Time Series Forecasting

Développez un outil de prévision de la consommation d'énergie

OBJECTIF

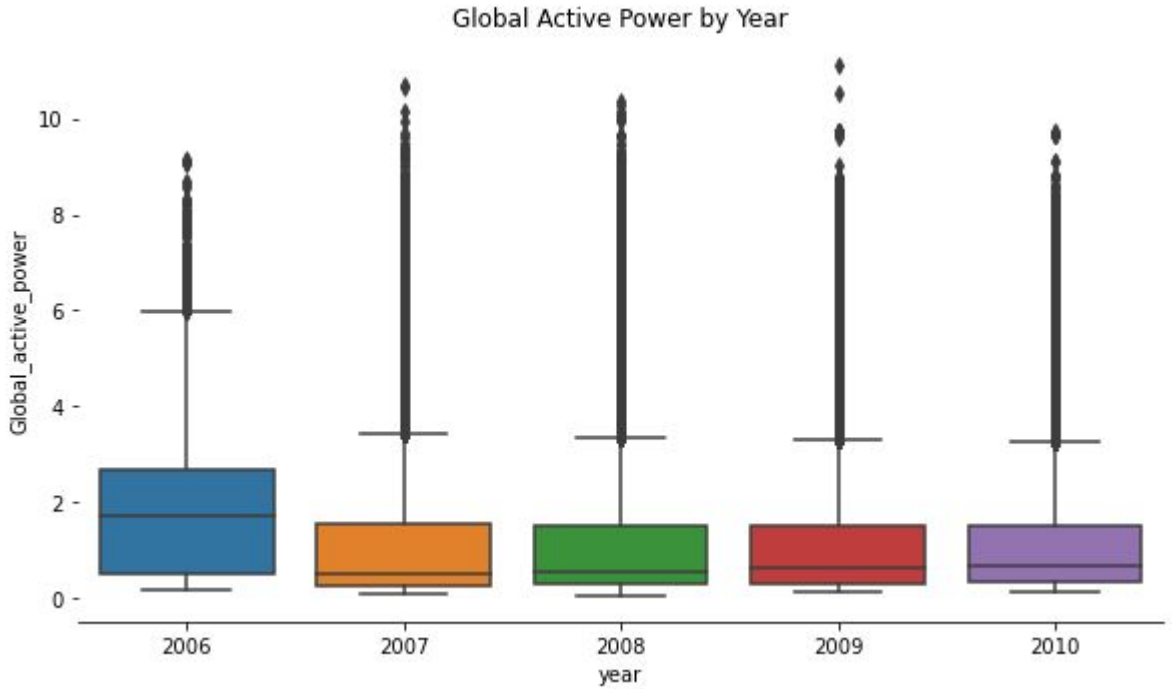
Créer un outil de prédiction de la consommation d'énergie de nos clients et anticiper les besoins du marché en utilisant des réseaux neuronaux récurrents

Le dataset

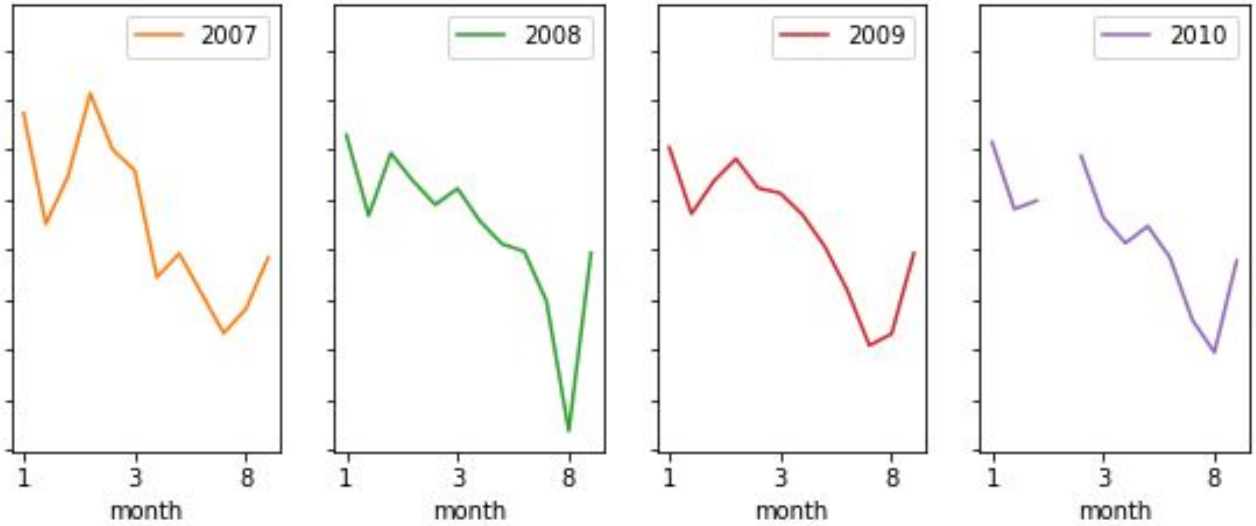
Ce dataset contient la consommation électrique d'un seul foyer par minute, sur une période de 4 ans, soit 3 millions de valeurs entre décembre 2006 , et novembre 2010.

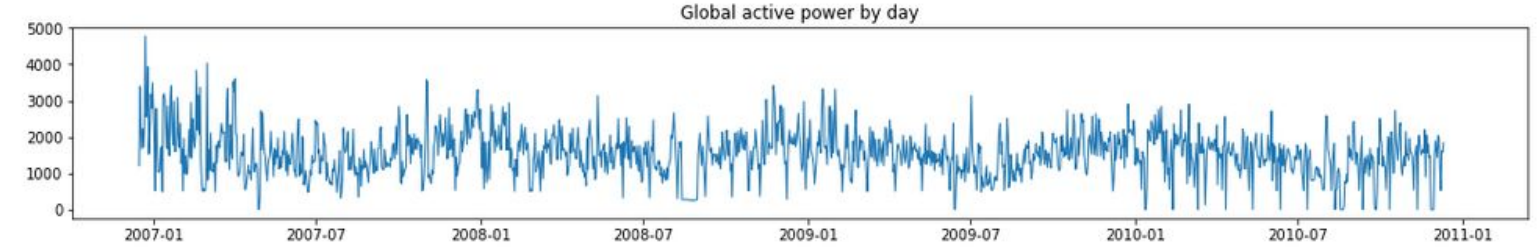
Il y a différentes mesures électriques (volt, ampère, kilowatt) mais nous restons concentré sur notre cible à prédire : la colonne "global activy power" qui elle, est en kilowatt.

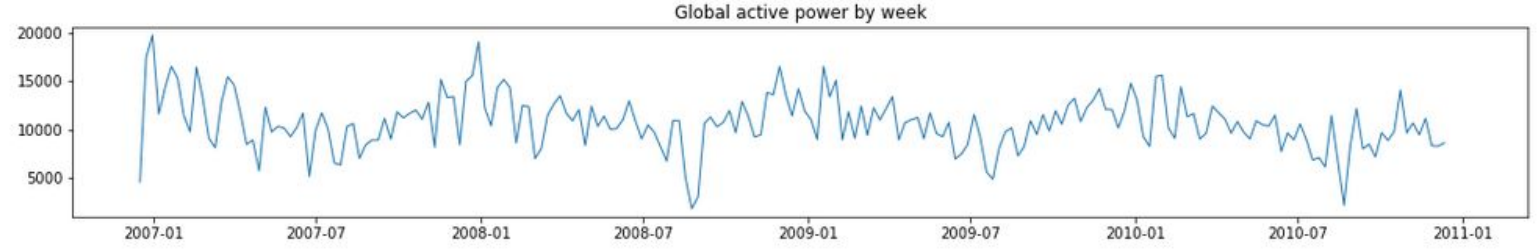
Notre cible : Global active power

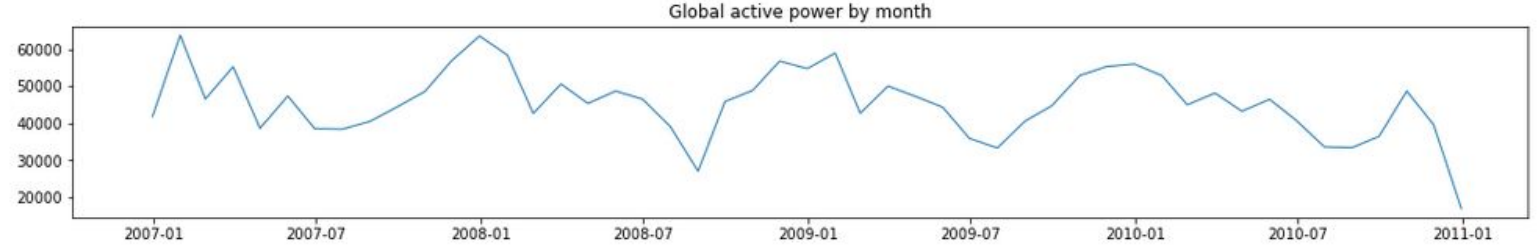


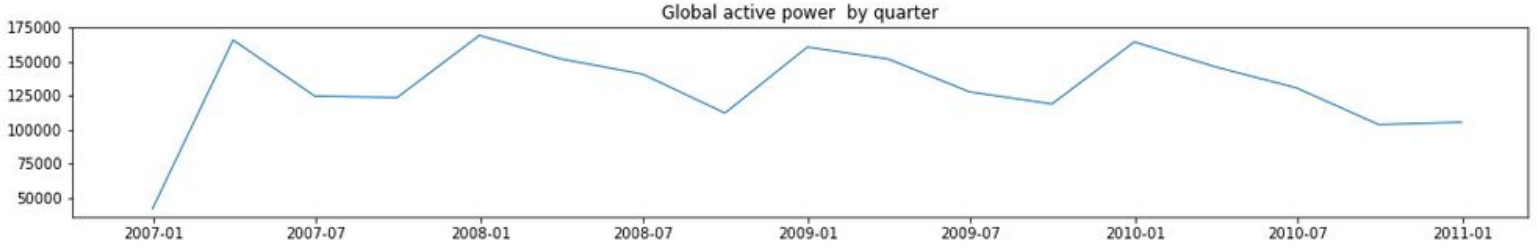
Global activity power par années




Global active power by day
5000
4000
3000
2000
1000
0
2007-01
2007-07
2008-01
2008-07
2009-01
2009-07
2010-01
2010-07
2011-01


Global active power by week
20000
15000
10000
5000
2007-01
2007-07
2008-01
2008-07
2009-01
2009-07
2010-01
2010-07
2011-01


Global active power by month
60000
50000
40000
30000
20000
2007-01
2007-07
2008-01
2008-07
2009-01
2009-07
2010-01
2010-07
2011-01


Global active power by quarter
175000
150000
125000
100000
75000
50000
2007-01
2007-07
2008-01
2008-07
2009-01
2009-07
2010-01
2010-07
2011-01

Les différentes étapes pour modéliser une série temporelle

- Set index sur le datetime
- Split le dataset (train / test)
- Normaliser les features
- Convertir les valeurs en matrice
- Reshape en X=t and Y=t+1.
- Reshape input en 3D : (sample size, time steps, features)

Résultats obtenus avec LSTM

- Resample par jour
- Variation de la sequence length
- Variation du Batch size
- Test en resample par heure

```
model = Sequential()
model.add(Input(shape=(sequence_length, num_features)))
model.add(LSTM(14, return_sequences=False, activation='tanh'))
model.add(Dense(1, activation='linear'))
```

```
sampling_rate = 1
sequence_length = 7
stride = 1
batch_size = 252
shuffle = False
```

loss: 0.4181 - mae: 0.4785

```
sampling_rate = 1
sequence_length = 14
stride = 1
batch_size = 252
shuffle = False
```

loss: 0.3659 - mae: 0.4479

```
sampling_rate = 1
sequence_length = 28
stride = 1
batch_size = 252
shuffle = False
```

loss: 0.3862 - mae: 0.4577

Variation Batch size

```
sampling_rate = 1
sequence_length = 14
stride = 1
batch_size = 32
shuffle = False
```

loss: 0.3513 - mae: 0.4267

```
sampling_rate = 1
sequence_length = 14
stride = 1
batch_size = 64
shuffle = False
```

loss: 0.3476 - mae: 0.4232

```
sampling_rate = 1
sequence_length = 14
stride = 1
batch_size = 128
shuffle = False
```

loss: 0.3559 - mae: 0.4202

```
sampling_rate = 1
sequence_length = 14
stride = 1
batch_size = 256
shuffle = False
```

loss: 0.3701 - mae: 0.4405

```
sampling_rate = 1
sequence_length = 14
stride = 1
batch_size = 512
shuffle = False
```

loss: 0.3520 - mae: 0.4268

Modification LSTM

```
model = Sequential()
model.add(Input(shape=(sequence_length, num_features)))
model.add(LSTM(28, return_sequences=False, activation='tanh'))
model.add(Dense(1, activation='linear'))
```

```
loss: 0.3377 - mae: 0.4150
```

Resample par heure

```
sampling_rate = 1
sequence_length = 24
stride = 1
batch_size = 512
shuffle = False
```

```
loss: 0.2870 - mae: 0.3797
```

Conclusion

- Meilleurs résultats en resampling par heure
- Difficultés de compréhension du preprocessing pour les time series en DL
- Next step : Continuer les tests avec LSTM et GRU, améliorer le modèle en jouant sur le preprocessing et les hyperparamètres