TP DE SERIE TEMPORELLE

REALISE PAR MEKA MOISE CHRISTIAN JUNIOR 21T2561

In [1]:

```
#pip install matplotlib numpy pandas
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
```

In [2]:

```
FILEPATH="Datasets/BTC-EUR.csv"
HORIZON=np.random.randint(1,1000)
FEATURE="Volume"
SEP=","
YLABEL="Volume"
XLABEL="Date"
TITLE="Evolution par jour du volume des BTC"
```

In [3]:

```
df=pd.read_csv(FILEPATH,sep=SEP)
```

In [4]:

```
df.head()
```

Out[4]:

	Date	Open	High	Low	Close	Adj Close	Volume
0	2014-09-17	359.546204	361.468506	351.586884	355.957367	355.957367	16389166
1	2014-09-18	355.588409	355.505402	319.789459	328.539368	328.539368	26691849
2	2014-09-19	328.278503	330.936707	298.921021	307.761139	307.761139	29560103
3	2014-09-20	307.665253	329.978180	303.931244	318.758972	318.758972	28736826
4	2014-09-21	318.120514	321.504517	306.502197	310.632446	310.632446	20702625

In [5]:

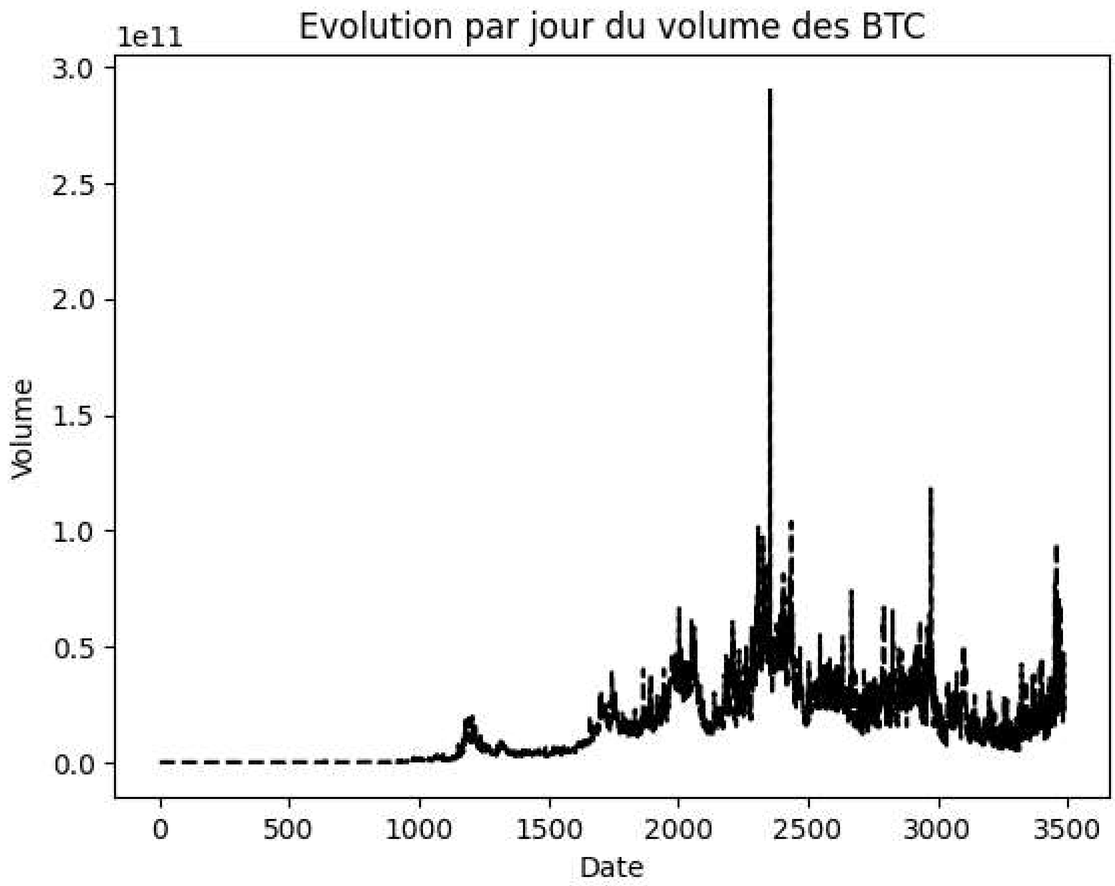
```
all_alpha=[0.001,0.005, 0.01, 0.05, 0.1, 0.3, 0.5, 0.9]
#all_alpha=np.linspace(0.001,0.9,10)
```

In [6]:

```
plt.plot(df[FEATURE], c="black", ls='--')
plt.xlabel(XLABEL)
plt.ylabel(YLABEL)
plt.title(TITLE)
```

Out[6]:

```
Text(0.5, 1.0, 'Evolution par jour du volume des BTC')
```



In [7]:

```
def predict_simple_expo_lissage(data, alpha, taille, horizon=1):
    results=[]
    for i in range(taille-horizon):
        if i==0:
            results.append((1-alpha)*data[i])
        else:
            tmp=(1-alpha)*data[i]+alpha*results[i-1]
            results.append(tmp)
    return results
```

In [8]:

```
alpha=all_alpha[-1]
res=predict_simple_expo_lissage(df[FEATURE], alpha, len(df[FEATURE]))
```

In [9]:

```
all_results={}
for alpha in all_alpha:
    all_results[f"alpha_{alpha}"]=predict_simple_expo_lissage(df[FEATURE], alpha
```

In [10]:

```
def sum_square_error(real, predic):
    result=real-predic
    result=result**2
    return np.sum(result)
```

In [11]:

```
all_error={}
for key in all_results.keys():
    real=np.array(df[FEATURE])
    predic=np.array(all_results[key])
    all_error[key]=sum_square_error(real[:real.shape[0]-HORIZON],predic)
```

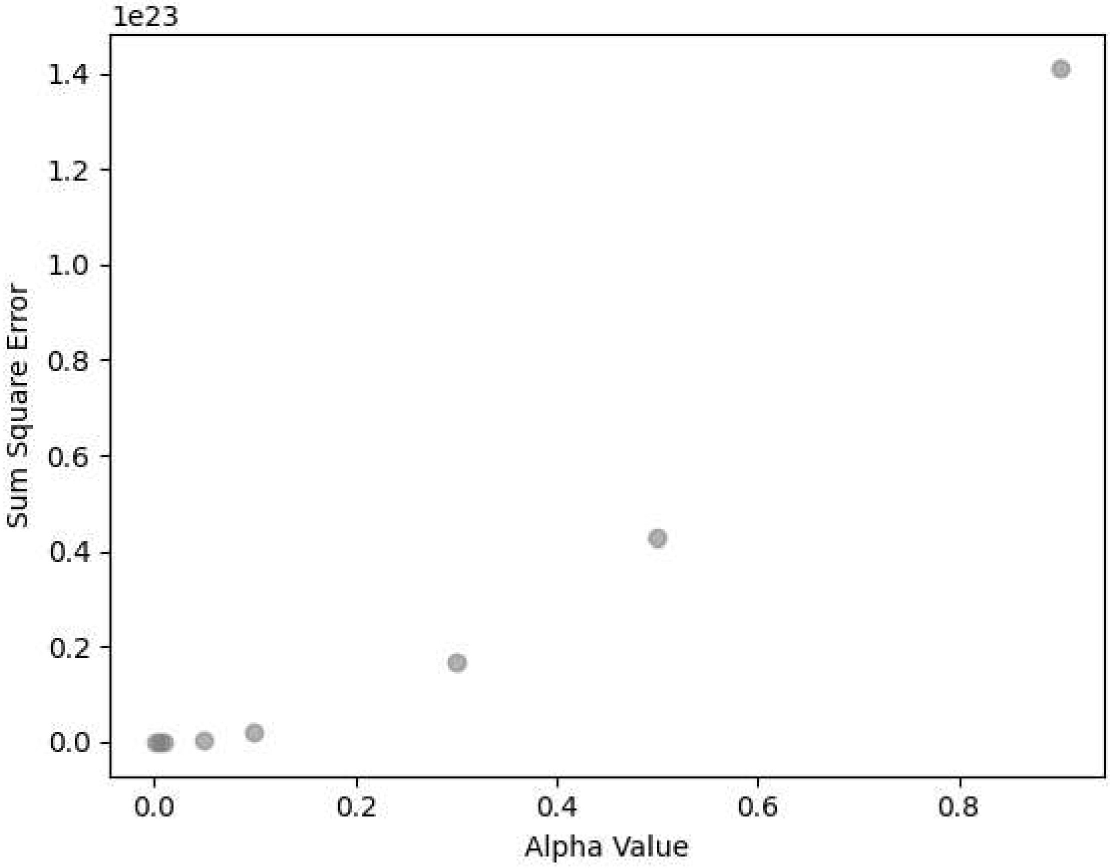
In [12]:

```
list_=[]
for key in all_error.keys():
    list_.append(all_error[key])
```

```
plt.scatter(all_alpha, list_, alpha=0.6, c="gray")
plt.xlabel("Alpha Value")
plt.ylabel("Sum Square Error")
```

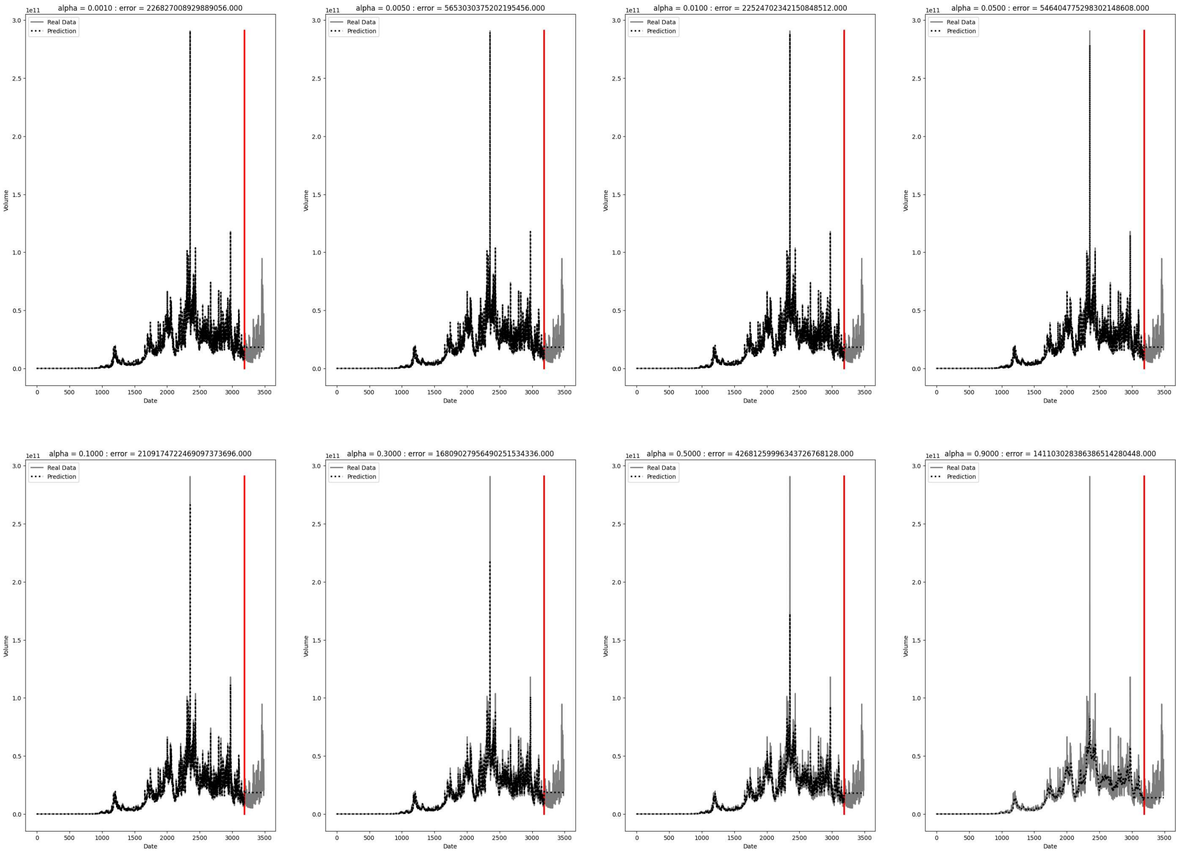
Out[12]:

```
Text(0, 0.5, 'Sum Square Error')
```



In [13]:

```
plt.figure(figsize=[35, 25])
for i in range(len(all_alpha)):
    plt.subplot(2, int(len(all_alpha)/2), i+1)
    plt.plot(df[FEATURE] , label='Real Data', c='gray', lw=2)
    m1=np.max(df[FEATURE])
    m2=np.min(df[FEATURE])
    x=len(df)-HORIZON
    y=np.linspace(m2,m1,100)
    x=np.ones(y.shape)*x
    plt.plot(x,y, c="red", lw=2.7, ls="-")
    last=all_results[f"alpha_{all_alpha[i]}"][-1]
    tmp=all_results[f"alpha_{all_alpha[i]}"]
    for j in range(HORIZON):
        tmp.append(last)
    plt.plot(tmp, label='Prediction', lw=2.5 , ls=':', c='black')
    alpha=f"alpha_{all_alpha[i]}"
    plt.title(f"alpha = {all_alpha[i]:.4f} : error = {all_error[alpha]:.3f}")
    plt.xlabel(XLABEL)
    plt.ylabel(YLABEL)
    plt.legend()
```


alpha = 0.0010 : error = 226827008929889056.000
alpha = 0.0050 : error = 5653030375202195456.000
alpha = 0.0100 : error = 22524702342150848512.000
alpha = 0.0500 : error = 54640477529830214860 8.000
alpha = 0.1000 : error = 210917472246909737 3696.000
alpha = 0.3000 : error = 168090279564902515 34336.000
alpha = 0.5000 : error = 426812599963437267 68128.000
alpha = 0.9000 : error = 141103028386386514 280448.000
1e11
Real Data
Prediction
Volume
Date
0.0
0.5
1.0
1.5
2.0
2.5
3.0
0
500
1000
1500
2000
2500
3000
3500

In [ ]: