Skin Disease Detection/Skin Cancer Detection

Overview :

We would like to create a webapp where the user can input a picture, when this started, ethnicity etc and the app will be able to say to him what type of skin disease he has.

I. ***<u>Steps for the project :</u>***

- ☐ **Define the problem and objectives (Create README.md file) :**
  - Find out the goal
  - What's the output ? (a classification model)
  - What's the deliverable ? (a web app or demo)

- ☐ **Data collection :**
  - Kaggle dataset link : //

- ☐ **Set up the development environment :**
  - TensorFlow or PyTorch for building deep learning models
  - Keras for high-level neural networks API (if using TensorFlow)
  - OpenCV or PIL for image preprocessing
  - Flask or Streamlit for web app
  - Matplotlib and Seaborn for visualizing data and results
  - Google colab for coding ?

- ☐ **Data Preprocessing :**
  - *Data Augmentation :* Apply transformations like flipping, zooming, rotating and shifting to increase data diversity.
  - *Label Encoding :* Convert categorical labels (eg : disease names) into numerical format for the model.
  - *Split Data :* 70/20/10 split for training, validation and testing.

- ☐ **Model Development :**
  - *Start with Transfer Learning :* Use pre-trained models (ResNet, EfficientNet, Inception) and fine-tune them for your classification task.
  - *Add Custom Layers :* Add layers like GlobalAveragePooling, Dense, and Dropout on top of the pre-trained model to adapt it to your problem.

    - *Compile the Model :* Use an appropriate loss function (e.g., categorical crossentropy) and an optimizer (e.g., Adam).
    - *Train the Model :* Train the model on your augmented dataset, and monitor performance on the validation set using metrics like accuracy, sensitivity, specificity, and AUC-ROC.

☐ **Model Evaluation :**
- *Evaluate on Test Data :* Test the model's performance on the holdout test set and compute metrics such as precision, recall, F1-score, and ROC-AUC.
- *Confusion Matrix :* Plot a confusion matrix to visually analyze how well the model distinguishes between different classes.
- *Tuning :* If the model isn't performing well, you can tune hyperparameters or experiment with different architectures.

☐ **Building a Web Application :**
- *Select Framework:* Choose between Flask or Streamlit for your web app:
    - Flask: A lightweight framework to build custom web apps.
    - Streamlit: Easier for building quick demos with minimal code.
- *Web App Features:*
    - Upload functionality for users to submit images.
    - A function to load your pre-trained model.
    - Display prediction results and confidence scores.

☐ **Deploy the Web App :**
- *Host the Web App:*
    - Use platforms like Heroku, AWS Lightsail, or Google Cloud to deploy your app.
    - For Streamlit, you can directly deploy on Streamlit Cloud.
- *Set Up Model Hosting:* If the model is large, ensure it's hosted and accessible efficiently, possibly using cloud storage services.

☐ **Final Presentation and Demo**

II. ***Timeline Breakdown :***

- ☐ **Weeks 1-2**: Data exploration, preprocessing, and model selection.
- ☐ **Weeks 3-5**: Model training, evaluation, and tuning.
- ☐ **Weeks 6-7**: Build and test the web app.
- ☐ **Week 8**: Deploy the app and prepare the final demo.

III. ***Strategies to stand out :***

- *New Data :* Use a dataset that hasn't been explored much or merge multiple datasets. For instance, focus on less common diseases, specific populations, or certain skin types, which can improve the relevance and impact of your model.

- *Model Enhancement:* Improve the accuracy of existing models by experimenting with different architectures (like transfer learning with pre-trained models such as ResNet, EfficientNet, etc.), hyperparameters, or optimization techniques.

- *Feature Engineering:* Incorporate additional features such as metadata (e.g., patient history, demographics) alongside the image data to improve predictions.

- *Explainability:* Implement explainable AI techniques to make the model's predictions more transparent. This could include visualizing which parts of the image contributed most to the model's decision, using techniques like Grad-CAM.

- *Application:* Extend the project beyond the model to build a web-based diagnostic tool with an intuitive interface for real-world use. You could focus on accessibility for healthcare professionals or even integration with telemedicine platforms.

- *Focus on Ethical AI:* Address ethical concerns in medical AI, like bias detection or fairness across diverse skin tones, which is an emerging issue in dermatology datasets.