HYBRID RENEWABLE ENERGY PREDICTION - SOURCE CODE

```
# ============================================
# HYBRID RENEWABLE ENERGY PREDICTION - ALL IN ONE
# ============================================
# This script:
# 1) Loads dataset (expects 'hybrid_energy_full_table.csv' in same folder)
# 2) Splits data, trains RandomForest model
# 3) Generates graphs (line + scatter)
# 4) Creates a PDF report (metrics, plots, full predictions table)
# 5) Saves all outputs in ./results
#
# ---------------------------
# DATA & FORMULAS INCLUDED
# ---------------------------
# Dataset columns (expected numeric features, names may vary in your CSV):
# - Temperature_C            : Ambient temperature ( C)
# - Humidity_%               : Relative humidity (%)
# - Solar_Irradiance_W_m2    : Solar irradiance (W/m^2)
# - Wind_Speed_m_s           : Wind speed (m/s)
# - Cloud_Cover              : Cloud fraction [0..1]
# - Pressure_hPa             : Air pressure (hPa)
# - Total_Output_kW          : Target variable (kW)
#
# Solar output (synthetic physics-inspired relationship used when generating data):
#   SolarEfficiency = 0.18 * (1 - 0.005*(Temperature_C - 25)) * (1 - 0.4*Cloud_Cover)
#   Solar_Output_kW   Solar_Irradiance_W_m2 * SolarEfficiency / 1000
#
# Wind output (synthetic physics-inspired relationship used when generating data):
#   AirDensity   1.225 * (1 - 0.0036*(Temperature_C - 15))
#   Wind_Output_kW   0.5 * AirDensity * (Wind_Speed_m_s^3) * K
# where K is a small scaling constant used during data generation.
#
# Total output:
#   Total_Output_kW = Solar_Output_kW + Wind_Output_kW
#
# Note: In this script we DO NOT recompute outputs; we PREDICT Total_Output_kW
# from numeric features using machine learning.
# ============================================

import os
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
from matplotlib.backends.backend_pdf import PdfPages
from sklearn.model_selection import train_test_split
from sklearn.ensemble import RandomForestRegressor
from sklearn.metrics import mean_absolute_error, mean_squared_error, r2_score

CSV_PATH = "hybrid_energy_full_table.csv"
OUTDIR = "results"
```

```
def main():
    # --------------------------
    # ### DATASET SECTION
    # --------------------------
    os.makedirs(OUTDIR, exist_ok=True)
    df = pd.read_csv(CSV_PATH)
    if "Total_Output_kW" not in df.columns:
        raise KeyError("Expected column 'Total_Output_kW' not found in CSV.")
    num_df = df.select_dtypes(include=[np.number]).copy()
    if "Total_Output_kW" not in num_df.columns:
        raise KeyError("After numeric selection, 'Total_Output_kW' missing. Ensure it is
numeric.")
    X = num_df.drop(columns=["Total_Output_kW"])
    y = num_df["Total_Output_kW"]

    # --------------------------
    # ### TRAIN/TEST SPLIT
    # --------------------------
    X_train, X_test, y_train, y_test = train_test_split(
        X, y, test_size=0.2, random_state=42
    )

    # --------------------------
    # ### MODEL TRAINING SECTION
    # --------------------------
    model = RandomForestRegressor(n_estimators=300, random_state=42)
    model.fit(X_train, y_train)

    # --------------------------
    # ### PREDICTION & METRICS
    # --------------------------
    y_pred_test = model.predict(X_test)
    y_pred_all = model.predict(X)

    mae = mean_absolute_error(y_test, y_pred_test)
    rmse = mean_squared_error(y_test, y_pred_test, squared=False)
    r2 = r2_score(y_test, y_pred_test)

    print("=== METRICS ===")
    print("MAE:", round(mae, 6))
    print("RMSE:", round(rmse, 6))
    print("R2 :", round(r2, 6))

    # Save predictions for ALL entries
    df_out = df.copy()
    df_out["Predicted_Total_Output_kW"] = y_pred_all
    out_csv = os.path.join(OUTDIR, "all_predictions.csv")
    df_out.to_csv(out_csv, index=False)

    # --------------------------
    # ### VISUALIZATION SECTION
    # --------------------------
```

```
    # Plot 1: first 100 samples line plot
    nshow = min(100, len(y))
    plt.figure(figsize=(12,6))
    plt.plot(range(nshow), y.iloc[:nshow].values, label="Actual", linewidth=2)
    plt.plot(range(nshow), y_pred_all[:nshow], label="Predicted RF", linewidth=2)
    plt.title("Actual vs Predicted Power Output (First 100 Samples)")
    plt.xlabel("Sample Index")
    plt.ylabel("Power (kW)")
    plt.legend()
    plt.grid(True)
    plt.tight_layout()
    plot1_path = os.path.join(OUTDIR, "line_first100.png")
    plt.savefig(plot1_path, dpi=200)
    plt.close()

    # Plot 2: all samples line plot
    plt.figure(figsize=(12,6))
    plt.plot(range(len(y)), y.values, label="Actual", linewidth=1.5, marker='o', markersize=2)
    plt.plot(range(len(y)), y_pred_all, label="Predicted", linewidth=1.5, marker='x',
markersize=2)
    plt.title("Random Forest: Actual vs Predicted Power Output (All Samples)")
    plt.xlabel("Sample")
    plt.ylabel("Total_Output_kW")
    plt.legend()
    plt.grid(True)
    plt.tight_layout()
    plot2_path = os.path.join(OUTDIR, "line_all.png")
    plt.savefig(plot2_path, dpi=200)
    plt.close()

    # Plot 3: scatter predicted vs actual (all samples)
    plt.figure(figsize=(8,6))
    plt.scatter(y, y_pred_all, alpha=0.6)
    minv, maxv = float(min(y.min(), y_pred_all.min())), float(max(y.max(), y_pred_all.max()))
    plt.plot([minv, maxv], [minv, maxv], linestyle="--")
    plt.xlabel("Actual Output (kW)")
    plt.ylabel("Predicted Output (kW)")
    plt.title("Predicted vs Actual (All Samples)")
    plt.grid(True)
    plt.tight_layout()
    plot3_path = os.path.join(OUTDIR, "scatter_pred_vs_actual.png")
    plt.savefig(plot3_path, dpi=200)
    plt.close()

    # ---------------------------
    # ### REPORT GENERATION SECTION
    # ---------------------------
    pdf_path = os.path.join(OUTDIR, "Hybrid_Energy_Report.pdf")
    with PdfPages(pdf_path) as pdf:
        # Page 1: Title + Metrics
        fig = plt.figure(figsize=(11.69,8.27))
        plt.axis("off")
        title = "Hybrid Renewable Energy Prediction Report"
```

```
        metrics_text = (
            f"Rows: {len(df)}\n"
            f"Features used: {list(X.columns)}\n\n"
            "Model: RandomForestRegressor (300 trees)\n\n"
            f"MAE: {mae:.6f}\n"
            f"RMSE: {rmse:.6f}\n"
            f"R : {r2:.4f}"
        )
        plt.text(0.05, 0.9, title, fontsize=18, weight="bold", va="top")
        plt.text(0.05, 0.8, metrics_text, fontsize=13, va="top")
        pdf.savefig(fig); plt.close(fig)

        # Page 2: Formulas and Dataset Overview
        fig = plt.figure(figsize=(11.69,8.27))
        plt.axis("off")
        formulas = (
            "Formulas & Dataset Overview\n\n"
            "Dataset fields (typical):\n"
            " - Temperature_C ( C), Humidity_% (%), Solar_Irradiance_W_m2 (W/m ),\n"
            "   Wind_Speed_m_s (m/s), Cloud_Cover (0..1), Pressure_hPa (hPa), Total_Output_kW
(kW)\n\n"
            "Solar (synthetic):\n"
            "  SolarEfficiency = 0.18 * (1 - 0.005*(Temperature_C - 25)) * (1 -
0.4*Cloud_Cover)\n"
            "  Solar_Output_kW   Solar_Irradiance_W_m2 * SolarEfficiency / 1000\n\n"
            "Wind (synthetic):\n"
            "  AirDensity   1.225 * (1 - 0.0036*(Temperature_C - 15))\n"
            "  Wind_Output_kW   0.5 * AirDensity * (Wind_Speed_m_s^3) * K\n\n"
            "Total:\n"
            "  Total_Output_kW = Solar_Output_kW + Wind_Output_kW\n\n"
            "Note: In this project, the ML model predicts Total_Output_kW from numeric features."
        )
        plt.text(0.05, 0.95, formulas, fontsize=12, va="top")
        pdf.savefig(fig); plt.close(fig)

        # Add plots
        for p in [plot1_path, plot2_path, plot3_path]:
            fig = plt.figure(figsize=(11.69,8.27))
            img = plt.imread(p)
            plt.imshow(img)
            plt.axis("off")
            pdf.savefig(fig); plt.close(fig)

        # Paginated predictions table (ALL entries)
        table_df = pd.DataFrame({
            "Index": np.arange(len(y)),
            "Actual_kW": y.values,
            "Predicted_kW": y_pred_all
        })
        rows_per_page = 30
        for i in range(0, len(table_df), rows_per_page):
            chunk = table_df.iloc[i:i+rows_per_page]
            fig = plt.figure(figsize=(11.69,8.27))
```

```
            plt.axis("off")
            plt.title(f"Predictions (Rows {i+1} {i+len(chunk)})", pad=20)
            tab = plt.table(cellText=chunk.values, colLabels=chunk.columns, loc="center")
            tab.scale(1, 1.3)
            pdf.savefig(fig); plt.close(fig)

if __name__ == "__main__":
    main()
```