Predicting Calories Burned Using Machine Learning

1.Project type:
This project falls under the implementation-flavor category. It aims to develop a deep learning model using PyTorch or TensorFlow to predict calories burned based on user characteristics and workout details. The implementation will be based on methodologies from existing research on caloric expenditure estimation, such as *"A Machine Learning Approach to Estimating Caloric Expenditure from Wearable Sensors."*

2.Problem statement:
Accurately estimating the number of calories burned during exercise is crucial for individuals tracking their fitness progress. Traditional methods, such as fitness trackers and metabolic equations, often provide generalized estimates that may not account for individual variations such as age, height, weight, and workout type. This project aims to leverage deep learning techniques to enhance calorie burn predictions based on available fitness data, optimizing the accuracy by experimenting with different neural network architectures and hyperparameters.

3.Project goal and motivation:
The goal of this project is to develop a deep learning model that estimates calorie expenditure more accurately than traditional methods by incorporating individual-specific attributes and workout details. By tuning hyperparameters such as network depth, learning rate, activation functions, and dropout rates, the model will be optimized for performance. The motivation behind this project stems from the increasing demand for personalized fitness insights and the limitations of existing calorie estimation techniques.

4.Methodolgy and plan:

- Data Collection & Preprocessing**:** Obtain datasets containing age, height, weight, session duration, total calories burned, and workout type. Clean the data by handling missing values, outliers, and standardizing features.
- Exploratory Data Analysis (EDA)**:** Identify patterns and correlations among the features.
- Model Selection & Training:
    - Implement a deep neural network (DNN) using PyTorch or TensorFlow**.**
    - Experiment with different architectures, including varying layer depth (e.g., 3-5 layers) and activations (ReLU, LeakyReLU, etc.)
    - Optimize learning rates, batch sizes, dropout rates, and regularization techniques.
- Model Evaluation: Compare model performance using Mean Absolute Error (MAE), Mean Squared Error (MSE), and R-squared metrics**.**
- Hyperparameter Tuning**:**
    - Adjust learning rates to observe convergence speed.
    - Test different optimizer choices (Adam, SGD, RMSprop).
    - Modify the number of neurons per layer to optimize performance.
- Deployment (Optional**)**: Integrate the model into a simple web or mobile interface for real-time predictions.

Milestones & Schedule:
Phase1:Data collection, cleaning, and preprocessing.
Phase2:Exploratory data analysis and feature engineer
Phase3: Deep learning model design, training, and evaluation.
Phase4: Model optimization and hyperparameter tuning.
Phase5:Integration and final testing.

5.Datasets:
Existing public datasets: Possible sources include Kaggle's fitness datasets, the Mhealth dataset, and publicly available exercise databases.
Custom Data Collection: Generating synthetic data or using wearable fitness tracker data.

6.Resources Needed:
Computational Resources**:** Python, TensorFlow/PyTorch, Scikit-learn, Jupyter Notebook.
Dataset Sources: Public repositories like Kaggle, UCI Machine Learning Repository.
Software & Tools: Google Colab or local machine with sufficient processing power (GPU optional for deep learning models).

7.Workload distribution:

TASK	WEI FAN WANG	DARSHAN NAIR	COLLABORATION
PHASE 1	Find and collect datasets from kaggle.	Clean the dataset, handle missing value.	Perform initial data exploration.
PHASE 2	Visualize data distribution,and correlations.	Analyze trends, feature importance.	Discuss key findings for model design.
PHASE 3	Implement initial DNN in Pytorch or TensorFLow.	Tune architecture (layers, neurons, activation functions.	Test and debug model performance.
PHASE 4	Test differnet optimizers(Adam, SGD, RMSprop)	Experiement with learning rates, dropout, batch sizes.	Compare model performance and optimize.
PHASE 5	Set up API for model deployment.	Create a simple UI.	Ensure model integraiton works.
FINAL REPORT & PRESENTATION	Write methodology, data preprocessing and EDA section.	Write model training, results, and conclusion sections.	Proofread, finalize, and prepare presentation.