Mini-Hackathon: Waste Management and Recycling in Indian Cities

Overview

Welcome to the Mini-Hackathon, organized by **PWSkills**, a leader in empowering aspiring data scientists with practical, hands-on learning experiences. Open to all enthusiasts, from beginners to seasoned practitioners, this bi-monthly Hackathon fosters innovation and technical expertise in data-driven solutions. This edition, themed around the **Waste Management and Recycling in Indian Cities** challenge, invites participants to tackle a critical real-world problem in urban sustainability using machine learning. By predicting recycling rates or exploring other waste management insights, participants will contribute to efficient waste management practices and sustainable urban development while honing their data science skills.

---

1. Problem Statement

The objective of this Hackathon is to develop a machine learning model that predicts the recycling rate (the target variable, “Recycling Rate (%)”) for a given set of city-specific waste management attributes, such as waste type, population density, and municipal efficiency. This is primarily a **regression problem**, where the goal is to minimize prediction error (e.g., Root Mean Squared Error, RMSE). The challenge leverages the **Waste Management and Recycling in India** dataset, enabling participants to explore data-driven insights that can optimize waste management systems, reduce environmental impact, and inform urban policy-making. Participants will analyze features like waste generation, cost of management, and landfill capacity to derive actionable solutions, simulating real-world decision-making in sustainable urban planning.

---

2. Data - Explanation

The **Waste Management and Recycling in India** dataset is a comprehensive collection of attributes related to waste generation, recycling, and disposal practices across various Indian cities. Sourced from simulated data based on real-world waste management trends in India, this dataset spans 2019–2023 and covers multiple waste types, including plastic, organic, electronic, construction, and hazardous waste. It is pivotal for developing predictive models that support sustainable waste management—a data-driven approach to optimize municipal systems, reduce landfill dependency, and enhance recycling efforts. Each row in the dataset represents a unique combination of city, waste type, and management attributes, paired with outcomes like recycling rates or disposal methods, making it an ideal resource for building and evaluating machine learning models.

Dataset Overview

- **Source**: Simulated dataset based on average waste management practices in Indian cities (inspired by municipal corporations and environmental reports) **(Link).**
- **Columns**:
  - **City/District (categorical)**: Name of the Indian city or district (e.g., Mumbai, Delhi).
  - **Waste Type (categorical)**: Type of waste generated (e.g., Plastic, Organic, E-Waste, Construction, Hazardous).
  - **Waste Generated (Tons/Day) (float)**: Amount of waste generated in tons per day.
  - **Recycling Rate (%) (float)**: Percentage of waste that is recycled, the primary target for regression tasks.
  - **Population Density (People/km^2) (float)**: Number of people per square kilometer in the city.
  - **Municipal Efficiency Score (1-10) (int)**: Score indicating municipal waste management effectiveness (e.g., segregation, collection, disposal).
  - **Disposal Method (categorical)**: Method used for waste disposal (e.g., Landfill, Recycling, Incineration, Composting), a potential target for classification tasks.
  - **Cost of Waste Management (₹/Ton) (float)**: Cost of managing one ton of waste in Indian Rupees.
  - **Awareness Campaigns Count (int)**: Number of waste management awareness campaigns organized in the year.
  - **Landfill Name (categorical)**: Name of the landfill site used by the city.
  - **Landfill Location (Lat, Long) (float, float)**: Geographical coordinates (latitude, longitude) of the landfill.
  - **Landfill Capacity (Tons) (float)**: Total waste capacity of the landfill in tons.

    - **Year (int)**: Year of the data entry (2019–2023).
- **Task**: Primarily, predict the "Recycling Rate (%)" based on the provided features (regression).
- **Anticipated Challenges & Considerations**:
    - Effectively handling and encoding categorical variables (e.g., City/District, Waste Type, Disposal Method).
    - Addressing potential class imbalances in classification tasks (e.g., Disposal Method).
    - Ensuring robust feature engineering to capture temporal trends (Year) and geospatial relationships (Landfill Location).
    - Managing missing or noisy data, if present, to maintain data integrity.
    - Incorporating multi-dimensional features (e.g., numerical, categorical, geospatial) into a cohesive model.

---

3. Tech Stack and Tools: Recommended Resources

Participants are encouraged to use a robust tech stack to tackle this challenge, supporting both machine learning model development and the creation of a Flask-based application for deployment. The following tools, libraries, and platforms are recommended but not mandatory, allowing flexibility for creative solutions:

- **- Programming Language:** Python (version ≥3.9)
- **- ML Libraries:** Scikit-learn, XGBoost/LightGBM/CatBoost, Joblib
- **- Web Development:** Flask, Streamlit (optional), Gunicorn
- **- Data Visualization:** Matplotlib, Seaborn,Plotly, Folium (optional)
- **- Platforms:** Google Colab, AWS, Render, Vercel, Hugging Face Spaces
- **- Other Tools:** Pandas, NumPy, GeoPandas (optional)
- **- Environment: Virtual environments (venv, conda) with requirements.txt for dependency management.**

---

4. Submission Criteria: Ensuring a Successful Entry

Participants must submit their work via **GitHub** or **Google Drive** to ensure accessibility and ease of evaluation. Submissions should include the components listed below, organized in a clear folder structure to support both experimentation and deployment of a Flask-based application. Participants are encouraged to deploy their application on one of the specified platforms (AWS, Render, Vercel, or Hugging Face Spaces) to demonstrate real-world applicability, though this is optional for bonus consideration.

Submission Components

- **Code**: A well-organized script or Jupyter Notebook containing the complete solution (data preprocessing, model training, and predictions), alongside modular code for a Flask-based application.
- **Model Output**: A .csv file with predictions on the test set (e.g., predicted recycling rates or disposal methods) in a format suitable for evaluation.
- **Documentation**: A detailed report (see Section 6) explaining the approach, methodology, findings, and deployment process (if applicable).
- **README**: A file explaining how to run the code and deploy the application, including dependencies, instructions, and platform-specific details.
- **Deployment Link** (optional): A working URL to the deployed Flask application (hosted on AWS, Render, Vercel, or Hugging Face Spaces), demonstrating model predictions.

Folder Structure

To ensure clarity and reproducibility, participants should organize their project using the following folder structure:

```
project_root/
 ├──── Notebooks/
 │    ├──── data_preparation.ipynb       # Data cleaning and preprocessing
 │    ├──── exploratory_data_analysis.ipynb  # EDA and visualizations
 │    ├──── model_training.ipynb         # Model training and evaluation
 ├──── src/
 │    ├──── __init__.py                 # Marks src as a Python package
 │    ├──── data/
 │    │    ├──── __init__.py
 │    │    ├──── preprocess.py          # Data preprocessing functions/classes
 │    ├──── models/
 │    │    ├──── __init__.py
 │    │    ├──── train.py               # Model training logic
 │    │    ├──── predict.py             # Prediction functions/classes
 │    ├──── utils/
 │    │    ├──── __init__.py
 │    │    ├──── helpers.py    # Utility functions (e.g., data loading, logging)
 │    ├──── app.py             # Flask application for serving predictions
 ├──── data/
 │    ├──── raw/           # Raw dataset files (e.g., train.csv, test.csv)
 │    ├──── processed/    # Processed data (e.g., cleaned or encoded datasets)
```

```
├──── models/
│     ├──── trained_model.pkl          # Saved trained model(s)
├──── static/             # Static files for Flask app (e.g., CSS, JS)
├──── templates/                 # HTML templates for Flask app
├──── requirements.txt             # Project dependencies
├──── README.md            # Instructions for running code and deployment
├──── predictions.csv              # Model output for submission
└──── report.pdf             # Detailed documentation (per Section 6)
```

Deployment Guidelines

To showcase real-world applicability, participants are encouraged to deploy their Flask-based application (built from src/app.py) on one of the following platforms: AWS, Render, Vercel, or Hugging Face Spaces. Below are high-level guidelines for each platform. Include deployment details in your README.md.

- **AWS (Amazon Web Services)**:
  - **Option**: Use AWS Elastic Beanstalk or EC2 for deploying the app, with S3 for storing data or models.
  - **Considerations**: Ensure proper IAM roles and security groups. AWS offers scalability but requires more setup.
- **Render**:
  - **Option**: Deploy the app as a web service on Render's free tier (Python ≥3.9).
  - **Considerations**: Free tier may spin down after 15 minutes of inactivity, impacting availability.
- **Vercel**:
  - **Option**: Deploy using Vercel's serverless functions, optimized for frontend but adaptable for Python apps.
  - **Considerations**: Free tier has a 10-second timeout, so optimize for quick responses.
- **Hugging Face Spaces**:
  - **Option**: Deploy using a Docker-based setup, ideal for ML applications.
  - **Considerations**: Supports GPU for inference but lacks custom domain support.

Submission Guidelines:

- **GitHub**: Push code to a public repository with the above folder structure. Include a README.md with setup and deployment instructions.

- **Google Drive**: Share a folder with all files, ensuring accessible view permissions.
- Ensure file names are descriptive (e.g., waste_management_model.ipynb, predictions.csv, app.py).
- Submissions must be received by the Hackathon deadline (15th August 4:00 pm). **Final submissions to be made to *abhishek.singh17@pw.live.***

---

5. Evaluation Criteria: How Your Solution Will Be Judged

Submissions will be evaluated based on the following criteria, ensuring a balance between technical performance, code quality, and practical deployment. A working deployment link is encouraged for bonus consideration.

- **Accuracy (Primary Metric)**: For regression tasks (e.g., predicting Recycling Rate), performance is measured by Root Mean Squared Error (RMSE) on the test set. For classification tasks (e.g., predicting Disposal Method), accuracy or F1-score may be used, depending on class balance.
- **Code Quality**:
  - **Clarity**: Code should be well-commented, modular, and easy to understand.
  - **Reproducibility**: Code must run without errors using the provided README.md instructions.
  - **Efficiency**: Efficient use of computational resources, optimized for deployment.
- **Deployment** (optional):
  - **Working Link**: A URL to the deployed Flask application (hosted on AWS, Render, Vercel, or Hugging Face Spaces).
  - **Functionality**: The app must correctly serve predictions based on input features.
- **Documentation**: The quality and clarity of the participant's report (see Section 6).
- **Innovation**: Creative approaches to feature engineering, model selection, visualization, or deployment strategies.

---

6. Documentation from the Participant Side

Participants must submit a detailed report (PDF or Markdown) alongside their code, providing a clear narrative accessible to both technical and non-technical audiences. The report should include:

- **Introduction**: Overview of the problem and its significance in urban sustainability and waste management.
- **Methodology**:
    - Data preprocessing steps (e.g., handling missing values, encoding categorical variables).
    - Feature engineering techniques (e.g., geospatial feature extraction, temporal trend analysis).
    - Model selection and justification.
    - Hyperparameter tuning process (if applicable).
    - Deployment procedure (if applicable).
- **Results**:
    - Model performance (e.g., RMSE for regression, accuracy/F1-score for classification).
    - Visualizations (e.g., recycling rate trends, landfill location maps).
    - Insights derived from the data or model.
- **Discussion**:
    - Challenges faced and how they were addressed.
    - Limitations and potential improvements.
    - Real-world implications of the solution.
    - Future Scope
- **Conclusion**: Summary of key findings and takeaways.

Additional Notes

- **Resources**: Participants are encouraged to explore municipal reports, environmental studies, or Kaggle competition pages to practice on similar datasets.
- **Ethics**: Ensure all code and ideas are original or properly cited. Plagiarism will result in disqualification.

We look forward to seeing your innovative solutions to the Waste Management and Recycling in Indian Cities challenge, brought to you by PWSkills! For any questions, contact the Hackathon organizers via **support@pwskills.com**. Happy coding!