EXTRACTION DE LA TENDANCE ET LA SAISONNALITE

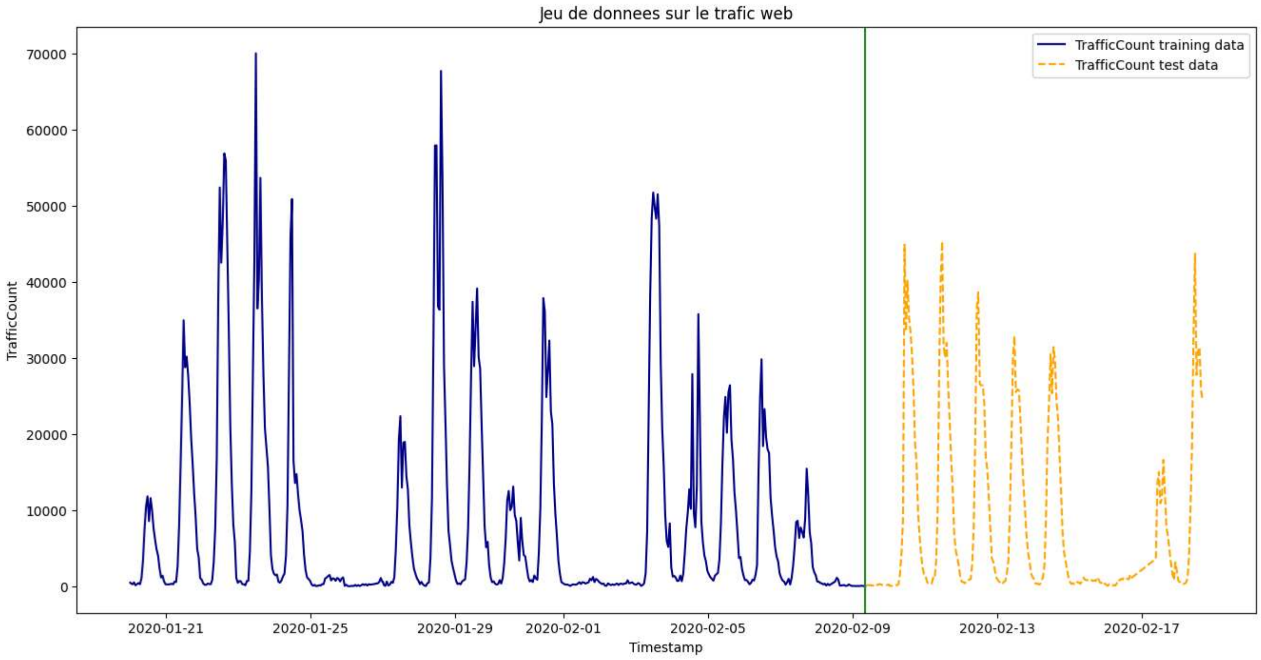
In [ ]:

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt

df = pd.read_csv("dataset/web_traffic.csv", nrows=700);
colors = ['crimson', 'deepskyblue', 'deeppink']
window_sizes = [3, 5, 10]
differences = ['Diff_1', 'Diff_2', 'Diff_3']

def seed(df, percentage=90):
    if percentage > 50:
        return {"training": df[:(percentage*len(df))//100], "test": df[((percentage*len(df))//100):]}
    else:
        return {"training": df[:((1 - percentage)*len(df))//100], "test": df[(((1 - percentage)*len(df))//100):]}
```

1 - Tracé de la serie

In [ ]:

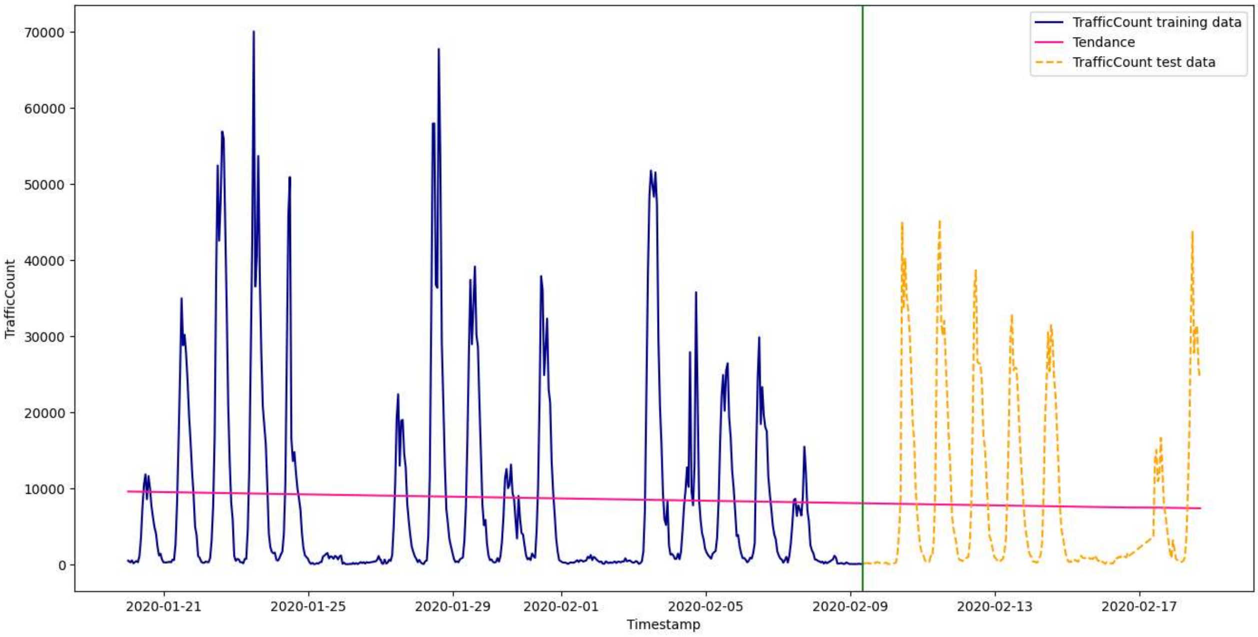
```
seed_df = seed(df, 70)
train_df = seed_df['training']
test_df = seed_df['test']
ycol = 'TrafficCount'
xcol = 'Timestamp'

plt.figure(figsize=(16, 8))
plt.title(label="Jeu de donnees sur le trafic web")
plt.plot(pd.to_datetime(train_df[xcol]), train_df[ycol], label=f"{ycol} training data", c='darkblue')
plt.axvline(x=pd.to_datetime(train_df[xcol][len(train_df) - 1]), c="forestgreen")
plt.plot(pd.to_datetime(test_df[xcol]), test_df[ycol], label=f"{ycol} test data", c='orange', linestyle="--")
plt.xlabel(xcol)
plt.ylabel(ycol)
plt.legend()
plt.show()
```



2 - Tracé de la serie en appliquant les moindres carres

In [ ]:

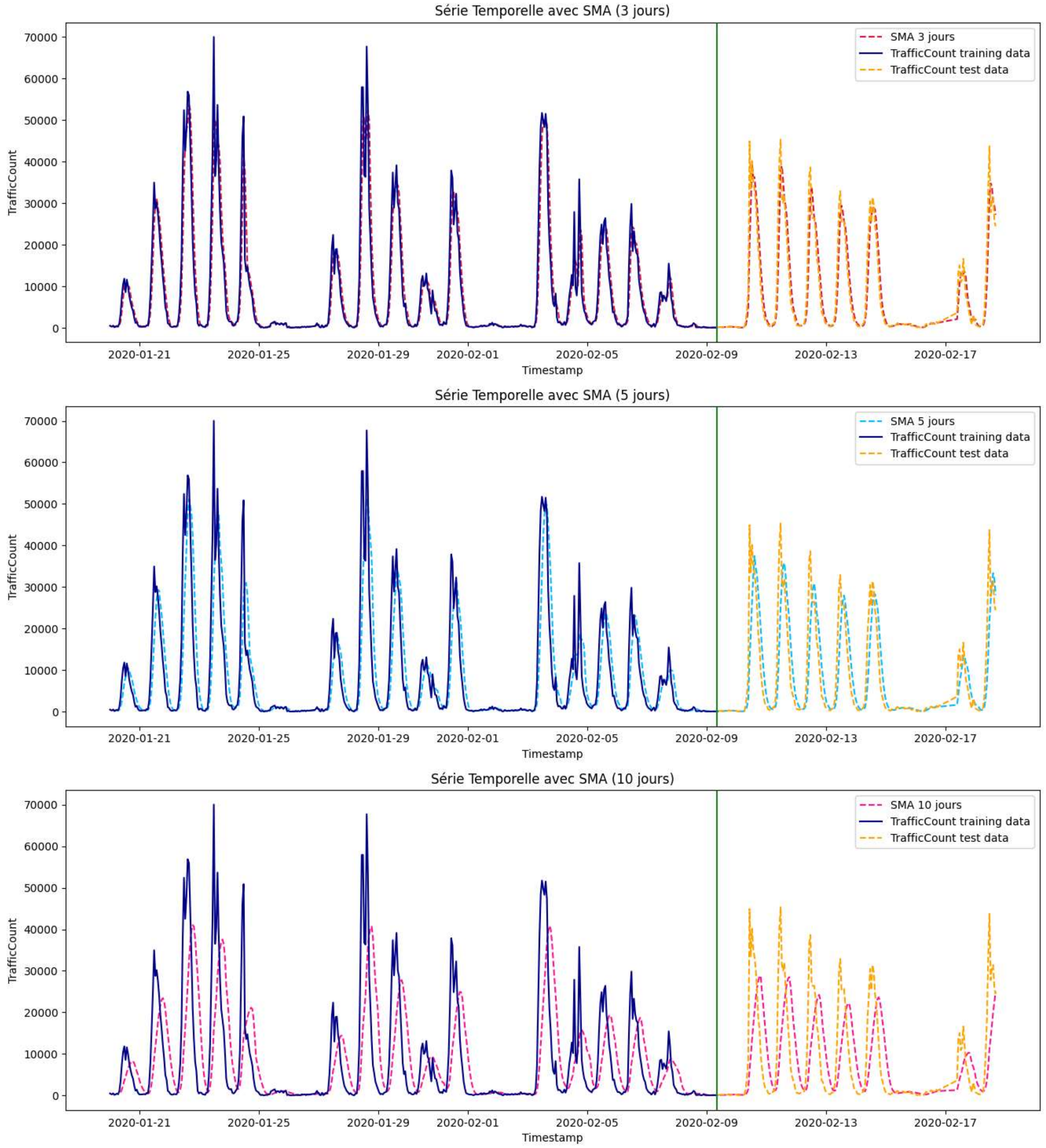
```
# Calcul des parametres de la droite de tendance
x = np.arange(len(df))
# Utilisation de l'index numerique comme variable independante
y = df[ycol].values

# Calcul de la pense (slope) et de l'intercept
x_mean = x.mean()
y_mean = y.mean()
xy_mean = (x * y).mean()
x_square_mean = (x * x).mean()
slope = (xy_mean - x_mean*y_mean)/(x_square_mean - x_mean**2)
intercept = y_mean - slope*x_mean

# Calcul des valeurs de la tendance
trend = slope * x + intercept

# Tracer de la serie et la tendance
plt.figure(figsize=(16, 8))
plt.plot(pd.to_datetime(train_df[xcol]), train_df[ycol], label=f"{ycol} training data", c='darkblue')
plt.plot(pd.to_datetime(df[xcol]), trend, label="Tendance", c="deeppink")
plt.axvline(x=pd.to_datetime(train_df[xcol][len(train_df) - 1]), c="forestgreen")
plt.plot(pd.to_datetime(test_df[xcol]), test_df[ycol], label=f"{ycol} test data", c='orange', linestyle="--")
plt.xlabel(xcol)
plt.ylabel(ycol)
plt.legend()
plt.show()
```



3 - Tracé de la serie en appliquant les moyennes mobiles

In [ ]:

```
# Fonction de calcul des moyennes mobiles simples (SMA)
def calculate_sma(df, window):
    sma = []
    for i in range(len(df)):
        if i < window - 1:
            sma.append(np.nan)  # Pas assez de données pour calculer la moyenne mobile
        else:
            window_data = df[i - window + 1:i + 1]
            sma.append(np.mean(window_data))
    return np.array(sma)

# Calcul des moyennes mobiles avec différentes fenêtres
smas = {window: calculate_sma(df[ycol].values, window) for window in window_sizes}

# Tracer la série temporelle et les moyennes mobiles
plt.figure(figsize=(16, 18))

for i, window in enumerate(window_sizes):
    plt.subplot(3, 1, i + 1)
    plt.plot(pd.to_datetime(df[xcol]), smas[window], label=f'SMA {window} jours', color=colors[i], linestyle='--')
    plt.plot(pd.to_datetime(train_df[xcol]), train_df[ycol], label=f"{ycol} training data", c='darkblue')
    plt.axvline(x=pd.to_datetime(train_df[xcol][len(train_df) - 1]), c="forestgreen")
    plt.plot(pd.to_datetime(test_df[xcol]), test_df[ycol], label=f"{ycol} test data", c='orange', linestyle="--")
    plt.title(f'Série Temporelle avec SMA ({window} jours)')
    plt.xlabel(xcol)
    plt.ylabel(ycol)
    plt.legend()

# plt.tight_layout()
plt.show()
```


Série Temporelle avec SMA (3 jours)

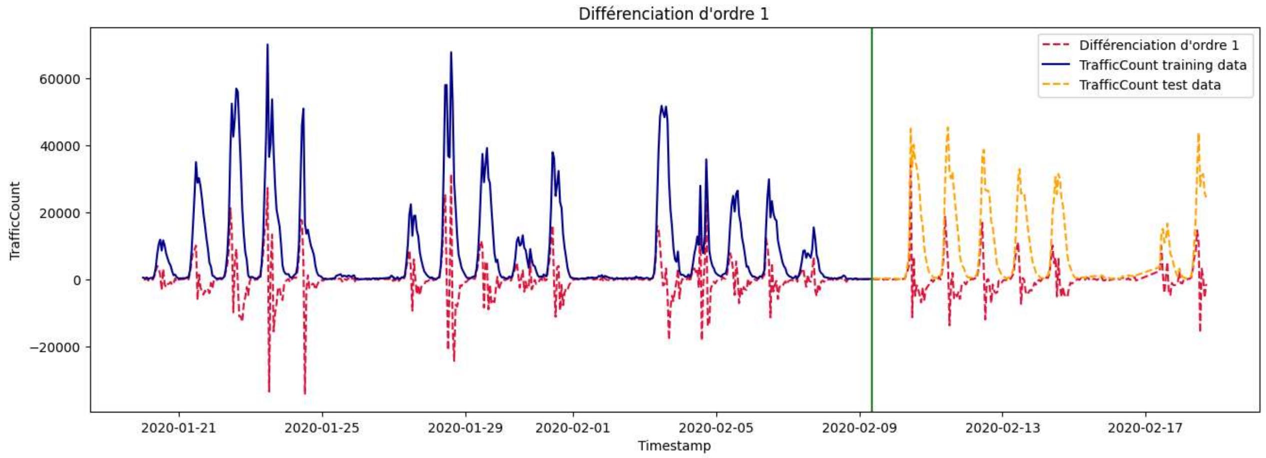
Série Temporelle avec SMA (5 jours)

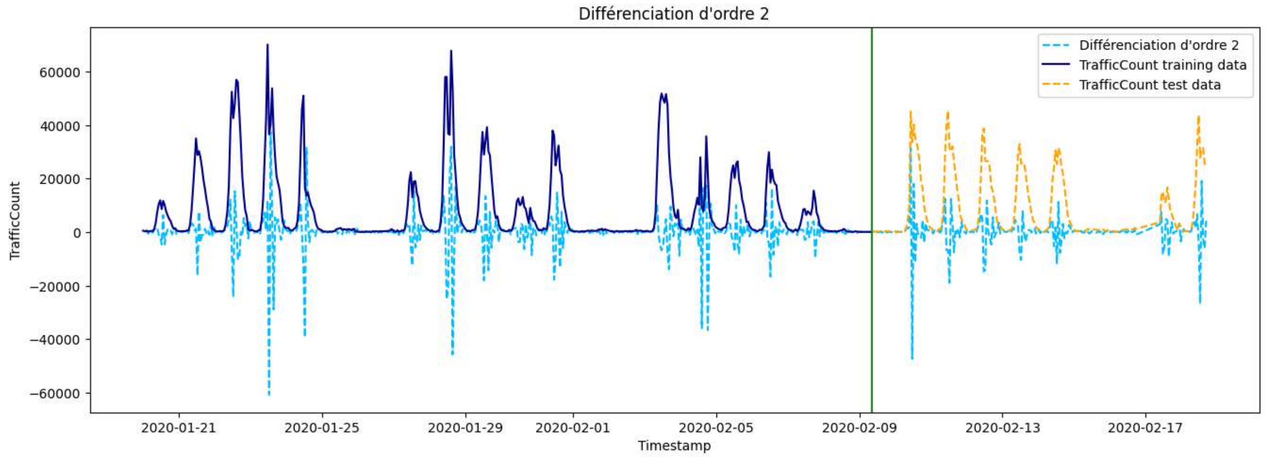
Série Temporelle avec SMA (10 jours)

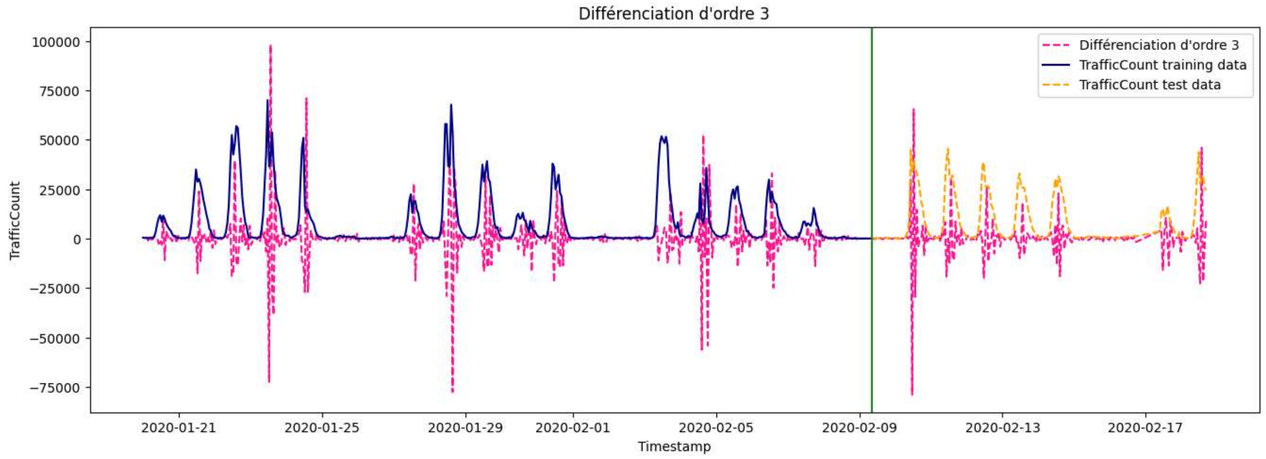
4 - Tracé de la serie en appliquant la différenciation

In [ ]:

```
# Fonction pour calculer la différenciation
def differentiate(df, order):
    diff_data = df.copy()
    for _ in range(order):
        diff_data = [j-i for i, j in zip(diff_data[:-1], diff_data[1:])]
    return [np.nan]*order + diff_data

# Calcul des différenciations pour chaque ordre
for i, difference in enumerate(differences):
    df[difference] = differentiate(df[ycol].tolist(), i + 1)

plt.figure(figsize=(16, 18))

for i, difference in enumerate(differences):
    plt.subplot(3, 1, i + 1)
    plt.plot(pd.to_datetime(df[xcol]), df[difference], label=f'Différenciation d\'ordre {i + 1}', color=colors[i], linestyle='--')
    plt.plot(pd.to_datetime(train_df[xcol]), train_df[ycol], label=f"{ycol} training data", c='darkblue')
    plt.axvline(x=pd.to_datetime(train_df[xcol][len(train_df) - 1]), c="forestgreen")
    plt.plot(pd.to_datetime(test_df[xcol]), test_df[ycol], label=f"{ycol} test data", c='orange', linestyle="--")
    plt.title(f'Différenciation d\'ordre {i + 1}')
    plt.xlabel(xcol)
    plt.ylabel(ycol)
    plt.legend()

# plt.tight_layout()
plt.show()
```


Différenciation d'ordre 1
Différenciation d'ordre 1
TrafficCount training data
TrafficCount test data
TrafficCount
60000
40000
20000
0
−20000
2020-01-21
2020-01-25
2020-01-29
2020-02-01
2020-02-05
2020-02-09
2020-02-13
2020-02-17
Timestamp


Différenciation d'ordre 2
Différenciation d'ordre 2
TrafficCount training data
TrafficCount test data
TrafficCount
60000
40000
20000
0
−20000
−40000
−60000
2020-01-21
2020-01-25
2020-01-29
2020-02-01
2020-02-05
2020-02-09
2020-02-13
2020-02-17
Timestamp


Différenciation d'ordre 3
Différenciation d'ordre 3
TrafficCount training data
TrafficCount test data
TrafficCount
100000
75000
50000
25000
0
−25000
−50000
−75000
2020-01-21
2020-01-25
2020-01-29
2020-02-01
2020-02-05
2020-02-09
2020-02-13
2020-02-17
Timestamp