import pandas as pd

import numpy as np

import matplotlib.pyplot as plt

import seaborn as sns

from statsmodels.tsa.arima.model import ARIMA

from sklearn.metrics import mean\_squared\_error

from math import sqrt

from sklearn.preprocessing import MinMaxScaler

from keras.models import Sequential

from keras.layers import Dense, LSTM

from tensorflow.keras.preprocessing.sequence import TimeseriesGenerator

df = pd.read\_csv("/content/ep.csv")

### df.head()

print("Missing values:\n")

df.isnull().sum()

df.fillna(method='ffill', inplace=True)

df.head()

df["site\_eui"] = pd.to\_numeric(df["site\_eui"], errors="coerce")

# Plot original distribution

plt.figure(figsize=(8, 4))

df["site\_eui"].hist(bins=50)

plt.title("Original Site EUI Distribution")

plt.xlabel("site\_eui")

plt.ylabel("Frequency")

plt.grid(True)

plt.show()

Q1 = df["site\_eui"].quantile(0.25)

Q3 = df["site\_eui"].quantile(0.75)

IQR = Q3 - Q1

# Define bounds

lower\_bound = Q1 - 1.5 \* IQR

upper\_bound = Q3 + 1.5 \* IQR

# Count outliers

outliers = df[(df["site\_eui"] < lower\_bound) | (df["site\_eui"] > upper\_bound)]

print(f"Number of outliers in 'site\_eui': {outliers.shape[0]}")

df\_filtered = df[(df["site\_eui"] >= lower\_bound) & (df["site\_eui"] <= upper\_bound)]

# Plot cleaned distribution

plt.figure(figsize=(8, 4))

df\_filtered["site\_eui"].hist(bins=50)

plt.title("Site EUI After Outlier Removal")

plt.xlabel("site\_eui")

plt.ylabel("Frequency")

plt.grid(True)

plt.show()

df['site\_eui'] = pd.to\_numeric(df['site\_eui'], errors='coerce')

df['Year\_Factor'] = pd.to\_numeric(df['Year\_Factor'], errors='coerce')

# Drop missing values in key fields

df = df.dropna(subset=['site\_eui', 'Year\_Factor'])

# Round year for grouping

df['year'] = df['Year\_Factor'].round().astype('Int64')

print("Basic statistics for Site Energy Use Intensity:")

print(df['site\_eui'].describe())

plt.figure(figsize=(10, 5))

plt.hist(df["site\_eui"], bins=50, color="skyblue", edgecolor="black")

plt.title("Histogram of Site Energy Use Intensity (site\_eui)")

plt.xlabel("Site EUI")

plt.ylabel("Frequency")

plt.grid(True)

plt.tight\_layout()

plt.show()

climate\_cols = [

"avg\_temp", "cooling\_degree\_days", "heating\_degree\_days",

"precipitation\_inches", "snowfall\_inches", "snowdepth\_inches",

"days\_below\_30F", "days\_above\_80F", "days\_with\_fog"

]

# Convert columns to numeric

for col in climate\_cols:

if col in df.columns:

df[col] = pd.to\_numeric(df[col], errors="coerce")

# Drop rows with all missing climate values

df\_climate = df[climate\_cols].dropna(how="all")

# Compute correlation matrix

correlation\_matrix = df\_climate.corr()

# Plot heatmap

plt.figure(figsize=(10, 8))

sns.heatmap(correlation\_matrix, annot=True, cmap="coolwarm", linewidths=0.5)

plt.title("Correlation Heatmap: Climate-Related Features")

plt.tight\_layout()

plt.show()

df["Year\_Factor"] = pd.to\_numeric(df["Year\_Factor"], errors="coerce")

# Drop missing values

df = df.dropna(subset=["site\_eui", "Year\_Factor"])

# Group by year and calculate mean site\_eui

yearly\_avg = df.groupby("Year\_Factor")["site\_eui"].mean().reset\_index()

# Plot

plt.figure(figsize=(10, 5))

plt.plot(yearly\_avg["Year\_Factor"], yearly\_avg["site\_eui"], marker="o", linestyle="-", color="teal")

plt.title("Year-wise Average Site Energy Use Intensity (EUI)")

plt.xlabel("Year")

plt.ylabel("Average Site EUI")

plt.grid(True)

plt.tight\_layout()

plt.show()

df\_bar = df.groupby("Year\_Factor")["site\_eui"].mean().reset\_index()

df\_bar["Year\_Factor"] = df\_bar["Year\_Factor"].astype(int)

# Generate colors using a colormap (e.g., viridis or tab20)

colors = plt.cm.tab20(np.linspace(0, 1, len(df\_bar)))

# Plot with multiple colors

plt.figure(figsize=(10, 6))

plt.bar(df\_bar["Year\_Factor"], df\_bar["site\_eui"], color=colors)

plt.title("Average Site EUI by Year")

plt.xlabel("Year")

plt.ylabel("Average Site EUI")

plt.grid(True)

plt.tight\_layout()

plt.show()

top\_facilities = df["facility\_type"].value\_counts().nlargest(5)

plt.figure(figsize=(6, 6))

plt.pie(top\_facilities.values, labels=top\_facilities.index, autopct='%1.1f%%', startangle=140)

plt.title("Top 5 Facility Types Distribution")

plt.axis("equal")

plt.tight\_layout()

plt.show()

unique\_years = df["Year\_Factor"].dropna().unique()[:5]

df\_subset = df[df["Year\_Factor"].isin(unique\_years)]

# Create the boxplot

plt.figure(figsize=(10, 6))

sns.boxplot(x="Year\_Factor", y="site\_eui", data=df\_subset)

plt.title("Site EUI Distribution for First 5 Unique Year\_Factor Values")

plt.xlabel("Year")

plt.ylabel("Site EUI")

plt.grid(True)

plt.tight\_layout()

plt.show()

plt.figure(figsize=(10, 6))

sns.scatterplot(x='floor\_area', y='site\_eui', data=df, hue='building\_class', palette='Set1')

plt.title('Floor Area vs Site EUI')

plt.show()

plt.figure(figsize=(12, 6))

sns.violinplot(x='facility\_type', y='site\_eui', data=df, palette='muted')

plt.title('Site EUI Distribution by Facility Type')

plt.xticks(rotation=90)

plt.show()

### 

### 