Tp3

May 27, 2024

1 - Charger les dataset

```
[ ]: import pandas as pd;
import numpy as np;
import matplotlib.pyplot as plt;

filepath_1 = "dataset/monthly-beer-production-in-austr.csv"
# filepath_2 = "dataset/BTC-EUR.csv"
filepath_2 = "dataset/AirPassengers.csv"
filepath_3 = "dataset/weather_data_kolkata_2015_2020.csv"
filepath_4 = "dataset/monthly-sunspots.csv"
filepath_5 = "dataset/Electric_Production.csv"

ds_1 = pd.read_csv(filepath_1)
ds_2 = pd.read_csv(filepath_2)
ds_3 = pd.read_csv(filepath_3, nrows=500)
ds_4 = pd.read_csv(filepath_4, nrows=500)
ds_5 = pd.read_csv(filepath_5)
```

2 - Division des donnees

- 90% pour l'entrainement
  - 10% pour le test

```
[ ]: def seed(ds, percentage=90):
    return ds[:(percentage*len(ds)//100)]

train_set_1 = seed(ds_1)
test_set_1 = seed(ds_1, 10)
train_set_2 = seed(ds_2)
test_set_2 = seed(ds_2, 10)
train_set_3 = seed(ds_3)
test_set_3 = seed(ds_3, 10)
train_set_4 = seed(ds_4)
test_set_4 = seed(ds_4, 10)
train_set_5 = seed(ds_5)
test_set_5 = seed(ds_5, 10)
```

```
used_train_set_att = {'Monthly beer production': train_set_1, 'Airpass':␣
↪train_set_2, 'TEMPERATURE': train_set_3, 'Sunspots': train_set_4,␣
↪'IPG2211A2N': train_set_5, }
used_test_set_att = {'Monthly beer production': test_set_1, 'Airpass':␣
↪test_set_2, 'TEMPERATURE': test_set_3, 'Sunspots': test_set_4, 'IPG2211A2N':␣
↪test_set_5, }

alpha_values = [0.5, 0.1, 0.9]
beta_values = gamma_values = alpha_values
```

A) Lissage simple

```
[ ]: def simple_exponential_smoothing(alpha):
         result_SEM = {}
         for ds in used_train_set_att:
             n = len(used_train_set_att[ds][ds])
             result = [used_train_set_att[ds][ds][0]]
             for i in range(1, n):
                 result.append((1 - alpha)*used_train_set_att[ds][ds][i] +␣
     ↪(alpha*result[i - 1]))

             result_SEM[ds] = result
         return result_SEM;
     # print(f"{len(used_train_set_att['Airpass']['Airpass'])},␣
     ↪{len(simple_exponential_smoothing(0.1)['Airpass'])}")
```

B) Lissage exponentiel double

```
[ ]: def double_exponential_smoothing(alpha, beta):
         results_DEM = {}
         for ds in used_train_set_att:
             n = len(used_train_set_att[ds][ds])
             level = [used_train_set_att[ds][ds][0]]
             trend = [used_train_set_att[ds][ds][1] - used_train_set_att[ds][ds][0]]
             result = [used_train_set_att[ds][ds][0]]

             for t in range(1, n):
                 level.append(alpha*used_train_set_att[ds][ds][t] + (1 -␣
     ↪alpha)*(level[t - 1] + trend[t - 1]))
                 trend.append(beta*(level[t] - level[t - 1]) + (1 - beta)*trend[t -␣
     ↪1])
                 result.append(level[t] + trend[t])

             results_DEM[ds] = result
         return results_DEM
     # print(f"{len(used_train_set_att['Airpass']['Airpass'])},␣
     ↪{len(double_exponential_smoothing(0.1, 0.1)['Airpass'])}")
```

C) Holt-Winters non saisonnier

[ ]:
```
def holt_winters_non_seasonal(alpha, beta):
    results_HWNS = {}
    for ds in used_train_set_att:
        n = len(used_train_set_att[ds][ds])
        level = [used_train_set_att[ds][ds][0]]
        trend = [used_train_set_att[ds][ds][1] - used_train_set_att[ds][ds][0]]
        result = [used_train_set_att[ds][ds][0]]

        for t in range(1, n):
            level.append(alpha*used_train_set_att[ds][ds][t] + (1 - alpha)*(level[t - 1] + trend[t - 1]))
            trend.append(beta*(level[t] - level[t - 1]) + (1 - beta)*trend[t - 1])
            result.append(level[t] + trend[t])

        results_HWNS[ds] = result
    return results_HWNS
```

D) Holt-Winters saisonnier additif

[ ]:
```
def holt_winters_additive(alpha, beta, gamma, periods):
    results_HWSA = {}
    for ds in used_train_set_att:
        n = len(used_train_set_att[ds][ds])
        level = [used_train_set_att[ds][ds][0]]
        trend = [used_train_set_att[ds][ds][1] - used_train_set_att[ds][ds][0]]
        result = [used_train_set_att[ds][ds][0]]
        seasonality = [used_train_set_att[ds][ds][i] - used_train_set_att[ds][ds][0] for i in range(periods)]

        for t in range(1, n):
            level.append(alpha*(used_train_set_att[ds][ds][t] - seasonality[t % periods]) + (1 - alpha)*(level[t - 1] + trend[t - 1]))
            trend.append(beta*(level[t] - level[t - 1]) + (1 - beta)*trend[t - 1])
            seasonality.append(gamma*(used_train_set_att[ds][ds][t] - level[t]) + (1 - gamma)*seasonality[t % periods])
            result.append(level[t] + trend[t] + seasonality[t % periods])

        results_HWSA[ds] = result
    return results_HWSA
```

E) Holt-Winters saisonnier multiplicatif

[ ]:
```
def holt_winters_multiplicative(alpha, beta, gamma, periods):
    results_HWSM = {}
```

```
    for ds in used_train_set_att:
        n = len(used_train_set_att[ds][ds])
        level = [used_train_set_att[ds][ds][0]]
        trend = [used_train_set_att[ds][ds][1] - used_train_set_att[ds][ds][0]]
        result = [used_train_set_att[ds][ds][0]]
        seasonality = [used_train_set_att[ds][ds][i] /␣
↪used_train_set_att[ds][ds][0] for i in range(periods)]

        for t in range(1, n):
            level.append(alpha*(used_train_set_att[ds][ds][t] / seasonality[t %␣
↪periods]) + (1 - alpha)*(level[t - 1] + trend[t - 1]))
            trend.append(beta*(level[t] - level[t - 1]) + (1 - beta)*trend[t -␣
↪1])
            seasonality.append(gamma*(used_train_set_att[ds][ds][t] / level[t])␣
↪+ (1 - gamma)*seasonality[t % periods])
            result.append((level[t] + trend[t])*seasonality[t % periods])

        results_HWSM[ds] = result
    return results_HWSM
```

3 - Calcul et representation des erreurs de prevision, determination du meilleur

[ ]:
```
best_alpha_SEM = {}
best_alpha_DEM = {}
best_alpha_HWNS = {}
best_alpha_HWSA = {}
best_alpha_HWSM = {}
best_beta_DEM = {}
best_beta_HWNS = {}
best_beta_HWSA = {}
best_beta_HWSM = {}
best_gamma_HWSA = {}
best_gamma_HWSM = {}
best_alpha_forecasts_SEM = {}
best_alpha_forecasts_DEM = {}
best_alpha_forecasts_HWNS = {}
best_alpha_forecasts_HWSA = {}
best_alpha_forecasts_HWSM = {}
periods = 12
errors = []

for ds in used_train_set_att:
    min_error_SEM = min_error_DEM = min_error_HWNS = min_error_HWSA =␣
↪min_error_HWSM = float('inf')
    ds_errors = {}
    for alpha in alpha_values:
        forecast_SEM = simple_exponential_smoothing(alpha);
```

```python
        error_SEM = np.mean((np.array(used_train_set_att[ds][ds]) - np.array(forecast_SEM[ds]))**2)
        ds_errors[f"LES (  = {alpha})"] = error_SEM

        if (error_SEM < min_error_SEM):
            min_error_SEM = error_SEM
            best_alpha_SEM[ds] = alpha
            best_alpha_forecasts_SEM[ds] = forecast_SEM

    for alpha in alpha_values:
        for beta in beta_values:
            forecast_DEM = double_exponential_smoothing(alpha, beta);
            error_DEM = np.mean((np.array(used_train_set_att[ds][ds]) - np.array(forecast_DEM[ds]))**2)
            ds_errors[f"LED (  = {alpha},   = {beta})"] = error_DEM

            if (error_DEM < min_error_DEM):
                min_error_DEM = error_DEM
                best_alpha_DEM[ds] = alpha
                best_beta_DEM[ds] = beta
                best_alpha_forecasts_DEM[ds] = forecast_DEM

    for alpha in alpha_values:
        for beta in beta_values:
            forecast_HWNS = holt_winters_non_seasonal(alpha, beta);
            error_HWNS = np.mean((np.array(used_train_set_att[ds][ds]) - np.array(forecast_HWNS[ds]))**2)
            ds_errors[f"HW non saisonnier (  = {alpha},   = {beta})"] = error_HWNS

            if (error_HWNS < min_error_HWNS):
                min_error_HWNS = error_HWNS
                best_alpha_HWNS[ds] = alpha
                best_beta_HWNS[ds] = beta
                best_alpha_forecasts_HWNS[ds] = forecast_HWNS

    for alpha in alpha_values:
        for beta in beta_values:
            for gamma in gamma_values:
                forecast_HWSA = holt_winters_additive(alpha, beta, gamma, periods);
                error_HWSA = np.mean((np.array(used_train_set_att[ds][ds]) - np.array(forecast_HWSA[ds]))**2)
                ds_errors[f"HW saisonnier additif (  = {alpha},   = {beta},   = {gamma})"] = error_HWSA
```

```python
                if (error_HWSA < min_error_HWSA):
                    min_error_HWSA = error_HWSA
                    best_alpha_HWSA[ds] = alpha
                    best_beta_HWSA[ds] = beta
                    best_gamma_HWSA[ds] = gamma
                    best_alpha_forecasts_HWSA[ds] = forecast_HWSA

    for alpha in alpha_values:
        for beta in beta_values:
            for gamma in gamma_values:
                forecast_HWSM = holt_winters_multiplicative(alpha, beta, gamma, periods);
                error_HWSM = np.mean((np.array(used_train_set_att[ds][ds]) - np.array(forecast_HWSM[ds]))**2)
                ds_errors[f"HW saisonnier multificatif (  = {alpha},   = {beta},  = {gamma})"] = error_HWSM

                if (error_HWSM < min_error_HWSM):
                    min_error_HWSM = error_HWSM
                    best_alpha_HWSM[ds] = alpha
                    best_beta_HWSM[ds] = beta
                    best_gamma_HWSM[ds] = gamma
                    best_alpha_forecasts_HWSM[ds] = forecast_HWSM

    errors.append(ds_errors)

errors_df = pd.DataFrame(errors)
errors_df.index = [ds for ds in used_train_set_att]
errors_df = errors_df.T
best_errors = errors_df.min(axis=0)

# def highlight_best_errors(val, min_values):
#     column = val.name
#     return ['font-weight: bold' if v == min_values[column] else '' for v in val]

errors_df = errors_df.map(lambda x: f'{x: .4f}')
# highlighted_errors_df = errors_df.style.apply(highlight_best_errors, axis=0, min_values=best_errors)
# ax = plt.subplot()
plt.figure(figsize=(14, 24))
plt.axis('tight')
plt.axis('off')

table = plt.table(cellText=errors_df.values, colLabels=errors_df.columns, rowLabels=errors_df.index, cellLoc='center', loc='center')
```

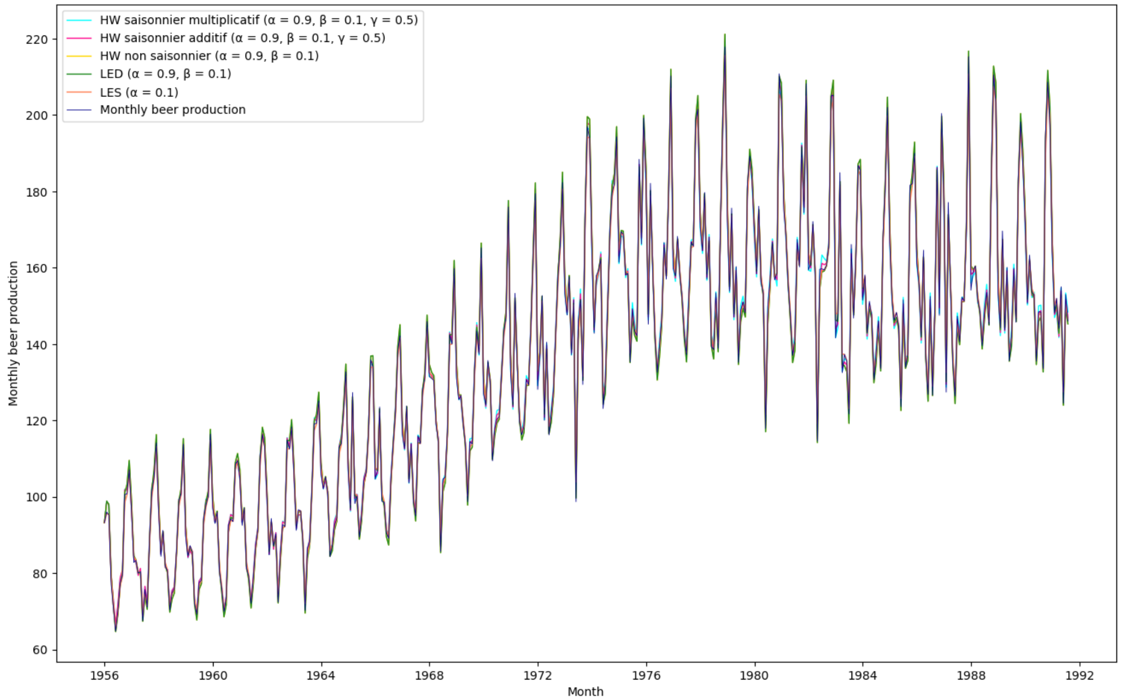
```
table.auto_set_font_size(False)
table.set_fontsize(14)
table.scale(1.2, 1.8)

# for (i, j), cell in table.get_celld().items():
#     if i == 0 or j == -1:
#         continue
#     if float(errors_df.iloc[i - 1, j]) == best_errors.iloc[j]:
#         cell.set_text_props(weight='bold')
# plt.title('Tableau des erreurs de lissage', loc='center')
plt.show()
```

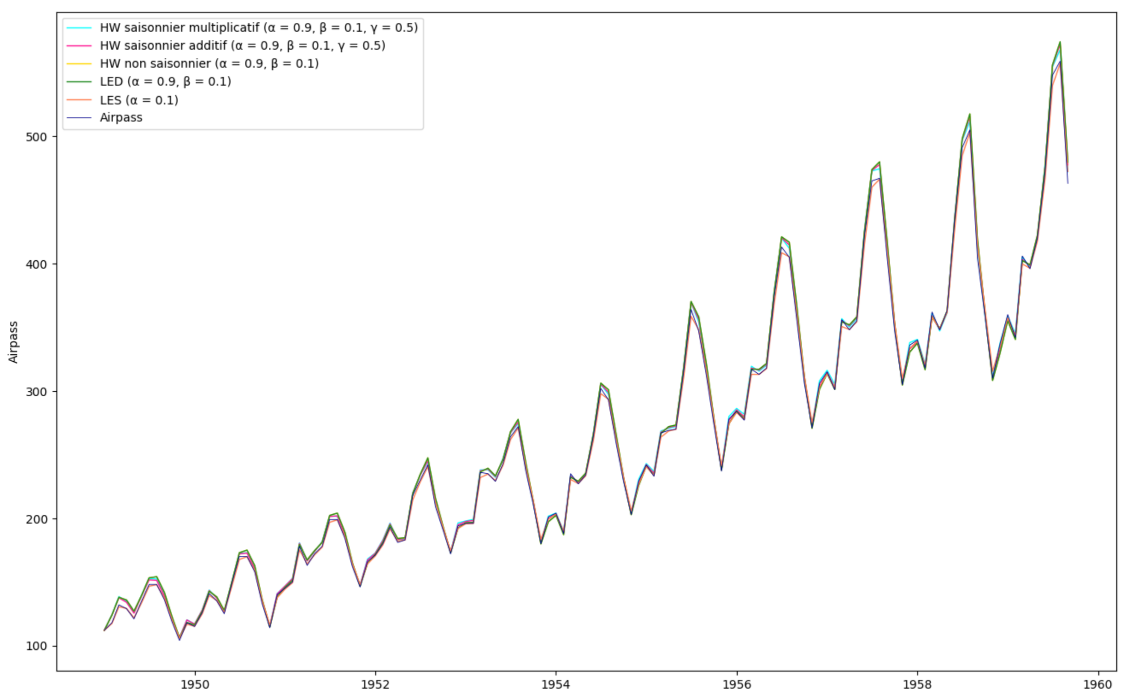
	Monthly beer production	Airpass	TEMPERATURE	Sunspots	IPG2211A2N
LES (α = 0.5)	88.8825	345.1257	0.7308	80.3059	17.7348
LES (α = 0.1)	3.6097	9.9758	0.0132	3.5886	0.5989
LES (α = 0.9)	303.4176	1908.1228	7.5439	488.0435	52.0565
LED (α = 0.5, β = 0.5)	126.6813	574.1385	1.1673	90.4107	23.2415
LED (α = 0.5, β = 0.1)	91.5796	359.7864	0.8549	80.9694	18.1438
LED (α = 0.5, β = 0.9)	147.8600	723.5922	1.1962	112.4362	41.6670
LED (α = 0.1, β = 0.5)	378.1203	1830.4747	34.4542	382.3697	57.6455
LED (α = 0.1, β = 0.1)	319.1567	1579.3399	9.4435	497.5373	57.5425
LED (α = 0.1, β = 0.9)	494.6388	2345.7213	26.0693	403.4945	63.5817
LED (α = 0.9, β = 0.5)	60.2794	293.3337	0.7109	49.2099	14.1814
LED (α = 0.9, β = 0.1)	3.8301	26.0906	0.0951	6.4976	0.7496
LED (α = 0.9, β = 0.9)	183.5351	673.8546	0.9749	174.2027	44.6530
HW non saisonnier (α = 0.5, β = 0.5)	126.6813	574.1385	1.1673	90.4107	23.2415
HW non saisonnier (α = 0.5, β = 0.1)	91.5796	359.7864	0.8549	80.9694	18.1438
HW non saisonnier (α = 0.5, β = 0.9)	147.8600	723.5922	1.1962	112.4362	41.6670
HW non saisonnier (α = 0.1, β = 0.5)	378.1203	1830.4747	34.4542	382.3697	57.6455
HW non saisonnier (α = 0.1, β = 0.1)	319.1567	1579.3399	9.4435	497.5373	57.5425
HW non saisonnier (α = 0.1, β = 0.9)	494.6388	2345.7213	26.0693	403.4945	63.5817
HW non saisonnier (α = 0.9, β = 0.5)	60.2794	293.3337	0.7109	49.2099	14.1814
HW non saisonnier (α = 0.9, β = 0.1)	3.8301	26.0906	0.0951	6.4976	0.7496
HW non saisonnier (α = 0.9, β = 0.9)	183.5351	673.8546	0.9749	174.2027	44.6530
HW saisonnier additif (α = 0.5, β = 0.5, γ = 0.5)	44.2273	313.3179	1.1006	294.3082	6.0831
HW saisonnier additif (α = 0.5, β = 0.5, γ = 0.1)	44.2273	313.3179	1.1006	294.3082	6.0831
HW saisonnier additif (α = 0.5, β = 0.5, γ = 0.9)	44.2273	313.3179	1.1006	294.3082	6.0831
HW saisonnier additif (α = 0.5, β = 0.1, γ = 0.5)	44.8926	198.5536	0.9114	274.8322	4.9411
HW saisonnier additif (α = 0.5, β = 0.1, γ = 0.1)	44.8926	198.5536	0.9114	274.8322	4.9411
HW saisonnier additif (α = 0.5, β = 0.1, γ = 0.9)	44.8926	198.5536	0.9114	274.8322	4.9411
HW saisonnier additif (α = 0.5, β = 0.9, γ = 0.5)	54.4629	389.4855	1.2014	368.9331	9.5672
HW saisonnier additif (α = 0.5, β = 0.9, γ = 0.1)	54.4629	389.4855	1.2014	368.9331	9.5672
HW saisonnier additif (α = 0.5, β = 0.9, γ = 0.9)	54.4629	389.4855	1.2014	368.9331	9.5672
HW saisonnier additif (α = 0.1, β = 0.5, γ = 0.5)	120.6035	1030.0803	33.8906	920.9018	18.0626
HW saisonnier additif (α = 0.1, β = 0.5, γ = 0.1)	120.6035	1030.0803	33.8906	920.9018	18.0626
HW saisonnier additif (α = 0.1, β = 0.5, γ = 0.9)	120.6035	1030.0803	33.8906	920.9018	18.0626
HW saisonnier additif (α = 0.1, β = 0.1, γ = 0.5)	123.0267	941.4760	9.2852	995.0668	19.9236
HW saisonnier additif (α = 0.1, β = 0.1, γ = 0.1)	123.0267	941.4760	9.2852	995.0668	19.9236
HW saisonnier additif (α = 0.1, β = 0.1, γ = 0.9)	123.0267	941.4760	9.2852	995.0668	19.9236
HW saisonnier additif (α = 0.1, β = 0.9, γ = 0.5)	129.0832	1272.5479	25.3244	1007.1991	20.5002
HW saisonnier additif (α = 0.1, β = 0.9, γ = 0.1)	129.0832	1272.5479	25.3244	1007.1991	20.5002
HW saisonnier additif (α = 0.1, β = 0.9, γ = 0.9)	129.0832	1272.5479	25.3244	1007.1991	20.5002
HW saisonnier additif (α = 0.9, β = 0.5, γ = 0.5)	23.2639	161.2496	0.7078	157.6030	3.4087
HW saisonnier additif (α = 0.9, β = 0.5, γ = 0.1)	23.2639	161.2496	0.7078	157.6030	3.4087
HW saisonnier additif (α = 0.9, β = 0.5, γ = 0.9)	23.2639	161.2496	0.7078	157.6030	3.4087
HW saisonnier additif (α = 0.9, β = 0.1, γ = 0.5)	1.5547	17.4698	0.0932	12.4373	0.2403
HW saisonnier additif (α = 0.9, β = 0.1, γ = 0.1)	1.5547	17.4698	0.0932	12.4373	0.2403
HW saisonnier additif (α = 0.9, β = 0.1, γ = 0.9)	1.5547	17.4698	0.0932	12.4373	0.2403
HW saisonnier additif (α = 0.9, β = 0.9, γ = 0.5)	109.7542	363.5045	1.2798	649.4113	11.0484
HW saisonnier additif (α = 0.9, β = 0.9, γ = 0.1)	109.7542	363.5045	1.2798	649.4113	11.0484
HW saisonnier additif (α = 0.9, β = 0.9, γ = 0.9)	109.7542	363.5045	1.2798	649.4113	11.0484
HW saisonnier multificatif (α = 0.5, β = 0.5, γ = 0.5)	63.8179	106.6904	1.1782	178.4705	5.7376
HW saisonnier multificatif (α = 0.5, β = 0.5, γ = 0.1)	63.8179	106.6904	1.1782	178.4705	5.7376
HW saisonnier multificatif (α = 0.5, β = 0.5, γ = 0.9)	63.8179	106.6904	1.1782	178.4705	5.7376
HW saisonnier multificatif (α = 0.5, β = 0.1, γ = 0.5)	54.5007	75.4595	0.9013	198.9105	3.8136
HW saisonnier multificatif (α = 0.5, β = 0.1, γ = 0.1)	54.5007	75.4595	0.9013	198.9105	3.8136
HW saisonnier multificatif (α = 0.5, β = 0.1, γ = 0.9)	54.5007	75.4595	0.9013	198.9105	3.8136
HW saisonnier multificatif (α = 0.5, β = 0.9, γ = 0.5)	76.1978	128.8099	1.2729	196.8464	6.8360
HW saisonnier multificatif (α = 0.5, β = 0.9, γ = 0.1)	76.1978	128.8099	1.2729	196.8464	6.8360
HW saisonnier multificatif (α = 0.5, β = 0.9, γ = 0.9)	76.1978	128.8099	1.2729	196.8464	6.8360
HW saisonnier multificatif (α = 0.1, β = 0.5, γ = 0.5)	183.4412	373.7476	35.8379	795.2547	19.5227
HW saisonnier multificatif (α = 0.1, β = 0.5, γ = 0.1)	183.4412	373.7476	35.8379	795.2547	19.5227
HW saisonnier multificatif (α = 0.1, β = 0.5, γ = 0.9)	183.4412	373.7476	35.8379	795.2547	19.5227
HW saisonnier multificatif (α = 0.1, β = 0.1, γ = 0.5)	170.5967	423.4449	9.8542	962.7186	18.9106
HW saisonnier multificatif (α = 0.1, β = 0.1, γ = 0.1)	170.5967	423.4449	9.8542	962.7186	18.9106
HW saisonnier multificatif (α = 0.1, β = 0.1, γ = 0.9)	170.5967	423.4449	9.8542	962.7186	18.9106
HW saisonnier multificatif (α = 0.1, β = 0.9, γ = 0.5)	222.1944	385.2682	24.4245	803.9088	25.0398
HW saisonnier multificatif (α = 0.1, β = 0.9, γ = 0.1)	222.1944	385.2682	24.4245	803.9088	25.0398
HW saisonnier multificatif (α = 0.1, β = 0.9, γ = 0.9)	222.1944	385.2682	24.4245	803.9088	25.0398
HW saisonnier multificatif (α = 0.9, β = 0.5, γ = 0.5)	31.0765	55.9513	0.7443	108.9585	2.6224
HW saisonnier multificatif (α = 0.9, β = 0.5, γ = 0.1)	31.0765	55.9513	0.7443	108.9585	2.6224
HW saisonnier multificatif (α = 0.9, β = 0.5, γ = 0.9)	31.0765	55.9513	0.7443	108.9585	2.6224
HW saisonnier multificatif (α = 0.9, β = 0.1, γ = 0.5)	2.0377	12.2110	0.0950	9.6416	0.2211
HW saisonnier multificatif (α = 0.9, β = 0.1, γ = 0.1)	2.0377	12.2110	0.0950	9.6416	0.2211
HW saisonnier multificatif (α = 0.9, β = 0.1, γ = 0.9)	2.0377	12.2110	0.0950	9.6416	0.2211
HW saisonnier multificatif (α = 0.9, β = 0.9, γ = 0.5)	121.7603	133.4819	1.3129	439.7844	7.0408
HW saisonnier multificatif (α = 0.9, β = 0.9, γ = 0.1)	121.7603	133.4819	1.3129	439.7844	7.0408
HW saisonnier multificatif (α = 0.9, β = 0.9, γ = 0.9)	121.7603	133.4819	1.3129	439.7844	7.0408

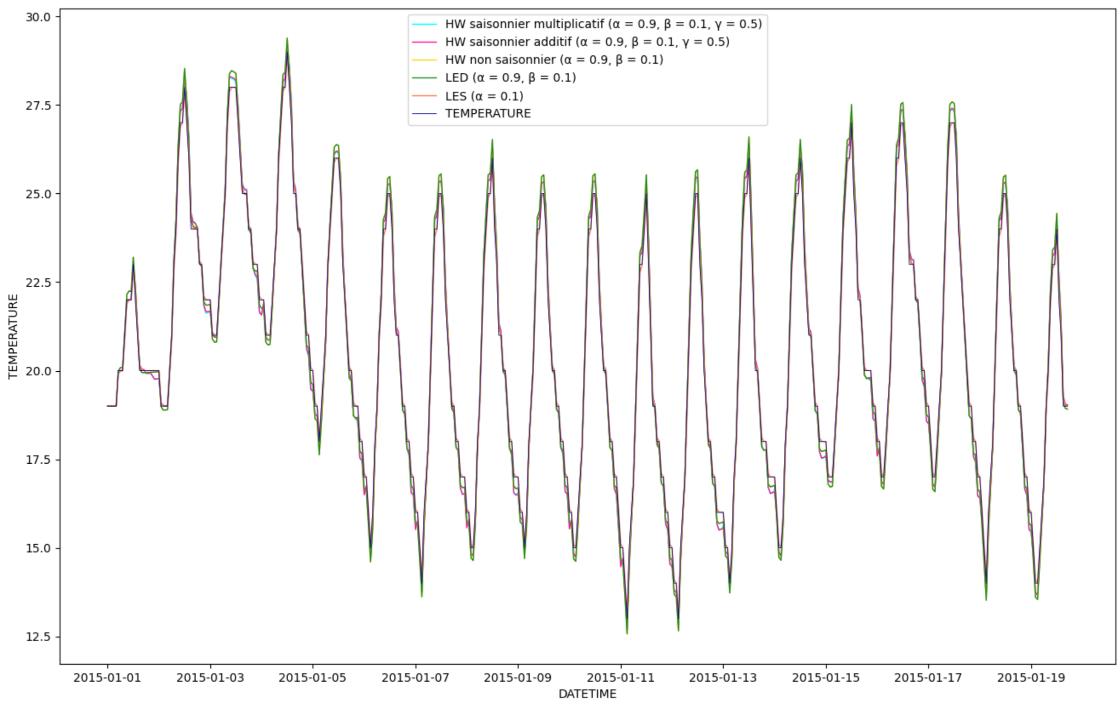
4 - Trace des courbes pour les meilleures valeurs de

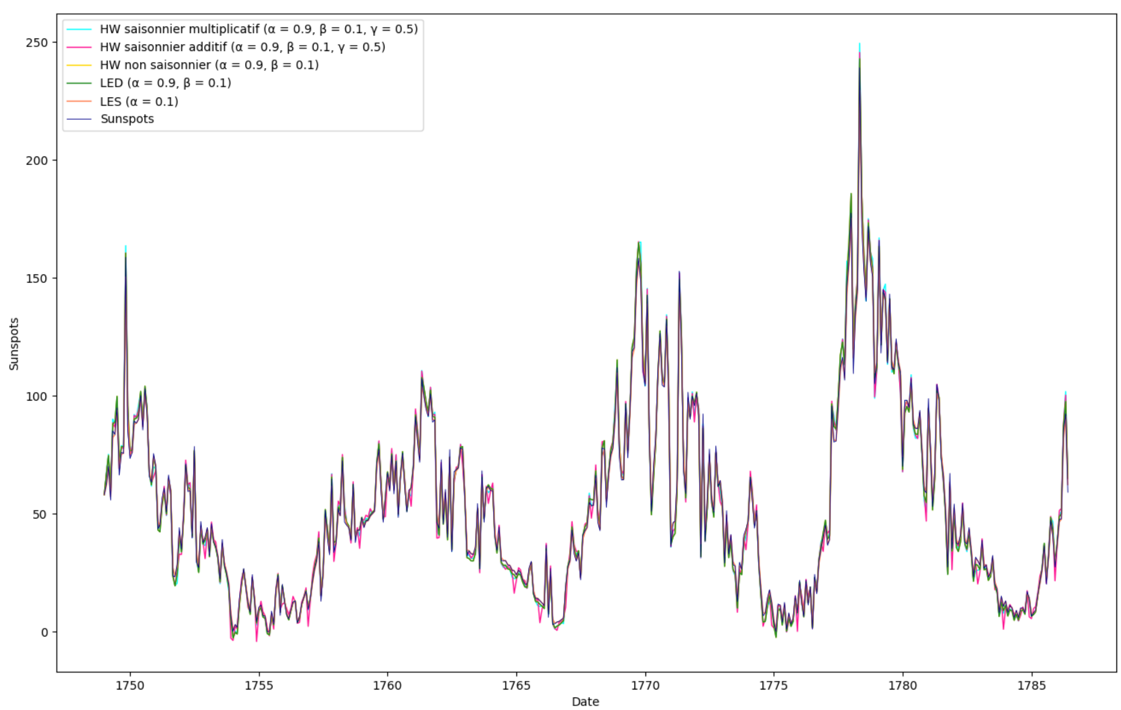
[ ]:
```
# i = 1
# plt.figure(figsize=(16, 32))
# plt.title('Lissage exponentiel simple, double et triple')
# plt.xticks([])
# plt.yticks([])
```

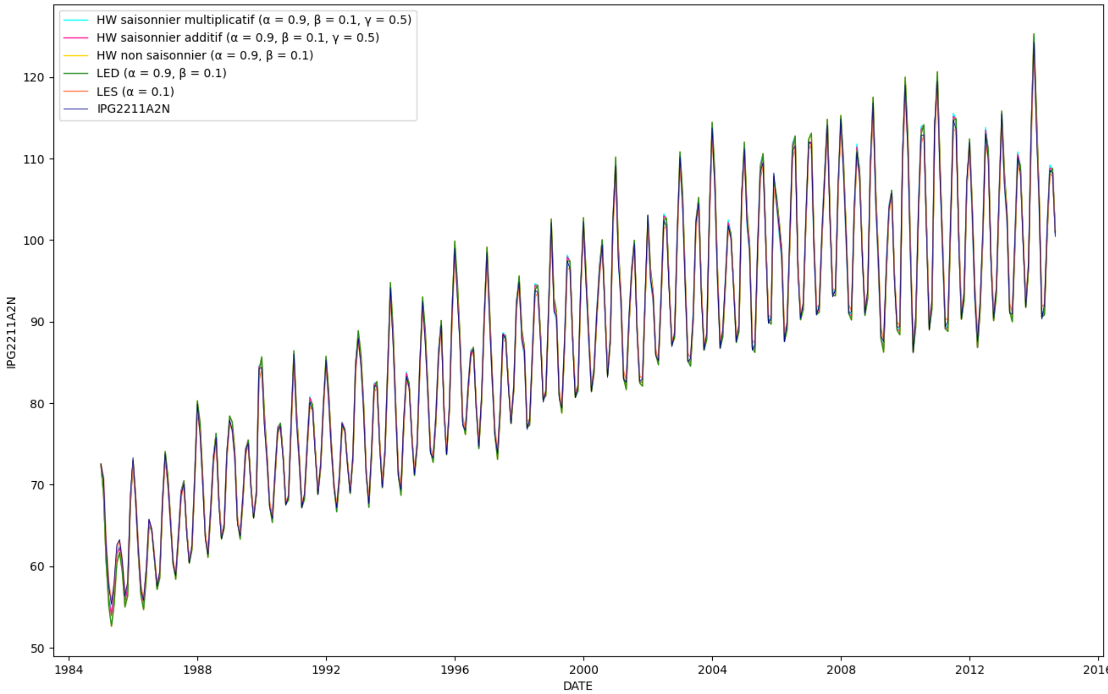
```
for ds in used_test_set_att:
    # ax = plt.subplot(5, 1, i)
    plt.figure(figsize=(16, 10))
    plt.plot(pd.to_datetime(used_train_set_att[ds][used_train_set_att[ds].
 ↪columns[0]]), best_alpha_forecasts_HWSM[ds][ds], linewidth=1, label=f"HW␣
 ↪saisonnier multiplicatif (  = {best_alpha_HWSM[ds]},   =␣
 ↪{best_beta_HWSM[ds]},   = {best_gamma_HWSM[ds]})", c="aqua")
    plt.plot(pd.to_datetime(used_train_set_att[ds][used_train_set_att[ds].
 ↪columns[0]]), best_alpha_forecasts_HWSA[ds][ds], linewidth=1, label=f"HW␣
 ↪saisonnier additif (  = {best_alpha_HWSA[ds]},   = {best_beta_HWSA[ds]},   =␣
 ↪{best_gamma_HWSA[ds]})", c="deeppink")
    plt.plot(pd.to_datetime(used_train_set_att[ds][used_train_set_att[ds].
 ↪columns[0]]), best_alpha_forecasts_HWNS[ds][ds], linewidth=1, label=f"HW non␣
 ↪saisonnier (  = {best_alpha_HWNS[ds]},   = {best_beta_HWNS[ds]})", c="gold")
    plt.plot(pd.to_datetime(used_train_set_att[ds][used_train_set_att[ds].
 ↪columns[0]]), best_alpha_forecasts_DEM[ds][ds], linewidth=1, label=f"LED (  =␣
 ↪{best_alpha_DEM[ds]},   = {best_beta_DEM[ds]})", c="forestgreen")
    plt.plot(pd.to_datetime(used_train_set_att[ds][used_train_set_att[ds].
 ↪columns[0]]), best_alpha_forecasts_SEM[ds][ds], linewidth=1, label=f"LES (  =␣
 ↪{best_alpha_SEM[ds]})", c="coral")
    plt.plot(pd.to_datetime(used_train_set_att[ds][used_train_set_att[ds].
 ↪columns[0]]), used_train_set_att[ds][ds], linewidth=0.6, label=ds,␣
 ↪c='darkblue')
    plt.legend()
    plt.xlabel(used_train_set_att[ds].columns[0])
    plt.ylabel(ds)
    # i = i + 1

plt.show()
```


HW saisonnier multiplicatif (α = 0.9, β = 0.1, γ = 0.5)
HW saisonnier additif (α = 0.9, β = 0.1, γ = 0.5)
HW non saisonnier (α = 0.9, β = 0.1)
LED (α = 0.9, β = 0.1)
LES (α = 0.1)
Monthly beer production
Monthly beer production
Month


HW saisonnier multiplicatif (α = 0.9, β = 0.1, γ = 0.5)
HW saisonnier additif (α = 0.9, β = 0.1, γ = 0.5)
HW non saisonnier (α = 0.9, β = 0.1)
LED (α = 0.9, β = 0.1)
LES (α = 0.1)
Airpass
Airpass
t


HW saisonnier multiplicatif (α = 0.9, β = 0.1, γ = 0.5)
HW saisonnier additif (α = 0.9, β = 0.1, γ = 0.5)
HW non saisonnier (α = 0.9, β = 0.1)
LED (α = 0.9, β = 0.1)
LES (α = 0.1)
TEMPERATURE
TEMPERATURE
DATETIME
30.0
27.5
25.0
22.5
20.0
17.5
15.0
12.5
2015-01-01
2015-01-03
2015-01-05
2015-01-07
2015-01-09
2015-01-11
2015-01-13
2015-01-15
2015-01-17
2015-01-19


HW saisonnier multiplicatif (α = 0.9, β = 0.1, γ = 0.5)
HW saisonnier additif (α = 0.9, β = 0.1, γ = 0.5)
HW non saisonnier (α = 0.9, β = 0.1)
LED (α = 0.9, β = 0.1)
LES (α = 0.1)
Sunspots
Sunspots
Date
250
200
150
100
50
0
1750
1755
1760
1765
1770
1775
1780
1785


HW saisonnier multiplicatif (α = 0.9, β = 0.1, γ = 0.5)
HW saisonnier additif (α = 0.9, β = 0.1, γ = 0.5)
HW non saisonnier (α = 0.9, β = 0.1)
LED (α = 0.9, β = 0.1)
LES (α = 0.1)
IPG2211A2N
IPG2211A2N
DATE
50
60
70
80
90
100
110
120
1984
1988
1992
1996
2000
2004
2008
2012
2016