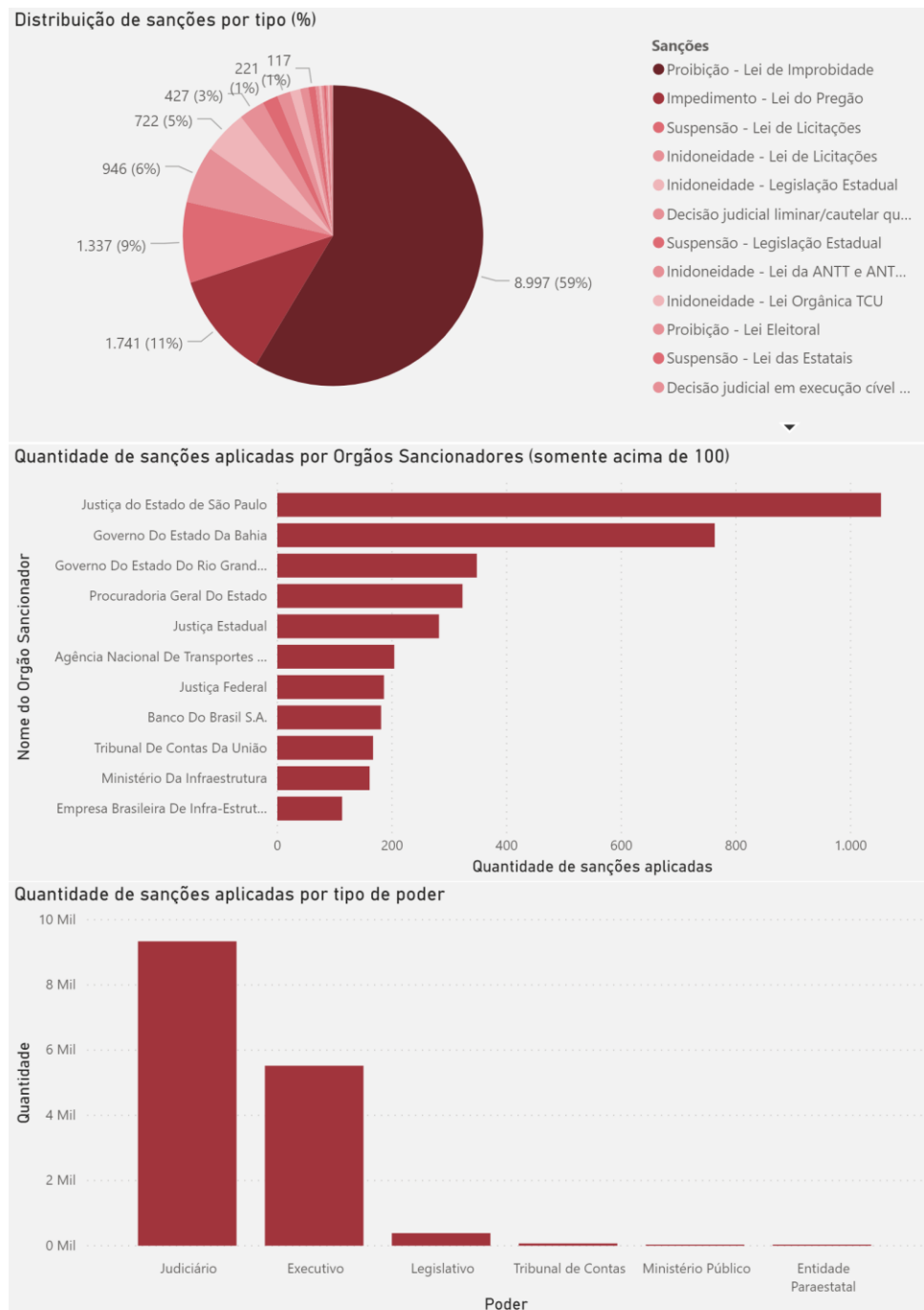
No primeiro gráfico da segunda página do “Data Story Telling” observa-se a distribuição de sanções por tipo de pessoa (Física e Jurídica). Percebe-se uma distribuição razoavelmente equilibrada. No segundo gráfico é possível verificar a quantidade de sanções por categorias de entidade. Nota-se uma predominância de pessoas físicas e de entidades empresariais privadas. O terceiro gráfico exibe uma quantidade de sanções aplicadas às Pessoas Jurídicas por tipo de atividade econômica (CNAE). É possível notar a predominância de atividades como: Comércio – Reparação de veículos automotores e motocicletas; Atividades administrativas e serviços complementares; Atividades profissionais, científicas e técnicas; Transporte, armazenagem e correio; e Indústrias de transformação.



Na terceira página encontram-se as análises com perspectivas temporais. No primeiro gráfico é possível verificar a evolução de aplicações de sanções nos últimos 10 anos. Já no segundo gráfico essa visão é detalhada por tipo de sanção, onde observa-se um aumento acentuado na aplicação de sanções relacionadas à Lei de Improbidade.



Na última página encontram-se as análises referentes às sanções. O primeiro gráfico exibe a distribuição das sanções por tipo, onde nota-se a predominância da sanção “Proibição – Lei de Improbidade”. No segundo gráfico exibe-se a quantidade de sanções aplicadas por órgão sancionadores e, mais uma vez, o estado de São Paulo figura no topo, através do órgão “Justiça do Estado de São Paulo”. No último gráfico observa-se a quantidade de sanções aplicadas por pela ótica das esferas dos poderes, onde os poderes judiciários e executivos são predominantes.



5.1. Validação dos resultados

Conforme o objetivo proposto o trabalho foi desenvolvido para avaliar a eficácia da aplicação das técnicas de Análise Exploratória de Dados (EDA) e Visualização de Dados na obtenção de insights. Para isso todo o “dataset” foi submetido a Análise Exploratória de Dados (EDA), que foi executada em cada série do conjunto de dados, em alguns casos de forma isolada em outros de forma agrupada ou comparativa. Com base no trabalho de EDA foi possível obter os seguintes insights:

- Observou-se uma distribuição balanceada de sanções entre pessoas físicas (55,2%) e jurídicas (44,8%), não sendo possível determinar alguma tendência nesse aspecto;
- No espectro das pessoas jurídicas a categoria de Entidades Empresariais Privadas é predominante, representando cerca de 97% desse universo de dados;
- Dentre as pessoas jurídicas, as que mais sofreram sanções são as classificadas com CNAE “Comércio; Reparação de Veículos Automotores e Motocicletas”, representando cerca de 34% desse universo, seguida de “Atividades Administrativas e Serviços Complementares”, com 15%, “Atividades Profissionais, Científicas e Técnicas”, com 7,9% e “Transporte, Armazenagem e Correio”, com cerca de 7%, compondo os cinco primeiros dessa lista;
- O estado de São Paulo é predominante no número de sanções aplicadas, possuindo quase 4 vezes mais sanções que o segundo colocado, o estado do Paraná.
- Os estados de São Paulo, Paraná, Rio Grande do Sul, Bahia e Minas Gerais (5 primeiros) totalizam cerca de 57% de todas as sanções do país;
- O estado que mais aplica sanções é São Paulo, totalizando cerca de 26% das sanções vigentes, seguido pelo Distrito Federal, com 8%, Rio Grande Sul, com 7%, Bahia, 6,9% e MG, com 6,2%, compondo os 5 primeiros colocados dessa classificação;
- O órgão sancionadores mais atuantes estão classificados como Justiça do Estado de São Paulo;
- Os poderes que mais aplicam sanções são o Judiciário, com cerca de 61% do total, e o Executivo, com cerca de 36%. Os dois juntos totalizam cerca de 97% de todas as sanções aplicadas;
- O prazo médio de vigência de uma sanção é de 59 meses. Porém, nessa série, percebeu-se, também, que o coeficiente de variação é de é muito alto (60%), o que faz com que a média não seja um métrica muito assertiva;
- Observou-se que o índice de reincidência é de cerca de 16,3%;

- A distribuição de reincidência entre Pessoas Físicas e Jurídicas é praticamente idêntica à distribuição de todo o universo de dados (55% PF; 45% PJ);
- Verificou-se que o estado de São Paulo também mantém a liderança dos reincidentes, com 36,5% do total;
- A sanção predominante entre os reincidentes é a “Proibição – Lei de Improbidade”;
- O tempo de vida média das empresas reincidentes é de cerca de 16 anos, mas aqui cabem duas observações: 1) O tempo de vida média das empresas reincidentes (16,3 anos) é praticamente igual das empresas não reincidentes (16,8 anos); 2) Aqui também o coeficiente de variação é muito alto (cerca de 64%), fazendo com que essa métrica não seja muito assertiva;

Após a obtenção dos insights elencados acima foi possível determinar que, as técnicas de Análise Exploratória de Dados e Visualização de Dados são eficazes para esse propósito, possibilitando a extração de métricas e estatísticas que fornecem insumos para os processos de tomada de decisão.

5.2. Trabalhos Futuros

Durante o desenvolvimento do trabalho e considerando os resultados obtido ao término foi possível elencar algumas propostas para trabalhos futuros dentro desse contexto:

- Utilização de mais conjuntos de dados das bases do governo para cruzamento das informações (ex: Dados Abertos - Compras Governamentais: <http://compras.dados.gov.br/docs/home.html>);
- Aplicação de técnicas de PCA para redução da dimensionalidade dos dados;
- Implementação de algoritmos de classificação (ex: árvore de decisão ensemble);
- Implementação de modelos de Machine Learning para detecção de padrões e predição de tendências.

6. Links

Scripts: <https://github.com/willsancheslima/tcc-puc-minas>

Vídeo: <https://youtu.be/IPJP6VOBO1k>

REFERÊNCIAS

<https://pandas.pydata.org/pandas-docs/stable/reference/api/pandas.DataFrame.html>

<https://medium.com/@hhuuggoo/handy-dandy-guide-to-working-with-time-in-pandas-738ca6eadbbd>

<https://medium.com/@amirzai/flatten-json-on-python-package-index-pypi-9e4951693a5a>

<https://www.dataquest.io/blog/python-api-tutorial/>

<https://transparenciainternacional.org.br/ipc/>

<https://comunidade.transparenciainternacional.org.br/asset/24:ucc-novas-medidas.pdf?stream=1>

<https://www.gov.br/cgu/pt-br/assuntos/transparencia-publica>

<https://www.gov.br/cgu/pt-br/assuntos/transparencia-publica/brasil-transparente>

<http://www.portaltransparencia.gov.br/sobre/o-que-e-e-como-funciona>

<http://www.portaltransparencia.gov.br/api-de-dados>