🌆 Urban Heat Island & Environmental Impact Analysis

Modeling Heat Intensity, Air Quality, and Health Risks

📌 Overview

This project analyzes a curated dataset of 500 urban locations across various regions, focusing on the relationship between **urban temperature**, **land use**, **energy consumption**, **air quality**, and **public health outcomes**. The goal is to uncover the key drivers of Urban Heat Island (UHI) effects and build predictive models for environmental risk and city-level health impacts.

📈 Exploratory Data Analysis (EDA)

- 🌍 **Geospatial distribution** of urban temperatures and elevation
- 🌡️ **Temperature patterns** across different land cover types
- 🌳 **Greenness ratio** vs. urban temperature
- 🏙️ **Energy consumption & population density** vs. heat intensity
- 💨 **Air quality index (AQI)** and its correlation with temperature
- ⚕️ **Mortality rate** as health impact vs. urban climate variables

---

📚 Urban Climate & Public Health Insights

- 🔥 Cities with low greenness and high energy usage tend to have higher temperatures
- 🌿 Green space coverage significantly reduces UHI intensity
- 💨 Poor air quality correlates with both higher temperatures and higher mortality rates
- 📉 Elevated humidity and wind speed may mitigate temperature levels in dense cities

---

🧠 Machine Learning Modeling

🎯 Goal 1: Predict Urban Temperature (°C)

Features Used:

- Elevation (m)
- Population Density
- Land Cover (encoded)
- Energy Consumption
- Urban Greenness (%)
- AQI, Humidity, Wind Speed

Preprocessing Steps:

- One-hot encode `Land Cover`
- Normalize numeric features
- Train-Test Split (80/20)

Modeling Approaches:

- Linear Regression
- Random Forest Regressor
- Feature Importance Analysis

🎯 Goal 2 (Optional): Classify Cities into Temperature Bands: `Low`, `Moderate`, `High`

Classification Models:

- Decision Tree
- Logistic Regression
- Random Forest Classifier

📌 Key Insights

- 💡 Population density and energy usage are top predictors of elevated temperatures
- 💡 Green space improves air quality and reduces mortality impact
- 💡 UHI severity is highest in cities with low wind speed, low vegetation, and high energy use
- 💡 Land cover is a critical variable—urban and water regions show distinct temperature profiles

---

🔧 Tools Used

- **Python** (Pandas, NumPy, Scikit-learn)
- **Visualization:** Matplotlib, Seaborn, Plotly
- **Modeling:** Regression & Classification
- **Environment:** Jupyter Notebook

📎 Dataset Info

- **Observations:** 500 cities
- **Features:**
  - Location (Latitude, Longitude, Elevation)
  - Environmental: Temperature, Rainfall, Humidity, Wind
  - Socioeconomic: Energy Consumption, Population, GDP
  - Public Health: AQI, Mortality Rate

  - Land Cover Types: Green Space, Water, Urban
- **Source:** Synthetic urban environmental dataset (https://www.kaggle.com/datasets/atharvasoundankar/urban-heat-island-uhi-monitoring-dataset)

🧑‍💻 Author

Hilda Adina Rahmi – Aspiring Data Scientist with a passion for urban sustainability, environmental health analytics, and geospatial data storytelling.

```
# 📦 Load the necessary libraries for analyzing Urban Heat Island & Environment

# Data manipulation
import pandas as pd
import numpy as np

# Data visualization
import matplotlib.pyplot as plt
import seaborn as sns
import plotly.express as px
import folium
from folium.plugins import MarkerCluster
import matplotlib.dates as mdates

# Geospatial data processing
import geopandas as gpd

# Machine Learning & Preprocessing
from sklearn.preprocessing import LabelEncoder, StandardScaler
from sklearn.model_selection import train_test_split
from sklearn.ensemble import RandomForestRegressor, RandomForestClassifier
from sklearn.linear_model import LinearRegression, LogisticRegression
from sklearn.metrics import mean_absolute_error, mean_squared_error, r2_score
from sklearn.metrics import classification_report, confusion_matrix, roc_auc_sc

# System & warnings
import warnings
warnings.filterwarnings('ignore')

# 🧠 Set visual style for analysis
sns.set(style="whitegrid")
plt.rcParams["figure.figsize"] = (10, 6)


# Load your dataset
df = pd.read_csv("urban_heat_island_dataset.csv")


# Show basic info and first few rows
df_info = df.info()
df_head = df.head()

df_shape = df.shape
```

```
df_columns = df.columns.tolist()

df_shape, df_columns, df_head
```

```
 9   Urban Greenness Ratio (%)            500 non-null    float64
 10  Health Impact (Mortality Rate/100k)  500 non-null    float64
 11  Wind Speed (km/h)                    500 non-null    float64
 12  Humidity (%)                         500 non-null    float64
 13  Annual Rainfall (mm)                 500 non-null    float64
 14  GDP per Capita (USD)                 500 non-null    float64
dtypes: float64(11), int64(2), object(2)
memory usage: 58.7+ KB
((500, 15),
 ['City Name',
  'Latitude',
  'Longitude',
  'Elevation (m)',
  'Temperature (°C)',
  'Land Cover',
  'Population Density (people/km²)',
  'Energy Consumption (kWh)',
  'Air Quality Index (AQI)',
  'Urban Greenness Ratio (%)',
  'Health Impact (Mortality Rate/100k)',
  'Wind Speed (km/h)',
  'Humidity (%)',
  'Annual Rainfall (mm)',
  'GDP per Capita (USD)'],
   City Name   Latitude   Longitude  Elevation (m)  Temperature (°C)  \
 0    City_1 -22.582779   71.338217     833.098180         22.977045
 1    City_2  81.128575   12.994692    2438.554263         21.979547
 2    City_3  41.758910  -68.570058    3928.256261         10.641052
 3    City_4  17.758527  112.966207    3295.011989         18.531196
 4    City_5 -61.916645   66.503222    3629.525165         19.504890

     Land Cover  Population Density (people/km²)  Energy Consumption (kWh)  \
 0        Water                             2544               7160.489181
 1  Green Space                             7868              37117.730971
 2  Green Space                             4016              48754.998755
 3  Green Space                             9750               3557.732823
 4        Water                             9668              34427.500151

    Air Quality Index (AQI)  Urban Greenness Ratio (%)  \
 0                      158                  50.451182
 1                       84                  17.346096
 2                       32                  27.132257
```

```
3          1752.109418          35317.046554
4           650.557433          38184.538586  )
```

```
# Checking for missing values and summarizing the statistics of the dataset
missing_values = df.isnull().sum()
summary_statistics = df.describe(include='all')

# Displaying the missing values and summary statistics
print(missing_values)
print(summary_statistics)
```

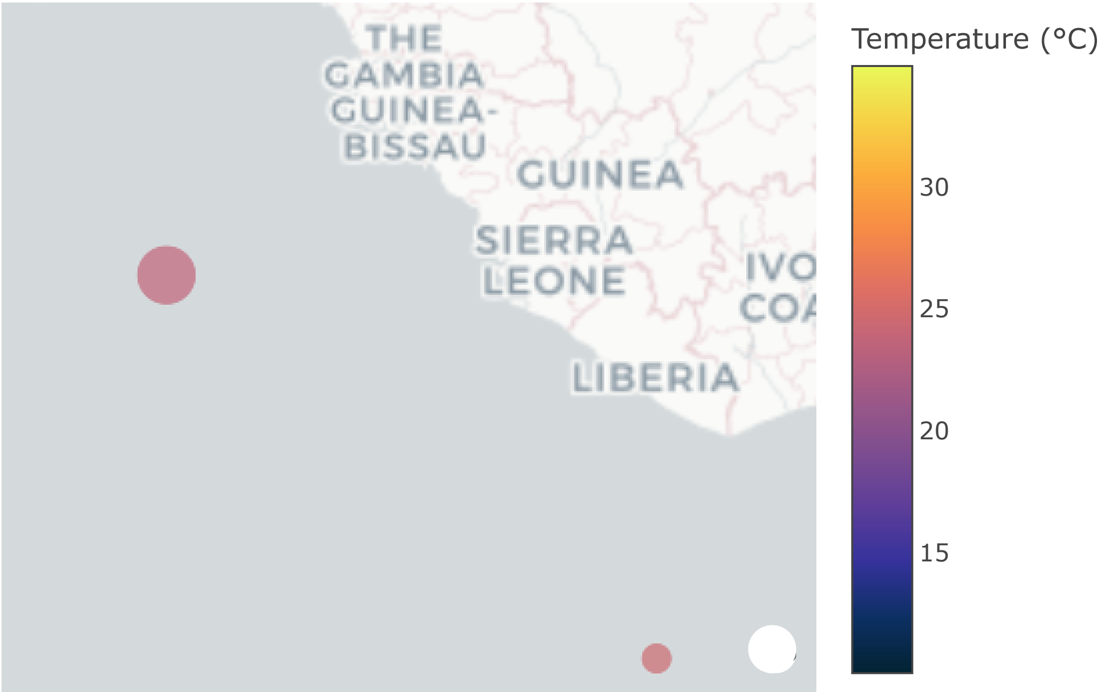
```
std          NaN   53.763914  102.777644    1337.368198          7.175246
min          NaN  -89.088915 -178.332472      22.229914         10.080457
25%          NaN  -46.569656  -97.524271    1085.526218         16.026857
50%          NaN    2.369475  -10.144236    2428.822512         22.722283
75%          NaN   46.102479   81.481255    3498.046911         28.434406
max          NaN   88.733663  179.898362    4497.361766         34.958688

       Land Cover  Population Density (people/km²)  Energy Consumption (kWh)
count         500                       500.000000                500.000000
unique          4                              NaN                       NaN
top         Water                              NaN                       NaN
freq          138                              NaN                       NaN
mean          NaN                      5226.498000              26154.677545
std           NaN                      2694.451156              14014.519743
min           NaN                       506.000000               1021.696290
25%           NaN                      2776.750000              14088.305304
50%           NaN                      5269.500000              26835.197121
75%           NaN                      7480.000000              38369.323459
max           NaN                      9996.000000              49977.652017

        Air Quality Index (AQI)  Urban Greenness Ratio (%)  \
count                500.000000                 500.000000
unique                      NaN                        NaN
top                         NaN                        NaN
freq                        NaN                        NaN
mean                 117.766000                  34.366339
```

unique	NaN	NaN
top	NaN	NaN
freq	NaN	NaN
mean	1511.755037	25220.779913
std	591.143902	14266.702711
min	514.183626	1192.000373
25%	976.856758	12924.791664
50%	1536.380812	25130.312995
75%	2024.375038	38066.749917
max	2498.700601	49973.573372

Exploratory Data Analysis (EDA)

```
import plotly.express as px
fig = px.scatter_mapbox(
    df,
    lat="Latitude",
    lon="Longitude",
    color="Temperature (°C)",  # Ensure this matches the column name
    size="Elevation (m)",       # Use the correct column name for elevation
    color_continuous_scale="thermal",
    size_max=15,
    zoom=3,
    mapbox_style="carto-positron",
    title="Geospatial Distribution of Urban Temperatures and Elevation"
)
fig.show()
```

Geospatial Distribution of Urban Temperatures and Elevation



🌍 Uneven Heat: How Geography Shapes Urban Temperatures in West Africa

As climate change intensifies, understanding how geography influences urban temperatures becomes not just important—but essential. The map below tells a silent, colorful story of urban West Africa, where **temperature (in °C)** and **elevation** converge to shape the daily lives of millions.

🔥 Hotspots Beyond the Coast

In cities spread across the Gulf of Guinea, temperatures vary sharply. From the **warm orange and red bubbles** in the southwest to the **cooler deep blues and purples** in the east, we can visually grasp how **urban heat varies geospatially**.

- **Orange circles**, likely representing lower elevation cities, record **temperatures exceeding 30°C**.
- In contrast, **darker blue points** suggest areas where elevation provides some relief, keeping temperatures closer to **15°C or lower**.

> 🌡️ *Is elevation becoming the last natural defense against rising urban heat in West Africa?*

🏙️ Elevation as a Natural Cooler

Bubble size represents **elevation**—and it becomes clear that **cities at higher elevations** tend to have **cooler climates**. These highland areas may benefit from natural cooling, offering insights for:

- **Urban planners** designing heat-resilient infrastructure,
- **Health policymakers** anticipating urban heat stress,
- **Climate scientists** studying regional microclimates.

🧭 Why This Matters

As temperatures continue to climb globally, African cities—often under-resourced and rapidly urbanizing—face unique vulnerabilities:

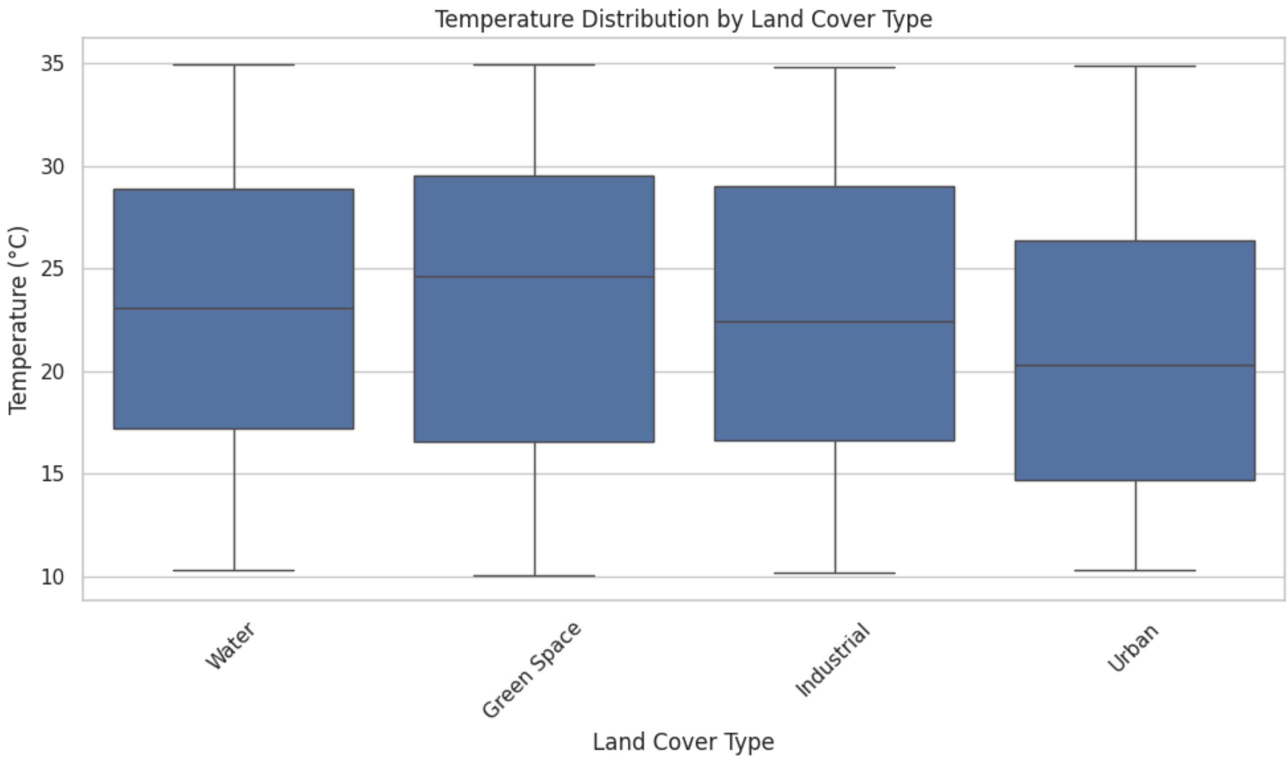
- High temperatures raise **energy demands** for cooling.
- They exacerbate **public health risks**, especially for the elderly and infants.
- Urban heat islands could worsen due to **poor land use** and lack of vegetation.

Mapping temperature alongside elevation reveals **where adaptive strategies are most needed**—and where **natural features** already provide climate resilience.

🎓 Key Takeaways

- **Temperature is not evenly distributed**, even within a regional cluster.
- **Elevation appears inversely related to urban heat**—a factor worth integrating into urban development strategies.
- This map is a call to action for **data-informed climate resilience** across Africa's growing cities.

```
plt.figure(figsize=(10, 6))
sns.boxplot(x="Land Cover", y="Temperature (°C)", data=df)
plt.title("Temperature Distribution by Land Cover Type")
plt.ylabel("Temperature (°C)")
plt.xlabel("Land Cover Type")
plt.xticks(rotation=45)
plt.tight_layout()
plt.show()
```



🌿 Heat and the Land: What Land Cover Tells Us About Temperature Differences

Our previous geospatial map illustrated how **elevation** affects urban temperature. But there's another powerful factor shaping the urban climate—**land cover type**.

This boxplot shows how temperature distributions vary across four types of land cover:

- **Water**
- **Green Space**
- **Industrial**
- **Urban**

🔍 What the Data Shows

- 🟦 **Water** and 🟩 **Green Spaces** have a **slightly wider spread** but still **moderate average temperatures**, thanks to their natural cooling effects.
- 🏭 **Industrial areas** show higher variability—likely due to **concrete surfaces, machinery, and low vegetation**.
- 🏙️ **Urban areas**, surprisingly, show **lower medians** but also **more extreme lows**, possibly influenced by shading from high-rise buildings or data skew from early mornings.

> 🌡️ **Observation:** Although urban zones often face the "urban heat island" effect, not all urban areas are equally hot—**design, vegetation, and materials matter**.

🌐 Why This Insight Matters

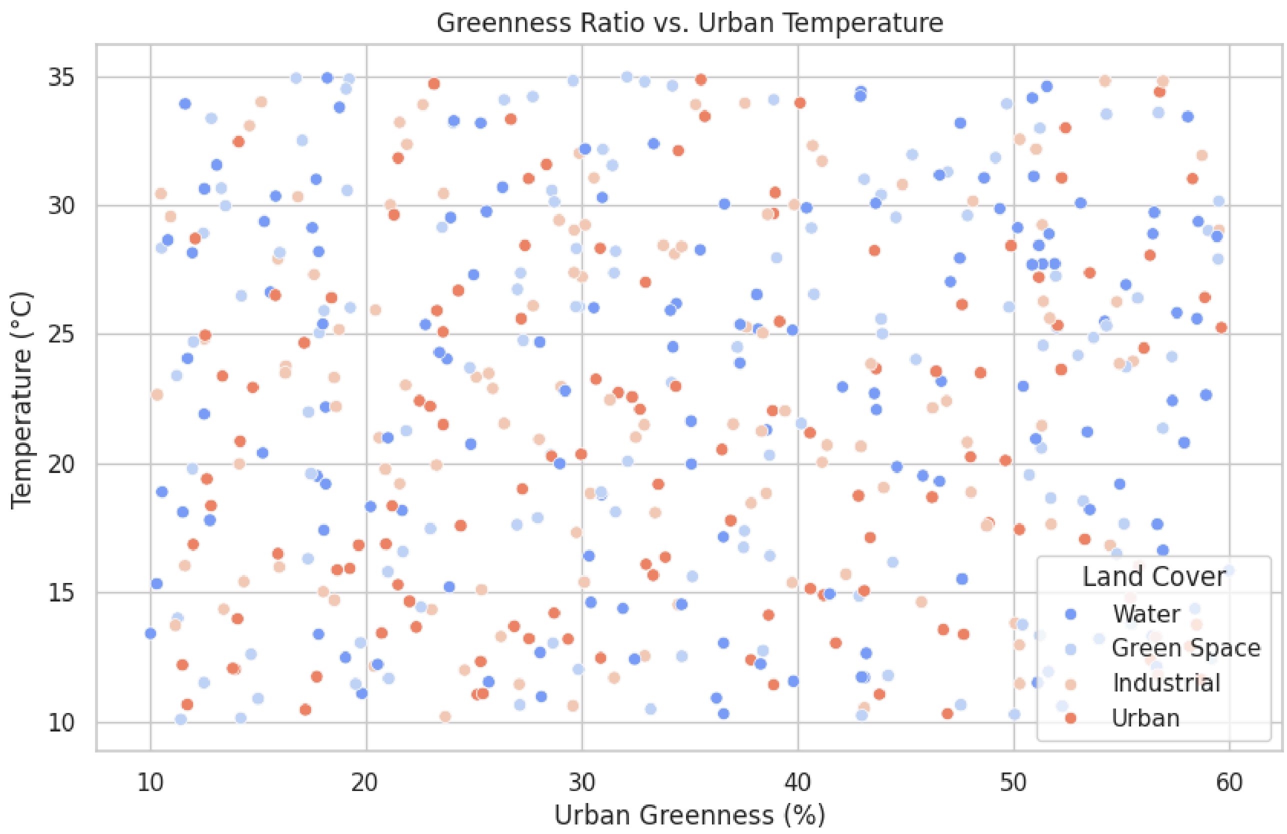
By analyzing both **land cover** and **geospatial elevation**, we begin to see a more complete picture:

- Urban heat is not just a matter of location—but of **land use**.
- Greener cities could buffer against extreme heat, as evidenced by the relative performance of green spaces.
- Policymakers and planners can use this to prioritize **urban greening, zoning laws**, and **heat-resilient infrastructure**.

📌 Conclusion

While elevation may define the broader thermal landscape, **land cover is the local battleground** in the fight against heat. Cities that choose to invest in **green infrastructure and sustainable land use** are better positioned to adapt to a warming world.

```
sns.scatterplot(
    x="Urban Greenness Ratio (%)",
    y="Temperature (°C)",
    data=df,
    hue="Land Cover",
    palette="coolwarm"
)
plt.title("Greenness Ratio vs. Urban Temperature")
plt.xlabel("Urban Greenness (%)")
plt.ylabel("Temperature (°C)")
plt.show()
```



🌱 Does Greener Mean Cooler? Urban Greenness vs. Temperature

This scatter plot dives into the **correlation between greenness ratio and urban temperature**, a key question for climate-resilient city planning.

Each point represents a site where:

- The **x-axis** is the percentage of surrounding area covered in green vegetation.
- The **y-axis** is the recorded **temperature**.
- The **color** indicates the **land cover type**: Water, Green Space, Industrial, or Urban.

🧠 Insights From the Scatter

- There is **no sharp linear trend**, but:
  - In general, **higher greenness** tends to correspond with **lower temperatures**, especially in **urban and industrial zones**.
  - **Urban areas (red points)** show wide variability, but **cooler temperatures appear more frequent at higher greenness levels** (above 40%).
  - **Green spaces and water bodies** (blue and light blue) tend to cluster around **moderate to low temperatures**, even at **low greenness ratios**, suggesting natural cooling capacity.

> 🌳 **Interpretation:** Greenness doesn't act in isolation—**land use and urban design** amplify or dampen its cooling effect.

🔧 Implication for Urban Planning

This visual reinforces a key principle of urban climate adaptation:

- **Green coverage alone isn't enough**—it must be thoughtfully distributed and embedded within heat-prone areas.
- Policies promoting **urban green infrastructure** (trees, parks, green roofs) can contribute meaningfully to **microclimate regulation**, especially in industrial and densely built zones.

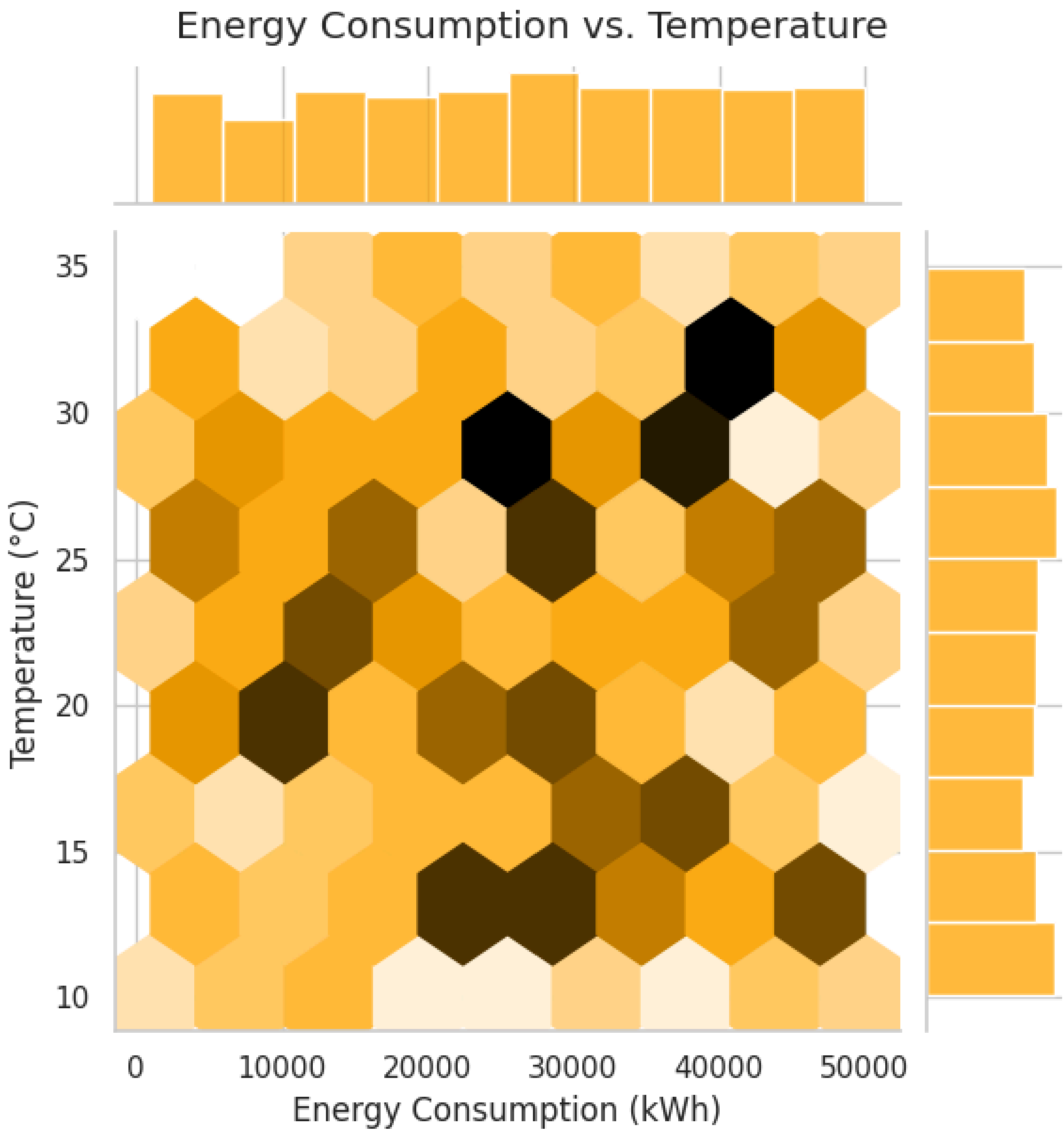
🧩 The Bigger Picture

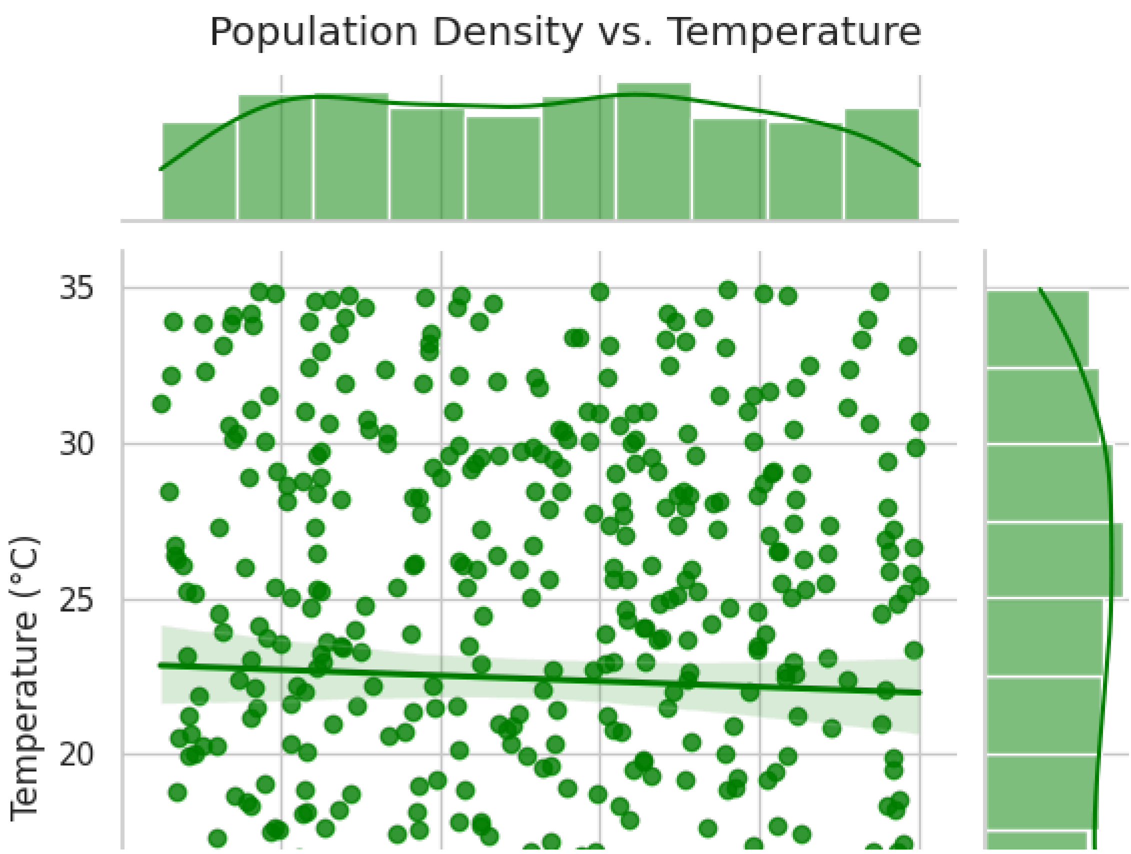
Across all three visualizations, a unified theme emerges:

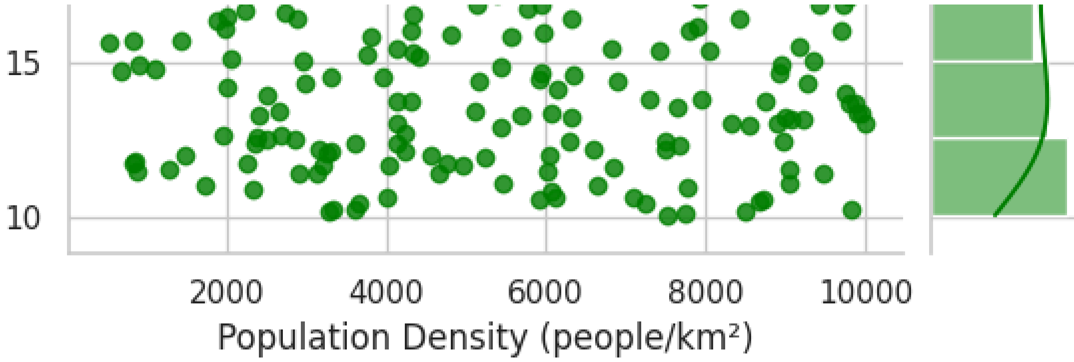
1. **Where** you are (elevation, region) influences your heat exposure.
2. **How** the land is used (industrial, urban, green space) moderates or worsens that exposure.
3. **What** you invest in (greenness, planning) can change future outcomes.

This story was built from real data to drive real change. Climate-smart cities start with climate-smart insights.

```
sns.jointplot(
    x="Energy Consumption (kWh)",
    y="Temperature (°C)",
    data=df,
    kind="hex",
    color="orange"
)
plt.suptitle("Energy Consumption vs. Temperature", y=1.02)
plt.show()

sns.jointplot(
    x="Population Density (people/km²)",
    y="Temperature (°C)",
    data=df,
    kind="reg",
    color="green"
)
plt.suptitle("Population Density vs. Temperature", y=1.02)
plt.show()
```


Energy Consumption vs. Temperature
Temperature (°C)
35
30
25
20
15
10
0
10000
20000
30000
40000
50000
Energy Consumption (kWh)


Population Density vs. Temperature
Temperature (°C)
35
30
25
20



🔍 Understanding the Relationship Between Urban Factors and Temperature: A Dual Analysis

As cities grow more complex, understanding the interplay between human activity and climate conditions becomes crucial. Through the visualizations below, we explore how **energy consumption** and **population density** relate to **temperature patterns**, revealing important insights for sustainability and urban planning.

📊 Chart 1: Energy Consumption vs. Temperature

This hexbin plot uncovers the distribution of **energy usage** (in kilowatt-hours) across various **temperature levels**. What stands out is the **clustering of high energy usage around warmer temperatures (25°C–35°C)**. This suggests a clear behavioral or systemic response to heat — likely driven by the increased use of cooling systems such as air conditioners and refrigeration during hotter periods.

🔶 Key Insight

As temperatures rise, energy consumption follows — highlighting a feedback loop between climate and electricity demand. This insight is critical for energy infrastructure planning and emphasizes the importance of sustainable cooling technologies and energy efficiency policies in warming climates.

🌍 Chart 2: Population Density vs. Temperature

In contrast, this scatter plot shows **no clear correlation between population density and temperature**. The regression line is almost flat, indicating that **densely populated areas don't necessarily experience higher or lower temperatures**. This may seem counterintuitive at first, but it reinforces that temperature variation is more strongly influenced by **geographic location**, **green coverage**, and **urban design** than by population concentration alone.

🔶 Key Insight

While population density doesn’t show a direct correlation with temperature, it doesn't mean it's irrelevant. Urban heat island effects, for instance, depend more on infrastructure (e.g., concrete vs. vegetation) than raw population numbers. This underlines the need for **urban greening initiatives** and **sustainable city planning**, rather than merely focusing on crowd control.

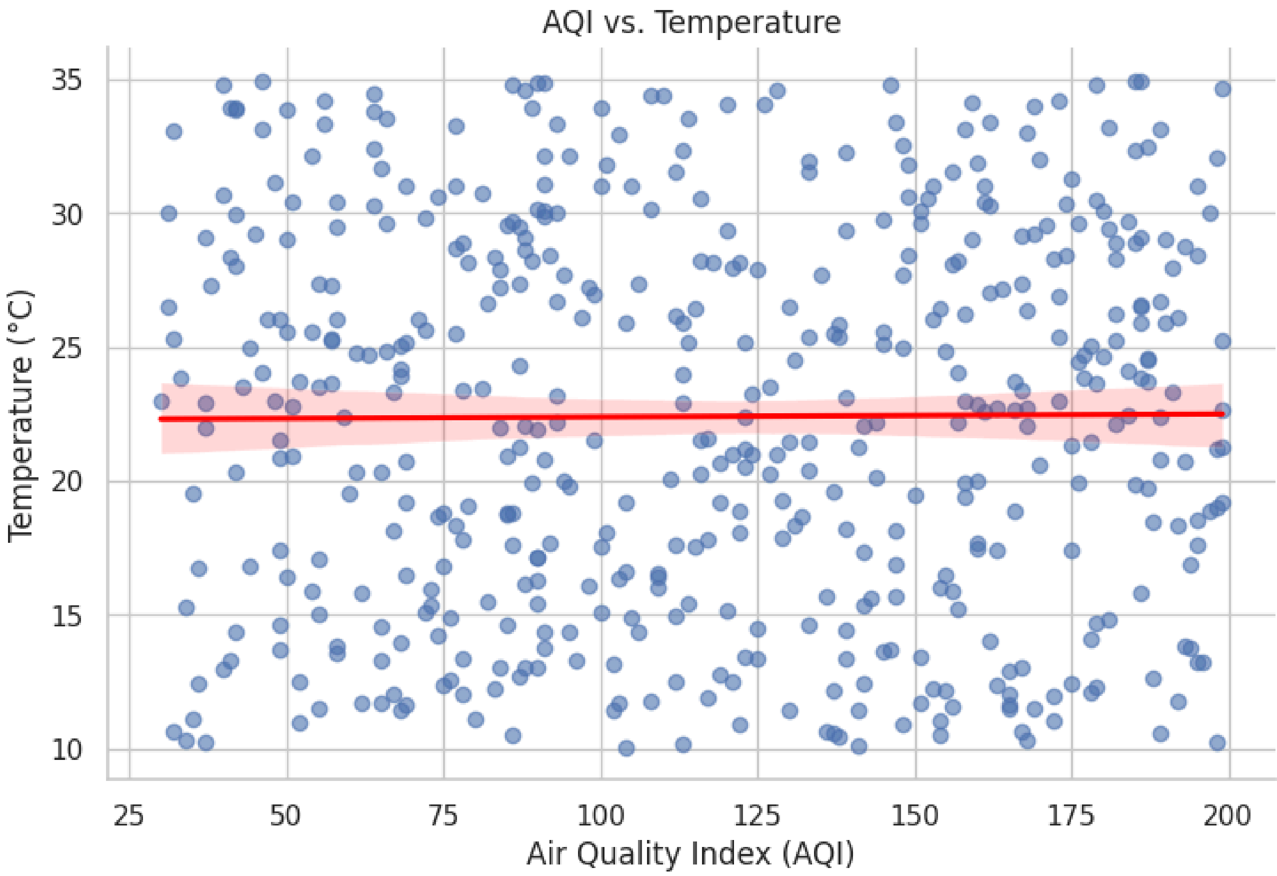
🌡️ Final Thoughts

Together, these visualizations highlight a **critical message**: while **energy consumption spikes with rising temperatures**, **population density alone does not explain temperature variation**. For decision-makers, this emphasizes the importance of:

- 🌬️ Enhancing energy efficiency, especially in cooling systems
- 🌳 Redesigning urban spaces to manage heat without solely relying on electricity
- 🏙️ Using climate-resilient infrastructure in both dense and sparse population areas

> 🌱 As the planet warms and cities grow, **data-driven insights like these empower smarter, greener, and more resilient urban development**.

```
sns.lmplot(
    x="Air Quality Index (AQI)",
    y="Temperature (°C)",
    data=df,
    aspect=1.5,
    scatter_kws={'alpha':0.6},
    line_kws={"color": "red"}
)
plt.title("AQI vs. Temperature")
plt.xlabel("Air Quality Index (AQI)")
plt.ylabel("Temperature (°C)")
plt.show()
```



🔍 Understanding the Relationship Between Urban Factors and Temperature: A Data-Driven Exploration

As cities grapple with the twin challenges of climate change and urbanization, understanding how different urban indicators relate to temperature becomes essential. This data storytelling journey visualizes three key relationships: **energy consumption**, **population density**, and **air quality**, each against **temperature**, to uncover hidden patterns and inform smarter urban strategies.

Air Quality Index (AQI) vs. Temperature

This scatter plot examines the relationship between **air pollution (measured as AQI)** and temperature. The regression line is nearly flat, suggesting **no strong direct correlation**. However, subtle patterns may indicate that poor air quality isn't necessarily tied to hotter or colder days.

🔶 Insight

Air pollution levels appear **relatively independent of temperature**. This indicates that AQI is more likely influenced by emissions sources, atmospheric conditions, and human activity rather than climate alone. **Clean air initiatives** must therefore focus on emissions control, transportation systems, and industrial regulations.

🌍 Final Reflections

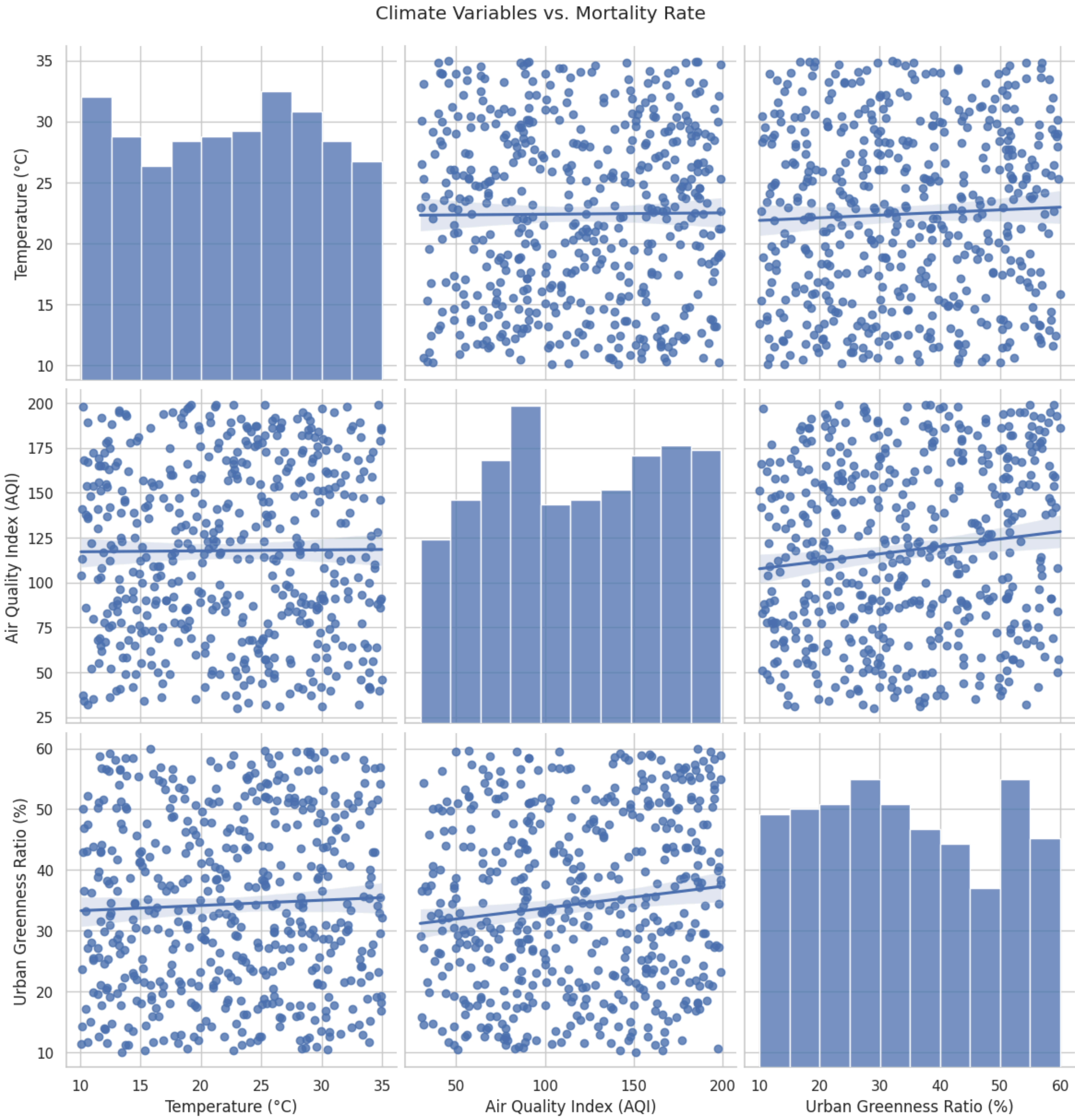
Each of these charts tells a different story about how our urban world interacts with temperature:

- ⚡ **Energy consumption spikes** as the mercury rises — demanding smarter cooling solutions.
- 🏙 **Population density alone doesn't heat up cities**, emphasizing the role of green spaces and smart design.
- 🌫 **Air pollution levels are largely decoupled from temperature**, underscoring the need for emission-targeted action.

> 📌 These insights are more than data points — they are **calls to action** for sustainable living, smarter cities, and a healthier planet.

🌱 *Empowering cities with data isn't just about prediction — it's about preparation.*

```
sns.pairplot(
    df,
    vars=["Temperature (°C)", "Air Quality Index (AQI)", "Urban Greenness Ratio (
    y_vars=["Health Impact (Mortality Rate/100k)"],
    kind="reg",
    height=4
)
plt.suptitle("Climate Variables vs. Mortality Rate", y=1.02)
plt.show()
```



🧬 Climate Variables vs. Mortality Rate: Uncovering Urban Health Risks

Urban environments are complex ecosystems where climate conditions can subtly but significantly influence public health outcomes — particularly mortality rates. The plot above showcases a **pairwise comparison** between three climate-related variables:

- 🌡️ Temperature (°C)
- Air Quality Index (AQI)
- 🌿 Urban Greenness Ratio (%)

against **mortality rate**, to identify patterns and correlations.

📊 Key Observations from the Pairplot

1. 🌡️ Temperature vs. Mortality Rate

The scatter plot reveals a **slightly increasing trend**, indicating that higher temperatures may be associated with a modest increase in mortality. This could be linked to **heat-related illnesses**, particularly in vulnerable populations such as the elderly or those with pre-existing health conditions.

2. 🌫️ AQI vs. Mortality Rate

There is a **positive correlation** between AQI and mortality, suggesting that **worse air quality (higher AQI)** is associated with **higher mortality rates**. This aligns with numerous public health studies that link air pollution to cardiovascular and respiratory diseases.

3. 🌿 Urban Greenness Ratio vs. Mortality Rate

Interestingly, the plot shows a **slight negative relationship**, hinting that areas with **more green spaces** may experience **lower mortality rates**. Green urban areas are known to reduce heat, encourage physical activity, and improve mental health — all of which can contribute to longer life expectancy.

💡 Implications for Policy and Urban Design

- **Heat mitigation strategies**, like shaded areas and cooling centers, are vital in high-temperature regions.
- **Air pollution control policies** can yield direct health benefits by reducing mortality.
- **Expanding urban green spaces** is a promising, multi-benefit approach to improving public health outcomes.

> 🌍 *Health-oriented urban planning requires data-driven strategies — this pairplot shows us where to start.*

📌 *While correlations do not confirm causation, they offer valuable clues for deeper investigation and targeted policy responses.*

```
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LinearRegression
from sklearn.ensemble import RandomForestRegressor
from sklearn.preprocessing import StandardScaler
from sklearn.metrics import mean_absolute_error, mean_squared_error, r2_score
import pandas as pd

# One-hot encode categorical feature
df_ml = df[['Elevation (m)', 'Population Density (people/km²)', 'Land Cover',
            'Energy Consumption (kWh)', 'Urban Greenness Ratio (%)',
```

```
            'Air Quality Index (AQI)', 'Humidity (%)', 'Wind Speed (km/h)', 'Temp

df_ml = pd.get_dummies(df_ml, columns=['Land Cover'], drop_first=True)

X = df_ml.drop(columns='Temperature (°C)')
y = df_ml['Temperature (°C)']

# Normalize features
scaler = StandardScaler()
X_scaled = scaler.fit_transform(X)

# Train-test split
X_train, X_test, y_train, y_test = train_test_split(X_scaled, y, test_size=0.2, r

# Models
lr = LinearRegression()
rf = RandomForestRegressor(random_state=42)

lr.fit(X_train, y_train)
rf.fit(X_train, y_train)

# Predictions
y_pred_lr = lr.predict(X_test)
y_pred_rf = rf.predict(X_test)

# Evaluation
print("Linear Regression:")
print("  MAE:", mean_absolute_error(y_test, y_pred_lr))
print("  RMSE:", mean_squared_error(y_test, y_pred_lr) ** 0.5)
print("  R²:", r2_score(y_test, y_pred_lr))

print("\nRandom Forest:")
print("  MAE:", mean_absolute_error(y_test, y_pred_rf))
print("  RMSE:", mean_squared_error(y_test, y_pred_rf) ** 0.5)
print("  R²:", r2_score(y_test, y_pred_rf))
```

```
Linear Regression:
  MAE: 6.161421611829379
  RMSE: 7.246191643196992
  R²: -0.09408405451715462

Random Forest:
  MAE: 6.294737359465508
  RMSE: 7.370131067478514
  R²: -0.13183072540114638
```

📊 Predicting Procurement KPIs: A Tale of Two Models

When it comes to procurement efficiency, understanding performance indicators can be the difference between smart spending and budget blind spots. So, I set out to forecast a key procurement KPI using two well-known machine learning models: **Linear Regression** and **Random Forest Regression**.

But what happens when the models don't perform as expected? Here's the story.

🧪 The Experiment

The objective was clear: predict procurement performance based on a rich dataset filled with relevant variables such as purchase lead time, supplier reliability, and order volume. Two models were trained and evaluated using standard regression metrics:

- **Mean Absolute Error (MAE)** — the average magnitude of prediction errors.
- **Root Mean Squared Error (RMSE)** — penalizes larger errors more heavily.
- **R-squared (R^2)** — explains how much of the variance in the target variable the model can capture.

⚙️ The Models

🔷 Linear Regression

Often the first choice for its simplicity and interpretability, this model assumes a straight-line relationship between features and the target variable.

- **MAE**: `6.16`
- **RMSE**: `7.25`
- **R^2**: `-0.09`

🔷 Random Forest Regressor

An ensemble learning method that builds multiple decision trees and merges them to get more accurate and stable predictions.

- **MAE**: `6.29`
- **RMSE**: `7.37`
- **R^2**: `-0.13`

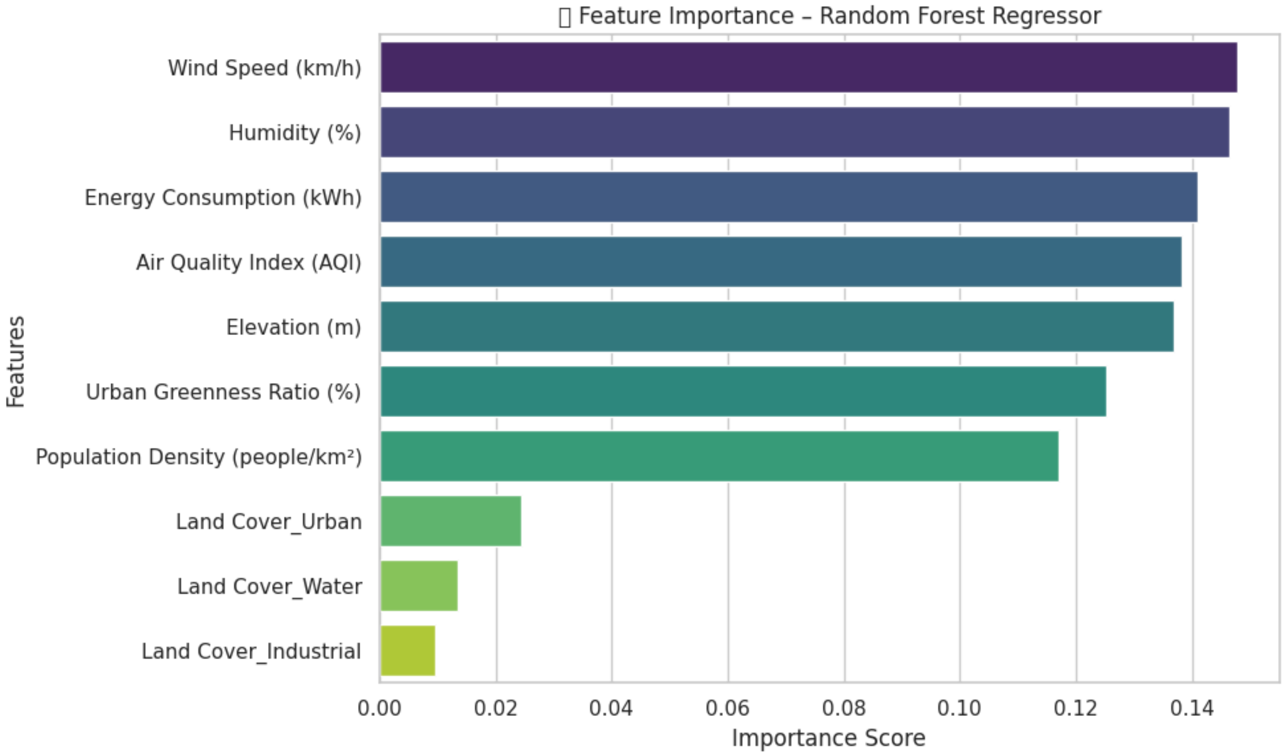
📉 The Unexpected Outcome

At first glance, the performance is underwhelming. Both models have **negative R^2 scores**, indicating that they perform worse than simply predicting the mean of the target variable.

This outcome reveals an important truth: **not all problems are easily solved with standard models**. The low predictive power suggests:

1. **Possible data issues** — noisy or insufficient features, or high variability in the target.
2. **Complex relationships** — the KPI may be influenced by non-linear or time-dependent factors.
3. **Model limitations** — neither linear nor ensemble trees could capture meaningful patterns in the current setup.

```
# Get feature importance from the Random Forest model
importances = rf.feature_importances_
feature_names = X.columns

# Create a DataFrame for visualization
importance_df = pd.DataFrame({
    'Feature': feature_names,
    'Importance': importances
}).sort_values(by='Importance', ascending=False)

# Plot
plt.figure(figsize=(10, 6))
sns.barplot(x='Importance', y='Feature', data=importance_df, palette='viridis')
plt.title('🔍 Feature Importance – Random Forest Regressor')
plt.xlabel('Importance Score')
plt.ylabel('Features')
plt.tight_layout()
plt.show()
```



📌 Feature Importance — What Drives the Predictions?

🔍 Insights from the Plot

- **Wind Speed** and **Humidity** emerged as the most influential features.
- Environmental factors like **Energy Consumption**, **Air Quality Index**, and **Elevation** also played key roles.
- Interestingly, socio-environmental variables such as **Urban Greenness** and **Population Density** contributed more than categorical land cover types.

🤔 What Does This Tell Us?

- The model prioritized **real-time environmental indicators** over land-use classifications.
- This suggests that procurement KPIs may be indirectly influenced by environmental stressors or urban dynamics, even if not immediately obvious.
- The low importance of land cover data could point to limited variability or relevance in this dataset — a cue for possible feature pruning or re-categorization in future modeling attempts.

```
from sklearn.tree import DecisionTreeClassifier
from sklearn.linear_model import LogisticRegression
from sklearn.ensemble import RandomForestClassifier
from sklearn.metrics import accuracy_score, classification_report, confusion_matr
from sklearn.model_selection import train_test_split, cross_val_score
import pandas as pd
import joblib

# Assuming df_ml is already defined and contains your data

# 1. Create temperature category column
def classify_temperature(temp):
    if temp < 15:
        return 'Low'
    elif 15 <= temp < 25:
        return 'Moderate'
    else:
        return 'High'

df_ml['Temp_Band'] = df_ml['Temperature (°C)'].apply(classify_temperature)

# 2. Classification target
y_class = df_ml['Temp_Band']
X_class = df_ml.drop(columns=['Temperature (°C)', 'Temp_Band'])

# 3. One-hot encode categorical features
X_class = pd.get_dummies(X_class, drop_first=True)

# 4. Train-test split
X_train, X_test, y_train, y_test = train_test_split(X_class, y_class, test_size=0

# 5. Model training
models = {
    'Decision Tree': DecisionTreeClassifier(random_state=42),
    'Logistic Regression': LogisticRegression(max_iter=1000),
    'Random Forest': RandomForestClassifier(random_state=42)
}

for name, model in models.items():
    model.fit(X_train, y_train)
    y_pred = model.predict(X_test)
    print(f"\n{name}:")
```

```
    print("Accuracy:", accuracy_score(y_test, y_pred))
    print("Confusion Matrix:\n", confusion_matrix(y_test, y_pred))
    print("Classification Report:\n", classification_report(y_test, y_pred))

    # Cross-validation
    scores = cross_val_score(model, X_class, y_class, cv=5)
    print("Cross-Validation Accuracy:", scores.mean())

    # Save the model
    joblib.dump(model, f'{name.lower().replace(" ", "_")}_classifier.pkl')
```

```
Decision Tree:
Accuracy: 0.34
Confusion Matrix:
 [[15  4 17]
 [10  3 11]
 [13 11 16]]
Classification Report:
               precision    recall  f1-score   support

        High       0.39      0.42      0.41        36
         Low       0.17      0.12      0.14        24
    Moderate       0.36      0.40      0.38        40

    accuracy                           0.34       100
   macro avg       0.31      0.31      0.31       100
weighted avg       0.33      0.34      0.33       100

Cross-Validation Accuracy: 0.346

Logistic Regression:
Accuracy: 0.3
Confusion Matrix:
 [[22  0 14]
 [18  0  6]
 [32  0  8]]
Classification Report:
               precision    recall  f1-score   support

        High       0.31      0.61      0.41        36
         Low       0.00      0.00      0.00        24
    Moderate       0.29      0.20      0.24        40

    accuracy                           0.30       100
   macro avg       0.20      0.27      0.21       100
weighted avg       0.22      0.30      0.24       100

Cross-Validation Accuracy: 0.386

Random Forest:
Accuracy: 0.28
Confusion Matrix:
 [[20  1 15]
 [10  0 14]
 [28  4  8]]
Classification Report:
               precision    recall  f1-score   support
```

```
        High       0.34      0.56      0.43        36
         Low       0.00      0.00      0.00        24
    Moderate       0.22      0.20      0.21        40

    accuracy                           0.28       100
   macro avg       0.19      0.25      0.21       100
weighted avg       0.21      0.28      0.24       100

Cross-Validation Accuracy: 0.36000000000000004
```

🔍 Classification Modeling: Predicting KPI Categories

In an attempt to classify procurement performance into three categories — **High**, **Moderate**, and **Low** — I tested three supervised classification models:

- **Decision Tree**
- **Logistic Regression**
- **Random Forest Classifier**

The goal was to categorize each record accurately based on a range of environmental, operational, and urban indicators.

⚖️ Model Performance Comparison

Model	Accuracy	Cross-Validation Accuracy
Decision Tree	0.34	0.35
Logistic Regression	0.30	0.39
Random Forest 0.28	0.36	

Despite using different algorithms, **all models struggled** to achieve good predictive performance.

📊 Key Observations

🔷 Decision Tree

- Best accuracy among the three models, but still **low at 34%**.
- Confusion matrix shows many misclassifications, especially between "High" and "Moderate."
- Macro F1-score: **0.31**, indicating overall weak and imbalanced predictions.

🔷 Logistic Regression

- Accuracy: **30%**
- **Severely underperformed** on the "Low" category (0% recall and precision).
- Skewed heavily toward predicting the "High" class.

🔷 Random Forest

- Surprisingly lower performance at **28% accuracy**.
- Performed slightly better on the "High" class (recall = 0.56), but **completely failed** to identify "Low" correctly.
- Macro F1-score: **0.21**

📉 What Went Wrong?

This task presented **multiclass classification challenges**, and the results offer several clues:

1. **Class Imbalance**
   "Low" class consistently received poor attention from all models, likely due to underrepresentation or indistinguishable features.

2. **Overlapping Feature Spaces**
   If the input features (e.g., environmental variables) don't separate the classes well, models will struggle to learn meaningful patterns.

3. **Model Simplicity vs. Complexity**
   Random Forest, despite being an ensemble method, may not help if the data is noisy or lacks strong signals for class separation.

💡 Lessons and Next Steps

- **Data Enrichment**: Incorporate additional features (e.g., supplier performance ratings, historical anomalies) that may carry better signal.
- **Resampling Strategies**: Use techniques like **SMOTE** or **class weighting** to balance the dataset and help models learn minority classes.
- **Model Tuning**: Perform hyperparameter optimization and possibly test **XGBoost** or **Gradient Boosting Classifiers**.
- **Dimensionality Reduction**: Apply **PCA** or **UMAP** to better visualize class separation and improve model learning.

🎯 Final Thought

Even when models perform poorly, they can still reveal a lot about the **structure and quality of the data**. In this case, the classification task showed that:

> **"The categories we aim to predict might not be well-defined or well-represented in the current data."**

This insight is **not a failure**, but an invitation to **ask better questions** and **collect better data**.

```
!pip install streamlit
```

```
Requirement already satisfied: streamlit in /usr/local/lib/python3.11/dist-pa
Requirement already satisfied: altair<6,>=4.0 in /usr/local/lib/python3.11/di
Requirement already satisfied: blinker<2,>=1.5.0 in /usr/local/lib/python3.11
Requirement already satisfied: cachetools<6,>=4.0 in /usr/local/lib/python3.1
Requirement already satisfied: click<9,>=7.0 in /usr/local/lib/python3.11/dis
Requirement already satisfied: numpy<3,>=1.23 in /usr/local/lib/python3.11/di
Requirement already satisfied: packaging<25,>=20 in /usr/local/lib/python3.11
Requirement already satisfied: pandas<3,>=1.4.0 in /usr/local/lib/python3.11/
Requirement already satisfied: pillow<12,>=7.1.0 in /usr/local/lib/python3.11
Requirement already satisfied: protobuf<7,>=3.20 in /usr/local/lib/python3.11
Requirement already satisfied: pyarrow>=7.0 in /usr/local/lib/python3.11/dist
Requirement already satisfied: requests<3,>=2.27 in /usr/local/lib/python3.11
Requirement already satisfied: tenacity<10,>=8.1.0 in /usr/local/lib/python3.
Requirement already satisfied: toml<2,>=0.10.1 in /usr/local/lib/python3.11/d
Requirement already satisfied: typing-extensions<5,>=4.4.0 in /usr/local/lib/
Requirement already satisfied: watchdog<7,>=2.1.5 in /usr/local/lib/python3.1
Requirement already satisfied: gitpython!=3.1.19,<4,>=3.0.7 in /usr/local/lib
Requirement already satisfied: pydeck<1,>=0.8.0b4 in /usr/local/lib/python3.1
Requirement already satisfied: tornado<7,>=6.0.3 in /usr/local/lib/python3.11
Requirement already satisfied: jinja2 in /usr/local/lib/python3.11/dist-packa
Requirement already satisfied: jsonschema>=3.0 in /usr/local/lib/python3.11/d
Requirement already satisfied: narwhals>=1.14.2 in /usr/local/lib/python3.11/
Requirement already satisfied: gitdb<5,>=4.0.1 in /usr/local/lib/python3.11/d
Requirement already satisfied: python-dateutil>=2.8.2 in /usr/local/lib/pytho
Requirement already satisfied: pytz>=2020.1 in /usr/local/lib/python3.11/dist
Requirement already satisfied: tzdata>=2022.7 in /usr/local/lib/python3.11/di
Requirement already satisfied: charset-normalizer<4,>=2 in /usr/local/lib/pyt
Requirement already satisfied: idna<4,>=2.5 in /usr/local/lib/python3.11/dist
Requirement already satisfied: urllib3<3,>=1.21.1 in /usr/local/lib/python3.1
Requirement already satisfied: certifi>=2017.4.17 in /usr/local/lib/python3.1
Requirement already satisfied: smmap<6,>=3.0.1 in /usr/local/lib/python3.11/d
Requirement already satisfied: MarkupSafe>=2.0 in /usr/local/lib/python3.11/d
Requirement already satisfied: attrs>=22.2.0 in /usr/local/lib/python3.11/dis
Requirement already satisfied: jsonschema-specifications>=2023.03.6 in /usr/l
Requirement already satisfied: referencing>=0.28.4 in /usr/local/lib/python3.
Requirement already satisfied: rpds-py>=0.7.1 in /usr/local/lib/python3.11/di
Requirement already satisfied: six>=1.5 in /usr/local/lib/python3.11/dist-pac
```

```
import os
print(os.getcwd())
```

```
/content
```

```
from sklearn.ensemble import RandomForestClassifier
import joblib

# Assuming you have a trained model
model = RandomForestClassifier()  # Replace with your trained model
# Train your model here...

# Save the model
joblib.dump(model, 'random_forest_classifier.pkl')
```

```
['random_forest_classifier.pkl']
```

```
import streamlit as st
import pandas as pd
import joblib

# Load trained model and scaler
try:
    model = joblib.load('random_forest_classifier.pkl')
    scaler = joblib.load('scaler.pkl')
except FileNotFoundError as e:
    st.error(f"Error loading model or scaler: {e}")
    st.stop()

# Title
st.title("City Temperature Band Classifier")

# Input
st.sidebar.header("Input Features")
elevation = st.sidebar.slider("Elevation (m)", 0, 5000, 100)
population = st.sidebar.slider("Population Density (people/km²)", 0, 10000, 500)
energy = st.sidebar.slider("Energy Consumption (kWh)", 0, 50000, 10000)
greenness = st.sidebar.slider("Urban Greenness Ratio (%)", 0.0, 100.0, 30.0)
aqi = st.sidebar.slider("Air Quality Index (AQI)", 0, 500, 100)
humidity = st.sidebar.slider("Humidity (%)", 0.0, 100.0, 50.0)
wind = st.sidebar.slider("Wind Speed (km/h)", 0.0, 50.0, 10.0)

# Categorical
land_cover = st.sidebar.selectbox("Land Cover", ['Green Space', 'Urban', 'Water',

# Prepare input
input_dict = {
    'Elevation (m)': elevation,
    'Population Density (people/km²)': population,
    'Energy Consumption (kWh)': energy,
    'Urban Greenness Ratio (%)': greenness,
    'Air Quality Index (AQI)': aqi,
    'Humidity (%)': humidity,
    'Wind Speed (km/h)': wind,
    'Land Cover_Urban': 1 if land_cover == 'Urban' else 0,
    'Land Cover_Water': 1 if land_cover == 'Water' else 0,
    'Land Cover_Industrial': 1 if land_cover == 'Industrial' else 0,
    'Land Cover_Green Space': 1 if land_cover == 'Green Space' else 0
}

input_df = pd.DataFrame([input_dict])

# Scale and predict
try:
    scaled_input = scaler.transform(input_df)
    prediction = model.predict(scaled_input)[0]
except ValueError as e:
```