Capstone Project — Human Activity Recognition (HAR)

This repository contains my final capstone project notebook on **Physical Activity Recognition**. The project applies machine learning techniques to identify and classify human activities from sensor data.

The workflow was designed to follow a structured approach — from **data exploration and preprocessing** to **model building, evaluation, optimization, and interpretation**. The aim was not just to achieve accuracy, but also to ensure fairness, reproducibility, and meaningful insights into which features drive activity recognition.

**Repository Contents**

**Physical\_Activity\_Recognition\_Vinitha.ipynb**  
The main notebook includes:

**Step 1: Data Exploration & Preprocessing**

* Handling missing values, encoding activity labels, and scaling features. Stratified train-test splitting to ensure balanced activity distribution.

**Step 2: Baseline Model Development**

* Implemented RandomForestClassifier as the primary model. Established baseline accuracy and interpretability.

**Step 3: Hyperparameter Tuning**

* Used GridSearchCV for parameter optimization and Improved performance and reduced overfitting.

**Step 4: Model Evaluation & Insights**

* Confusion matrix, precision, recall, F1-score. Feature importance ranking for interpretability.

**Step 5: Finalization & Saving**

* Exported trained model and reports for reuse. Documented findings and reflection for future improvements.
* **README.md** (this file) has Project overview, methodology, and repository guide.

**How to Run**

**Open in Colab or locally.**

Colab recommended. If local, ensure Python 3.9+ with pandas, numpy, scikit-learn, matplotlib, and seaborn.

**Point to the dataset.**

* In the load cell, update the CSV path (e.g., file\_path = "/content/train.csv" or your local path).

**Run cells top → bottom.**

* Sections: Setup → EDA/cleaning → encoding/scaling → split → model train & GridSearch → evaluation → artifacts/report.

**(Optional) Tweak config.**

* Drop columns (e.g., subject), change test split size, or adjust the GridSearch parameter grid.
* A **quality gate** is applied with MIN\_F1—raise/lower it to match your bar.

**Find outputs.**

* Trained model, metrics, confusion matrix, and feature-importance plots are saved under artifacts/week8 and artifacts/week9 (includes an HTML summary report).

**Project Overview:**

**Goal:** Build a practical Human Activity Recognition (HAR) classifier from wearable/sensor data that is accurate, interpretable, and easy to re-run.

**Data & Prep:**

* Label-encoded Activity, dropped subject to avoid leakage, scaled features, and used stratified train/test splits. Quick EDA for shape, types, and missing values.

**Modelling:**

* Strong baseline with RandomForestClassifier. GridSearchCV tuned depth, estimators, and split/leaf parameters. A “light” RF variant and simple latency check for efficiency.

**Evaluation:**

* Accuracy + macro-F1, classification report, and a confusion matrix to surface class-wise errors. Feature importance plot to see which signals drive predictions.

**Artifacts & Reporting:**

* Saves best\_model.joblib, metrics\_\*.json, confusion-matrix and feature-importance PNGs, and a Week-9 HTML report. A quality gate (MIN\_F1) fails the run if performance dips below the threshold.

**What to try next:**

* Subject-wise splits for realism, compact models or gradient boosting for speed/accuracy trade-offs, and robustness tests (noise/missing values).

**Reflections**

Through this project, I learned:

* **Random Forests are reliable baselines** – they capture patterns well without heavy tuning, but can be resource-intensive.
* **Feature scaling and stratified splits are key** – these steps ensured fairness across activity classes and prevented bias.
* **GridSearchCV improves robustness** – tuning hyperparameters systematically gave a noticeable boost in balanced accuracy.
* **Feature importance offers clarity** – seeing which signals contribute most improved interpretability and trust in results.
* **Saving models and reports ensures reproducibility** – good practice for teamwork and future deployment.

**Next Steps**

* **Explore hybrid models** – try combining tree-based methods with temporal models (e.g., RandomForest + LSTM).
* **Subject-wise testing** – check generalization by splitting data across individuals, not just random splits.
* **Optimize for efficiency** – use pruning, feature selection, or lightweight models (e.g., XGBoost, LightGBM) for faster deployment.
* **Robustness checks** – test the model on noisy, imbalanced, or real-world activity data.
* **Future scope** – extend beyond the current dataset to broader physical activity contexts for stronger generalization.