Assignment 5: Classification Task on the SVHN Dataset

DSCI 6601: Data Science

Objective

The goal of this assignment is to train and evaluate various classifiers on the **Street View House Numbers (SVHN)** dataset. This dataset contains images of house numbers that will be used for classification into digits (0-9). You will implement and compare different classifiers to analyze their performance.

Data Download

Download the dataset from the following links:

- **Training Data**: `http://ufldl.stanford.edu/housenumbers/train_32x32.mat`
- **Testing Data**: `http://ufldl.stanford.edu/housenumbers/test_32x32.mat`

These files are in the MATLAB `.mat` format and can be loaded in Python using the `scipy.io.loadmat` function.

Data Format

The dataset consists of color images with dimensions $N \times H \times W \times C$, where:

- N: Number of images in the dataset.
- H: Height of each image (32 pixels).
- W: Width of each image (32 pixels).
- C: Number of color channels (3, for RGB images).

For example, the training data has the dimensions $73,257 \times 32 \times 32 \times 3$, representing 73,257 RGB images of size 32×32.

Preprocessing Requirement

Since the models we use do not natively handle multi-dimensional image data, you must **flatten** the images into one-dimensional vectors before training these models. This means reshaping each image from $32 \times 32 \times 3$ into a vector of size 3072 $(H \cdot W \cdot C)$.
For example:

- A single image of shape $32 \times 32 \times 3$ will be reshaped into a vector of length 3072.
- The entire dataset of N images will be transformed to have the shape $N \times 3072$.

This preprocessing step is necessary for compatibility with tree-based models.

Tasks

You are required to train a classifier using the following models:

1. **Decision Tree**
2. **SVM**
3. **Random Forest**
4. **Neural Network (NN)** with varying number of parameters:
   - About 5,000 parameters
   - About 10,000 parameters
   - About 20,000 parameters

   Experiment with the architecture (e.g., layers and neurons) to achieve the specified parameter counts. Use any other hyperparameters to ensure the model converges effectively (regularization, dropout, etc).

Evaluation and Analysis

For each classifier, model should be trained in the train data and validated on the test data. Perform the following:

- Identify the label in the **test data** that is predicted with the highest confidence. Plot the corresponding image and display its predicted class.
- Analyze misclassifications:
  - Identify any pair of classes that are frequently misclassified as each other.
  - Provide insights on why these misclassifications might occur.
- Compare the overall performance of each classifier in terms of metrics like **accuracy**, **precision**, **recall**, and **F1-score**. Plot the **confusion matrix** of the final model on the test data.

Submission Instructions

Your submission should include the following:

- Comprehensive responses to all the tasks outlined above, accompanied by the Python notebook code and outputs from your Jupyter Notebook.
- Clear and concise explanations of each step taken, with well-documented text and comments included in your Jupyter Notebook for every step.
- A thorough analysis of the results within the Jupyter Notebook.
- Plots of the image with the highest confidence prediction and the identified misclassified classes.