TD Wind Energy

Description:

Dans ce TD, vous allez mettre en pratique vos nouvelles connaissances pour analyser un jeu de données réel provenant d'une éolienne. Pour réussir ce TD, vous devrez importer le dataset dans Google Colab ou autre outil convenable, et charger les bibliothèques nécessaires telles que Numpy,Pandas,Matplotlib et Seaborn. Vous répondrez ensuite à une série de questions portant sur la manipulation,l'analyse et visualisation des données, afin d'identifier et de comprendre les tendances et motifs présents dans les données.

Les données sont collectées à l'aide de capteurs intelligents et de systèmes SCADA (Supervisory Control and Data Acquisition), puis agrégées sur des intervalles de 10 minutes.

Les variables du dataset sont les suivantes :

- **Date/Time** : date et heure des mesures, avec un pas de 10 minutes.
- **LV Active Power (kW)** : puissance réellement produite par l'éolienne à un instant donné.
- **Theoretical Power (kWh)** : puissance théorique que l'éolienne pourrait produire pour une vitesse de vent donnée, fournie par le fabricant. Elle correspond à la **puissance maximale possible**, calculée selon la **loi de Betz**.
- **Wind Speed (m/s)** : vitesse du vent mesurée à la hauteur du moyeu de l'éolienne (vitesse utilisée pour la production d'électricité).
- **Wind Direction (°)** : direction du vent à la hauteur du moyeu (l'éolienne s'oriente automatiquement dans cette direction).

Questions :

1.1 Chargement des données

- Votre première tâche consiste à **importer les bibliothèques Python nécessaires** pour réaliser ce TD.
- Ensuite, chargez le fichier de données `T1.csv` dans un DataFrame Pandas nommé `df_wind`.
  ➢ **Quelle colonne devez-vous utiliser comme index ? Pourquoi ?**

1.2 Exploration des données

Dans cette partie, vous allez explorer le jeu de données afin de **comprendre sa structure et son contenu**.

- Quelle est la **taille** du DataFrame chargé ?
- Affichez les **5 premières lignes** et les **5 dernières lignes** du DataFrame `df_wind`.
  ➤ Que pouvez-vous en déduire ? Pour **quelle(s) année(s)** l'éolienne a-t-elle enregistré des données ?
- Affichez le **type de données** de chaque colonne du dataset.
- Vérifier la présence de valeurs manquantes.
- Extraire la colonne Wind Speed sous forme NumPy.
- Calculez la **moyenne** et l'**écart-type** de la variable **vitesse du vent** (*Wind Speed*).

1.3 Manipulation des données

- **Renommer les colonnes** comme suit :

  ➤ `Power` au lieu de `LV Active Power (kW)`

  ➤ `Wind speed` au lieu de `Wind Speed (m/s)`

  ➤ `Theoretical power` au lieu de `Theoretical Power (kWh)`

  ➤ `Wind direction` au lieu de `Wind Direction (°)`

- Tracez l'**évolution de la puissance produite** (par intervalles de 10 minutes) en fonction du **temps**.
- Créez une nouvelle colonne appelée **`loss`**, correspondant à la **différence entre la puissance théorique et la puissance réellement produite**. Puis, tracez la **perte d'énergie en fonction du temps**.
  ➤ **Que pouvez-vous en conclure ?**
- Une éolienne commence généralement à produire de l'électricité pour une vitesse de vent comprise entre **12 et 14 km/h** (soit **3,3 à 3,8 m/s**). Si la vitesse du vent est **supérieure à 3,3 m/s** et que la puissance produite est **nulle ou négative**, on peut supposer que l'éolienne est **en maintenance**.
  ➤ Identifiez les lignes correspondant à cette condition.
  ➤ Créez un nouveau DataFrame appelé **`df_wind_nm`** (sans maintenance), excluant ces périodes.

```
df_wind_nm = df_wind[ ~((df_wind['Power']<=0)
& (df_wind['Wind_speed'] > 3.3))]
```

- Ensuite, tracez la **perte d'énergie en fonction du temps** pour le DataFrame `df_wind_nm`.
- Tracez la **différence de perte d'énergie** entre les DataFrames **avec maintenance (`df_wind`)** et **sans maintenance (`df_wind_nm`)**.

  ➤ **Quelle conclusion peut-on en tirer ?**

- Tracer la densité de probabilité (KDE) de la puissance et de la vitesse du vent du DataFrame sans maintenance (**`df_wind_nm`**).
- On souhaite maintenant créer les **composantes x et y du vent** à partir de la vitesse et de la direction du vent.

  ➤ Les équations utilisées sont :

$$x = Ws \times \cos\left(\frac{Wd \times \pi}{180}\right)$$

$$y = Ws \times \sin\left(\frac{Wd \times \pi}{180}\right)$$

où :

- Ws représente la **vitesse du vent**
- Wd représente la **direction du vent**

Ajoutez ces deux nouvelles colonnes (**x** et **y**) au dataset.

```
def x_y_component(wind_direction, wind_speed):
    """
    Convert degrees to x,y components
    """
    #convert to radians
    radians = (wind_direction * np.pi)/180
    # give the x, y compenents
    x = wind_speed * np.cos(radians)
    y = wind_speed * np.sin(radians)

    return x,y
```

- Afin de réduire l'effet du **bruit aléatoire**, les données doivent être **rééchantillonnées**.

```
X_df_wind = df_wind.resample('X').mean()
```

➤ Créez des DataFrames **horaire**, **journalier**, **hebdomadaire** et **mensuel** à partir du DataFrame sans maintenance, en utilisant la **moyenne**.

➤ Utilisez la méthode **`resample()`** de Pandas, avec :

  - `'h'` pour l'heure
  - `'d'` pour le jour
  - `'W'` pour la semaine
  - `'ME'` pour le mois

➤ Tracez ensuite l'évolution de la **puissance produite** pour chaque fréquence sous forme de **sous-graphiques(sub-plots)**.

➤ Vous pouvez également utiliser la fonction **`asfreq()`**.

➤ **Quelle est la différence entre `resample()` et `asfreq()` ?**

- Remplacez toutes les valeurs NaN par des valeurs interpolées dans les DataFrames horaires, journaliers, hebdomadaires et mensuels, en utilisant la fonction interpolate().
- Ajoutez et utilisez une fonction nommée **`direction`** afin de créer une nouvelle colonne **catégorielle** appelée **`Direction`**. Cette fonction prend en entrée la **vitesse du vent**.
- Ajoutez cette colonne aux DataFrames **journalier**, **hebdomadaire** et **mensuel** issus du DataFrame sans maintenance (`df_wind_nm`).

```
def direction(x):
    if x > 348.75 or x<11.25: return 'N'
    if x < 33.75: return 'NNE'
    if x < 56.25: return 'NE'
    if x < 78.75: return 'ENE'
    if x < 101.25: return 'E'
    if x < 123.75: return 'ESE'
    if x < 146.25: return 'SE'
    if x < 168.75: return 'SSE'
    if x < 191.25: return 'S'
    if x < 213.75: return 'SSW'
    if x < 236.25: return 'SW'
    if x < 258.75: return 'WSW'
    if x < 281.25: return 'W'
    if x < 303.75: return 'WNW'
    if x < 326.25: return 'NW'
    else: return 'NNW'
```

- Tracez l'évolution de la **puissance produite** du DataFrame horaire durant le **mois de décembre**. Puis, tracez un second graphique correspondant aux **deux dernières semaines de décembre**.
- Tracez la **puissance moyenne produite** à l'échelle **hebdomadaire et mensuelle** sur un **même graphique**.
- Appliquez le code fourni.
  ➢ **Que pouvez-vous en conclure ?**

```
m_s_t_se = df_wind_nm.groupby('Wind_speed')['Theoretical_power'].mean()
m_s_p_se = df_wind_nm.groupby('Wind_speed')['Power'].mean()
plt.figure(figsize=(10,5))
plt.plot(m_s_t_se,label='Theoretical')
plt.plot(m_s_p_se,label='Actual Power')
plt.xlabel('Wind Speed (m/s)')
plt.ylabel('Power (kW)')
plt.title('Theoretical Power vs Actual Power Curve')
plt.grid()
plt.legend()
plt.show()
```

- Tracer un pair-plot pour le DataFrame filtré. Quelles conclusions pouvez-vous en tirer ?
- Tracer la carte de chaleur des corrélations. Que pouvez-vous en conclure ?
  - ★ Quelles variables ont une corrélation plus forte avec la variable 'Power' ? Pouvez-vous les lister ?
  - ★ Quelles variables n'ont pas de corrélation avec 'Power' ? Pouvez-vous les lister ?
  - ★ Pourrions-nous obtenir les mêmes résultats si d'autres méthodes de corrélation telles que le « coefficient de corrélation de Kendall Tau » ou la « corrélation par rang de Spearman » étaient utilisées

- Tracer la perte d’énergie en fonction de la direction du vent. Que pouvez-vous en conclure ?
- Créer un nuage de points 3D avec les variables (x, y, z) : où ‘wind speed’ (vitesse du vent), ‘wind direction’ (direction du vent) et ‘Power’ (puissance) représentent respectivement les axes x, y et z. Que pouvez-vous en conclure ?