**Status Report on Predicting Calories Burned Using Machine Learning**

* **Problem Statement**

The goal of this project is to develop a machine learning model to accurately predict calories burned based on various physiological and exercise-related factors. Current fitness tracking applications rely on general estimations that do not consider personalized data effectively. By leveraging machine learning, we aim to enhance the accuracy of calorie predictions.

* **Project Goal**

The primary objective is to build a predictive model that estimates calories burned based on input features such as age, gender, weight, height, heart rate metrics (Max\_BPM, Avg\_BPM, Resting\_BPM), session duration, workout type, body fat percentage, water intake, workout frequency, experience level, and BMI. The model will be evaluated for accuracy and optimized for performance to provide better estimations compared to standard fitness formulas.

* **Assumption and Methods**
  + Assumptions
    - The data is clean and representative of real-world user.
    - BMI are strong indicators of calorie burn.
    - Different workout types impact calories burned differently.
    - BMI and body fat percentage influence the rate at which calories are burned.
  + Methods
    - **1. Understanding the Data (EDA)**
      * We began with visual and statistical analysis to understand the relationships between features and the target variable (Calories\_Burned). Techniques included:
        + Correlation Matrix to inspect feature correlations (e.g., Avg\_BPM and Session\_Duration show strong correlation with calories burned).
    - **2. Cleaning and Preparing the Data**
      * To prepare the datasets for model training, we performed the following (we performed more techniques, these are just a few):
        + Removed Missing Values

For Dataset 2, we dropped any rows with missing data to make sure the models aren't affected by incomplete information.

* + - * + Created New Features

In Dataset 3, we calculated BMI using the formula:

* + - * + Encoded Categorical Data

For the Gender column, we converted text labels to numbers:

Male → 0

Female → 1

* + - **3. Training Models (Initial training, methods can change)**
      * XGBoost:
        + Trained using these parameters so far:

Learning rate: 0.05

This controls how fast the model learns. A smaller number like 0.05 means the model learns more slowly, which can help prevent overfitting and improve accuracy.

Max depth: 6

This decides how complex each decision tree can be. A depth of 6 allows the model to make detailed decisions, but not so complex that it overfits.

Number of estimators: 500

This means the model builds 500 trees. This should have good performance but will take some more time to train.

* + - * Feedforward Neural Network (FNN):
        + Input layer matching the number of features
        + Two hidden layers (ReLU activation) and dropout for regularization
        + Output layer with a single node (target)
        + Optimized using Adam and MSE loss function
      * Convolutional Neural Network (CNN):
        + Next model to create, no information for this model.
    - **4. Checking Model Performance**
      * MAE (Mean Absolute Error): Tells us the average error in predictions.
      * MSE (Mean Squared Error): Like MAE, but it gives more weight to bigger mistakes.
      * Score: Tells us how well the model explains the variation in calories burned.
* **Software, Tools and Datasets**
  + Programming languages and libraries:
    - Python, Pandas, NumPy, Matplotlib, Seaborn (EDA & Preprocessing)
    - Scikit Learn (AE, MSE, R2, Train Test Split)
    - TensorFlow/Pytorch (NN Models)
* Dataset: Fitness data including heart rates, workout type and calories burned.
* **Experimental Plan**
  + Step 1 - Data Analysis and Preprocessing:
    - Perform EDA to understand feature distribution and correlations.
    - Clean data by removing inconsistencies and handling missing values.
    - Encode categorical variables.
    - Normalize numerical features.
  + Step 2 - Model training and Benchmarking:
    - Train and compare XGBoost, FNN and CNN
    - Tune hyperparameters (learning rates, dropout layers, batch size)
    - Evaluate performance using MAE, MSE and R2 score.
  + Step 3 - Optimization and fine-tuning:
    - Identify and address model biases.
    - Use feature importance analysis to refine model input.
    - Implement ensemble method if needed.
  + Step 4 - Testing and Deployment:
    - Validate the model with test data.
* **Current Status and Partial Results**
  + EDA completed: Identified correlations.
  + Data preprocessing: Handling missing values and encoding categorical data (one-hot encoding applied).
  + Feature Engineering: Normalization of numeric features.
  + Model training:
    - Initial Benchmarking: Evaluated XGBoost and FNN models.
    - Findings: We have obtained the initial MAE, MSE, and R2 scores for both models. However, during training, we observed that predictions for our third dataset were highly inaccurate in both XGBoost and FNN, with an extraordinarily low accuracy score.
  + XGBoost Initial Results:

****

* + FNN Initial Results:

****

* **Plan for the remaining month**
* Finalize hyperparameter tuning models.
* Investigate the dataset for possible errors or inconsistencies.
* Train our third model CNN to compare its results with XGBoost and FNN.
* Conduct cross validation and compare models thoroughly.
* Finalize results, create a presentation/report, and prepare for the potential deployment.