**Task 1: Deep Learning Food Classification using Multi Models**

**Objective:**

Develop a deep learning-based food classification model using multiple architectures to classify 34 food categories. The project includes dataset balancing, model training, validation, deployment using Flask, and documentation.

**Project Details:**

**1. Dataset Information:**

* **Total Classes:** 34
* **Dataset Link:** [Food Image Classification Dataset](https://www.kaggle.com/datasets/harishkumardatalab/food-image-classification-dataset)
* **Data Preprocessing:**
  + Balance the dataset.
  + Select **200 images per class**.
  + Upload selected images to **Google Drive**.

**2. Model Development:**

Train three different models for classification and evaluate their performance.

**A. Custom Deep Learning Model**

* Train a custom model from scratch.
* **Training Parameters:**
  + **Epochs:** 30
  + **Save Trained Model**
  + **Generate Validation Report:**
    - Accuracy
    - True Positives (TP)
    - True Negatives (TN)
    - False Positives (FP)
    - False Negatives (FN)
    - Precision
    - Recall
    - F1-Score
  + **Save Report to:** Custom\_Model.txt

**B. VGG16 Model (Transfer Learning)**

* Train a **VGG16** pre-trained model.
* **Training Parameters:**
  + **Epochs:** 30
  + **Save Trained Model**
  + **Generate Validation Report:**
    - Accuracy
    - TP, TN, FP, FN
    - Precision, Recall, F1-Score
  + **Save Report to:** VGG16\_Model.txt

**C. ResNet Model (Transfer Learning)**

* Train a **ResNet** pre-trained model.
* **Training Parameters:**
  + **Epochs:** 30
  + **Save Trained Model**
  + **Generate Validation Report:**
    - Accuracy
    - TP, TN, FP, FN
    - Precision, Recall, F1-Score
  + **Save Report to:** ResNet\_Model.txt

**3. Coding Guidelines:**

* **Use Object-Oriented Programming (OOP):**
  + Implement Classes, Objects, and Constructors.
* **Implement Exception Handling** to ensure smooth execution.
* **Development Environment:** Use **PyCharm**.

**4. Data Collection & JSON Creation:**

* **Extract Nutritional Information**
  + Each class should have associated **protein, fiber, calories, carbohydrates, and fat** data.
  + Store this information in a **JSON file**.

**5. Deployment (Flask & HTML Frontend)**

Create a user-friendly web interface for prediction.

**Frontend Features:**

* **User Input:**
  + Image upload option.
  + Dropdown menu to select one of the **three trained models** (Custom, VGG16, ResNet).
  + Display **34 class names** on the right side.
* **Backend Processing (Flask API):**
  + Load the selected model.
  + Perform prediction and return results.
* **Output Display:**
  + Predicted class.
  + Model validation report (Accuracy, TP, TN, FP, FN, Precision, Recall, F1-Score).

**6. Project Submission Requirements:**

1. **Upload all code to GitHub**
   * Use Viharatech GitHub repository.
   * Include a well-structured README.md file.
2. **Send Submission Mail to:** hr.viharatech@gmail.com
3. **Create Documentation:**
   * A detailed **Word Document** describing the process followed.
   * A **PowerPoint Presentation (PPT)** summarizing the project.
4. **Deadline:** 12th February 2025
   * After the deadline, students will be required to **present their project**.

**Final Deliverables:**

✔ **Trained Models:** Custom, VGG16, ResNet.  
✔ **Validation Reports:** Custom\_Model.txt, VGG16\_Model.txt, ResNet\_Model.txt.  
✔ **JSON File:** Nutritional data for all 34 classes.  
✔ **Flask Web Application:** Prediction functionality.  
✔ **GitHub Repository:** Code with README.  
✔ **Documentation & PPT:** Steps followed in project implementation.  
✔ **Email Submission to HR.**

**Important Notes:**

📌 Ensure proper model training and validation.

📌 Code should follow best practices in **object-oriented programming**.

📌 Complete submission is mandatory by **12th February 2025**.

📌 Presentation after submission is required.

**Best of Luck!**  
**Viharatech EdTech**