QUALITE DE L'AIR DANS LA STATION DE CHATELET

Présenté par Joshua Nijimbere

Objectifs

Comment évolue et évoluera la qualité de l'air dans la station de chatelet

Evolution du CO2

Data

```
df.head(), df.tail()
```

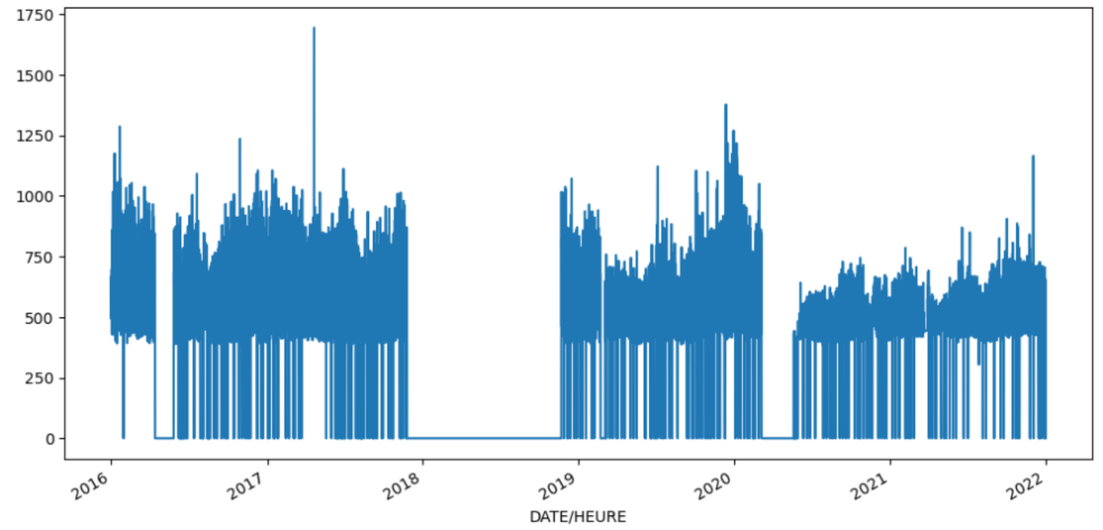
```
(                           NO  NO2  PM10  CO2  TEMP  HUMI
 DATE/HEURE
 2016-01-01 00:00:00+01:00  32   37   139  649  23.1  36.1
 2016-01-01 01:00:00+01:00  16   32   132  575  22.9  35.6
 2016-01-01 02:00:00+01:00  13   25   128  651  23.2  36.3
 2016-01-01 03:00:00+01:00  14   27   125  668  23.4  36.2
 2016-01-01 04:00:00+01:00  17   32    93  652  23.0  36.7,
                            NO  NO2  PM10  CO2  TEMP  HUMI
 DATE/HEURE
 2021-12-31 19:00:00+01:00  17   30    49  628  18.8  64.1
 2021-12-31 20:00:00+01:00  21   33    49  589  18.7  64.7
 2021-12-31 21:00:00+01:00  37   36    51  589  18.9  64.4
 2021-12-31 22:00:00+01:00  18   28    39  526  18.5  65.0
 2021-12-31 23:00:00+01:00   9   23    31  519  19.1  62.1)
```

```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Index: 52391 entries, 2016-01-01 00:00:00+01:00 to 2021-12-31 23:00:00+01:00
Data columns (total 6 columns):
 #   Column  Non-Null Count  Dtype
---  ------  --------------  -----
 0   NO      52391 non-null  int64
 1   NO2     52391 non-null  int64
 2   PM10    52391 non-null  int64
 3   CO2     52391 non-null  int64
 4   TEMP    52391 non-null  float64
 5   HUMI    52391 non-null  float64
dtypes: float64(2), int64(4)
memory usage: 2.8+ MB
```

```
df['CO2'].plot(figsize=(12,6))
```

```
<AxesSubplot:xlabel='DATE/HEURE'>
```



Réseaux de neurones

Long Short-Term Memory network (LSTM)

```
results= seasonal_decompose(df['CO2'], extrapolate_trend='freq', period=1)
results.plot();
```

```
scaler= MinMaxScaler()
```

```
train_data = df2.iloc[:-8760]
test_data = df2.iloc[-8760:]
train_data.info()

scaler.fit(train_data)
scaled_train= scaler.transform(train_data)
scaled_test= scaler.transform(test_data)
```

```
<class 'pandas.core.frame.DataFrame'>
Index: 43631 entries, 2016-01-01 00:00:00+01
Data columns (total 1 columns):
 #   Column  Non-Null Count  Dtype
---  ------  --------------  -----
 0   CO2     43631 non-null  int64
dtypes: int64(1)
memory usage: 681.7+ KB
```

```
n_input= 8760
n_features = 1
generator = TimeseriesGenerator(scaled_train, scaled_train, length=n_input, batch_size=1)
```

```
#definition du model
model = Sequential()
model.add(LSTM(1000,input_shape=(n_input,n_features)))
model.add(Dense(1))
#model.compile()
model.compile(optimizer='adam', loss='mse')
```

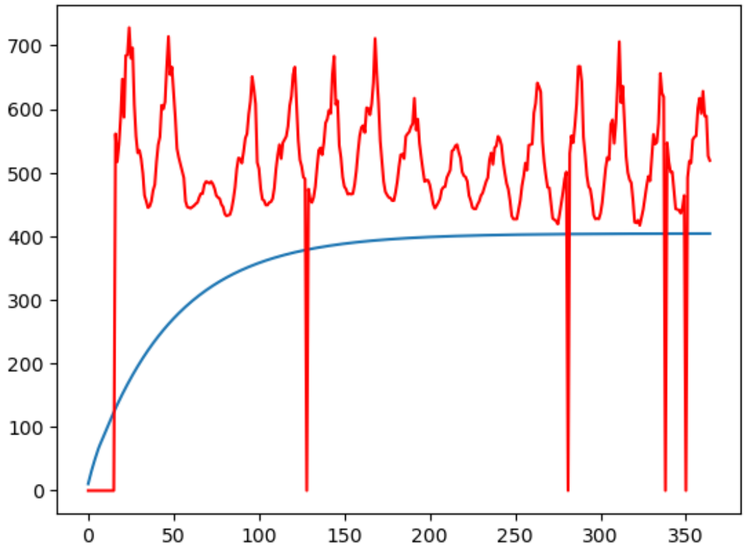
```
model.summary()
```

```
predictions_tests=[]

premier_groupe = scaled_train[-n_input:]
groupe_actuel= premier_groupe.reshape((1, n_input, n_features))

for i in range(len(test_data)):
    #prediction du premier groupe
    groupe_actuel= model.predict(groupe_actuel)[0]

    #ajouter la prediction dans la liste
    predictions_tests.append(groupe_actuel)

    #on utilise la prediction pour re-initialiser le groupe et enlever la premiere valeur
    groupe_actuel= np.append(groupe_actuel[:,1:,:], [[groupe_actuel]], axis=1)
```

Modèle Autoregréssion

```
dftest = adfuller(df['CO2'], autolag = "AIC")
print('ADF:', dftest[0])
print('P-Value:', dftest[1])
print("Num of lags:", dftest[2])
print('Num of observation used for ADF regression and critical values calculation:', dftest[3])
print('Critical Values:')
for key, val in dftest[4].items():
    print("\t", key, ": ", val)
```

```
ADF: -7.2598211499700325
P-Value: 1.6930124032892653e-10
Num of lags: 58
Num of observation used for ADF regression and critical values calculation: 52332
Critical Values:
	1% :  -3.43047496409059
	5% :  -2.8615952316158593
	10% :  -2.5667993979530026
```




```
print(rmse)
```

```
184.10671604570706
```

```
model= AutoReg(train, lags=10).fit()
print(model.summary())
```

```
                            AutoReg Model Results
==============================================================================
Dep. Variable:                      y   No. Observations:                52026
Model:                     AutoReg(10)   Log Likelihood             -290800.251
Method:               Conditional MLE   S.D. of innovations             64.823
Date:                Mon, 07 Nov 2022   AIC                        581624.502
Time:                        01:05:33   BIC                        581730.814
Sample:                            10   HQIC                       581657.743
                                52026
==============================================================================
                 coef    std err          z      P>|z|      [0.025      0.975]
------------------------------------------------------------------------------
const         10.6949      0.551     19.399      0.000       9.614      11.776
y.L1           0.9901      0.004    225.957      0.000       0.982       0.999
y.L2          -0.0185      0.006     -3.000      0.003      -0.031      -0.006
y.L3          -0.0644      0.006    -10.470      0.000      -0.077      -0.052
y.L4           0.0401      0.006      6.508      0.000       0.028       0.052
y.L5           0.0212      0.006      3.443      0.001       0.009       0.033
y.L6          -0.0313      0.006     -5.078      0.000      -0.043      -0.019
y.L7          -0.0611      0.006     -9.927      0.000      -0.073      -0.049
y.L8           0.0483      0.006      7.845      0.000       0.036       0.060
y.L9           0.0837      0.006     13.596      0.000       0.072       0.096
y.L10         -0.0346      0.004     -7.888      0.000      -0.043      -0.026
                                    Roots
==============================================================================
                  Real          Imaginary           Modulus         Frequency
```

Modèle ARIMA

```
stepwise_fit = auto_arima(df['CO2'], trace=True)
stepwise_fit.summary()
```

```
Best model:  ARIMA(4,1,4)(0,0,0)[0]
Total fit time: 75.720 seconds
```

```
model=ARIMA(train_data, order=(4, 1, 4))
model=model.fit()
model.summary()
```

```
rmse2=sqrt(mean_squared_error(pred1, test_data))
print(rmse2)
```

```
90.18592502542586
```

```
start=len(train_data)
end=len(train_data)+len(test_data)-1
pred1=model.predict(start=len(train_data), end=end,type='levels')
print(pred)
```

```
2022-12-01    526.970807
2022-12-02    515.088035
2022-12-03    523.060057
2022-12-04    513.066241
2022-12-05    520.624110
2022-12-06    510.941785
2022-12-07    517.723829
2022-12-08    509.442593
2022-12-09    516.390771
2022-12-10    508.706318
2022-12-11    514.557527
2022-12-12    507.422905
2022-12-13    513.438352
2022-12-14    507.365810
```

Enseignements

Merci !