INSTITUTO FEDERAL
Brasília
Campus Brasília

TECNOLOGIA EM SISTEMAS PARA INTERNET

Gustavo William
Hugo César Alves da Silva
João Paulo Dantas
Polyana Cristina Sousa

RELATÓRIO DE PRÁTICA INTEGRADA DE CIÊNCIA DE DADOS E APRENDIZADO DE MÁQUINA

Brasília - DF
26/08/2021

Sumário

1. Objetivos

Os objetivos desta pesquisa consistem em aplicar métodos e saberes advindos das áreas de ciências de dados e aprendizagem de máquina com o intuito de coletar um conjunto de dados do meio web, mais especificamente da plataforma Nuforc, de forma tabular, sendo que, dentro deste conjunto foram considerados os relatos registrados na plataforma cujo o período de ocorrência está na faixa temporal de setembro de 1997 e agosto de 2017, aproximadamente 20 anos.

Nesta etapa da sprint; tendo concluído a coleta, exploração e preparação dos dados; tem por objetivo explorar o armazenamento dos dados em um banco de dados e proceder com uma análise detalhada dos dados com o intuito de extrair valor das informações coletadas.

2. Descrição do problema

Trabalhar o armazenamento dos dados em um banco de dados seguro e adequado para os tipos de dados que estão sendo utilizados nesta pesquisa e sequentemente efetuar uma análise detalhada dos dados extraindo valor dos mesmos, utilizando conhecimentos de machine learning para este feito. O problema então consiste em armazenar os dados e construir modelos de análise temporal.

Para isso esta etapa do projeto consiste em:

1. Criar uma conta no mlab;
2. familiarizar com o funcionamento do mongoDB;
3. Estabelecer conexão com o banco de dados;
4. Criar um banco de dados chamado ovni;
5. criar uma coleção com o nome ovnis;
6. Inserir na coleção criada todos os registros de df_OVNI_preparado;
7. Contar e mostrar quantos documentos há na coleção ovnis;
8. Resgatar todos os documentos (registros) da coleção ovnis e ordenar por tio (shape);
9. Verificar quantas ocorrências existem por estado;
10. Buscar todas as ocorrências da cidade Phenix;
11. Buscar as ocorrências do estado da Califórnia e ocultar o id de cada documento (registro);
12. Proceder com a recuperação dos dados no banco de dados e ordenar as visualizações sobre a cidade de Phoenix agrupados por dia, por mês e por ano;
13. Ordenar as observações de forma ascendente e temporalmente(da mais antiga para a mais recente)
14. Construir um gráfico de barras para visualizar os dados em forma de série temporal;
15. Construir um gráfico de linha para visualizar os dados em forma de série temporal;
16. Construir um conjunto de treinamento e um conjunto de teste;
17. Investigar os parâmetros apresentados para discriminar o melhor modelo;
18. Realizar uma previsão utilizando o melhor modelo.

3. Desenvolvimento

Para esta etapa do projeto trabalhou-se o armazenamento dos dados em banco de dados, lidando assim o mecanismo de persistência dos dados, para tal utilizamos um sistema gerenciador de banco de dados MongoDB, uma vez que os critérios de segurança, facilidade de manutenção e acesso são de de extrema valia. Para interagir com o MongoDB foi usado um servidor, este forneceu o serviço de armazenamento com o MongoDB, também foi feito o uso da interface da ferramenta MongoDB Atlas para interagir com os dados armazenados.

O módulo para conexão com o banco de dados foi o PyMongo que possui ferramentas próprias para o trabalho com banco de dados MongoDB e Python. Uma vez conectado ao banco, foram criados o banco de dados chamado ovni e uma coleção com nome ovnis, em seguida os registros de “df_OVNI_preparado.csv” foram inseridos na tabela.

Utilizando as funções do módulo pyMongo, aplicamos suas operações nos registros a fim de explorar e conhecer os dados armazenados.

Para análise temporal foi necessário, além de recuperar os dados do banco, agrupar e ordenar os mesmos de maneira cronológica tornando assim mais simples a construção dos artefatos visuais de análise, para a construção do modelo preditivo os dados foram segmentados em dados de treinamento e teste e em seguida, utilizado o pacote statsmodel, fez-se a implementação de métodos que tornaram possível trabalhar com previsões de séries temporais

As tecnologias utilizadas no projeto foram a linguagem de programação PYTHON e o ambiente de desenvolvimento do Google Collaboratory, sendo as bibliotecas, API e banco de dados, os seguintes:

- MongoDB: programa de banco de dados orientado a documentos multiplataforma;
- MongoDB Atlas: Banco de dados em nuvem, construído e gerenciado pelos desenvolvedores do MongoDB.
- PyMongo: Módulo que contém ferramentas apropriadas para trabalhar com MongoDB e Python;

- Statemodels: Pacote Python que permite aos usuários explorar modelos, estimar modelos estatísticos e realizar testes estatísticos.

Após implementado e ajustado o modelo, foi feita a medição de qualidade com base no Critério de Informação de Akaike (AIC) que é uma métrica para mensurar a qualidade de modelos estatísticos.

3.1 Respostas aos problemas da pesquisa

3.2 Código implementado

Dados no MongoDB

```
!pip install dnspython
```

```
Collecting dnspython
  Downloading dnspython-2.1.0-py3-none-any.whl (241 kB)
     |████████████████████████████████| 241 kB 7.8 MB/s
Installing collected packages: dnspython
Successfully installed dnspython-2.1.0
```

```
!pip install pymongo
import pymongo

myclient =
pymongo.MongoClient("mongodb+srv://jppd:m0ng02021@clusterprojetointegrado.
eaolt.mongodb.net/ovni?retryWrites=true&w=majority")
mydb = myclient["ovni"]
mycol = mydb["ovnis"]

x = mycol.find()
```

1 - Contar e mostrar quantos documentos há na coleção ovnis.

```
y = mycol.count()
print(y)

78217
/usr/local/lib/python3.7/dist-packages/ipykernel_launcher.py:1:
DeprecationWarning: count is deprecated. Use estimated_document_count or
count_documents instead. Please note that $where must be replaced by $expr,
$near must be replaced by $geoWithin with $center, and $nearSphere must be
replaced by $geoWithin with $centerSphere
  """Entry point for launching an IPython kernel.
```

2 - Resgatar todos os documentos (registros) da coleção ovnis e ordenar por tipo (shape).

```
w = mycol.find().sort("formato").limit(10)
for x in w:
  print(x)
```

```
{'_id': ObjectId('611d8c333e2c494514770de7'), '': '215', 'cidade': 'Louisville', 'estado': 'KY', 'formato': 'Changing', 'visualizacao_data': '1997-11-08', 'visualizacao_horario': '19:30:00', 'weekdays': 'Saturday', 'visualizacao_dia': '8', 'visualizacao_mes': '11'}
{'_id': ObjectId('611d8c333e2c494514770e81'), '': '369', 'cidade': 'Wyoming', 'estado': 'MI', 'formato': 'Changing', 'visualizacao_data': '1998-02-23', 'visualizacao_horario': '07:30:00', 'weekdays': 'Monday', 'visualizacao_dia': '23', 'visualizacao_mes': '2'}
{'_id': ObjectId('611d8c333e2c494514770e99'), '': '393', 'cidade': 'Sharon (to Brookfield, OH)', 'estado': 'PA', 'formato': 'Changing', 'visualizacao_data': '1998-02-12', 'visualizacao_horario': '22:15:00', 'weekdays': 'Thursday', 'visualizacao_dia': '12', 'visualizacao_mes': '2'}
{'_id': ObjectId('611d8c333e2c494514770e5d'), '': '333', 'cidade': 'Satellite Point (near, midway between Orlando & Boca)', 'estado': 'FL', 'formato': 'Changing', 'visualizacao_data': '1998-01-08', 'visualizacao_horario': '20:00:00', 'weekdays': 'Thursday', 'visualizacao_dia': '8', 'visualizacao_mes': '1'}
{'_id': ObjectId('611d8c333e2c494514770e88'), '': '376', 'cidade': 'Kerrville', 'estado': 'TX', 'formato': 'Changing', 'visualizacao_data': '1998-02-20', 'visualizacao_horario': '19:00:00', 'weekdays': 'Friday', 'visualizacao_dia': '20', 'visualizacao_mes': '2'}
{'_id': ObjectId('611d8c333e2c494514770e67'), '': '343', 'cidade': 'Milwaukee/West Allis/West Milwaukee', 'estado': 'WI', 'formato': 'Changing', 'visualizacao_data': '1998-01-01', 'visualizacao_horario': '23:00:00', 'weekdays': 'Thursday', 'visualizacao_dia': '1', 'visualizacao_mes': '1'}
{'_id': ObjectId('611d8c333e2c494514770f08'), '': '504', 'cidade': 'Columbus', 'estado': 'GA', 'formato': 'Changing', 'visualizacao_data': '1998-03-08', 'visualizacao_horario': '22:00:00', 'weekdays': 'Sunday', 'visualizacao_dia': '8', 'visualizacao_mes': '3'}
{'_id': ObjectId('611d8c333e2c494514770ebb'), '': '427', 'cidade': 'Yakima', 'estado': 'WA', 'formato': 'Changing', 'visualizacao_data': '1998-03-29', 'visualizacao_horario': '02:29:00', 'weekdays': 'Sunday', 'visualizacao_dia': '29', 'visualizacao_mes': '3'}
{'_id': ObjectId('611d8c333e2c494514770e93'), '': '387', 'cidade': 'Yakima', 'estado': 'WA', 'formato': 'Changing', 'visualizacao_data': '1998-02-15', 'visualizacao_horario': '12:00:00', 'weekdays': 'Sunday', 'visualizacao_dia': '15', 'visualizacao_mes': '2'}
{'_id': ObjectId('611d8c333e2c494514770d22'), '': '18', 'cidade': 'West Manchester', 'estado': 'OH', 'formato': 'Changing', 'visualizacao_data': '1997-09-18', 'visualizacao_horario': '04:15:00', 'weekdays': 'Thursday', 'visualizacao_dia': '18', 'visualizacao_mes': '9'}
```

3 - Verificar quantas ocorrências existem por estado.

```
filtro_estado = mycol.aggregate(
   [
      {
         "$group":
```

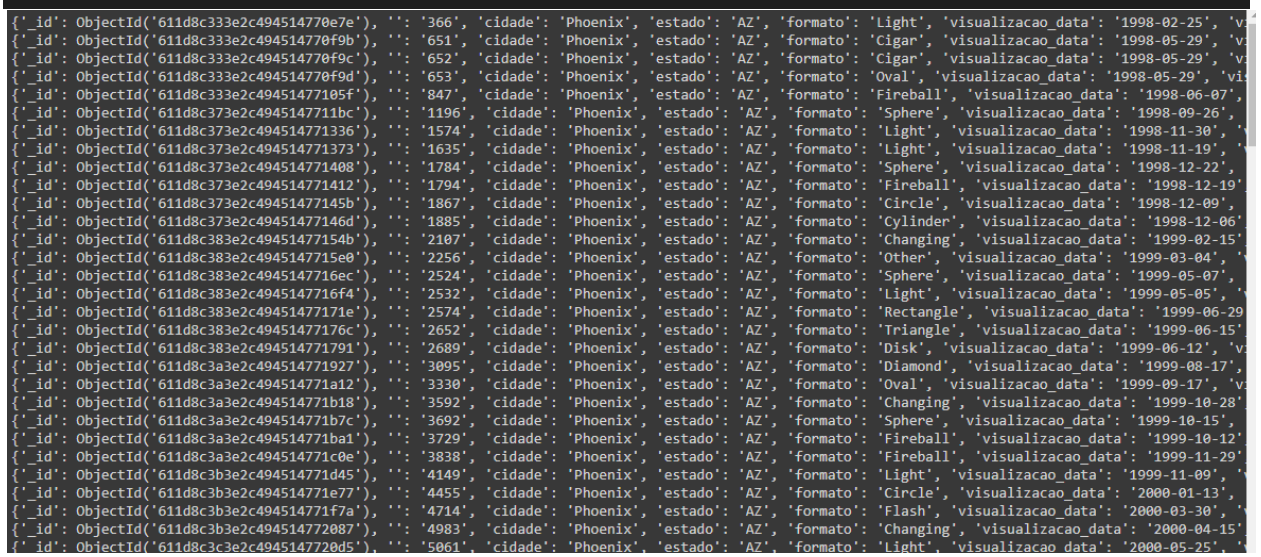
```
                    {
                        "_id": "$estado",
                        "count":{"$sum":1}
                    }
        },
        {
            "$sort":
                    {"count":-1}
        }
    ]
)

for i in filtro_estado:
  print(i)
```

```
{'_id': 'CA', 'count': 10197}
{'_id': 'FL', 'count': 4969}
{'_id': 'WA', 'count': 4277}
{'_id': 'TX', 'count': 3640}
{'_id': 'NY', 'count': 3435}
{'_id': 'AZ', 'count': 3067}
{'_id': 'PA', 'count': 2875}
{'_id': 'IL', 'count': 2780}
{'_id': 'OH', 'count': 2643}
{'_id': 'NC', 'count': 2234}
{'_id': 'MI', 'count': 2190}
{'_id': 'OR', 'count': 2132}
{'_id': 'CO', 'count': 1898}
{'_id': 'MO', 'count': 1670}
{'_id': 'NJ', 'count': 1665}
{'_id': 'GA', 'count': 1595}
{'_id': 'VA', 'count': 1590}
{'_id': 'MA', 'count': 1546}
{'_id': 'IN', 'count': 1502}
{'_id': 'WI', 'count': 1495}
{'_id': 'SC', 'count': 1411}
{'_id': 'TN', 'count': 1376}
{'_id': 'MN', 'count': 1254}
{'_id': 'MD', 'count': 1103}
{'_id': 'CT', 'count': 1041}
{'_id': 'NV', 'count': 975}
{'_id': 'KY', 'count': 965}
{'_id': 'NM', 'count': 914}
{'_id': 'UT', 'count': 876}
{'_id': 'OK', 'count': 825}
{'_id': 'AL', 'count': 790}
{'_id': 'IA', 'count': 759}
{'_id': 'ID', 'count': 721}
{'_id': 'ME', 'count': 686}
{'_id': 'KS', 'count': 682}
{'_id': 'LA', 'count': 677}
{'_id': 'AR', 'count': 668}
{'_id': 'NH', 'count': 658}
{'_id': 'MT', 'count': 570}
{'_id': 'WV', 'count': 535}
```

```
{'_id': 'MS', 'count': 450}
{'_id': 'NE', 'count': 407}
{'_id': 'HI', 'count': 405}
{'_id': 'AK', 'count': 403}
{'_id': 'VT', 'count': 366}
{'_id': 'RI', 'count': 361}
{'_id': 'DE', 'count': 243}
{'_id': 'WY', 'count': 231}
{'_id': 'SD', 'count': 214}
{'_id': 'ND', 'count': 141}
{'_id': 'DC', 'count': 110}
```

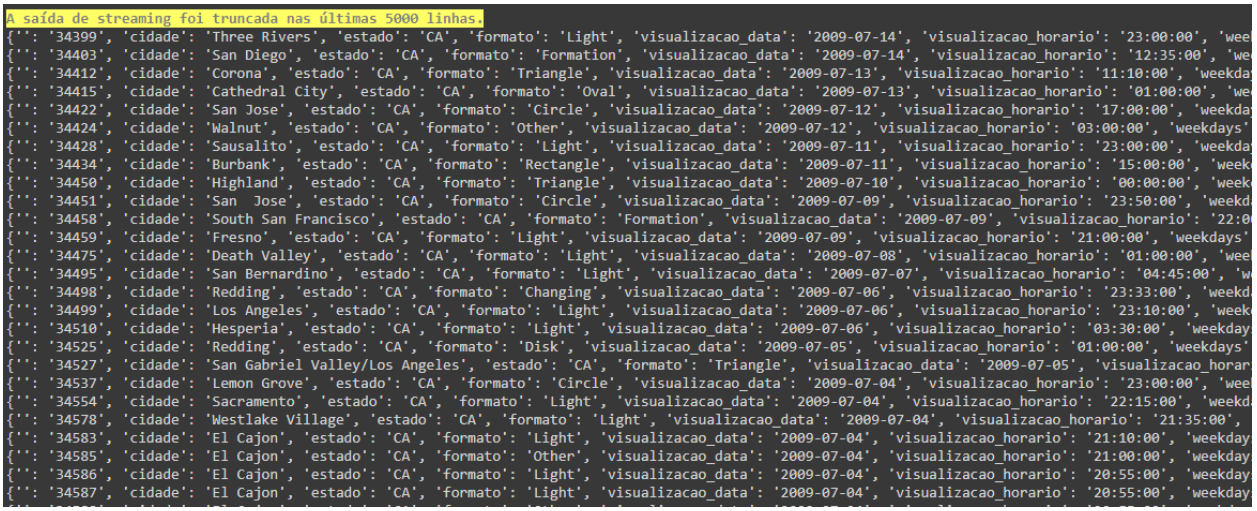
4 - Buscar todas as ocorrências da cidade Phoenix.

```
filtro_cidade = mycol.find({"cidade":"Phoenix"});

for i in filtro_cidade:
  print(i)
```



5 - Buscar as ocorrências do estado da Califórnia e ocultar o id de cada documento (registro).

```
filtro_id = mycol.find({"estado":"CA"}, {'_id': False})

for i in filtro_id:
  print(i)
```



Análise Temporal

1 - Recuperação dos dados

1.1 - Recuperar os dados (do MongoDB a partir do arquivo df_OVNI_preparado) de visualização sobre a cidade de Phoenix agrupados por dia, por mês e por ano;

```
!pip install dnspython
import pymongo
import pandas as pd
from pymongo import MongoClient
client =
pymongo.MongoClient("mongodb+srv://jppd:m0ng02021@clusterprojetointegrado.
eaolt.mongodb.net/ovni?retryWrites=true&w=majority")
db = client["ovni"]
collection = db["ovnis"]
df = pd.DataFrame(list(collection.find()))
df.drop(df.columns[0:2], axis=1, inplace=True)
df.head()
```

	cidade	estado	formato	visualizacao_data	visualizacao_horario	weekdays	visualizacao_dia	visualizacao_mes
0	Madison	WI	Light	1997-09-30	22:00:00	Tuesday	30	9
1	San Francisco	CA	Triangle	1997-09-28	23:15:00	Sunday	28	9
2	Egan	SD	Other	1997-09-27	23:00:00	Saturday	27	9
3	Crestwood	KY	Disk	1997-09-27	05:00:00	Saturday	27	9
4	Clearfield	UT	Triangle	1997-09-25	22:00:00	Thursday	25	9

```
df['visualizacao_ano'] = pd.to_datetime(df['visualizacao_data']).dt.year
df.head()
```

	cidade	estado	formato	visualizacao_data	visualizacao_horario	weekdays	visualizacao_dia	visualizacao_mes	visualizacao_ano
0	Madison	WI	Light	1997-09-30	22:00:00	Tuesday	30	9	1997
1	San Francisco	CA	Triangle	1997-09-28	23:15:00	Sunday	28	9	1997
2	Egan	SD	Other	1997-09-27	23:00:00	Saturday	27	9	1997
3	Crestwood	KY	Disk	1997-09-27	05:00:00	Saturday	27	9	1997
4	Clearfield	UT	Triangle	1997-09-25	22:00:00	Thursday	25	9	1997

```
filtro = df['cidade'] == 'Phoenix'
df_phoenix = df[filtro].sort_values(by='visualizacao_data', ascending=True).reset_index(drop=True)
df_phoenix
```

	cidade	estado	formato	visualizacao_data	visualizacao_horario	weekdays	visualizacao_dia	visualizacao_mes	visualizacao_ano
0	Phoenix	AZ	Light	1998-02-25	03:00:00	Wednesday	25	2	1998
1	Phoenix	AZ	Cigar	1998-05-29	05:10:00	Friday	29	5	1998
2	Phoenix	AZ	Cigar	1998-05-29	05:10:00	Friday	29	5	1998
3	Phoenix	AZ	Oval	1998-05-29	05:00:00	Friday	29	5	1998
4	Phoenix	AZ	Fireball	1998-06-07	21:05:00	Sunday	7	6	1998
...	...	...	...	...	...	...	...	...	...
486	Phoenix	AZ	Other	2017-06-15	15:35:00	Thursday	15	6	2017
487	Phoenix	AZ	Oval	2017-07-06	21:25:00	Thursday	6	7	2017
488	Phoenix	AZ	Fireball	2017-07-26	04:20:00	Wednesday	26	7	2017
489	Phoenix	AZ	Flash	2017-08-04	21:15:00	Friday	4	8	2017
490	Phoenix	AZ	Light	2017-08-14	00:20:00	Monday	14	8	2017

491 rows × 9 columns

1.2 - Ordenar as observações de forma ascendente temporalmente (da observação mais antiga para a observação mais recente).

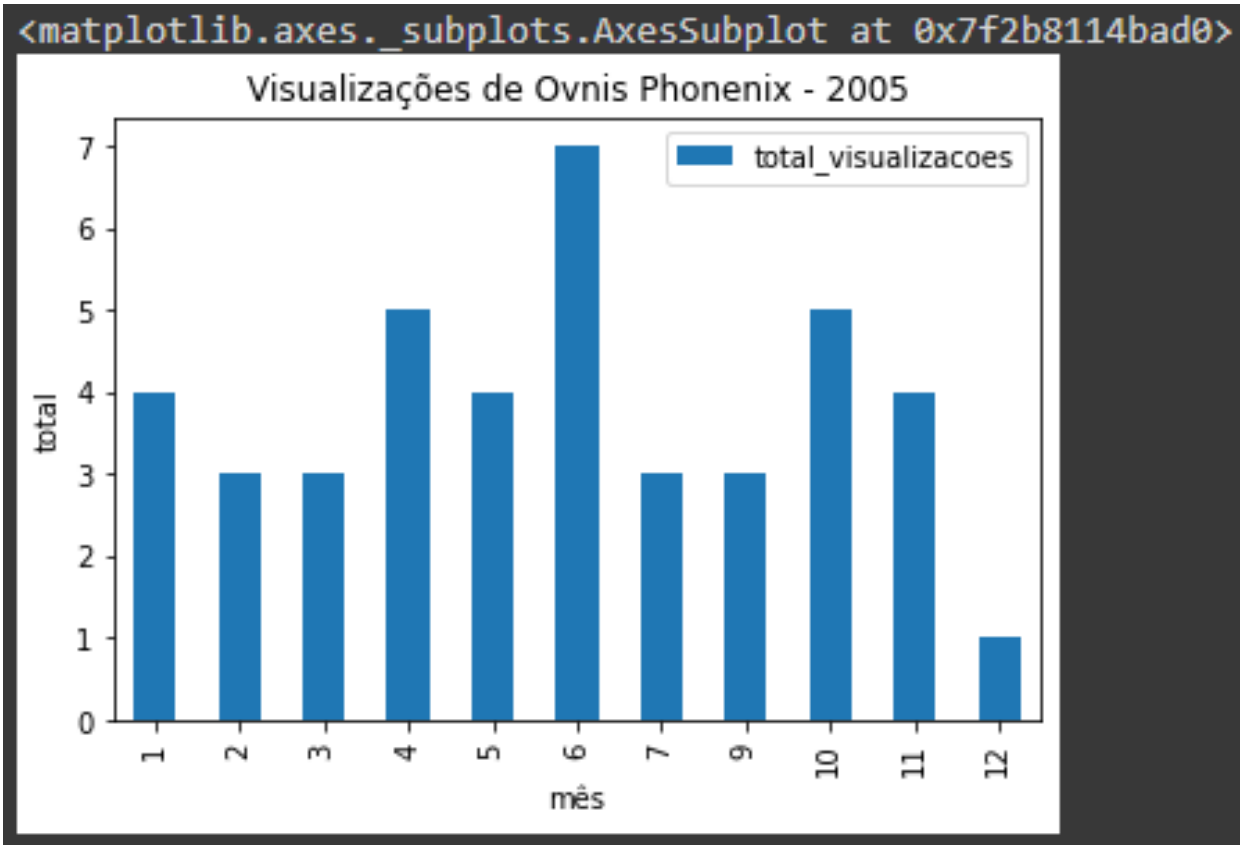
	visualizacao_data	total_visualizacoes
0	1998-02-25	1
1	1998-05-29	3
2	1998-06-07	1
3	1998-09-26	1
4	1998-11-19	1
...	...	...
439	2017-06-15	2
440	2017-07-06	1
441	2017-07-26	1
442	2017-08-04	1
443	2017-08-14	1

444 rows × 2 columns

2 - Visualização dos dados em forma de Série Temporal

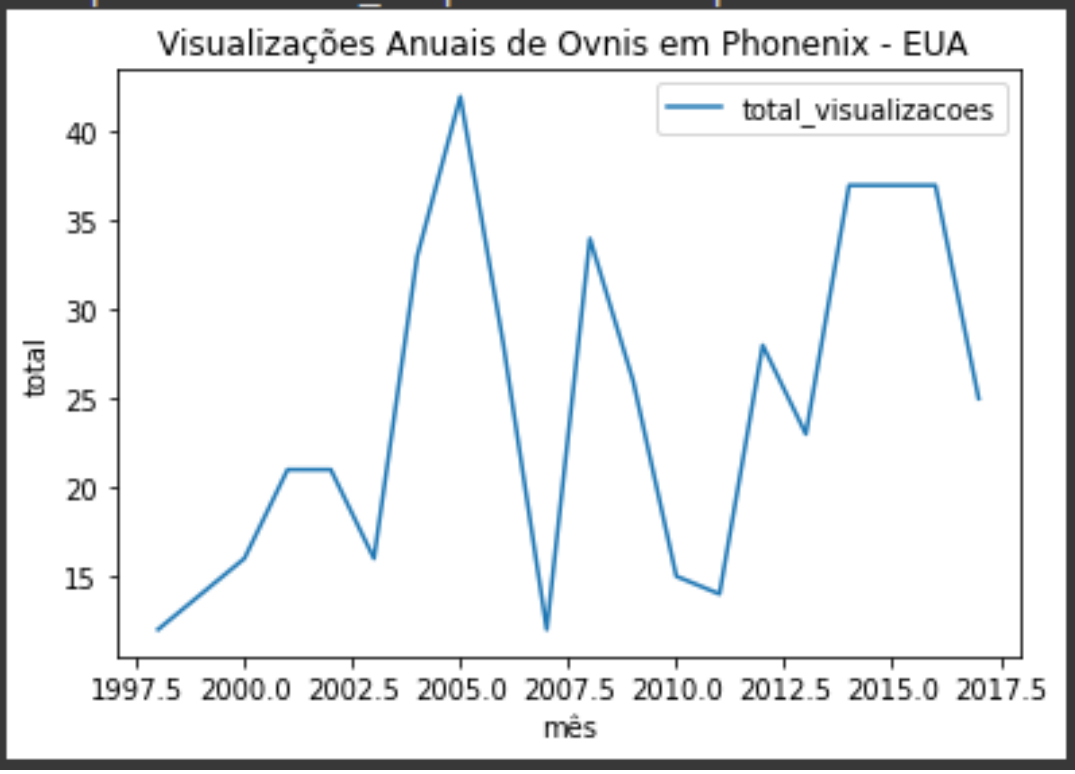
2.1 - Observar o gráfico em barras da série temporal para o ano x de forma a investigar como se comporta a distribuição das visualizações.

```
filtro = df_phoenix['visualizacao_ano'] == 2005
df_phoenix_2005 = df_phoenix[filtro]



#Tive que dar um cast nesses dois campos pois não estava funcionando o
sort_values, provavelmente estavam sendo interpretados como text
df_phoenix_2005.visualizacao_mes                                    =
df_phoenix_2005.visualizacao_mes.astype(int)
df_phoenix_2005.visualizacao_dia                                    =
df_phoenix_2005.visualizacao_dia.astype(int)



df_phoenix_mensal                                                   =
df_phoenix_2005.groupby(['visualizacao_mes']).size().reset_index()
df_phoenix_mensal.rename(columns={0:
"total_visualizacoes"},inplace='True')
df_phoenix_mensal.sort_values(by='visualizacao_mes',       ascending=True,
inplace=True)
df_phoenix_mensal.plot(                'visualizacao_mes'                ,
'total_visualizacoes',kind='bar',       xlabel='mês',       ylabel='total',
title='Visualizações de Ovnis Phonenix - 2005')
```




```
df_phoenix_anual =
df_phoenix.groupby(['visualizacao_ano']).size().reset_index()
df_phoenix_anual.rename(columns={0: "total_visualizacoes"},inplace='True')
df_phoenix_anual.sort_values(by='visualizacao_ano', ascending=True,
inplace=True)

df_phoenix_anual.plot( 'visualizacao_ano' , 'total_visualizacoes',
xlabel='mês', ylabel='total', title='Visualizações Anuais de Ovnis em
Phonenix - EUA ')
```

```
<matplotlib.axes._subplots.AxesSubplot at 0x7f2b80a1d910>
```



Construção dos conjuntos de Treinamento e Teste

3.1 - Separar 70% das observações para treinamento e 30% das observações para teste (como se trata de uma informação temporal, não podemos pegar uma amostra aleatória, sugestão: calcular o índice que corresponde a 70% das observações e considerar da primeira amostra até ele para treinamento; e do índice seguinte até o final para teste).

```
df_phoenix_data =
df_phoenix.groupby(['visualizacao_data']).size().reset_index()
df_phoenix_data.rename(columns={0: "total_visualizacoes"},inplace='True')
df_phoenix_data.sort_values(by='visualizacao_data', ascending=True,
inplace=True)
df_phoenix_data.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 444 entries, 0 to 443
Data columns (total 2 columns):
 #   Column              Non-Null Count  Dtype
---  ------              --------------  -----
 0   visualizacao_data   444 non-null    object
 1   total_visualizacoes  444 non-null    int64
dtypes: int64(1), object(1)
memory usage: 10.4+ KB
```

```
treino = (0.7*444)
```

```
teste = 0.3*444
print('70% da base corresponde a '+ str(treino) + ' e 30% corresponde a '
+ str(teste) )
```

```
70% da base corresponde a 310.79999999999995 e 30% corresponde a 133.2
```

```
df_treino = df_phoenix_data[df_phoenix_data.index < '2013-06-17']
df_teste = df_phoenix_data[df_phoenix_data.index >= '2013-06-17']
```

4 - Investigar os parâmetros para discriminar o melhor modelo:

4.1 - Utilizando o pacote statsmodels, vamos testar uma família de métodos apropriados para lidar com previsão de séries temporais chamados conjuntamente de SARIMAX (Links para um site externo.), ou seja, utilize a função SARIMAX para criar um modelo;

```
import statsmodels.api as sm
```

4.2 - Em seguida, chame a função fit() para ajustar o modelo

```
mod = sm.tsa.statespace.SARIMAX(df_treino.values, trend='c',
order=(1,1,1), seasonal_order=(0,1,1,12))
res = mod.fit(disp=False)
```

```
/usr/local/lib/python3.7/dist-packages/statsmodels/base/model.py:512:
ConvergenceWarning: Maximum Likelihood optimization failed to converge. Check
mle_retvals
  "Check mle_retvals", ConvergenceWarning)
```

4.3 - Para medir a qualidade do modelo ajustado, acesse a propriedade AIC do resultado. O Critério de Informação de Akaike (AIC em inglês) é uma métrica "quanto menor melhor", dessa forma, ao comparar modelos diferentes, aquele que possuir o menor valor de AIC é o melhor.

```
print(res.summary())
```

```
                              Statespace Model Results
==========================================================================================
Dep. Variable:                                 y   No. Observations:                  311
Model:             SARIMAX(1, 1, 1)x(0, 1, 1, 12)   Log Likelihood                -205.349
Date:                            Sat, 28 Aug 2021   AIC                            420.697
Time:                                    19:06:29   BIC                            439.183
Sample:                                         0   HQIC                           428.097
                                              - 311
Covariance Type:                              opg
==============================================================================
                 coef    std err          z      P>|z|      [0.025      0.975]
------------------------------------------------------------------------------
intercept  -4.527e-05      0.000     -0.294      0.769      -0.000       0.000
ar.L1          0.1118      0.090      1.244      0.214      -0.064       0.288
ma.L1         -0.9986      0.558     -1.791      0.073      -2.091       0.094
ma.S.L12      -0.9986      2.639     -0.378      0.705      -6.172       4.175
sigma2         0.1979      0.519      0.381      0.703      -0.819       1.215
===================================================================================
Ljung-Box (Q):                       16.87   Jarque-Bera (JB):            106892.25
Prob(Q):                              1.00   Prob(JB):                         0.00
Heteroskedasticity (H):               0.76   Skew:                             7.74
Prob(H) (two-sided):                  0.18   Kurtosis:                        94.48
===================================================================================

Warnings:
[1] Covariance matrix calculated using the outer product of gradients (complex-step).
```

4.4 - Exemplo de qualidade do modelo de acordo com o parâmetro AIC:

```
print('A qualidade do modelo estimada pelo AIC é: 420.697')
```

```
A qualidade do modelo estimada pelo AIC é: 420.697
```

```
print(res.params)
```

```
[-4.52687097e-05  1.11799220e-01 -9.98575347e-01 -9.98620168e-01
  1.97940567e-01]
```

5 - A última etapa é realizar uma previsão utilizando o melhor modelo:

5.1 - Utilizando a função forecast sobre o modelo ajustado, faça uma previsão apropriada para a quantidade de dias que existem no seu conjunto de teste;

```
sarima_pred = res.forecast(133)
sarima_pred
```

```
array([1.23246139, 1.07509921, 1.07479813, 1.0357217 , 1.30428575,
       1.03434416, 0.99519174, 1.07142538, 1.10921311, 1.07006115,
       1.10781123, 1.03020424, 1.23071706, 1.06973779, 1.06898705,
       1.02981508, 1.29832318, 1.02833007, 0.98912662, 1.06530928,
       1.10304605, 1.06384312, 1.10154223, 1.02388428, 1.22434613,
       1.06331589, 1.06251419, 1.02329125, 1.29174839, 1.02170431,
       0.98244989, 1.05858158, 1.09626739, 1.05701349, 1.09466163,
       1.01695271, 1.2173636 , 1.05628239, 1.05542972, 1.01615582,
       1.28456199, 1.01446694, 0.97516155, 1.05124228, 1.08887712,
       1.04957226, 1.08716943, 1.00940954, 1.20976946, 1.04863729,
```

```
       1.04773366, 1.00840879, 1.27676399, 1.00661797, 0.96726162,
       1.04329138, 1.08087525, 1.04151942, 1.07906563, 1.00125478,
       1.20156373, 1.04038059, 1.03942599, 1.00005015, 1.26835439,
       0.99815741, 0.95875009, 1.03472888, 1.07226179, 1.03285499,
       1.07035023, 0.99248841, 1.19274639, 1.03151229, 1.03050672,
       0.99107992, 1.25933318, 0.98908524, 0.94962695, 1.02555478,
       1.06303672, 1.02357895, 1.06102323, 0.98311044, 1.18331746,
       1.02203239, 1.02097585, 0.98149808, 1.24970038, 0.97940147,
       0.93989221, 1.01576907, 1.05320005, 1.01369131, 1.05108462,
       0.97312087, 1.17327692, 1.01194088, 1.01083338, 0.97130464,
       1.23945598, 0.9691061 , 0.92954588, 1.00537177, 1.04275177,
       1.00319208, 1.04053442, 0.9625197 , 1.16262478, 1.00123778,
       1.00007931, 0.96049961, 1.22859997, 0.95819912, 0.91858794,
       0.99436286, 1.0316919 , 0.99208124, 1.02937261, 0.95130693,
       1.15136104, 0.98992307, 0.98871364, 0.94908297, 1.21713237,
       0.94668055, 0.9070184 , 0.98274236, 1.02002043, 0.9803588 ,
       1.01759921, 0.93948255, 1.1394857 ])
```

5.2 - Calcule o erro médio e o desvio-padrão com relação ao seu conjunto de testes.

```
from sklearn.metrics import mean_squared_error
import numpy as np

erro_medio = mean_squared_error(df_teste.values, sarima_pred)
erro_medio
0.1991035863307196

desvio_padrao = np.sqrt(erro_medio)
desvio_padrao
0.4462102490202568
```

4. Considerações finais

Compreendemos que através de uma análise profunda dos dados conseguimos extrair informações, que em ocasiões precisas podem ser capazes de amparar decisões importantes para o futuro.

O uso do MongoDb foi uma novidade para equipe e só foi possível vencer o desafio com a ajuda de todos e o aprendizado mostrou a relevância do uso da tecnologia como mais uma ferramenta útil no processo de organização dos dados.

O uso do SARIMAX despertou o interesse da equipe em aprender mais sobre a ferramenta, uma vez que percebemos o seu potencial de aplicações em outras áreas, como por exemplo, previsão de preços de ações.

Havia uma proporção de conhecimento desigual entre os integrantes da equipe sobre as ferramentas e bibliotecas utilizadas.

De modo geral, a experiência no desenvolvimento das soluções agregou conhecimento à equipe.

Referências

MongoDB. Documentação MongoDB. Disponível em: <https://docs.mongodb.com>. Acesso em: 27 de agosto de 2021.

MongoDBAtlas. Documentação MongoDBAtlas. Disponível em: <https://docs.atlas.mongodb.com>. Acesso em: 27 de agosto de 2021.

WikiPedia. Critério de Informação de Akaike. Disponível em: <https://pt.wikipedia.org/wiki/Critério_de_informação_de_Akaike>. Acesso em: 27 de agosto de 2021.