



PRÉDICTION DE TEMPÉRATURE AVEC ALGORITHMES DE MACHINE LEARNING

PRÉPARÉ PAR: HABIB SAMYA
RAMI HALA
BOUDRIBILA KAOUTAR

Date de remise: 31/12/2024

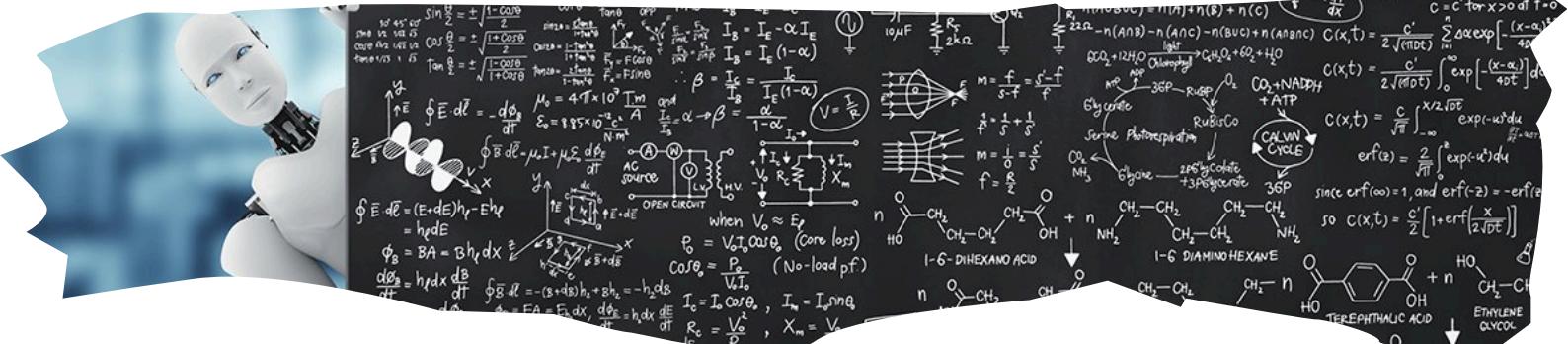


TABLE DE MATIÈRE:

• INTRODUCTION :

- Objectif du projet.....01
- Contexte et probmématiche.....01
- Importance de l'approche uilisée.....02

• DONNÉES & MÉTHODOLOGIE :

- Description des données.....03
- Méthode adptée.....03

• IMPLÉMENTATION & CODE :

- Bibliothèques utilisées.....04
- Présentation des codes.....04

• ANALYSE DES RÉSULTATS :

- Représentation graphique.....10
- Résultats de la console.....15

• CONCLUSION & DISCUSSION:.....19



INTRODUCTION:

- **OBJECTIF DU PROJET :**

L'objectif principal de ce projet est de prédire avec précision la température moyenne d'un mois donné en se basant sur des données météorologiques historiques, en utilisant des approches avancées de machine learning. Ce travail vise à explorer et comparer l'efficacité de différents algorithmes de régression, tels que la régression linéaire, et les forêts aléatoires, afin d'identifier celui qui offre les meilleures performances pour cette tâche spécifique. En détail, le projet met l'accent sur la collecte et le traitement rigoureux des données météorologiques provenant de sources fiables comme OpenWeatherMap, NOAA ou Kaggle. Les données, une fois nettoyées et préparées, serviront à extraire des caractéristiques pertinentes, telles que des moyennes mobiles et des tendances temporelles, afin d'alimenter les modèles prédictifs. Une analyse exploratoire approfondie sera également réalisée pour visualiser les corrélations entre les variables comme la température, l'humidité et les précipitations, ce qui permettra de mieux comprendre les facteurs influents. Enfin, le projet aboutira à la construction d'un pipeline de machine learning complet, intégrant le prétraitement, la sélection des caractéristiques, l'entraînement des modèles, ainsi que leur évaluation selon des métriques comme le RMSE, le MAE et le R². Ce processus permettra de comparer les forces et les limites de chaque modèle tout en offrant des visualisations claires des prédictions et des erreurs, renforçant ainsi la pertinence et l'applicabilité des résultats pour des secteurs impactés par les conditions climatiques.

- **CONTEXTE ET PROBLÉMATIQUE:**

Avec les impacts croissants du changement climatique, prévoir les températures futures est devenu essentiel pour de nombreux secteurs comme l'agriculture, l'énergie et les transports. Ce projet vise à utiliser des données météorologiques historiques et des algorithmes de machine learning pour prédire efficacement la température moyenne d'un mois donné, tout en comparant les performances de plusieurs modèles afin d'identifier le plus adapté.



• IMPORTANCE DE L'APPROCHE UTILISÉE:

L'approche utilisée dans ce projet repose sur l'application de techniques avancées de machine learning pour prédire la température moyenne d'un mois donné à partir de données météorologiques historiques. Cette méthodologie est particulièrement importante pour plusieurs raisons liées à l'énoncé du projet :

1. **Exploitation de données climatiques historiques** : Ces données sont riches en informations sur les tendances passées et les variations saisonnières, ce qui permet de modéliser les comportements climatiques avec précision. L'utilisation d'algorithmes de machine learning permet de détecter des relations complexes entre les variables (température, humidité, précipitations) qui pourraient être difficiles à identifier avec des méthodes traditionnelles.
2. **Comparaison des performances des modèles** : En mettant en œuvre différents algorithmes (régression linéaire, random forest, etc.), l'approche permet d'évaluer leurs forces et leurs limites. Cela aide à sélectionner l'algorithme le plus performant pour ce type de prédiction, en tenant compte de critères tels que la précision des prédictions et la simplicité du modèle.
3. **Pipeline automatisé** : L'intégration d'un pipeline complet, depuis la collecte des données jusqu'à l'évaluation des modèles, garantit une reproductibilité et une efficacité dans le traitement. Cela facilite également l'extension ou la modification du processus pour d'autres types de données ou prédictions.
4. **Mesures d'évaluation claires** : Les métriques comme le RMSE, le MAE et le R² fournissent des indications précises sur la qualité des prédictions, permettant de quantifier l'impact de chaque modèle et d'optimiser leurs hyperparamètres pour améliorer les résultats.

En s'appuyant sur ces étapes bien structurées et les principes énoncés, le code fourni met en pratique ces concepts de manière concrète. Par exemple, il traduit le processus de nettoyage des données, de sélection des caractéristiques et d'entraînement des modèles en un workflow automatisé et reproductible. Cela montre l'efficacité de l'approche dans la gestion des données réelles et dans l'obtention de résultats exploitables pour des cas d'utilisation pratiques.



DONNÉES & MÉTHODOLOGIE:

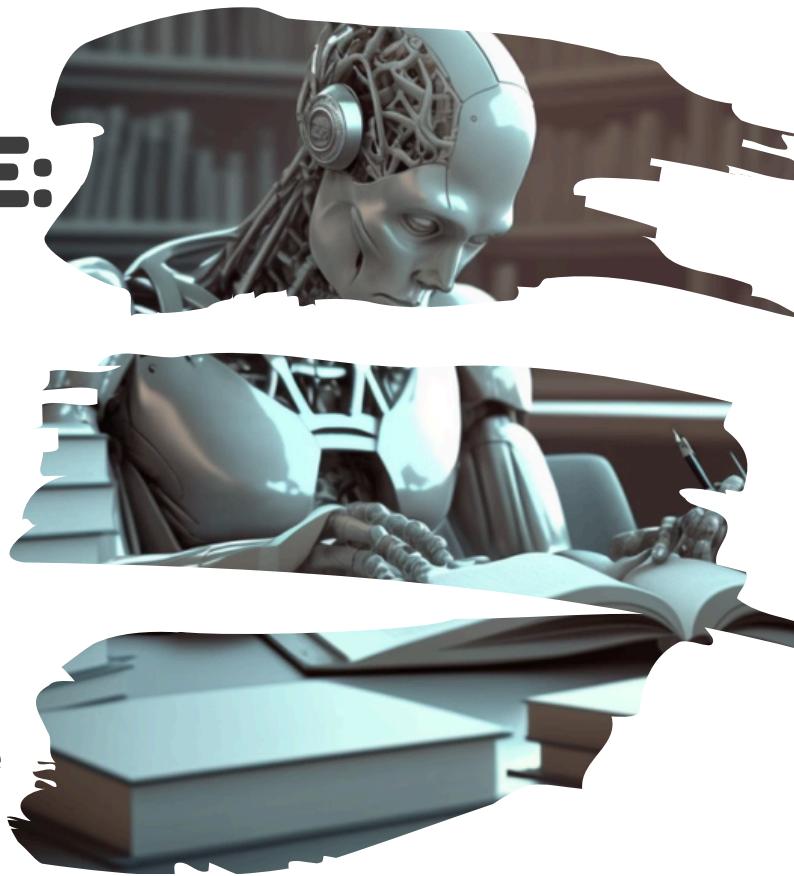
• DESCRIPTION DE DONNÉES:

Le jeu de données utilisé pour ce projet a été récupéré depuis la plateforme Kaggle, reconnue pour sa richesse en données variées et fiables. Il contient des informations météorologiques historiques, telles que la température moyenne journalière, l'humidité et les précipitations, couvrant une période de 10 ans. La méthodologie adoptée se décompose en plusieurs étapes clés : collecte et structuration des données en format CSV, nettoyage pour gérer les valeurs manquantes et anomalies, extraction de caractéristiques pertinentes pour enrichir l'analyse, et enfin, mise en œuvre d'un pipeline de machine learning pour entraîner et évaluer différents modèles prédictifs. Cette approche garantit une exploitation optimale des données pour répondre à la problématique posée.

• MÉTHODE ADOPTÉE:

La méthode adoptée dans le code suit un pipeline structuré en plusieurs étapes :

1. Chargement et nettoyage des données : Les données sont importées depuis un fichier CSV, puis nettoyées pour gérer les valeurs manquantes et éliminer les anomalies.
2. Extraction des caractéristiques : De nouvelles variables sont générées, comme les moyennes mobiles et les indicateurs temporels (mois, jour), afin d'enrichir le jeu de données.
3. Division des données : Le jeu de données est séparé en ensembles d'entraînement et de test pour évaluer les performances des modèles.
4. Modélisation : Différents algorithmes de machine learning, tels que la régression linéaire et la forêt aléatoire, sont appliqués pour créer des modèles prédictifs.
5. Évaluation des performances : Les modèles sont évalués à l'aide de métriques comme le RMSE, le MAE et le R², permettant une comparaison objective de leur efficacité.



IMPLÉMENTATION & CODE:



- **BIBLIOTHÈQUES UTILISÉES:**

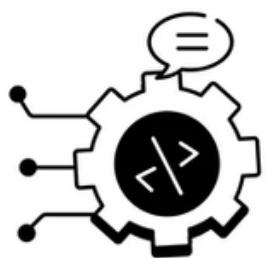
Les bibliothèques Python utilisées pour ce TP sont :
pandas, matplotlib, sklearn, numpy et glob

- **PRÉSENTATION DES CODES :**

```
import glob
import pandas as pd
import matplotlib.pyplot as plt
from sklearn.model_selection import train_test_split, cross_val_score
from sklearn.linear_model import LinearRegression
from sklearn.ensemble import RandomForestRegressor
from sklearn.metrics import mean_squared_error, mean_absolute_error, r2_score
import numpy as np
```

Dans cette partie du code, on importe toutes les librairies dont on aura besoin.

- **pandas** : Permet de manipuler et d'analyser facilement des données structurées sous forme de tableaux, comme les DataFrames.
- **matplotlib** : Utilisée pour créer des visualisations graphiques, telles que des courbes, des histogrammes et des diagrammes.
- **sklearn (scikit-learn)** : Fournit des outils performants pour l'apprentissage automatique, notamment pour la modélisation et l'évaluation des algorithmes.
- **numpy** : Facilite les calculs numériques avancés, en particulier pour manipuler des tableaux multidimensionnels et effectuer des opérations mathématiques.
- **glob** : Permet de rechercher des fichiers ou des chemins selon un motif spécifique, utile pour automatiser la gestion de fichiers.



- Cette section utilise glob pour trouver tous les fichiers CSV dans le répertoire courant, puis les charge et les combine en un seul DataFrame avec pandas.

```
csv_files = glob.glob('*.{!}.'.format('csv'))
print(csv_files)
weather = pd.concat([pd.read_csv(f) for f in csv_files], ignore_index=True)
```

- Définit la colonne datetime comme index pour faciliter la manipulation temporelle.
- Affiche des statistiques descriptives et des informations sur les colonnes pour analyser les données.
- Supprime les colonnes inutiles.

```
#set the datetime column as the index of the DataFrame
weather.set_index(keys='datetime', inplace=True)

# print the head and tail of the DataFrame to make sure that all 11 years of data are present.
print(weather.head())
print(weather.tail())

#get statistical summary of the data
print(weather.describe())

#check the data types of the columns
print(weather.info())

# print(weather.loc["2018-01-01", :]

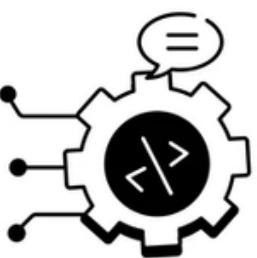
#Drop columns that are not needed
#drop name because we are working on one city
weather.drop(columns=['name', 'feelslikemax', 'feelslikemin', 'severerisk'], inplace=True)
```

- Identifie les valeurs manquantes et les remplit avec des valeurs appropriées pour éviter les erreurs dans les étapes suivantes.

```
#cleaning data
#check missing values
#Adding "/weather.shape[0]" to the end of the code will give you the proportion of missing values in each column.
print(weather.apply(pd.isnull).sum()/weather.shape[0])
#the results show that most of the columns are not null values,
#and therefore, we won't have to drop any columns

# Fill missing values
print(weather["preciptype"].value_counts())
print(pd.isnull(weather["preciptype"]).sum())

#fill all missing values of preciptype with "rain/snow"
weather["preciptype"].fillna("None", inplace=True)
#fill all missing values of windgust with 0, considering no windgust was recorded
weather['windgust'].fillna(0, inplace=True)
```



- Transforme des colonnes en données numériques ou catégoriques pour optimiser la mémoire et la performance du modèle.
- Convertit des colonnes et l'index en format datetime pour une meilleure manipulation temporelle.

```
#convert the columns that are not numerical to numerical
weather['preciptype'] = weather['preciptype'].map({'rain': 1, 'snow': 2, 'rain,snow': 3})
weather['sunrise'] = pd.to_datetime(weather['sunrise'], errors='coerce')
weather['sunset'] = pd.to_datetime(weather['sunset'], errors='coerce')

#turning data into categorical data, to make it easier for the model to interpret
# and to optimize the model's performance, and the memory usage of the DataFrame.
categorical_columns = ['conditions', 'description', 'icon', 'stations']
for col in categorical_columns:
    weather[col] = weather[col].astype('category')

#turn the datetime column into a datetime object, for better manipulation of the data
# we can even use 'weather.index.year' to search for data from a specific year as an example
weather.index = pd.to_datetime(weather.index)
```

- Génère des graphiques pour analyser les tendances de température, précipitations et les totaux annuels.

```
#-----Move to the Analysis step-----

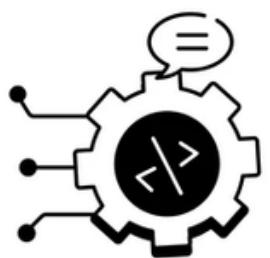
#save the cleaned data to a new csv file
# weather.to_csv("weather_cleaned.csv")

#plot the temperature min and max over time
weather[["tempmin", "tempmax"]].plot(kind='line', figsize=(10, 5), title='Temperature min and max over time')
plt.show()

#plot the precipitation over time
weather["precip"].plot(kind='line', figsize=(10, 5), title='Precipitation over time')
plt.show()

#plot the snow over time
weather["snow"].plot(kind='line', figsize=(10, 5), title='Snow over time')
plt.show()

#How much did it rain each year?
weather.groupby(weather.index.year)[["precip"]].sum().plot(kind='bar', figsize=(10, 5), title='Total precipitation each year')
plt.show()
```

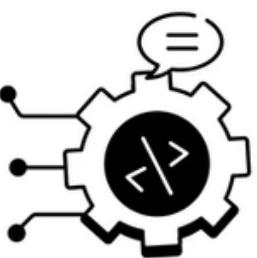


- Ajoute des colonnes basées sur des informations temporelles (mois, jour, année).
- Calcule des caractéristiques supplémentaires, comme les décalages (lag) et les moyennes glissantes.

```
# -----Start of the Modeling step-----  
  
# Feature extraction  
weather['month'] = weather.index.month  
weather['day'] = weather.index.day  
weather['year'] = weather.index.year  
  
# Lag features  
weather['tempmin_lag1'] = weather['tempmin'].shift(1)  
weather['tempmax_lag1'] = weather['tempmax'].shift(1)  
  
# Rolling averages  
weather['precip_rolling_mean'] = weather['precip'].rolling(window=7).mean()  
  
#Splitting the data into training and testing sets  
  
# Drop rows with NaN values introduced by lag and rolling calculations  
weather.dropna(inplace=True)
```

- Sépare les données en ensembles de caractéristiques (features) et cible (target).
- Divise les données en ensembles d'entraînement et de test avec une proportion de 80/20.

```
# Select features and target  
features = ['tempmin', 'tempmax', 'humidity', 'precip', 'month', 'day', 'tempmin_lag1', 'tempmax_lag1', 'precip_rolling_mean']  
target = 'temp' # Predicting maximum temperature  
  
X = weather[features]  
y = weather[target]  
  
# Split data  
X_train, X_test, y_train, y_test = train_test_split(*arrays: X, y, test_size=0.2, shuffle=False)  
  
#-----End of the Modeling step-----
```

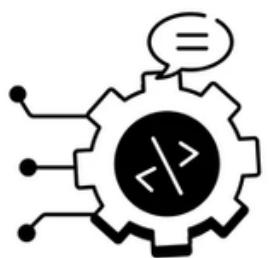


- Entraîne deux modèles, une régression linéaire et un régresseur à forêts aléatoires, sur l'ensemble d'entraînement.

```
#-----Start of the Model Training step-----  
  
#Linear Regression  
  
lr_model = LinearRegression()  
lr_model.fit(X_train, y_train)  
lr_predictions = lr_model.predict(X_test)  
  
#Random Forest  
  
rf_model = RandomForestRegressor(random_state=42)  
rf_model.fit(X_train, y_train)  
rf_predictions = rf_model.predict(X_test)
```

- Effectue une validation croisée pour évaluer la performance des modèles.
- Calcule des métriques d'évaluation telles que l'erreur quadratique moyenne (RMSE), l'erreur absolue moyenne (MAE), et le coefficient de détermination (R^2).

```
# Cross-validation for Linear Regression  
linear_scores = cross_val_score(lr_model, X_train, y_train, cv=5, scoring='r2')  
print(f"Linear Regression CV R2: {linear_scores.mean():.2f}")  
  
# Cross-validation for Random Forest  
rf_scores = cross_val_score(rf_model, X_train, y_train, cv=5, scoring='r2')  
print(f"Random Forest CV R2: {rf_scores.mean():.2f}")  
  
#-----End of the Model Training step-----  
#-----Start of the Model Evaluation step-----  
  
def evaluate_model(y_true, y_pred): 2 usages  
    rmse = np.sqrt(mean_squared_error(y_true, y_pred))  
    mae = mean_absolute_error(y_true, y_pred)  
    r2 = r2_score(y_true, y_pred)  
    return rmse, mae, r2  
  
# Evaluate Linear Regression  
lr_rmse, lr_mae, lr_r2 = evaluate_model(y_test, lr_predictions)  
print(f"Linear Regression - RMSE: {lr_rmse}, MAE: {lr_mae}, R2: {lr_r2}")  
  
# Evaluate Random Forest  
rf_rmse, rf_mae, rf_r2 = evaluate_model(y_test, rf_predictions)  
print(f"Random Forest - RMSE: {rf_rmse}, MAE: {rf_mae}, R2: {rf_r2}")  
  
#-----End of the Model Evaluation step-----
```



- Compare les prédictions des deux modèles avec les valeurs réelles à l'aide de graphiques.
- Analyse la distribution des erreurs pour évaluer les performances.

```
#-----Start of the Model Visualisation & Comparison step-----
# After training your models and generating predictions:
lr_predictions = lr_model.predict(X_test)
rf_predictions = rf_model.predict(X_test)

# Ensure indices align before plotting
y_test = y_test.reset_index(drop=True) # Reset the index of y_test
lr_predictions = pd.Series(lr_predictions, index=y_test.index) # Align predictions with y_test
rf_predictions = pd.Series(rf_predictions, index=y_test.index)

# Plotting the results

plt.figure(figsize=(10, 6))
plt.plot(*args: y_test.index, y_test, label='Actual', color='blue', linewidth=2) # Actual data
plt.plot(*args: y_test.index, lr_predictions, label='Linear Regression Predictions', color='orange') # LR predictions
plt.plot(*args: y_test.index, rf_predictions, label='Random Forest Predictions', color='green') # RF predictions
plt.legend()
plt.title('Temperature Predictions vs Actual')
plt.show()

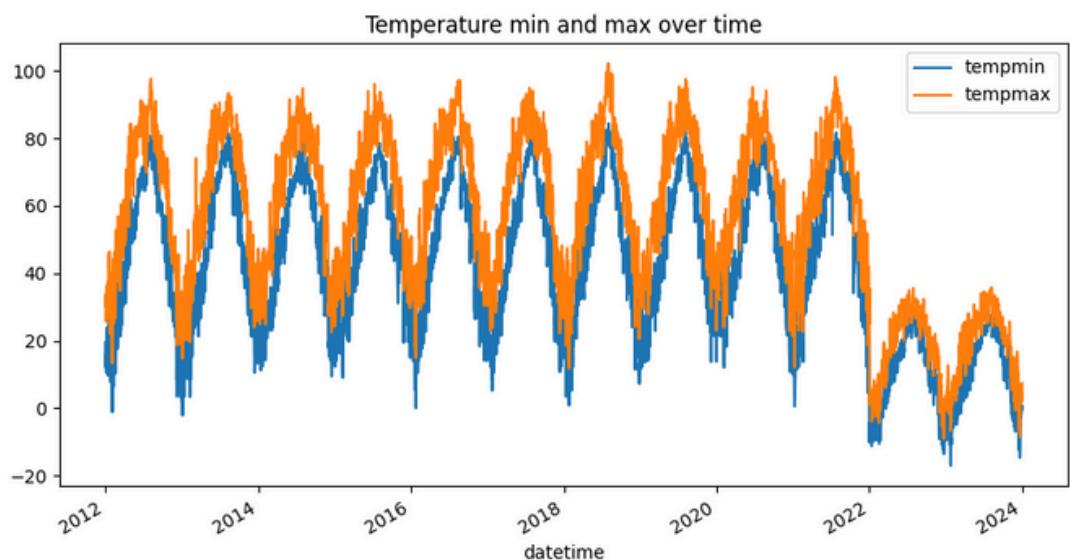
#plot the error distribution
errors = y_test - rf_predictions
plt.hist(errors, bins=20, alpha=0.7)
plt.title('Error Distribution (Random Forest)')
plt.xlabel('Prediction Error')
plt.ylabel('Frequency')
plt.show()
```



ANALYSE DES RÉSULTATS:

- Le code établi dans ce projet nous a permis de traiter et d'analyser les données météorologiques sur une période de 11 ans pour la ville étudiée. Plusieurs résultats ont été générés : un fichier CSV contenant les données nettoyées, plusieurs visualisations représentant l'évolution des températures, des précipitations et des chutes de neige au fil du temps, ainsi qu'une analyse des précipitations annuelles. Enfin, des modèles de régression linéaire et de forêts aléatoires ont été entraînés pour prédire les températures, et leurs performances ont été évaluées à l'aide de métriques telles que le RMSE, le MAE et le R².

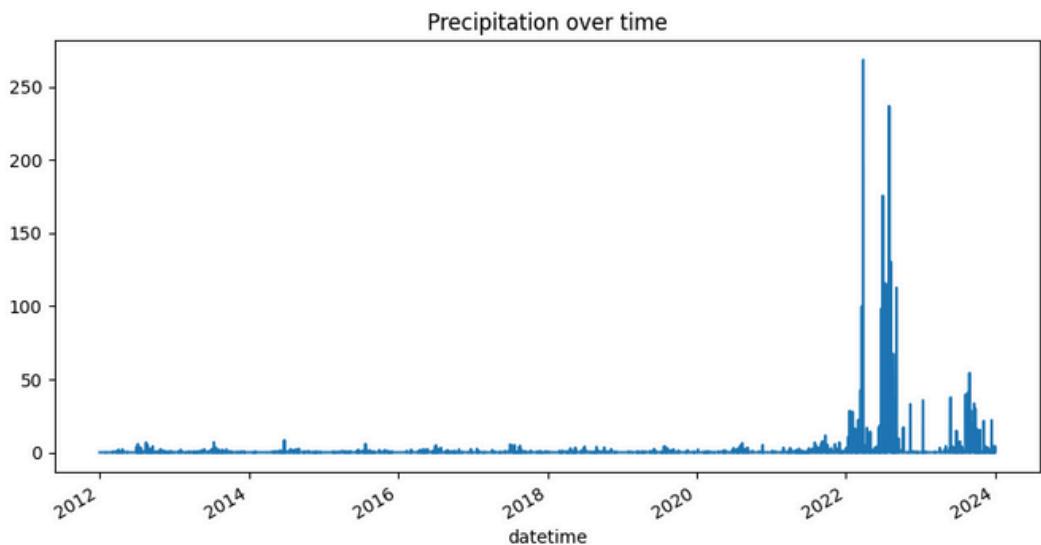
- REPRÉSENTATION GRAPHIQUE:**



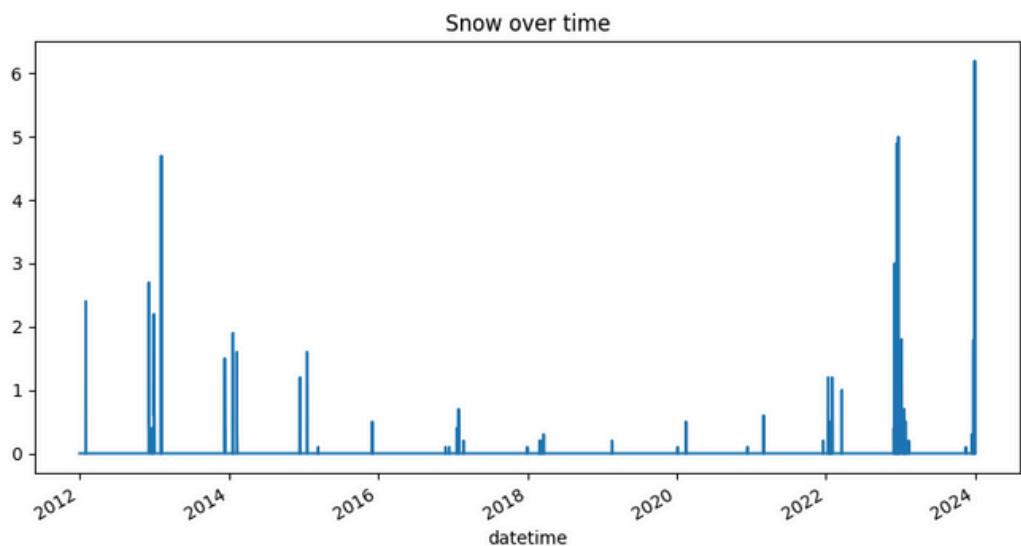
- Ce graphique montre les températures minimales (en bleu) et maximales (en orange) de 2012 à 2024. On observe une tendance saisonnière cyclique, avec des pics en été et des creux en hiver, reflétant des variations annuelles régulières. L'amplitude thermique entre les minima et maxima reste généralement stable. Le code utilisé a permis d'extraire et de visualiser les données météorologiques, avec une légende pour distinguer les courbes. Ce graphique met en évidence les tendances climatiques à long terme et les variations saisonnières.



• REPRÉSENTATION GRAPHIQUE:



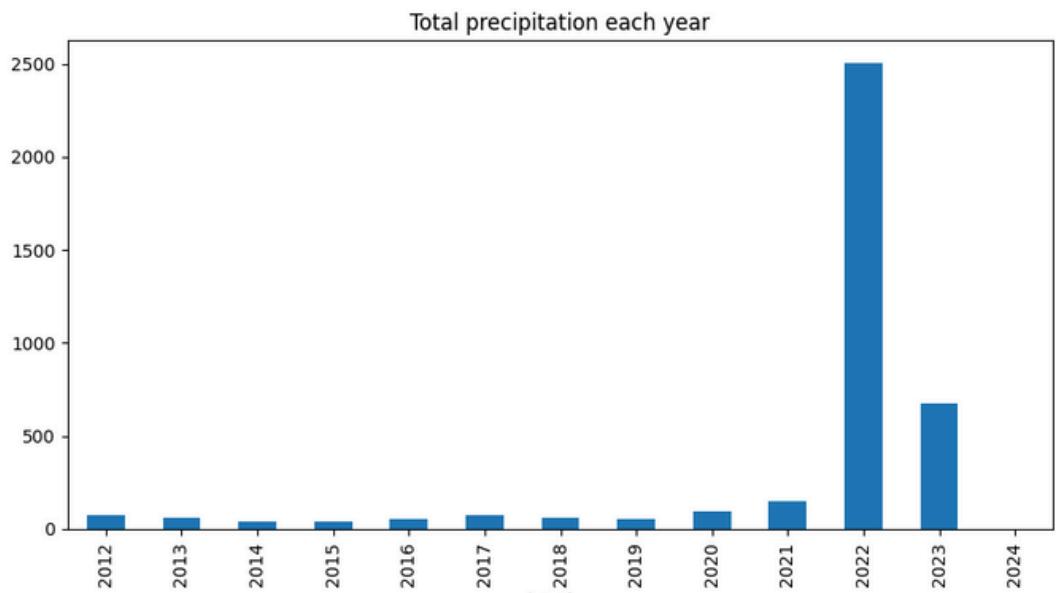
- Ce graphique illustre les précipitations sur une période de 2012 à 2024. On constate une quasi-absence de précipitations significatives avant 2021, suivie d'une augmentation marquée en 2022, avec des pics dépassant les 250 unités. Ces données traduisent un changement soudain des conditions climatiques ou météorologiques. Le code a permis d'extraire ces informations et de visualiser les variations au fil du temps. Ce graphique met en évidence une anomalie importante à partir de 2022, nécessitant une analyse approfondie pour en comprendre les causes.



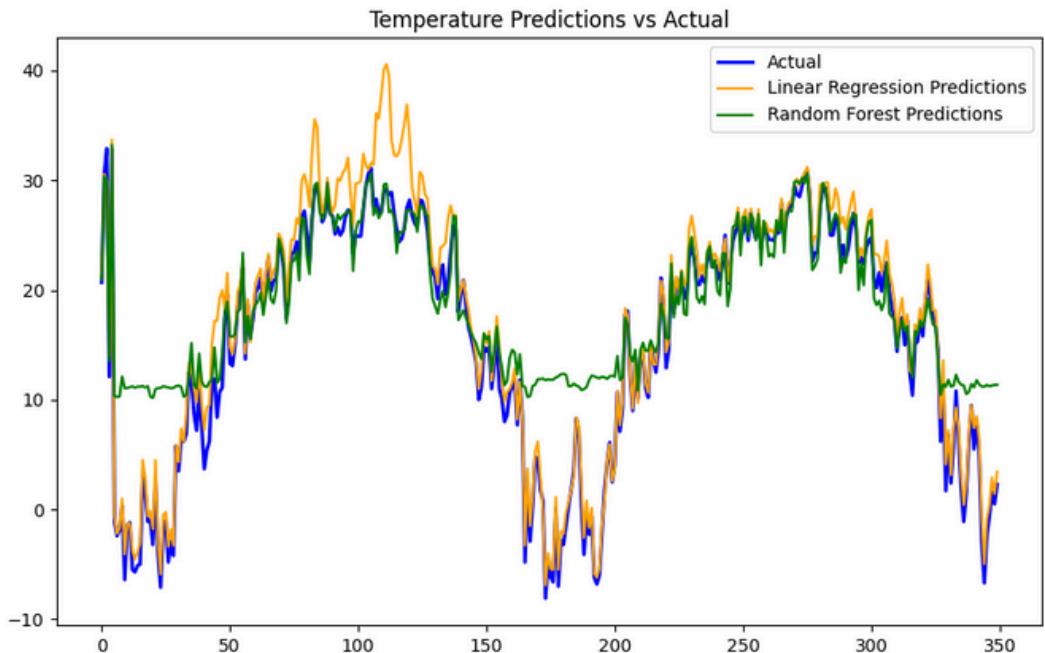


• REPRÉSENTATION GRAPHIQUE:

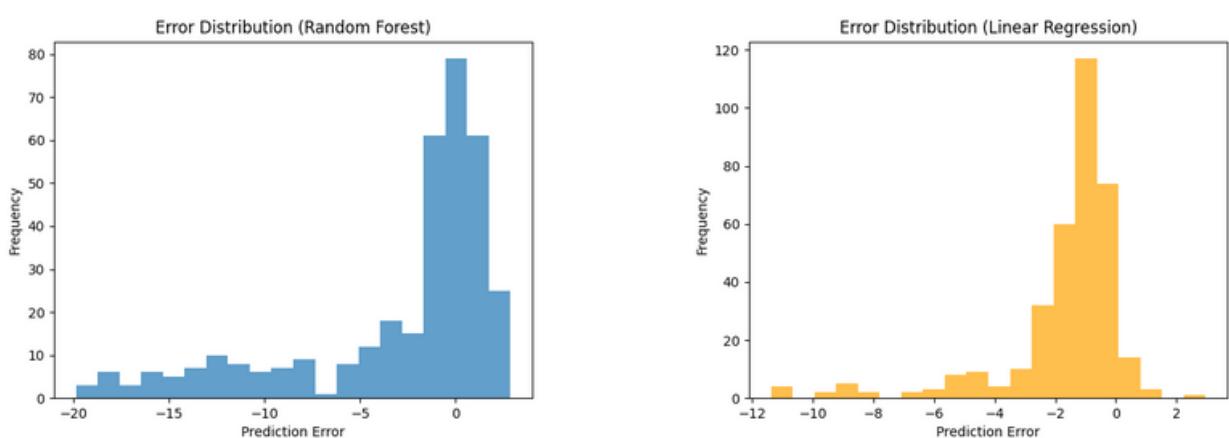
- Ce graphique représente les chutes de neige au fil du temps entre 2012 et 2024. Les données montrent des épisodes de neige significatifs entre 2012 et 2014, suivis d'une période relativement calme jusqu'en 2020. À partir de 2022, une recrudescence des chutes de neige est visible, atteignant des niveaux élevés, notamment en 2024 avec des valeurs dépassant 6 unités. Ces variations indiquent des changements climatiques ou environnementaux notables. Le code a permis de collecter et de visualiser ces informations pour mieux analyser les tendances des chutes de neige dans la région étudiée.



- Le graphique montre les précipitations totales par année sur une période de 12 ans. On observe un pic exceptionnel en 2022, avec des précipitations atteignant un maximum sans précédent. Cet événement correspond aux fortes pluies enregistrées à Séoul, notamment en août 2022. Les autres années présentent des précipitations nettement inférieures, indiquant un caractère anormal de l'année 2022.



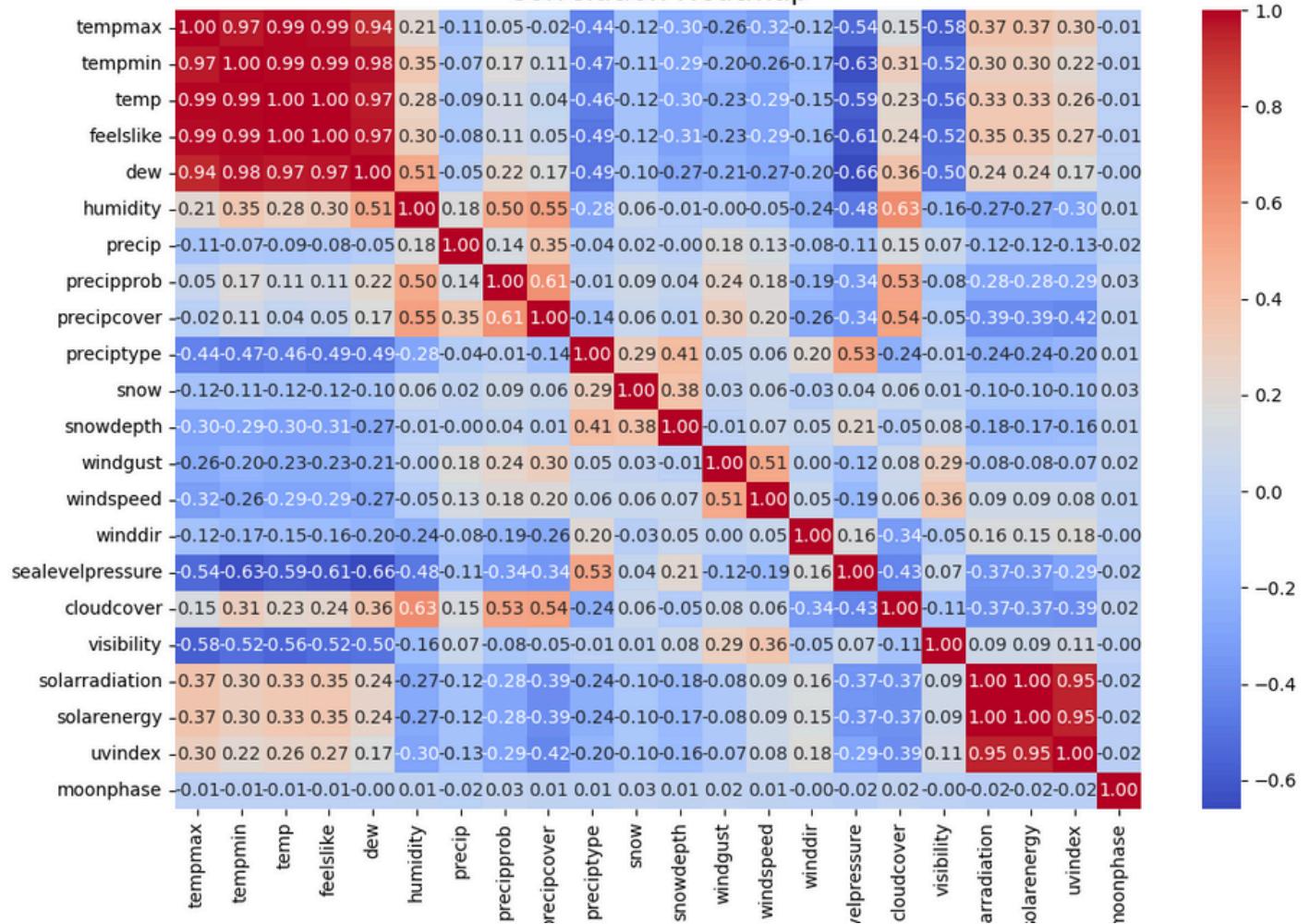
- Ce graphique compare les prédictions de température faites par deux modèles (régression linéaire en orange et forêt aléatoire en vert) avec les valeurs réelles en bleu. La courbe de la regression linéaire semble suivre les données réelles de manière plus précise, tandis que la foret aléatoire présente des écarts plus importants. Cette visualisation illustre l'efficacité relative des modèles pour prédire les températures quotidiennes.



- Ces deux graphiques montrent la distribution des erreurs de prédiction des modèles de forêt aléatoire et de la regression linéaire. Cette différence indique une bonne précision globale, avec un peu de bruit, par le deuxième modèle, et une marge d'erreur plus grande concernant le modèle de foret aléatoire.



Correlation Heatmap



- La matrice de corrélation illustre les relations entre différentes variables météorologiques. Les valeurs proches de 1 (en rouge) indiquent une forte corrélation positive, comme entre tempmax, tempmin, et temp. Cela signifie que ces variables augmentent ou diminuent ensemble. Inversement, les valeurs proches de -1 (en bleu foncé) montrent une forte corrélation négative. Les cases proches de 0 indiquent peu ou pas de relation. On remarque que des variables comme solarenergy et solarradiation ont une corrélation élevée, ce qui est logique car elles sont liées au rayonnement solaire. En revanche, certaines variables comme precip et humidity présentent des corrélations plus faibles avec d'autres.



• RÉSULTATS DE LA CONSOLE:

```
C:\Users\boudr\AppData\Local\Programs\Python\Python312\python.exe "C:\Users\boudr\Documents\GitHub\seoul-weather\seoul.py"
['seoul 2012-01-01 to 2014-01-01.csv', 'seoul 2014-01-01 to 2016-01-01.csv']

      name  ...
0  seoul  ...  47111099999,47098099999,47112099999,RKSM,47110...
1  seoul  ...  47111099999,47098099999,47112099999,RKSM,47110...
2  seoul  ...  47111099999,47098099999,47112099999,RKSM,47110...
3  seoul  ...  47111099999,47098099999,47112099999,RKSM,47110...
4  seoul  ...  47111099999,47098099999,47112099999,RKSM,47110...

[5 rows x 33 columns]

      name  ...  stations
datetime
2012-01-01  seoul  ...  47111099999,47098099999,47112099999,RKSM,47110...
2012-01-02  seoul  ...  47111099999,47098099999,47112099999,RKSM,47110...
2012-01-03  seoul  ...  47111099999,47098099999,47112099999,RKSM,47110...
2012-01-04  seoul  ...  47111099999,47098099999,47112099999,RKSM,47110...
2012-01-05  seoul  ...  47111099999,47098099999,47112099999,RKSM,47110...

[5 rows x 32 columns]

      name  ...  stations
datetime
2023-12-28  seoul  ...  47111099999,47098099999,47112099999,4711909999...
2023-12-29  seoul  ...  47111099999,47098099999,47112099999,4711909999...
2023-12-30  seoul  ...  47111099999,47098099999,47112099999,4711909999...
2023-12-31  seoul  ...  47111099999,47098099999,47112099999,4711909999...
2024-01-01  seoul  ...  47111099999,47098099999,47112099999,4711909999...
```

- Les résultats affichés dans la console représentent les sorties des méthodes ‘**head()**’ deux fois, la première pour visualiser le dataframe, et la deuxième après assigner **Date** comme index du dataframe, et la troisième sortie revient à l’appelation de la méthode ‘**tail()**’ pour s’assurer que notre data commence par l’année 2012 jusqu’à l’année 2022.



```
[8 rows x 24 columns]
<class 'pandas.core.frame.DataFrame'>
Index: 4389 entries, 2012-01-01 to 2024-01-01
Data columns (total 32 columns):
 #   Column           Non-Null Count Dtype  
--- 
 0   name             4389 non-null   object  
 1   tempmax          4389 non-null   float64 
 2   tempmin          4389 non-null   float64 
 3   temp              4389 non-null   float64 
 4   feelslikemax     4389 non-null   float64 
 5   feelslikemin     4389 non-null   float64 
 6   feelslike         4389 non-null   float64 
 7   dew               4389 non-null   float64 
 8   humidity          4389 non-null   float64 
 9   precip            4389 non-null   float64 
 10  precipprob       4389 non-null   int64   
 11  precipcover      4389 non-null   float64 
 12  preciptype       1750 non-null   object  
 13  snow              4389 non-null   float64 
 14  snowdepth         4389 non-null   float64 
 15  windgust          3432 non-null   float64 
 16  windspeed         4389 non-null   float64 
 17  winddir           4389 non-null   float64 
 18  sealevelpressure  4389 non-null   float64 
 19  cloudcover        4389 non-null   float64 
 20  visibility        4389 non-null   float64 
 21  solarradiation   4389 non-null   float64 
 22  solarenergy       4389 non-null   float64 
 23  uvindex           4389 non-null   int64   
 24  severerisk        722 non-null    float64 
 25  sunrise            4389 non-null   object  
 26  sunset             4389 non-null   object  
 27  moonphase          4389 non-null   float64 
 28  conditions         4389 non-null   object  
 29  description        4389 non-null   object  
 30  icon               4389 non-null   object  
 31  stations           4389 non-null   object  
dtypes: float64(22), int64(2), object(8)
memory usage: 1.1+ MB
```





• RÉSULTATS DE LA CONSOLE:

None	0.000000
tempmax	0.000000
tempmin	0.000000
temp	0.000000
feelslike	0.000000
dew	0.000000
humidity	0.000000
precip	0.000000
precipprob	0.000000
precipcover	0.000000
preciptype	0.601276
snow	0.000000
snowdepth	0.000000
windgust	0.218045
windspeed	0.000000
winddir	0.000000
sealevelpressure	0.000000
cloudcover	0.000000
visibility	0.000000
solarradiation	0.000000
solarenergy	0.000000
uvindex	0.000000
sunrise	0.000000
sunset	0.000000
moonphase	0.000000
conditions	0.000000
description	0.000000
icon	0.000000

stations	0.000000
dtype:	float64
preciptype	
rain	1454
rain,snow	198
snow	98
Name:	count, dtype: int64
	2639

- Ce résultat console montre deux informations importantes :
- Proportion de valeurs manquantes par colonne :
- La plupart des colonnes ont 0.000000 comme proportion de valeurs manquantes, ce qui indique des données complètes.
- Les colonnes preciptype (60,13 %) et windgust (21,80 %) présentent des valeurs manquantes, nécessitant un traitement.
- Distribution des types de précipitations dans preciptype :
- Les occurrences enregistrées montrent : rain (1454), rain,snow (198), et snow (98), sur un total de 2639 observations.
- Cela confirme la nécessité de remplir les valeurs manquantes pour garantir des analyses cohérentes.



• RÉSULTATS DANS LA CONSOLE:

```
Linear Regression CV R2: 1.00
Random Forest CV R2: 1.00
Linear Regression - RMSE: 2.6249839635109304, MAE: 1.7239632606487707, R2: 0.9396620097039555
Random Forest - RMSE: 6.192656905446273, MAE: 3.741205714285713, R2: 0.6641915886040919
```

Ce résultat console présente les performances des modèles entraînés :

Validation croisée (R^2) :

- Les deux modèles, régression linéaire et forêt aléatoire, obtiennent un score parfait de 1.00 en validation croisée, indiquant un surajustement possible.

Évaluation des modèles sur les données de test :



Régression linéaire :

- RMSE : 2.62, MAE : 1.72, R^2 : 0.94.
- Cela indique une bonne précision avec des erreurs faibles et une excellente corrélation entre prédictions et valeurs réelles.

Forêt aléatoire :

- RMSE : 6.19, MAE : 3.74, R^2 : 0.66.
- Ce modèle est moins performant, avec des erreurs plus élevées et une corrélation plus faible.
- La régression linéaire semble mieux adaptée aux données dans ce cas.

DISCUSSION & CONCLUSION :

Voici une discussion détaillée des performances des deux modèles dans le cadre de ce projet, basée sur les métriques et les cas d'utilisation :

1. Performances des deux modèles : régression linéaire et forêt aléatoire

Les deux modèles ont été évalués à l'aide de plusieurs métriques : R^2 , RMSE (Root Mean Square Error) et MAE (Mean Absolute Error). Voici une analyse des résultats :

Régression linéaire

- R^2 : 0.94 indique que le modèle explique 94 % de la variance des données. C'est une excellente performance en termes de prédiction.
- RMSE : 2.62, ce qui montre que les erreurs moyennes quadratiques entre les prédictions et les valeurs réelles sont faibles.
- MAE : 1.72, ce qui indique que, en moyenne, le modèle se trompe de 1.72 unités sur les prédictions.

Forêt aléatoire

- R^2 : 0.66, ce qui est significativement plus faible que la régression linéaire. Cela signifie que le modèle n'explique que 66 % de la variance des données.
- RMSE : 6.19, traduisant une erreur quadratique moyenne plus élevée.
- MAE : 3.74, montrant que les prédictions s'écartent davantage des valeurs réelles en moyenne.

2. Comparaison détaillée basée sur les métriques

- R^2 (coefficient de détermination) :
 - La régression linéaire surpassé la forêt aléatoire avec une meilleure corrélation entre les prédictions et les valeurs réelles.
- Précision des prédictions (RMSE et MAE) :
 - La régression linéaire produit des prédictions nettement plus précises que la forêt aléatoire, avec des erreurs significativement plus faibles.
- Robustesse aux non-linéarités :
 - Bien que la régression linéaire fonctionne mieux ici, cela peut indiquer que les données présentent une relation principalement linéaire entre les variables. La forêt aléatoire, qui excelle dans la capture des non-linéarités complexes, ne semble pas bénéficier de ce type de relation dans ce cas précis.

3. Cas où chaque modèle est le plus adapté

Régression linéaire :

- Quand l'utiliser :Lorsque les données présentent une relation linéaire claire entre les variables indépendantes et dépendantes.
- Lorsque la simplicité et l'interprétabilité du modèle sont prioritaires.
- Pour des datasets propres, avec peu de bruit ou de valeurs aberrantes.
- Avantages :Facile à entraîner et interpréter.
- Performant sur des relations linéaires ou quasi-linéaires.
- Moins gourmand en termes de calcul et de mémoire.

Forêt aléatoire :

- Quand l'utiliser :Lorsque les données présentent des relations non linéaires complexes ou des interactions entre les variables.
- Pour des données plus bruyantes ou avec des valeurs aberrantes, car la forêt aléatoire est plus robuste.
- Lorsque des prédictions fiables sont nécessaires malgré des données incomplètes ou moins cohérentes.
- Avantages :Captation des non-linéarités et interactions complexes.
- Plus robuste aux variations des données et au surajustement.
- Fournit des informations sur l'importance des variables.

Conclusion

Pour ce projet, la régression linéaire s'est montrée plus performante, suggérant une relation linéaire entre les variables et la température. Cependant, la forêt aléatoire pourrait être utile si des facteurs non linéaires ou des données plus complexes étaient intégrés dans le futur. Le choix du modèle dépend donc des caractéristiques des données et des besoins spécifiques du projet (précision, interprétabilité, complexité).

