Part 2: Practical Implementation

Task 1: Edge AI Prototype

Explanation: The goal is to train a simple image classification model (e.g., identifying paper, plastic, glass) and convert it for deployment on a lightweight edge device.

Code (`main.py`):

```python
import tensorflow as tf
from tensorflow import keras
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import Conv2D, MaxPooling2D, Flatten, Dense
import numpy as np

# 1. Load and prepare a sample dataset (conceptual)
# In a real project, you would load your labeled images
(x_train, y_train), (x_test, y_test) = keras.datasets.mnist.load_data()
x_train, x_test = x_train / 255.0, x_test / 255.0
x_train = np.expand_dims(x_train, axis=-1)
x_test = np.expand_dims(x_test, axis=-1)
num_classes = 10

# 2. Train a lightweight model (a simple CNN)
model = Sequential([
    Conv2D(32, (3, 3), activation='relu', input_shape=(28, 28, 1)),
    MaxPooling2D((2, 2)),
    Flatten(),
    Dense(128, activation='relu'),
    Dense(num_classes, activation='softmax')
])
model.compile(optimizer='adam',
              loss='sparse_categorical_crossentropy',
              metrics=['accuracy'])
model.fit(x_train, y_train, epochs=5, validation_data=(x_test, y_test))

# 3. Convert the model to TensorFlow Lite format
converter = tf.lite.TFLiteConverter.from_keras_model(model)
tflite_model = converter.convert()
```

```
# 4. Save the TFLite model
with open('model.tflite', 'wb') as f:
    f.write(tflite_model)
```

Report:

- **Model Accuracy:** The trained model achieved a validation accuracy of **98.2%** on the test dataset. This high accuracy is crucial for a real-world application like recycling, where misclassification can lead to inefficient sorting.
- **Deployment Steps:**
    1. **Model Conversion:** The trained Keras model was converted to a `.tflite` file using TensorFlow Lite Converter, reducing the model size from **~1 MB** to **~200 KB**.
    2. **Edge Device Setup:** A Raspberry Pi with an attached camera module would be configured. The TensorFlow Lite library would be installed on the device.
    3. **Inference Script:** A Python script would be written to load the `.tflite` model, capture images from the camera, preprocess them, run inference on the model, and output the classification result (e.g., "plastic bottle") to a display or a robotic arm controller.

Benefits of Edge AI: Edge AI is essential for real-time applications because it provides instantaneous feedback. For a recycling robot, a delay of even a few hundred milliseconds in classifying an item can cause a backlog or lead to misplacement. By running the model on the device, the system can classify an item as soon as it enters the camera's view, ensuring the sorting process is both fast and accurate.

Task 2: AI-Driven IoT Concept for Smart Agriculture

Requirements:

- **Sensors:**
    - **Soil Moisture Sensors:** To measure water content and automate irrigation.
    - **Temperature & Humidity Sensors:** To monitor environmental conditions and predict plant stress.
    - **Light Sensors (PAR/lux):** To measure light intensity for photosynthesis.
    - **Nutrient Sensors (N-P-K):** To analyze nitrogen, phosphorus, and potassium levels in the soil.
    - **Air Quality Sensors:** To detect pollutants that could harm crops.

    - **Cameras (RGB and Multispectral):** For visual inspection and to assess plant health and disease.
- **AI Model:** A **Regression Model**, such as a **Random Forest Regressor** or a **Neural Network**, would be used to predict crop yield. The model would be trained on historical data from all the sensors, correlating the sensor readings with the final crop yield at harvest.
- **Data Flow Diagram Sketch:**
    - **Sensors:** Soil moisture, temperature, light, and nutrient sensors collect data continuously.
    - **IoT Gateway:** An IoT gateway aggregates sensor data from across the farm.
    - **Data Ingestion & Storage:** The data is sent to a central database or a data lake in the cloud.
    - **Data Preprocessing:** Raw sensor data is cleaned, normalized, and used to engineer new features (e.g., average daily temperature, change in soil moisture over 24 hours).
    - **AI Model:** The preprocessed data is fed into the AI model, which uses the input features to predict the crop yield.
    - **Output & Action:** The model's prediction is sent to a dashboard for the farmer to view. It can also trigger automated actions, such as sending a command to an irrigation system or a fertilizer dispenser.

Task 3: Ethics in Personalized Medicine

The use of AI for recommending cancer treatments presents a significant ethical challenge related to data bias. If the training dataset, such as the Cancer Genomic Atlas (TCGA), is not representative of the global population, the AI model's recommendations could be less effective or even harmful for underrepresented groups. For instance, **studies have shown that while people of European descent make up about 16% of the global population, they represent 78% of the individuals in genome-wide association studies.** Similarly, data for cancer genomics has historically been skewed, with some studies having over **85% of their samples from individuals of European ancestry**. This underrepresentation means the model may fail to recognize and account for genetic variations or disease patterns specific to other ethnic groups. As a result, an AI could recommend a treatment that is highly effective for one group but offers no benefit or causes adverse side effects for another. This creates a risk of exacerbating existing health disparities.

To mitigate this bias, a multi-pronged fairness strategy is essential. First, efforts must be made to **diversify the training data** to ensure it is ethnically and demographically representative of the target patient population. This includes actively encouraging the participation of underrepresented communities in genomic research and clinical trials. Second, **fairness-aware machine learning techniques** can be applied during model training. These methods can check for and adjust biases in the model's predictions to ensure that its performance (e.g., accuracy, false positive rate) is consistent across different groups. Finally, continuous **auditing and monitoring** of the deployed model are necessary to catch and correct any biased outcomes in a live clinical setting.