

2.2/2.2 MB 24.4 MB/s eta 0:00:00

```

Choose Files kaggle.json
• kaggle.json(application/json) - 64 bytes, last modified: 4/11/2025 - 100% done
Saving kaggle.json to kaggle.json
!kaggle json --h '1' --u 'https://www.kaggle.com' --k 'k' --c 'c2b096702d90b4c2a161549734c60c' '1'

```

```
!unzip climate-change-earth-surface-temperature-data.zip
```

```
import pandas as pd
import numpy as np
import os
from sklearn.metrics import mean_squared_error
from sklearn.preprocessing import MinMaxScaler
from tqdm import tqdm
import matplotlib.pyplot as plt
```

```
# Load and clean data
df = pd.read_csv("GlobalLandTemperaturesByCity.csv")
df['dt'] = pd.to_datetime(df['dt'])
df = df[df['AverageTemperature'].notnull()]
df['City_Country'] = df['City'] + ', ' + df['Country']
cities = df['City_Country'].unique()
```

```
for city in tqdm(cities, desc="Processing cities"):
    city_data = df[df['City_Country'] == city].copy()
    city_data.set_index('dt', inplace=True)
    # city_data = city_data.resample('M').mean()
    city_data = city_data.resample('M').agg({'AverageTemperature': 'mean'})
```

```
try:
    # Scale
    scaler = MinMaxScaler()
    y_scaled = scaler.fit_transform(city_data[['AverageTemperature']])
```

```
# Supervised learning
def create_sequences(data, look_back=12):
    X, y = [], []
    for i in range(len(data) - look_back):
        X.append(data[i:i+look_back])
        y.append(data[i+look_back])
    return np.array(X), np.array(y)
```

```
look_back = 12
X, y = create_sequences(y_scaled, look_back)
split = int(len(X) * 0.8)
X_train, X_test = X[:split], X[split:]
y_train, y_test = y[:split], y[split:]
```

```
# LSTM
model = Sequential()
model.add(LSTM(50, input_shape=(X_train.shape[1], X_train.shape[2])))
model.add(Dense(1))
model.compile(optimizer='adam', loss='mse')
model.fit(X_train, y_train, epochs=10, batch_size=32, verbose=0)
```

```
# ARIMA
arma_series = city_data['AverageTemperature']
arma_model = ARIMA(arma_series, order=(3,1,2))
arma_fitted = arma_model.fit()
arma_forecast = arma_model.fitted.forecast(steps=len(y_test))
```

```
# Combine
y_test_actual = scaler.inverse_transform(y_test)
arima_forecast_flat = arima_forecast.squeeze()
residual_correction = model.predict(X_test)[:len(arima_forecast_flat)]
residual_correction = scaler.inverse_transform(residual_correction)
```

```
hybrid forecast = arima forecast flat + residual correction.squeeze()
```

```
# RMSE
arma_rmse = np.sqrt(mean_squared_error(y_test_actual, arma_forecast_flat))
lstm_rmse = np.sqrt(mean_squared_error(y_test_actual, residual_correction))
hybrid_rmse = np.sqrt(mean_squared_error(y_test_actual, hybrid_forecast))
```

```
results.append({
    "City": city,
    "ARIMA_RMSE": arima_rmse,
    "LSTM_Residual_RMSE": lstm_rmse,
    "Hybrid_RMSE": hybrid_rmse
})
```

```
except Exception as e:
    continue
```

```
# Save
results_df = pd.DataFrame(results)
results_df.to_csv("citywise_forecast_rmse.csv", index=False)
print("✅ Saved to citywise_forecast_rmse.csv")

# Visualization: Top 10 cities by Hybrid RMSE
results_df_sorted = results_df.sort_values(by="Hybrid_RMSE")

plt.figure(figsize=(12, 6))
plt.barh(results_df_sorted['City'].head(10), results_df_sorted['Hybrid_RMSE'].head(10), color='green')
plt.xlabel("Hybrid RMSE")
plt.title("Top 10 Cities with Lowest Hybrid Forecast Error")
plt.gca().invert_yaxis()
plt.tight_layout()
plt.show()
```

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]