Rapport du projet en Sciences de données

TE54MI - Science des données 3

Thème:

«Réchauffement climatique et Consommation énergétique en France »

Équipe:

- INGABIRE Alvin
- MEDOM SADEFO Michele
- NIYONKURU Berline Cléria

Encadrants:

- Servajean Maximilien
- Cesar Leblanc

Table des Matières

INTRODUCTION

La science de données est un domaine qui permet de découvrir et d'analyser des données avec différents outils afin de trouver des informations intéressantes et d'en tirer une conclusion.
Dans le cadre de notre cours, nous devions mener un projet d'étude en groupe où la seule contrainte était de trouver des données en lien avec le réchauffement climatique.
Après plusieurs recherches, nous en avons trouvé sur la production et la consommation d'énergie en France(Lien vers les données).

Aujourd'hui l'électricité est un élément essentiel de notre vie. Presque devenu vitale, On l'utilise au quotidien, dans tout, pour tout et partout! Actuellement, nous consommons à chaque seconde cet élément qui est presque devenue aussi précieux que de l'oxygène. Le besoin de plus en plus excessif de cette denrée a obliger l'homme à trouver des solutions pour une production massive de celle-ci, et ceux malgré les lourds impacts environnementaux.

Les conséquences nocives de cette production massive ont détruit notre environnement et cela est devenu une des causes de ce qu'on appelle aujourd'hui le **réchauffement climatique,** qui a à son tour créé des conséquences dangereuses pour l'homme tels que de longues canicules ou encore la fonte de glaces et bien d'autres effets.

Nous avons trouvé ce jeu de données sur le site **ODRÉ(OPENDATA RÉSEAUX-ÉNERGIES)** et nous l'avons analysé afin de tirer des informations en relation avec le dérèglement climatique. Ces données comportent les informations sur la consommation par région de la France depuis le 01-06-2022 jusqu' aujourd'hui.

Ce jeu de données, rafraîchi une fois par heure, présente des données régionales "temps réel" issues de l'application éCO2mix. Elles proviennent des télémesures des ouvrages complétées par des forfaits et estimations.

Vous y trouverez au pas quart d'heure:

- 12 régions de la France à savoir: *Auvergne-Rhône-Alpes, Bourgogne-Franche-Comté, Bretagne, Centre-Val de Loire, Grand-Est, Hauts-de-France, Île-de-France, Normandie, Nouvelle-Aquitaine, Occitanie, Pays de la Loire et Provence-Alpes-Côte d'Azur.*
- La consommation totale réalisée
- La production selon les différentes filières composant le mix énergétique
- La consommation des pompes dans les Stations de Transfert d'Energie par Pompage (STEP)
- Le solde des échanges physiques avec les régions limitrophes
- **TCO :** le Taux de COuverture (TCO) d'une filière de production au sein d'une région représente la part de cette filière dans la consommation de cette région
- **TCH :** le Taux de CHarge (TCH) ou facteur de charge (FC) d'une filière représente son volume de production par rapport à la capacité de production installée et en service de cette filière
- La date et l'heure de la prise des mesures

NETTOYAGE DES DONNÉES

Concernant le nettoyage des données, nous avons supprimé les colonnes qui à notre sens n'étaient pas essentielles, ces colonnes étaient soit vides, soit inutiles pour notre analyse (à savoir le déstockage et le stockage de batterie, la nature des données et une colonne nommée Column 68). Puis, nous avons gardé les colonnes suivantes: consommation totale, la production de chaque filière d'énergie , la date et l'heure, le solde des échanges avec les régions limitrophes, la part de chaque filière d'énergie dans la consommation de cette région et le volume de production par rapport à la capacité de production de cette région.

```
colonnes_a_supprimer=["Déstockage batterie","Stockage batterie",'Column 68','Nature']
data_nettoye= data.drop(colonnes_a_supprimer, axis = 1)
```

RÉALISATION DES GRAPHIQUES

Pour lire et traiter nos données tabulaires et pour tracer nos graphiques, on s'est servi des bibliothèques Python à savoir :

- *pandas*
- *seaborn*
- *geopandas*
- *matplotlib*
- *numpy*
- *scikit-learn (sklearn)*
- *calendar*

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
import geopandas as gpd
import calendar
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LinearRegression
from sklearn.metrics import mean_squared_error
```

Cette partie est répartie sur 3 parties et chacune représente le traitement, le tracé des graphiques de chaque membre du groupe et l'interprétation de chaque graphique.

Partie Alvin

Dans la première partie des graphiques de notre fichier notebook, on retrouve 2 graphiques représentant la consommation énergétique par région sur 2 dates différents sur une seule même heure et la corrélation entre la production d'une filière et sa part dans la consommation totale d'énergie.

Pour pouvoir avoir une même heure et une date précise j'ai utilisé des masques pour des données comportant une seule heure et date puis j'ai gardé que les premières observations de ce sous ensemble.

```
#Prenons une même date et une même heure pour afficher nos graphiques
heure= '15:00'
date= '2022-07-15'
date2= '2023-07-15'

#nettoyage pour les graphiques de la première partie
masque = (data_nettoye['Heure'] == heure)

data_15= data_nettoye[masque]

masque22= (data_15['Date'] == date)
masque23= (data_15['Date'] == date2)

data_15_juillet_2022= data_15[masque22]
data_15_juillet_2023= data_15[masque23]

data_first = data_15_juillet_2022.drop_duplicates(subset='Région', keep='first')
data_second = data_15_juillet_2023.drop_duplicates(subset='Région', keep='first')
```

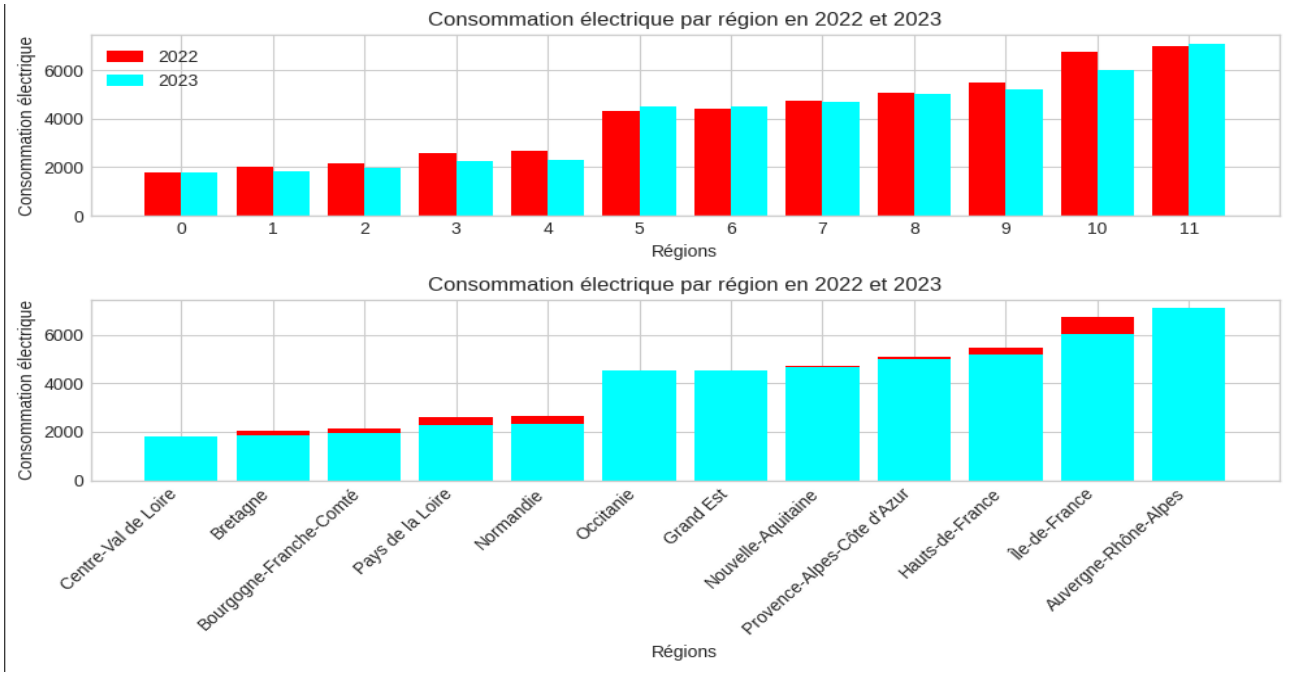
Avec ce masque, j'ai trié les données pour les afficher dans l'ordre croissant, puis j'ai divisé la fenêtre et rajouté un deuxième graphique juste en dessous pour bien visualiser la comparaison de la consommation des 2 dates(laquelle est supérieure à l'autre pour chaque région).

```
data_first = data_first.sort_values(by='Consommation (MW)')
data_second = data_second.sort_values(by='Consommation (MW)')

x = data_first['Région']
x2= data_second['Région']
y1 = data_first['Consommation (MW)']
y2 = data_second['Consommation (MW)']

# Création de positions pour les barres
x_pos = np.arange(len(x))
x_pos_1 = np.arange(len(x2))

#diviser la fenêtre pour afficher les graphiques
fig, ax = plt.subplots(2,1,figsize=(10, 6))  # 2 lignes, 1 colonne

# Largeur des barres
bar_width = 0.4
```

Puis j'ai enfin tracé le premier graphique(un diagramme en barre) à l'aide de la bibliothèque matplotlib.

```
# Premier graphique
ax[0].bar(x_pos - bar_width/2, y1, color="red", width=bar_width, label='2022')
ax[0].bar(x_pos_1 + bar_width/2, y2, color="cyan", width=bar_width, label='2023')

ax[0].set_xlabel("Régions")
ax[0].set_ylabel("Consommation électrique")
ax[0].set_title("Consommation électrique par région en 2022 et 2023")
ax[0].set_xticks(x_pos)
ax[0].set_xticks(x_pos_1)
ax[0].legend()

# Deuxième graphique
ax[1].bar(x_pos, y1, color="red", label='2022')
ax[1].bar(x_pos_1, y2, color="cyan", label='2023')

ax[1].set_xlabel("Régions")
ax[1].set_ylabel("Consommation électrique")
ax[1].set_title("Consommation électrique par région en 2022 et 2023")
ax[1].set_xticks(x_pos)
ax[1].set_xticklabels(x, rotation=45, ha='right')
ax[1].set_xticks(x_pos_1)
ax[1].set_xticklabels(x2, rotation=45, ha='right')

plt.tight_layout() # Pour éviter que les graphiques se chevauchent
plt.show()
```

Voici le résultat du code ci haut :



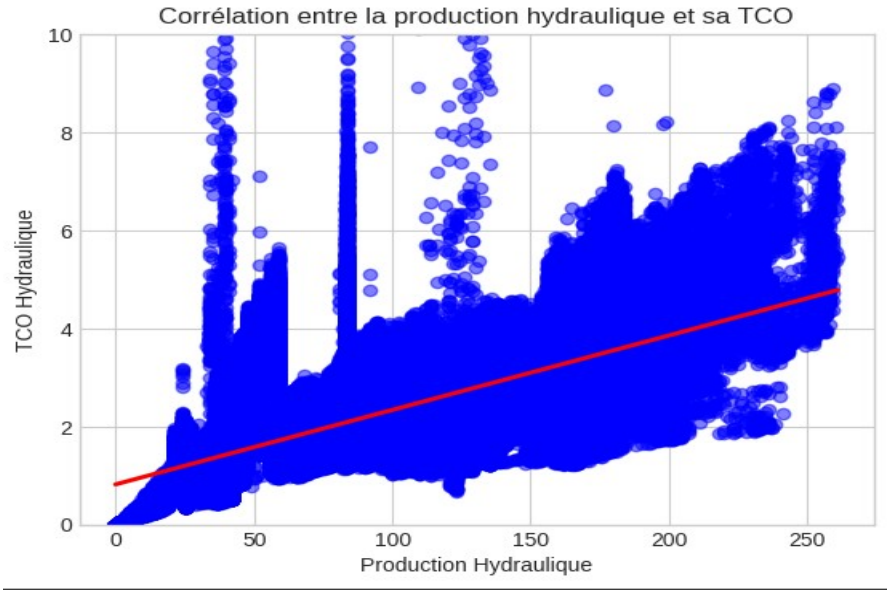
Ce premier graphique montre qu'à l'heure choisie (en l'occurrence 15h) pour les 2 journées choisies (15/07/2022 et 15/07/2023), on a une différence de la consommation totale d'énergie relevé qui montre que pour certaines régions, elles ont consommé moins que l'année d'avant et le contraire pour les autres sauf en Centre-Val de Loire qui semble avoir consommé la même quantité qu'il y a

un an. Les villes sont affichées par ordre croissant grâce à la méthode **sort_values()** pour mieux visualiser la région qui consomme moins et celle qui consomme plus.
On a globalement l'Île de France et Auvergne-Rhône-Alpes qui consomment beaucoup plus que les autres régions. En île de France, cela peut s'expliquer par le fait qu'elle contient la capitale de la France aussi connue comme la ville lumière(tout est dans le nom) et pour Auvergne-Rhône-Alpes, vue la date choisie, cela peut s'expliquer par son patrimoine naturel exceptionnel(grands espaces naturels, la beauté de ses paysages, ses volcans), considérée comme témoins uniques de l'histoire géologique. Elle s'avère aussi être la première région gastronomique en France ce qui peut attirer beaucoup de touristes surtout en été.

Puis on a le second graphique qui représente la corrélation entre la production hydraulique et sa part dans la consommation totale d'énergie. Je l'ai tracé à l'aide de la bibliothèque seaborn avec comme style « seaborn-whitegrid » et j'ai différencié la couleur de la droite de régression avec celle du nuage de points pour bien la distinguer.

```
plt.style.use('seaborn-whitegrid')  # Style des graphiques

# Création d'un nuage de points avec régression linéaire
sns.regplot(x='Bioénergies (MW)', y='TCO Bioénergies (%)', data=data_nettoye, scatter_kws={'alpha':0.5}, color='blue',line_kws={'color': 'red'})

# Ajout de titres et d'étiquettes d'axes
plt.title('Corrélation entre la production hydraulique et sa TCO')
plt.xlabel('Production Hydraulique')
plt.ylabel('TCO Hydraulique')
plt.ylim(0,10)

plt.show()
```

Pour enfin avoir le tracé suivant :



Sur ce second graphique, on remarque que de plus la production hydraulique augmente, de plus que sa part dans la consommation totale augmente ce qui est une bonne nouvelle, vu le thème autour duquel tourne cette analyse, car de plus qu'on augmentera la production hydraulique de plus sa TCO augmentera et ainsi les français consommeront de l'énergie de plus en plus verte.

Partie Michele

J’ai décidé de faire une analyse sur un an et j’ai choisi une heure précise , une date de début ,une date de fin et j’ai créé un masque pour extraire uniquement les données correspondant à mes critères.

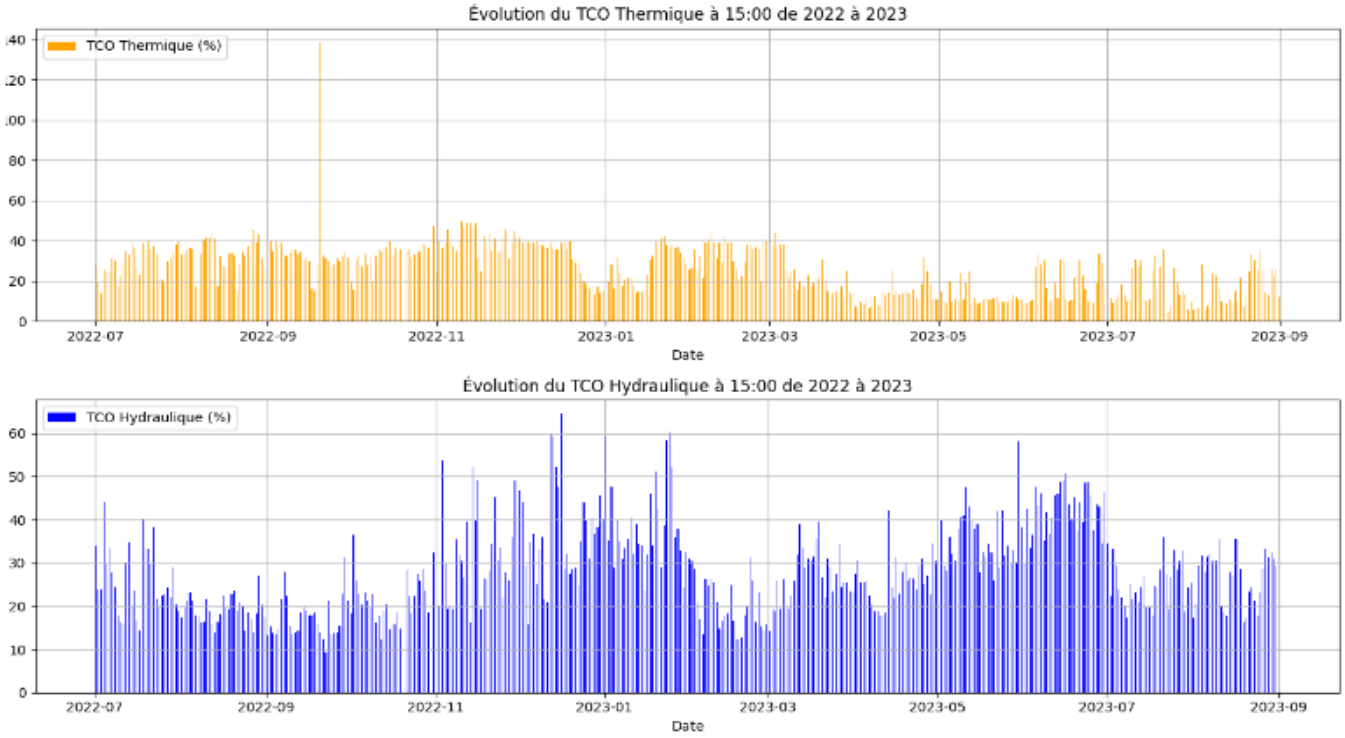
```
#intervalle de temps pour montrer l'évolution
date_debut = '2022-09-01'
date_fin = '2023-09-01'

#nettoyage pour les graphiques de la deuxième partie
mask = (data['Heure'] == heure) & (data['Date'] >= date_debut) & (data['Date'] <= date_fin)
data_selected_hour = data[mask]
```

J’ai ensuite converti la colonne 'Date' en type datetime pour faciliter la manipulation des dates.
J’ai créé une figure avec deux sous-graphiques rangés en deux lignes , une au dessus de l’autre afin d’avoir des résultats propres et bien lisibles. J’ai défini la largeur des barres dans le graphique. J’ai ajouté des éléments au graphique avec des lignes qui ajoutent un titre, des étiquettes d'axe, une légende et une grille au premier graphique. J’ai fait pareil pour le deuxième et pour finir j’ai ajusté la mise en page pour éviter le chevauchement des deux graphiques , puis je les ai affichés

```
# conversion de la colonne 'Date' en type datetime
data_selected_hour['Date'] = pd.to_datetime(data_selected_hour['Date'], errors='coerce')

fig, ax = plt.subplots(2, 1, figsize=(14, 8))
bar_width = 0.4

# Premier graphique
ax[0].bar(data_selected_hour['Date'], data_selected_hour["TCO Thermique (%)"], color="orange", width=bar_width, label="TCO Thermique (%)", align='center')
ax[0].set_title(f'Évolution du TCO Thermique à {heure} de 2022 à 2023')
ax[0].set_xlabel('Date')
ax[0].set_ylabel('Quantité (%)')
ax[0].legend()
ax[0].grid(True)

# Deuxième graphique
ax[1].bar(data_selected_hour['Date'] + pd.Timedelta(bar_width, unit='d'), data_selected_hour["TCO Hydraulique (%)"], color="blue", width=bar_width, label="TCO Hydraulique (%)", align='center')
ax[1].set_title(f'Évolution du TCO Hydraulique à {heure} de 2022 à 2023')
ax[1].set_xlabel('Date')
ax[1].set_ylabel('Quantité (%)')
ax[1].legend()
ax[1].grid(True)

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

Pour mon premier graphique, j’ai opté pour une représentation(à l’aide de matplotlib) en bâtons de l’évolution du TCO thermique et de l’ évolution du TCO Hydraulique. Sur l’axe des X sont marqués

les dates comme marqueur de temps et sur l'axe des Y le pourcentage de TCO. Pour rappel le TCO d'une énergie représente la part que ce type d'énergie représente sur la consommation d'énergie totale.
Nous savons que les énergies non renouvelables sont les plus utilisées mais toute fois les conséquences de notre pollution nous ont poussés vers les énergies renouvelables et nous voulons peu à peu améliorer notre impact environnementale.
Lorsque j'ai lancé mon code, j'ai obtenu ces graphiques :

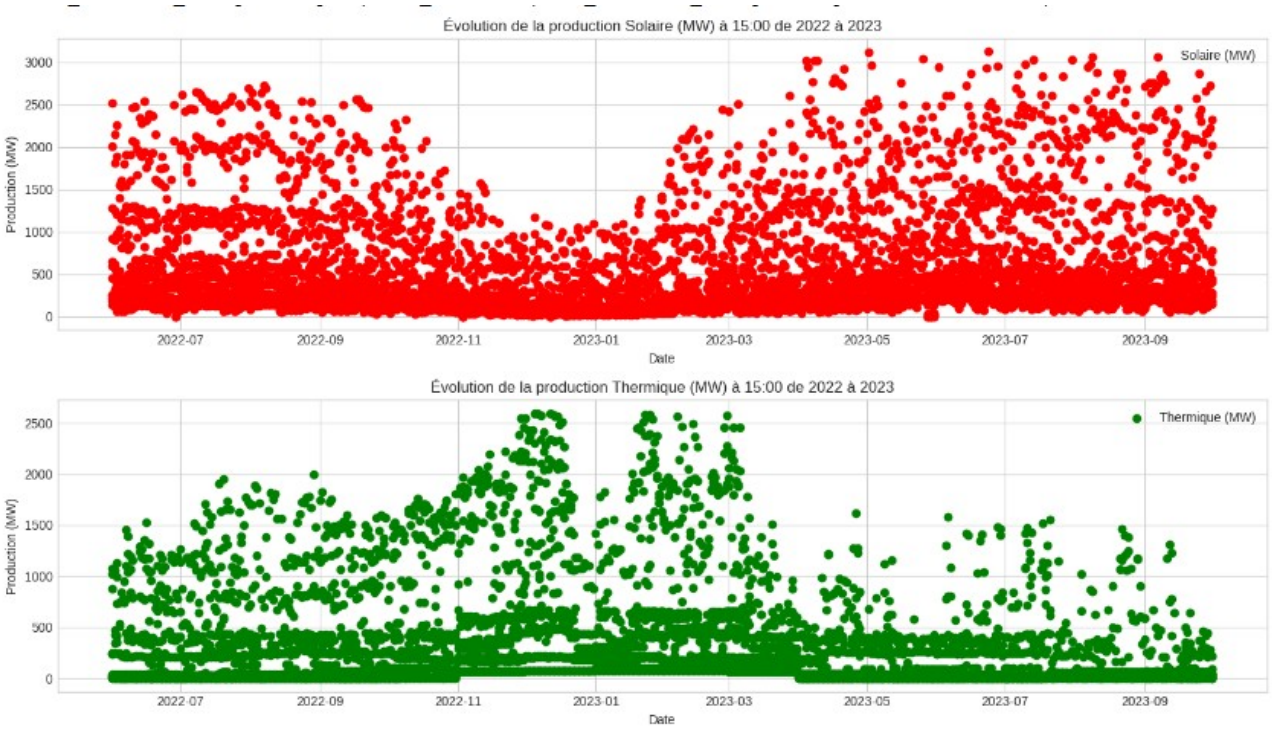


Nous voyons sur le premier graphique en orange que le taux de TCO thermique a diminué d'environ 10 - 20 % sur un an. À l'inverse, on remarque que le taux de TCO Hydraulique a augmenté aussi d'environ 10 % sur un an. Nous savons que la production d'énergie thermique pollue presque autant que la production d'énergie nucléaire. Ces données montre que la France de manière général est en train de s'améliorer d'année en année et d'intégrer de plus en plus les énergies renouvelables dans leur consommation. Puisque le taux d'énergie hydraulique dans le taux d'énergie total consommé en France a augmenté et le TCO thermique diminue , ça signifie que la France consomme un peu moins d'énergie thermique qu'il y a un an et consomme un peu plus d'énergie renouvelable comme l'hydraulique. Donc la France améliore son impact environnemental et contribue de moins en moins à la pollution et donc au réchauffement climatique.

Pour le deuxième graphique , j'ai trié mes données de la même façon que le premier ,en choisissant des données à une heure et sur une période précise. Mais j'ai opté comme graphique un nuage de points tracé à l'aide de la bibliothèque matplotlib:

```python
#modifions la durée d'affichage
date_debut = '2022-04-01'
date_fin = '2023-10-01'

mask = (data['Heure'] == heure) & (data['Date'] >= date_debut) & (data['Date'] <= date_fin)
data_selected_hour = data[mask]

data_selected_hour['Date'] = pd.to_datetime(data_selected_hour['Date'], errors='coerce')

fig, ax = plt.subplots(2, 1, figsize=(14, 8))

# Assigner une couleur différente à chaque énergie produite
colors = { 'Solaire (MW)': 'red', 'Thermique (MW)':'green'}  # Ajoutez d'autres énergies au besoin

for i, (energy, color) in enumerate(colors.items()):
    ax[i].scatter(data_selected_hour['Date'], data_selected_hour[energy], label=energy, color=color)
    ax[i].set_title(f'Évolution de la production {energy} à {heure} de 2022 à 2023')
    ax[i].set_xlabel('Date')
    ax[i].set_ylabel('Production (MW)')
    ax[i].legend()
    ax[i].grid(True)

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

J’ai obtenu comme résultat ces graphiques:



Le premier graphique montre que sur une période d’un an et 2 mois, La production d’énergie solaire a augmenté d’environ 500MV par mois. Nous savons que plus il fait chaud, plus la production d’energie solaire augmente. Avec ce graphique en rouge ,on voit très clairement une preuve de présence du réchauffement climatique.

Le deuxième graphique en vert nous montre l'évolution de la production d'énergie thermique qui a baissé d'environ 500MW. Cela est une bonne nouvelle car ça montre une fois de plus que la France diminue peu à peu la production d'énergie non renouvelable et s'améliore dans la production d'énergie renouvelable.

Partie Berline

Je me suis intéréssée à la façon dont la consommation Bioénergies varie au cours des mois et à la prédiction des consommations pour le futur.

Pour ce faire j'ai utilisé deux bibliothèques Python : Calendar et Scikit-learn.

- *Calendar* est une librairie de fonctions qui m'a été utile à l'extraction du mois de la colonne 'Date' en une nouvelle colonne 'Mois'.

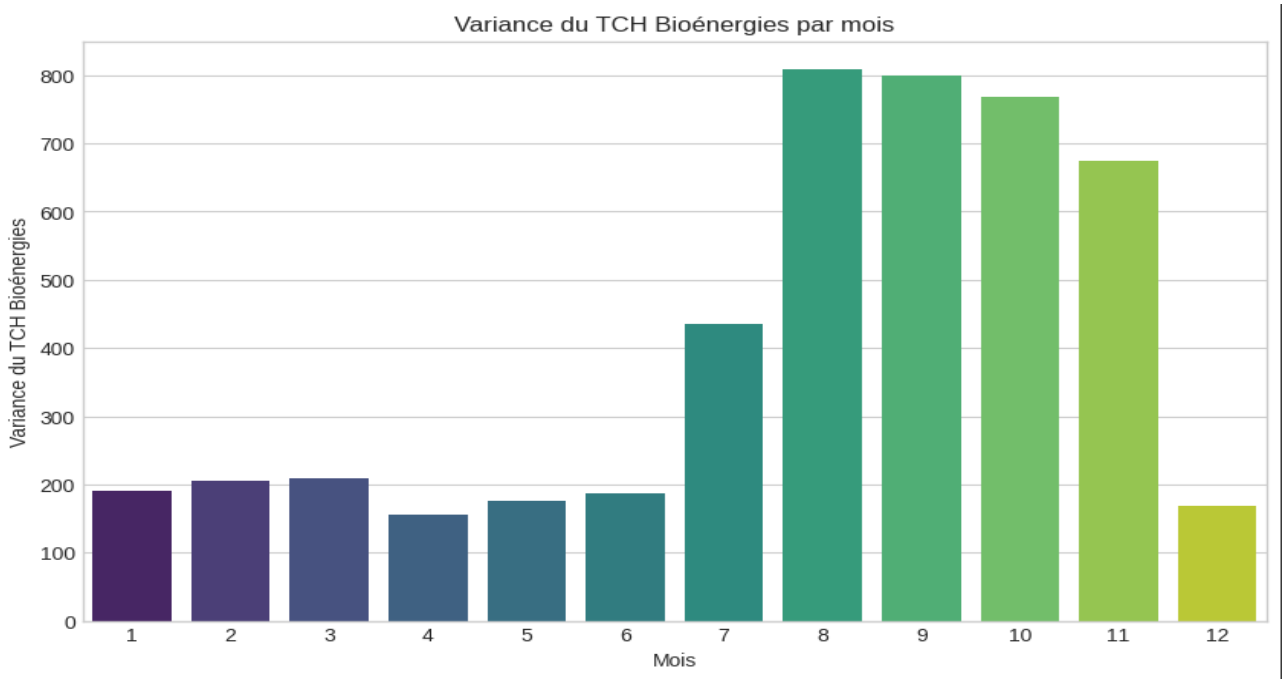
```
data['Mois'] = pd.to_datetime(data['Date']).dt.month
```

Après avoir une colonne de mois, le calcul de la variance de la consommation Bioénergies a été possible en groupant les taux de productions par mois.

```
moyenne_tch_par_mois = data.groupby('Mois')['TCH Bioénergies (%)'].mean()
variance_tch_par_mois = data.groupby('Mois')['TCH Bioénergies (%)'].var()
variance_df = variance_tch_par_mois.reset_index(name='Variance TCH Bioénergies')
```

La visualisation des résultats a pu être possible grâce au graphique à barres de la bibliothèque seaborn.

```
plt.figure(figsize=(10, 6))
sns.barplot(x='Mois', y='Variance TCH Bioénergies', data=variance_df, palette='viridis')

plt.xlabel('Mois')
plt.ylabel('Variance du TCH Bioénergies')
plt.title('Variance du TCH Bioénergies par mois')

plt.show()
```



Au cours de l'année 2023, on a observé une hausse de température exagérée pendant l'été par rapport à l'année 2022, d'où une variance élevée pour de Juillet à Novembre.
De Janvier à Juin, la variance n'est pas aussi grande car notre jeu de données concerne une période commençant le 1er juin 2022 à aujourd'hui. Et vu qu'on n'est pas encore arrivé au mois de Décembre on peut faire une comparaison de Décembre 2022 et Décembre 2023.

- *Scikit-learn* est une des librairies utiliser en python pour faire des prédictions grâce au machine learning
  Source: https://www.youtube.com/watch?v=oLlN5SYZr5g,
  https://www.youtube.com/watch?v=Eqv98w1ukZk

Pour la prédiction j'ai du séparer le jeu de données en 2 de façon aléatoire et appliquer le modèle de régression linéaire sur le jeu de données entrain. De là j'ai pu calculer les prédictions sur la consommation.

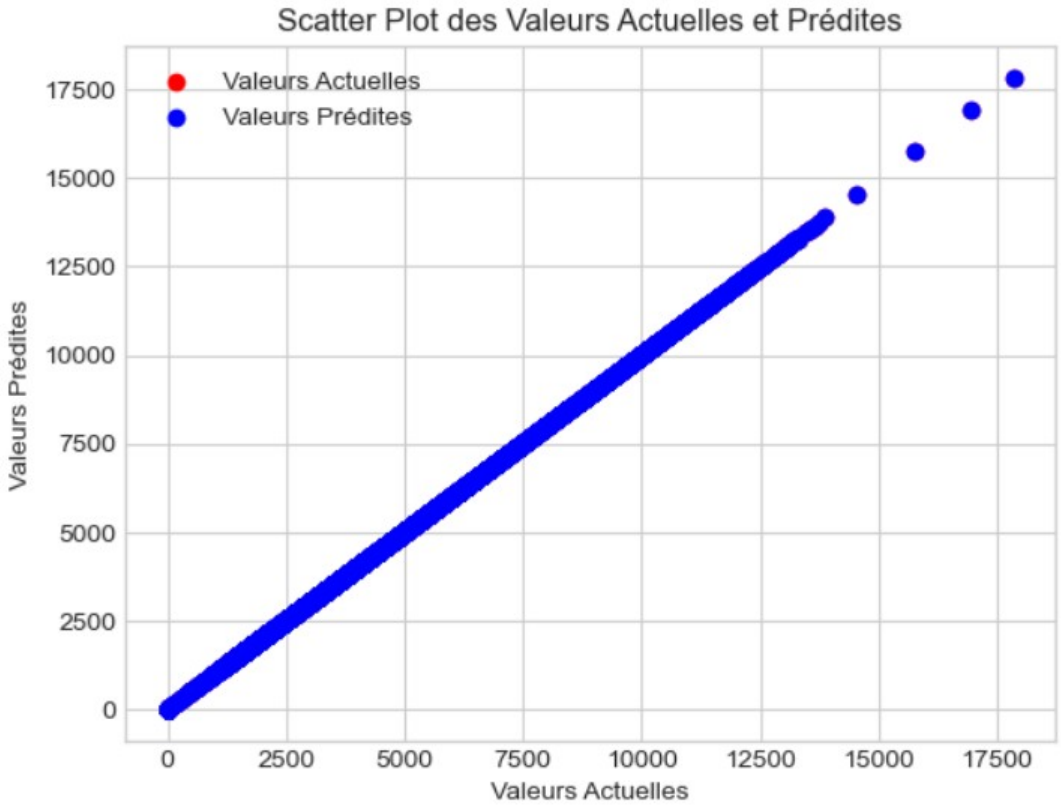
```
colonnes= ['Consommation (MW)', 'Mois']
data_a_predire = data[colonnes]
variable_a_predire = data["Consommation (MW)"]

X =data_a_predire['Consommation (MW)'].dropna()
y=variable_a_predire.dropna()

X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=123)

X_train = X_train.values.reshape(-1, 1)
X_test = X_test.values.reshape(-1, 1)

model = LinearRegression()
model.fit(X_train, y_train)
y_pred_test = model.predict(X_test)
error_metric = mean_squared_error(y_pred = y_pred_test, y_true = y_test)
print(error_metric)
```

```
1.816673423813159e-23
```

Par après, il fallait évaluer le modèle et le code ci-dessus m'a permis de calculer l'erreur moyenne entre les valeurs prédites et les valeurs réelles, l'error_metric (Root Mean Squarred Error), mesure qui est de 1.816673423813159e-23 pour mon modèle. Cela prouve que mon modèle est bien performant car la mesure d'erreur est presque égale à zéro.

Pour interpreter cela j'ai trouvé intéressant de regarder la corrélation entre les valeurs actuelles et les valeurs prédites avec le graphique de nuage de points de la bibliothèque matplotlib.

```
plt.scatter(X_test, y_test, color='red', label='Valeurs Actuelles')

plt.scatter(X_test, y_pred_test, color='blue', label='Valeurs Prédites')

plt.xlabel('Valeurs Actuelles')
plt.ylabel('Valeurs Prédites')
plt.title('Scatter Plot des Valeurs Actuelles et Prédites')
plt.legend()

plt.show()
```



Avec cette figure, on prouve qu'il y a corrélation entre les valeurs actuelles (de 2022 à 2023) et les valeurs prédites. Une corrélation positive suggère que les valeurs prédites augmentent ou diminuent de manière cohérente avec les valeurs réelles, cela signifie généralement que le modèle de régression linéaire construit est capable de capturer une partie de la variation dans les données. Une corrélation positive entre les valeurs réelles et prédites suggère que le modèle suit la tendance générale des données.