Nome: João Emanuel da Silva Lins

Matrícula: 162080263 ¶

Açude de Epitácio Pessoa

Regressao Linear - Açudes da Paraíba

Açude de Boqueirão - Com Recarga do Rio São Francisco (Abril/2017)

http://www.aesa.pb.gov.br/aesa-website/monitoramento/volume-acude/?id_acude=531 (http://www.aesa.pb.gov.br/aesa-website/monitoramento/volume-acude/?id_acude=531)

Acude de Boqueirão com Recarga do Rio São Francisco - 03/05/2017 ate 01/05/2018

In [1]:

```
import pandas as pd
```

In [2]:

```
# leitura do dataset
# Acude de Boqueirão com Recarga do Rio São Francisco - 03/05/2017 ate 01/05/201
8
df = pd.read_excel("Boqueirao-2017-2018.xlsx")
df.head()
```

Out[2]:

	Unnamed: 0	Açude	Data do registro	Volume (%)	Volume (m³)	Aporte (m²)
0	0	Epitácio Pessoa	20/04/2017	2.92	12033034.63	58974.26
1	1	Epitácio Pessoa	21/04/2017	2.94	12121496.02	88461.39
2	2	Epitácio Pessoa	22/04/2017	2.98	12268931.67	147435.65
3	3	Epitácio Pessoa	23/04/2017	3.02	12445854.45	176922.78
4	4	Epitácio Pessoa	24/04/2017	3.08	12681751.49	235897.04

In [3]:

```
del df["Açude"]
del df["Unnamed: 0"]
df.head()
```

Out[3]:

	Data do registro	Volume (%)	Volume (m³)	Aporte (m²)
0	20/04/2017	2.92	12033034.63	58974.26
1	21/04/2017	2.94	12121496.02	88461.39
2	22/04/2017	2.98	12268931.67	147435.65
3	23/04/2017	3.02	12445854.45	176922.78
4	24/04/2017	3.08	12681751.49	235897.04

In [4]:

```
df.tail()
```

Out[4]:

	Data do registro	Volume (%)	Volume (m³)	Aporte (m²)
374	30/04/2018	35.13	1.446395e+08	0.00
375	01/05/2018	35.40	1.457439e+08	1104342.24
376	02/05/2018	35.58	1.464801e+08	736228.16
377	03/05/2018	35.63	1.466641e+08	184057.04
378	04/05/2018	35.67	1.468482e+08	184057.04

In [5]:

```
len(df)
```

Out[5]:

379

In [6]:

```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 379 entries, 0 to 378
Data columns (total 4 columns):
 #   Column            Non-Null Count  Dtype  
---  ------            --------------  -----  
 0   Data do registro  379 non-null    object 
 1   Volume (%)        379 non-null    float64
 2   Volume (m³)       379 non-null    float64
 3   Aporte (m²)       379 non-null    float64
dtypes: float64(3), object(1)
memory usage: 12.0+ KB
```

In [7]:

```
# converter atributo para data
df['Data do registro'] =  pd.to_datetime( df['Data do registro'], format="%d/%m/%Y")
df.head()
```

Out[7]:

	Data do registro	Volume (%)	Volume (m³)	Aporte (m²)
0	2017-04-20	2.92	12033034.63	58974.26
1	2017-04-21	2.94	12121496.02	88461.39
2	2017-04-22	2.98	12268931.67	147435.65
3	2017-04-23	3.02	12445854.45	176922.78
4	2017-04-24	3.08	12681751.49	235897.04

In [8]:

```
# colocar a data como indice para fazer uma série temporal
df2 = df.set_index(pd.DatetimeIndex(df['Data do registro']))
del df2['Data do registro']
df2.head()
```

Out[8]:

Data do registro	Volume (%)	Volume (m³)	Aporte (m²)
2017-04-20	2.92	12033034.63	58974.26
2017-04-21	2.94	12121496.02	88461.39
2017-04-22	2.98	12268931.67	147435.65
2017-04-23	3.02	12445854.45	176922.78
2017-04-24	3.08	12681751.49	235897.04

In [9]:

```
#del df2["Data do registro"]
df2.describe()["Volume (%)"]
```

Out[9]:

```
count    379.000000
mean      10.493404
std        6.468186
min        2.920000
25%        7.535000
50%        9.240000
75%       10.220000
max       35.670000
Name: Volume (%), dtype: float64
```

In [10]:

```
df2.info()
```

```
<class 'pandas.core.frame.DataFrame'>
DatetimeIndex: 379 entries, 2017-04-20 to 2018-05-04
Data columns (total 3 columns):
 #   Column       Non-Null Count  Dtype  
---  ------       --------------  -----  
 0   Volume (%)   379 non-null    float64
 1   Volume (m³)  379 non-null    float64
 2   Aporte (m²)  379 non-null    float64
dtypes: float64(3)
memory usage: 11.8 KB
```

In [11]:

```
df2.head()
```

Out[11]:

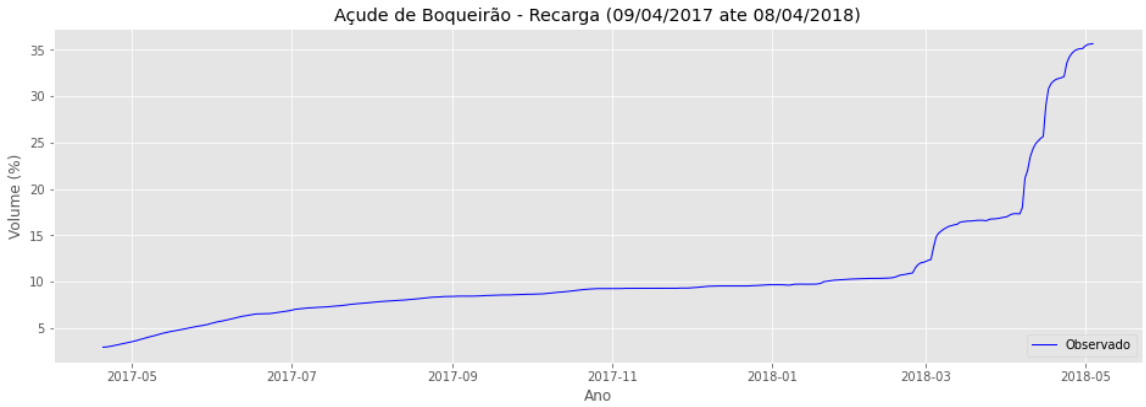
Data do registro	Volume (%)	Volume (m³)	Aporte (m²)
2017-04-20	2.92	12033034.63	58974.26
2017-04-21	2.94	12121496.02	88461.39
2017-04-22	2.98	12268931.67	147435.65
2017-04-23	3.02	12445854.45	176922.78
2017-04-24	3.08	12681751.49	235897.04

Plota dados do açude de Boqueirão - 20/04/2017 ate 04/05/2018

In [12]:

```
import matplotlib, matplotlib.pyplot as plt
%matplotlib inline
plt.figure(figsize=(16,5))
matplotlib.style.use("ggplot")

plt.plot(df2["Volume (%)"], color='blue', linewidth=1)
# adcionando textos
plt.title("Açude de Boqueirão - Recarga (09/04/2017 ate 08/04/2018)")
plt.xlabel("Ano")
plt.ylabel("Volume (%)")
plt.legend(["Observado"], loc="lower right")
plt.show()
```



In [13]:

```
# criar uma coluna com os dias
df3 = df2
numero_de_dias_observados = len(df3)
df3["dia"] = range(1,numero_de_dias_observados+1)
df3.head()
```

Out[13]:

Data do registro	Volume (%)	Volume (m³)	Aporte (m²)	dia
2017-04-20	2.92	12033034.63	58974.26	1
2017-04-21	2.94	12121496.02	88461.39	2
2017-04-22	2.98	12268931.67	147435.65	3
2017-04-23	3.02	12445854.45	176922.78	4
2017-04-24	3.08	12681751.49	235897.04	5

In [14]:

```
print("numero_de_dias_observados = ",len(df3)) # numero de tuplas ou linhas ou m
edições
```

```
numero_de_dias_observados =  379
```

In [15]:

```
df3.tail()
```

Out[15]:

Data do registro	Volume (%)	Volume (m³)	Aporte (m²)	dia
2018-04-30	35.13	1.446395e+08	0.00	375
2018-05-01	35.40	1.457439e+08	1104342.24	376
2018-05-02	35.58	1.464801e+08	736228.16	377
2018-05-03	35.63	1.466641e+08	184057.04	378
2018-05-04	35.67	1.468482e+08	184057.04	379

Preparar os dados para aplicar o modelo de regressao

In [16]:

```
import numpy, pandas as pd
import matplotlib, matplotlib.pyplot as plt
import sklearn.linear_model as lm
import warnings
warnings.filterwarnings('ignore')
#X = numpy.array([x.toordinal() for x in df3.index])[:, numpy.newaxis]

X = df3['dia'].values.reshape(-1, 1)
y = df3['Volume (%)']

print(X[:3])
print(y[:3])
```

```
[[1]
 [2]
 [3]]
Data do registro
2017-04-20    2.92
2017-04-21    2.94
2017-04-22    2.98
Name: Volume (%), dtype: float64
```

Aplicar o modelo de Regressao

In [17]:

```
# Aplicar o modelo de Regressao
olm = lm.LinearRegression()
olm.fit(X, y)
olm
```

Out[17]:

```
LinearRegression()
```

Métrica para avaliar o modelo

É uma métrica que mede os o quanto dos futuros examplos são previstos corretamente.
Varia entre 0 e 1. Quanto mais o R^2 se aproximar de 1, melhor a previsão.
Um R^2 próximo de 0, não reflete o modelo.

In [18]:

```
from sklearn.metrics import r2_score
import numpy as np

y_pred = olm.predict(df3['dia'].values.reshape(-1, 1))
# Evaluate the model

dfp = pd.DataFrame()
dfp['dia'] = df3['dia']
dfp['volume'] = df3['Volume (%)']
dfp['Vol_prev'] = y_pred
dfp['erro'] = np.abs(dfp['volume'] - dfp['Vol_prev'])
dfp['quadrado_erro'] = np.abs(dfp['volume'] - dfp['Vol_prev']) ** 2
dfp.head()
```

Out[18]:

Data do registro	dia	volume	Vol_prev	erro	quadrado_erro
2017-04-20	1	2.92	1.990586	0.929414	0.863811
2017-04-21	2	2.94	2.035574	0.904426	0.817986
2017-04-22	3	2.98	2.080563	0.899437	0.808987
2017-04-23	4	3.02	2.125551	0.894449	0.800039
2017-04-24	5	3.08	2.170540	0.909460	0.827118

In [19]:

```
import numpy as np

np.sqrt(dfp.quadrado_erro.sum())
```

Out[19]:

```
81.44069961913722
```

In [20]:

```
np.sqrt(dfp.erro.sum())
```

Out[20]:

```
31.871490181116236
```

In [21]:

```
r2 = r2_score(y, y_pred)
print('r2 = ', r2 )
```

```
r2 =  0.5806021874539069
```

Plota os Dados Observados e Previstos

preparando a saida para plotar

In [22]:

```
# a reta de regressao
a = olm.coef_[0]
b = olm.intercept_
print(' y = {0} * x + {1}'.format(a, b))
```

```
 y = 0.04498845398456563 * x + 1.9455974368639293
```

In [23]:

```
x1 = 366 * 4
y1 = 0.04498845398456564 * x1 + 1.9455974368639257
y1
```

Out[23]:

```
67.80869407026802
```

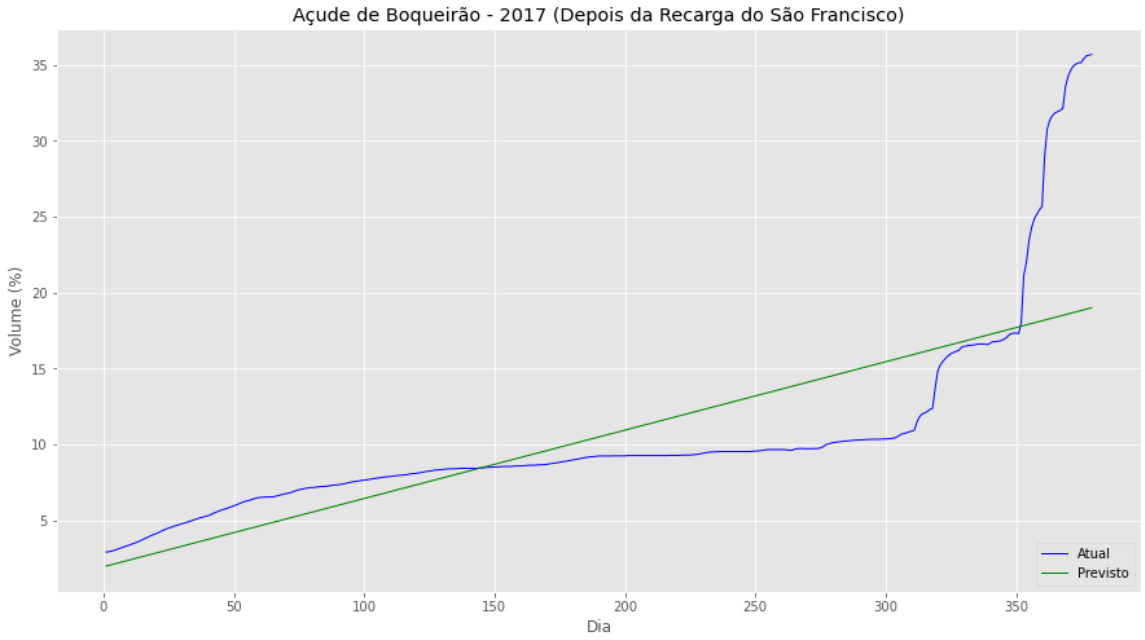
In [24]:

```
matplotlib.style.use("ggplot")
plt.figure(figsize=(15,8))

# Plot both data sets
plt.plot(X, y, color='blue',
         linewidth=1)
plt.plot(X, olm.predict(X), color='green',
         linewidth=1)

# Add decorations
plt.title("Açude de Boqueirão - 2017 (Depois da Recarga do São Francisco)")
plt.xlabel("Dia")
plt.ylabel("Volume (%)")
plt.legend(["Atual", "Previsto",], loc="lower right")
print('reta de regressão')
print(' y = {0} * x + {1}'.format(a, b))
plt.show()
```

```
reta de regressão
 y = 0.04498845398456563 * x + 1.9455974368639293
```



In [25]:

```
list(olm.predict(X))[:3]
```

Out[25]:

```
[1.9905858908484948, 2.0355743448330603, 2.0805627988176263]
```

In [26]:

```
import numpy as np
```

In [27]:

```
numero_de_dias_observados
```

Out[27]:

```
379
```

In [28]:

```
dados3meses = np.array(numero_de_dias_observados + 90)
dados3meses
```

Out[28]:

```
array(469)
```

In [29]:

```
olm.predict(dados3meses.reshape(-1, 1))[0]
```

Out[29]:

```
23.045182355625208
```

In [30]:

```
# predicao para 1 ano
dados = np.array(numero_de_dias_observados + 365).reshape(-1, 1)
pred = olm.predict(dados)[0]
print ("Volume Boqueirão depois de um ano = %6.2f %%" % (pred))
```

```
Volume Boqueirão depois de um ano =  35.42 %
```

In [31]:

```
# predicao para 2 anos
dados = np.array((numero_de_dias_observados+365)*2).reshape(1, -1)
pred = olm.predict(dados)[0]

print ("Volume Boqueirão depois de dois anos = %6.2f %%" % (pred))
```

```
Volume Boqueirão depois de dois anos =  68.89 %
```

In [32]:

```
numero_de_dias_observados
```

Out[32]:

```
379
```

In [33]:

```
# predicao até verter água - sangrar (100%) - Capacidade Máxima do Açude
ano=2018
dados = np.array((numero_de_dias_observados)).reshape(1, -1)
pred = olm.predict(dados)[0]
print ("Volume Boqueirão: Ano %d -> %-5.2f%%" %
       (ano,pred))
dia = 1
while (True):
    x = np.array([(numero_de_dias_observados + dia)]).reshape(1, -1)
    pred = olm.predict(x)[0]
    if pred >= 100:
        break
    if dia % 365 == 0:
        ano = ano +  1
        print ("Volume Boqueirão: Ano %d -> %-5.2f%%" % (ano,pred))
    dia = dia + 1
print ("Volume Boqueirão: Ano %d -> %-5.2f%%" % (ano,pred))
```

```
Volume Boqueirão: Ano 2018 -> 19.00%
Volume Boqueirão: Ano 2019 -> 35.42%
Volume Boqueirão: Ano 2020 -> 51.84%
Volume Boqueirão: Ano 2021 -> 68.26%
Volume Boqueirão: Ano 2022 -> 84.68%
Volume Boqueirão: Ano 2022 -> 100.02%
```

Com os dados de recarga do São Francisco e dados de chuva, representados pelo Volume diário,
O Açude de Boqueirão encherá em 4 anos (2022)