Prévoir la consommation électrique pour favoriser une distribution énergétique optimale.

Eloi Kling - Téodore Autuly

Problématique

Contexte

Dataset

Consommation électrique aux Etats-Unis

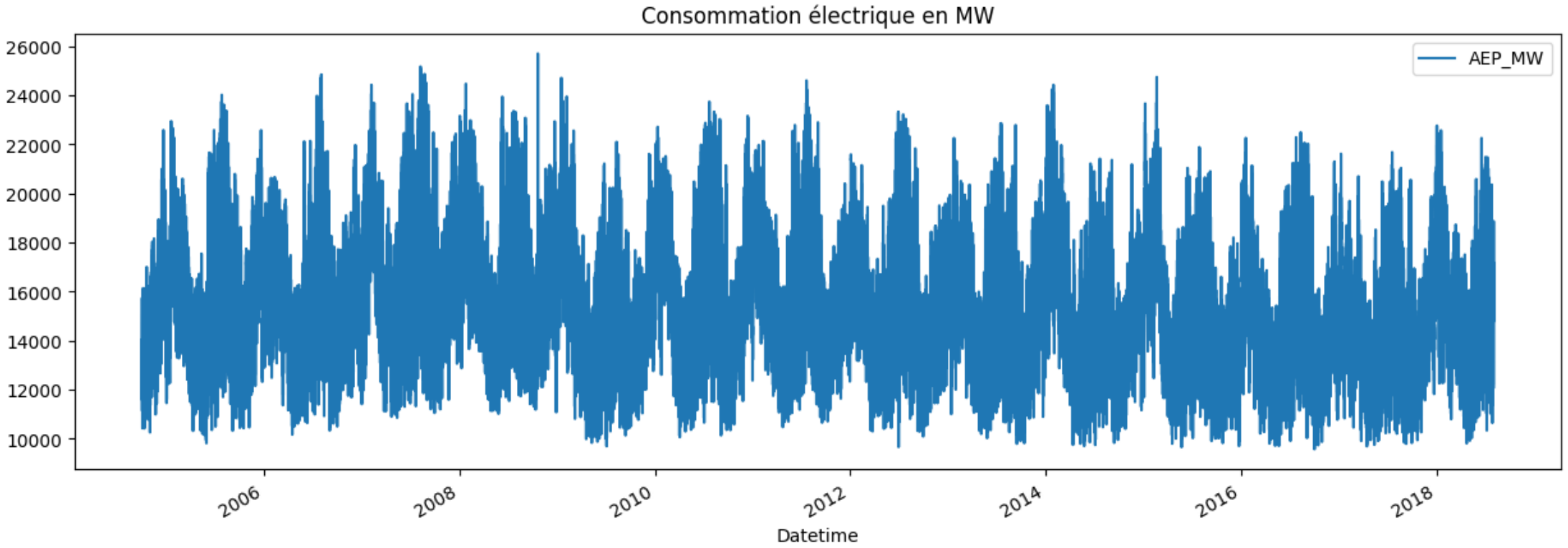
```
dataset.info()
dataset.shape
```

✓ 0.1s

```
<class 'pandas.core.frame.DataFrame'>
DatetimeIndex: 121273 entries, 2004-12-31 01:00:00 to 2018-01-02 00:00:00
Data columns (total 1 columns):
 #   Column  Non-Null Count   Dtype  
---  ------  --------------   -----  
 0   AEP_MW  121273 non-null  float64
dtypes: float64(1)
memory usage: 1.9 MB

(121273, 1)
```

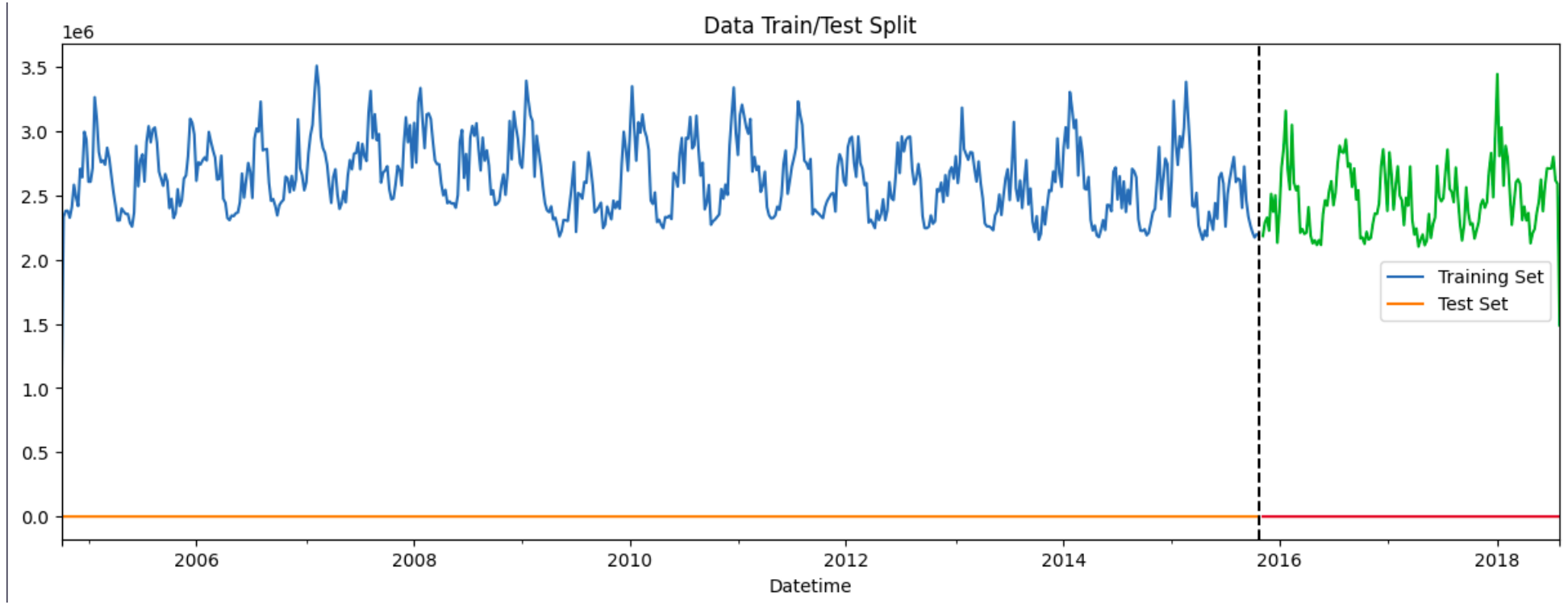
- 12 ans de data
- 31 décembre 2004 au 2 janvier 2018
- Type: float
- Fréquence: heure par heure
- 2 colonnes
- 121273 lignes

Dataset


Consommation électrique en MW
AEP_MW
26000
24000
22000
20000
18000
16000
14000
12000
10000
2006
2008
2010
2012
2014
2016
2018
Datetime

Split train/test

```
# split train et test
nb_lines_w = weekly_data.shape[0]
train_w = weekly_data.iloc[:int(nb_lines_w*0.8)]
test_w = weekly_data.iloc[int(nb_lines_w*0.8)+1:]
```



Modèle statistique

```
# Groupes pour le resampling
daily_groups = dataset.resample('D')
weekly_groups = dataset.resample('W')

# Jeux de données resamplés
daily_data = daily_groups.sum()
weekly_data = weekly_groups.sum()
```

Modèle statistique

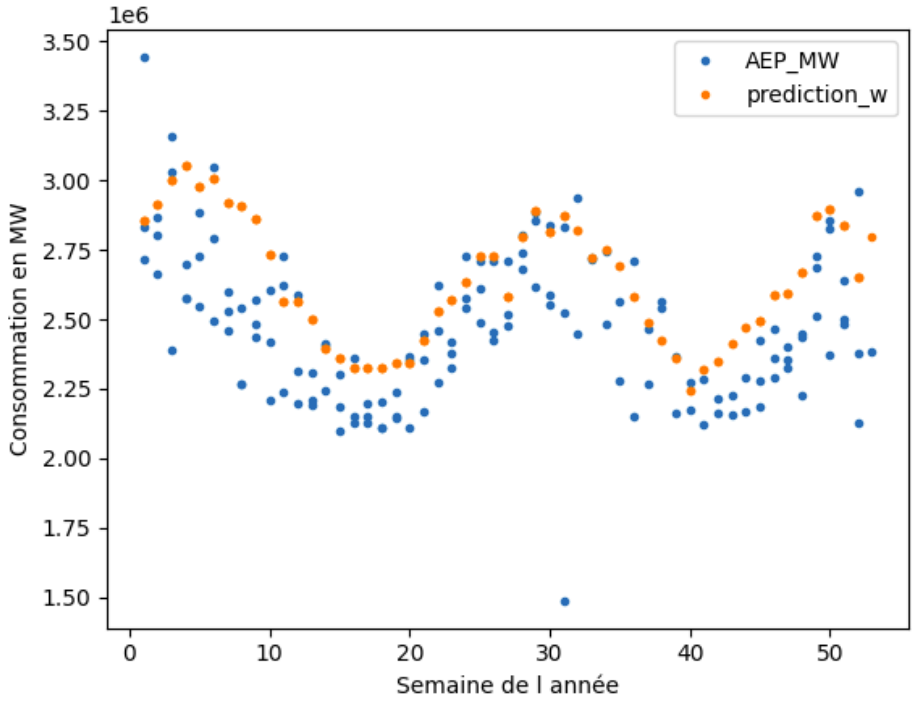
Consommation par semaine

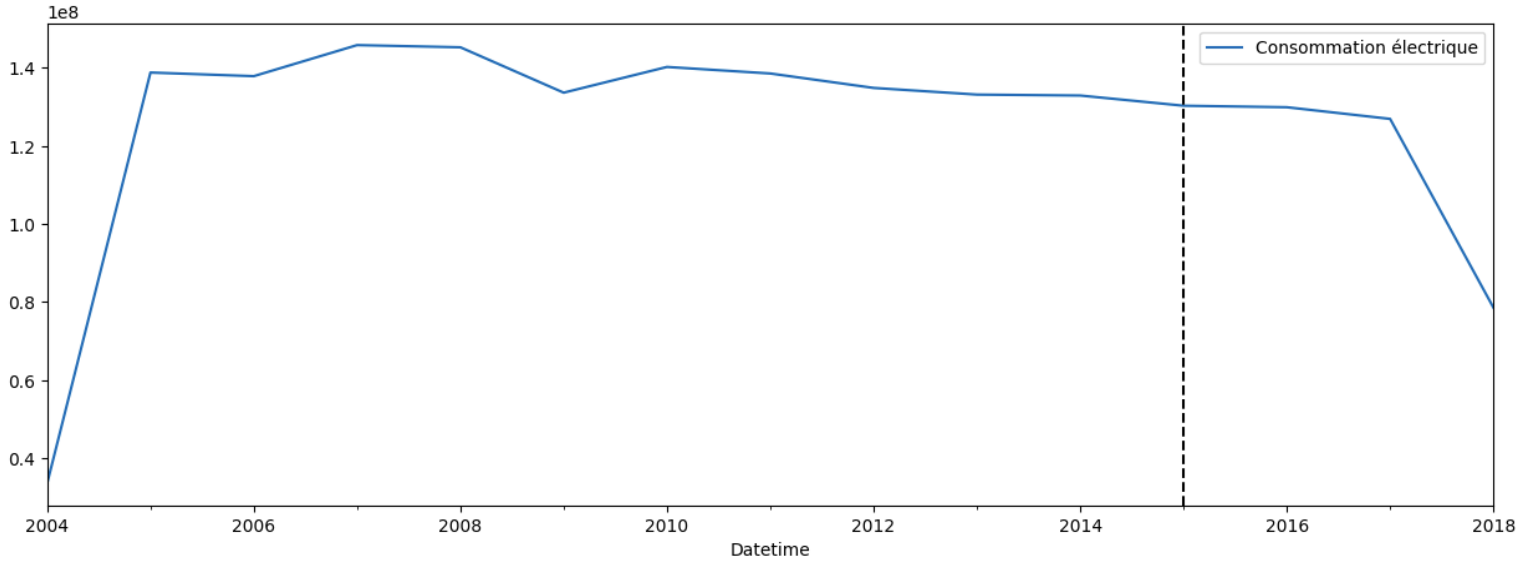
```
#moyenne de la consommation des années précédentes
train_model_week = train_w.groupby(["week_of_year"]).mean()
train_model_week = train_model_week.rename(columns={"AEP_MW":"prediction_w"})
```

```
#renvoie les prédictions dans une colonne
def predict (df,model):
    return df.merge(model, on="week_of_year",how="left")
test_predictions_week = predict(test_w,train_model_week)
```

Modèle statistique

Consommation par semaine



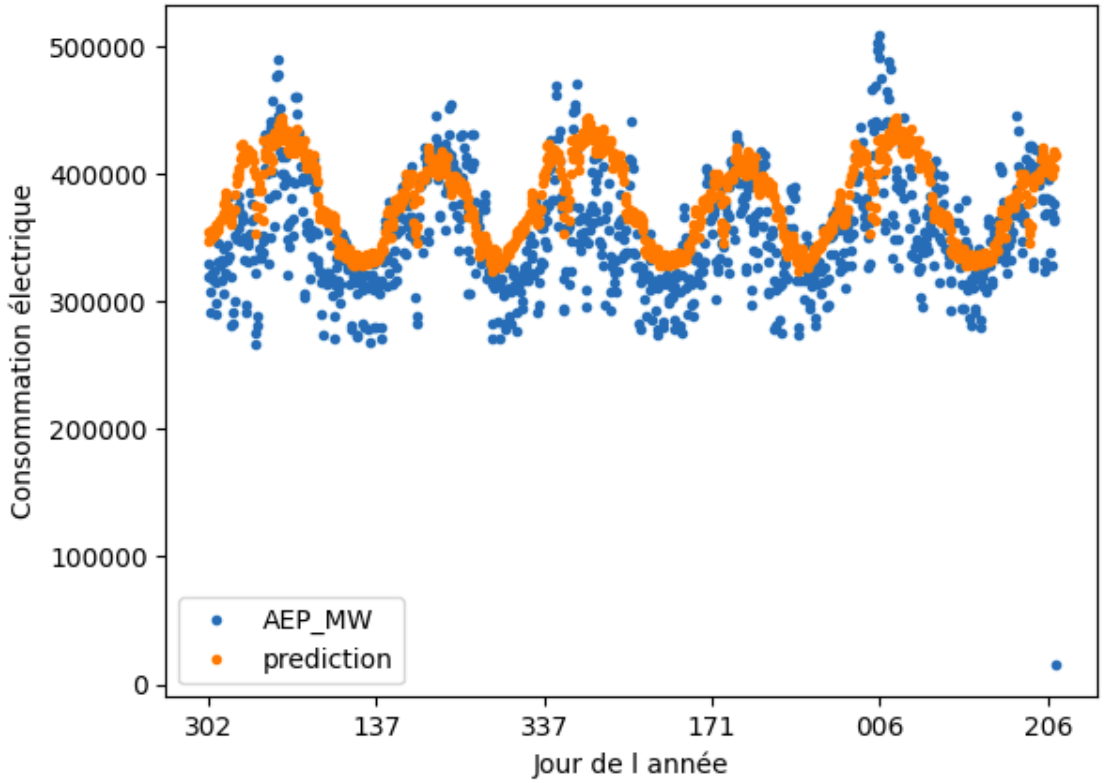


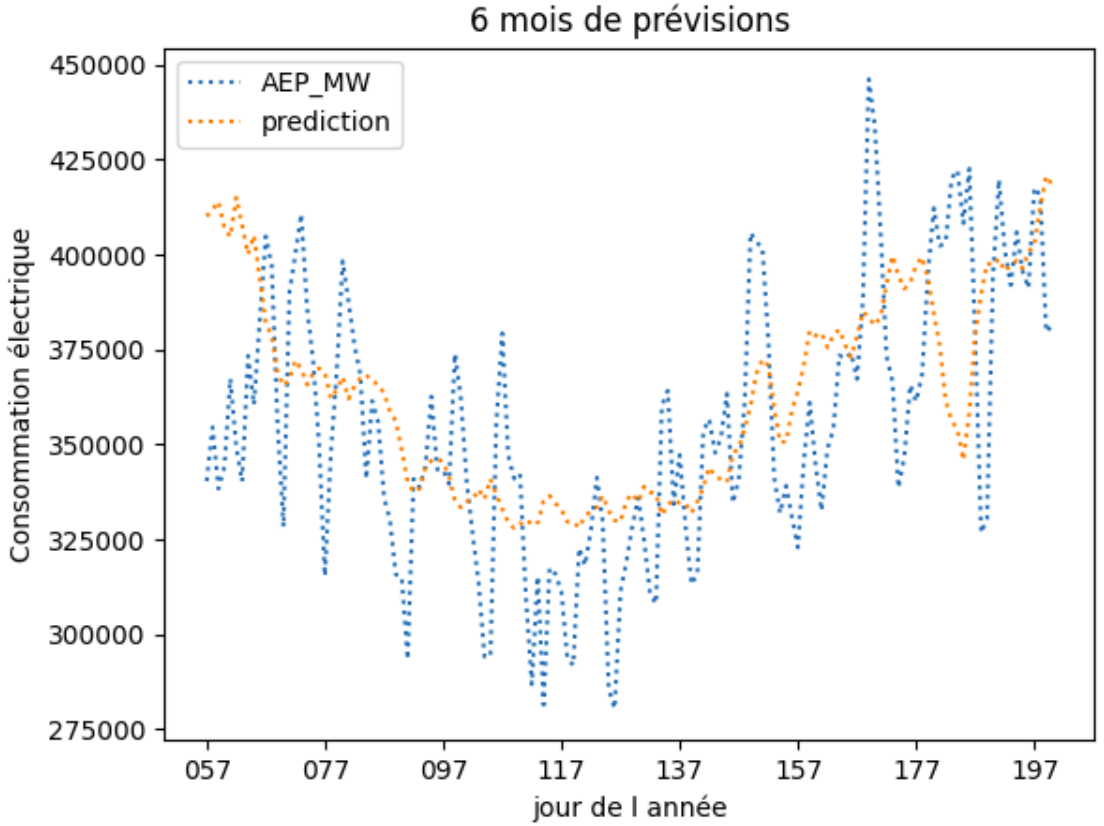

```
#evaluation des prédictions
print("RMSE %s" %mean_squared_error(test_predictions_week["AEP_MW"],test_predictions_week["prediction_w"],squared= False))
print("MAE %s" %mean_absolute_error(test_predictions_week["AEP_MW"],test_predictions_week["prediction_w"]))
```

```
RMSE 279583.5628271335
MAE 214121.01388888888
```

Modèle statistique

Consommation par jour






```
RMSE 45819.38893986183
MSE 36082.749579958
```

Ainsi notre modèle se trompe en moyenne de 45 819 MW par jour, par rapport à une consommation electrique moyenne par jour de 371 844 MW . (Soit une erreur de 14%)

Modèle statistique

Modèle de machine learning

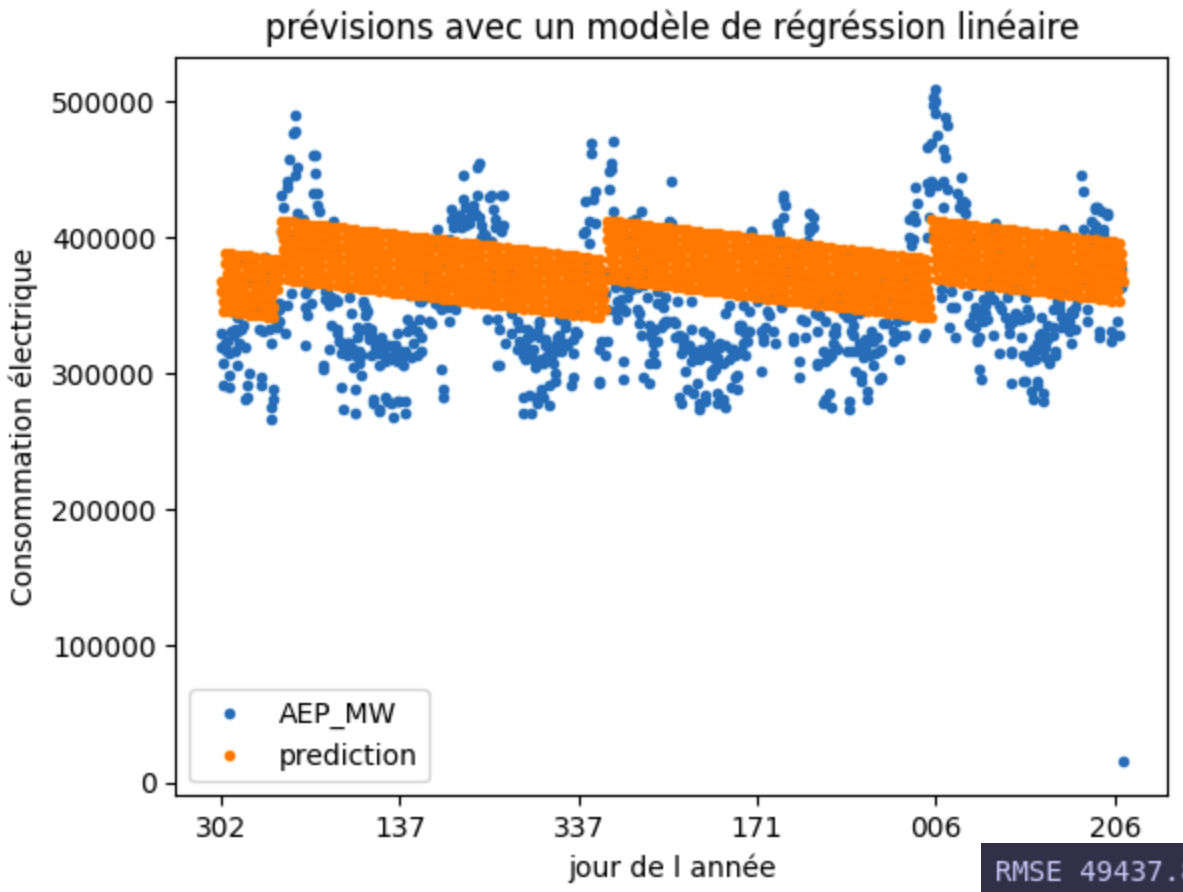
Modèle de machine learning

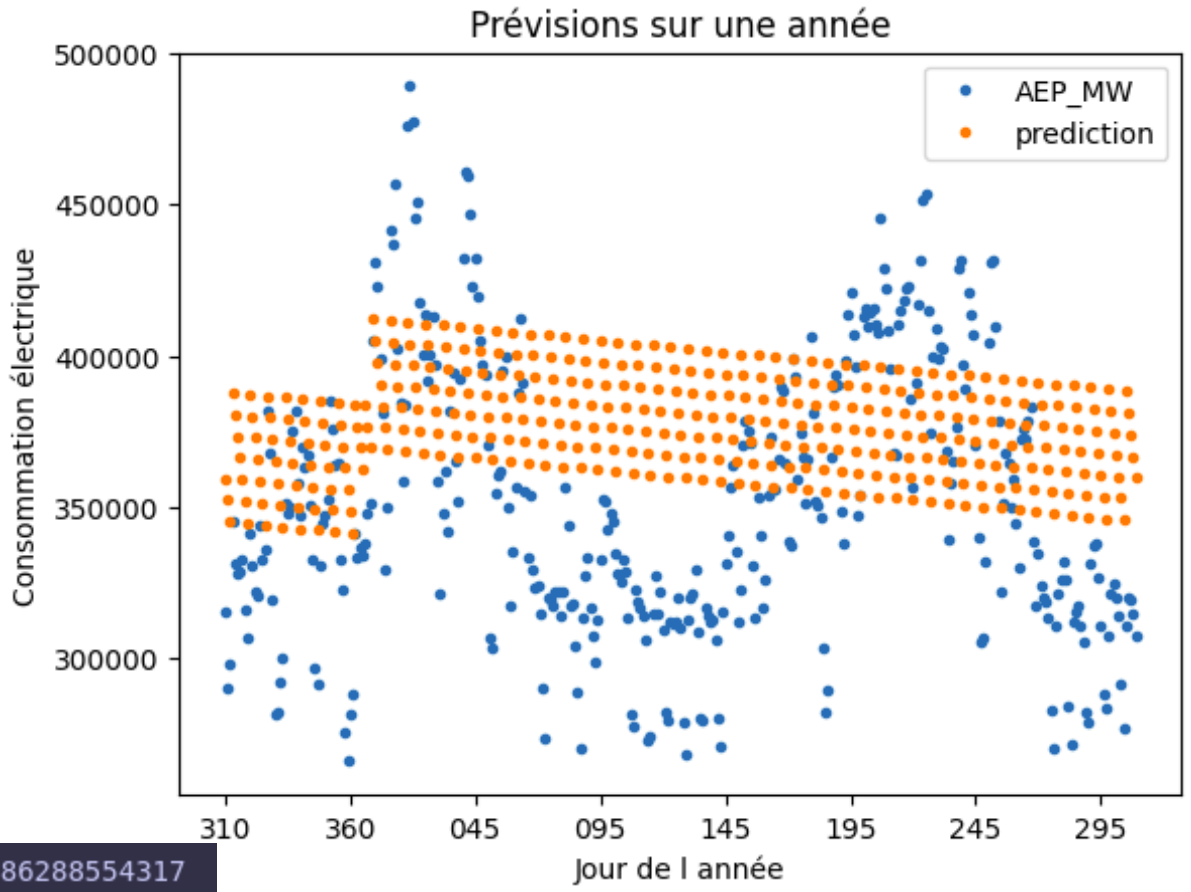
Modèle linéaire

```
# fit du modèle
test_predictions_l = test
val = train[["day_of_year", "day_of_week"]].values
model = LinearRegression().fit(val, train["AEP_MW"].values)
model.fit(val, train["AEP_MW"].values)
test_predictions_l["prediction"] = model.predict(test[["day_of_year", "day_of_week"]].values)
```

Modèle de machine learning

Modèle linéaire






```
RMSE 49437.86288554317
MSE 41066.621305642686
```

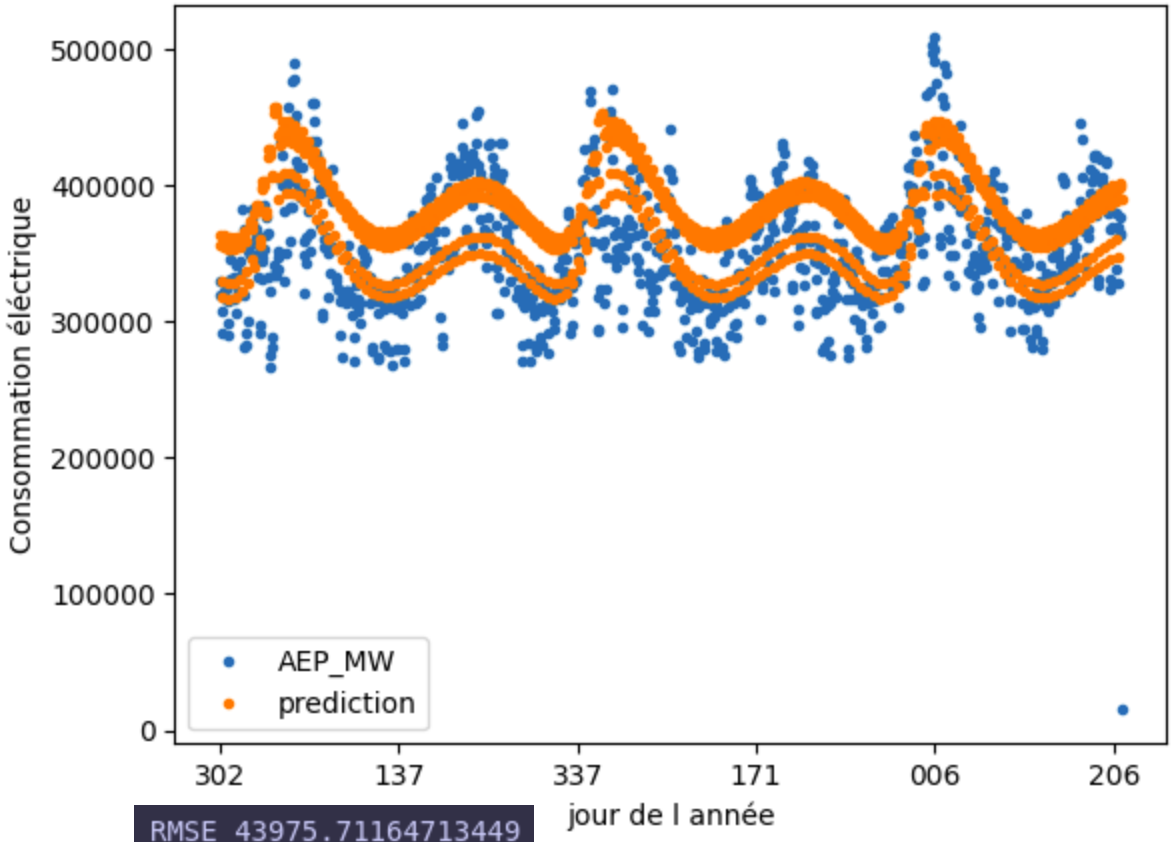
Modèle de machine learning

Modèle Polynomiale de degré 5

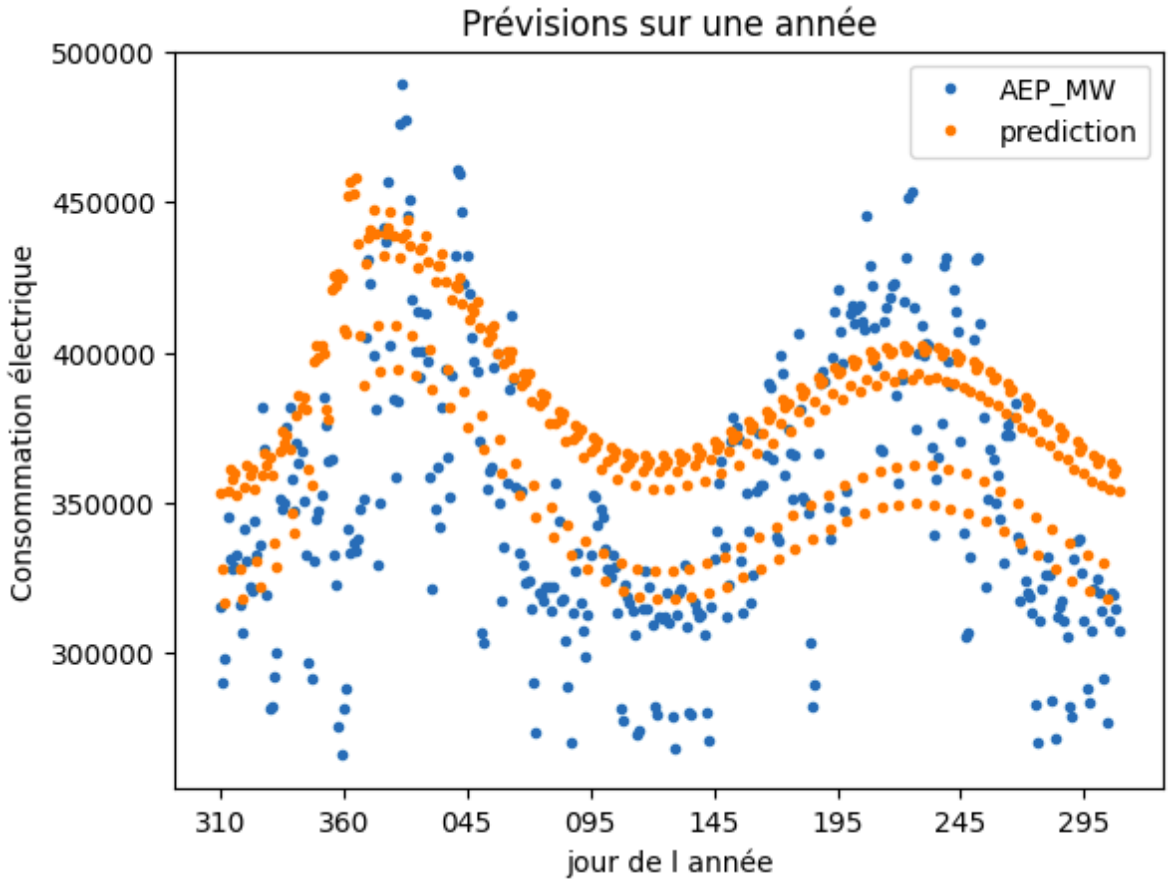
```
# fit du modèle de degré 5
test_predictions_p = test

X = train[["day_of_year", "day_of_week"]].values

model = Pipeline([('poly', PolynomialFeatures(degree=5)),
                  ('linear', LinearRegression(fit_intercept=True))])
model.fit(X, train["AEP_MW"].values)
test_predictions_p["prediction"] = model.predict(test[["day_of_year", "day_of_week"]].values)
```

Modèle de machine learning

Modèle Polynomiale de degré 5

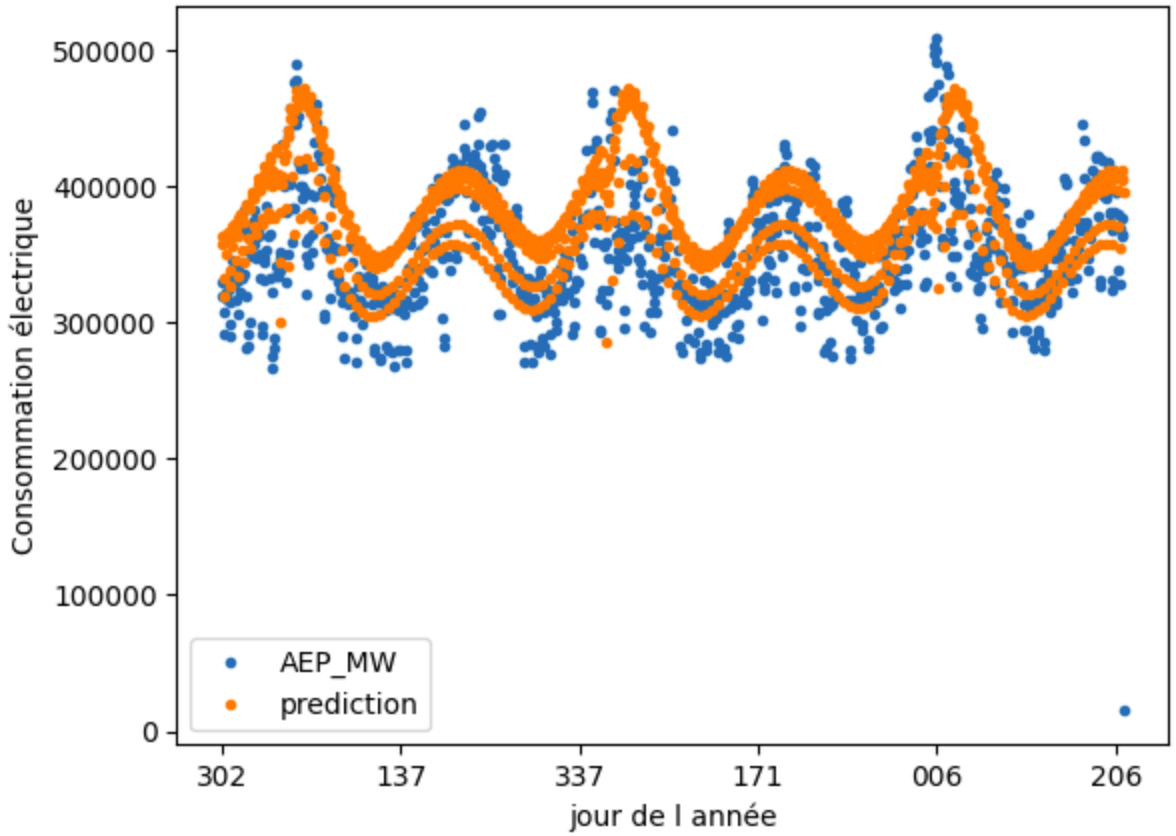


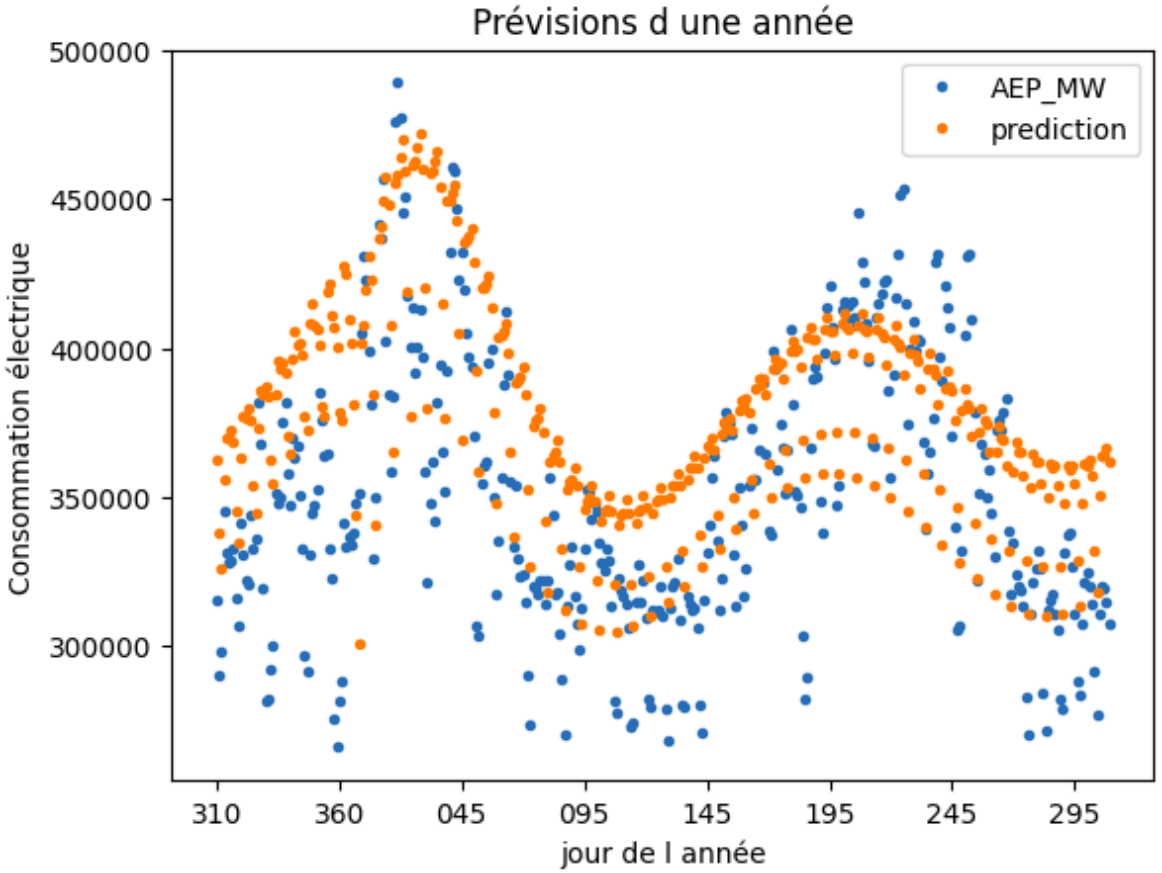

```
RMSE 43975.71164713449
MSE 34961.37300042017
```



Modèle de machine learning

Modèle Polynomiale de degré 6






```
RMSE 44714.75763384663
MSE 35070.344371712876
```

Nous avons affaire ici à un sur-apprentissage: à un degré supplémentaire l'algorithme surapprends sur les particularités de chaque donnée, et donc ses prédictions sont moins précises