Problem Definition

What I did:
I defined the key problem to be solved: detecting malignant skin lesions from images and metadata in a real-world telehealth application. The problem involves distinguishing between benign and malignant lesions, with a strong focus on accurately identifying malignant cases to avoid missed diagnoses.

How I did it:
I conducted research on the prevalence and impact of skin cancer, focusing on its relevance in clinical and telehealth settings. I framed the classification task as a binary supervised learning problem, where the target variable is "1" for malignant and "0" for benign lesions. I also defined the need for an accessible tool for underserved populations, which led me to select images and user-provided metadata as the main predictors.

Why I did it:
Melanoma, the deadliest form of skin cancer, has a significant impact on public health, with over 200,000 projected cases in the US in 2024. Early detection of malignant lesions can improve patient outcomes. Since telehealth platforms are increasingly used in underserved communities, providing an accessible image-based diagnostic tool can significantly enhance early detection efforts.

Data Collection and Ingestion

What I did:
I collected a comprehensive dataset consisting of lesion images and metadata from the ISIC 2024 Skin Cancer Detection Challenge. This dataset includes labeled images (benign/malignant) along with patient information like age, sex, and anatomical location of the lesion.

How I did it:
I ingested the image data (stored in JPEG and HDF5 formats) and metadata from CSV files. I then merged them into a single dataset to facilitate training. The images were converted from bytes to RGB format, and metadata columns were checked for missing values and inconsistencies. I stored the processed data in a format compatible with PyTorch's DataLoader for efficient batch processing.

Why I did it:
The dataset provides a real-world simulation of what users would provide on the telehealth app — an image and personal information. By merging images with metadata, I enabled the

development of a multi-input neural network that could process both data types simultaneously. This approach ensures that all available information is utilized to improve classification accuracy.

Exploratory Data Analysis (EDA)

What I did:
I explored the data to identify key patterns, detect missing values, analyze distributions, and visualize relationships between predictors and the target variable.

How I did it:
I visualized the distribution of the target variable to identify class imbalance (99.902% benign, 0.098% malignant). I analyzed numerical columns like `age_approx` and `clin_size_long_diam_mm` for potential outliers and trends. For categorical features like `sex` and `anatom_site_general`, I created bar plots to see their distribution. I also generated a correlation heatmap to identify relationships between features.

Why I did it:
This step was essential to understand the structure of the data and guide decisions about preprocessing, resampling, and model architecture. The class imbalance analysis prompted me to plan for resampling strategies to prevent model bias toward benign cases. Identifying outliers and missing values also influenced decisions about imputation strategies and data cleaning.

Data Preprocessing

What I did:
I handled missing values, imputed categorical and numerical variables, encoded categorical features, scaled numerical features, and applied feature transformations to image and metadata inputs.

How I did it:

1. **Missing Values**: For numerical features, I imputed missing values using the median. For categorical variables, I used the most frequent value.
2. **Encoding**: I used one-hot encoding for categorical variables like `sex` and `anatom_site_general` to ensure they were represented numerically for model input.
3. **Scaling**: I initially used StandardScaler but later switched to MinMaxScaler, which is better suited for neural networks.

4. **Image Resizing**: Images were resized to (128, 128) to meet the input size requirements of my CNN models. I later realized that ResNet and EfficientNet require (224, 224) input sizes, so I adjusted the pipeline accordingly.

Why I did it:
Missing values, unencoded categorical features, and unscaled numerical features could hinder model convergence. Data preprocessing ensures that input features are in a consistent format, improving model training stability. Resizing images aligns them with the input size expected by the CNN models. Switching to MinMaxScaler improved neural network convergence, as smaller values work better with activation functions like ReLU and Sigmoid.

Feature Engineering

What I did:
I applied random image transformations to create variability and improve model generalization. Specifically, I used random rotation and random resized crop for image data.

How I did it:

1. **Random Rotation**: Images were randomly rotated by ±10 degrees.
2. **Random Resized Crop**: Images were randomly cropped with a scale of 0.8 to 1.0 and resized to (128, 128).

Why I did it:
These transformations make the model more robust to changes in image orientation and scale. Since users may capture images at different angles, rotations simulate real-world variability. Cropping focuses the model on random portions of the image, forcing it to learn diverse spatial representations and preventing overfitting.

Model Development

What I did:
I developed three multi-input neural network models using CNN, ResNet-18, and EfficientNet for image processing, along with linear layers for metadata.

How I did it:

1. **Model 1-3**: Used 3-layer CNN for image input with linear layers for metadata.
2. **Model 4-6**: Used ResNet-18 for image input, allowing for deeper feature extraction.

3. **Model 7-9**: Used EfficientNet, which uses compound scaling for width, depth, and resolution. Each model concatenated image and metadata features before the final prediction.

Why I did it:
I wanted to explore how different architectures (simple CNN, ResNet-18, and EfficientNet) affected performance. The increasing complexity of ResNet and EfficientNet was expected to extract more refined image features, leading to better classification. The multi-input design lets the model leverage both image and metadata to improve prediction accuracy.

Model Training

What I did:
I trained the models using binary cross-entropy loss, early stopping, and performance monitoring (accuracy, AUROC, partial AUC-aboveTPR).

How I did it:

1. **Training Loop**: Images and metadata were processed in batches, and forward passes were computed for predictions.
2. **Backward Propagation**: Loss was calculated using BCEWithLogitsLoss, and gradients were backpropagated using the Adam optimizer.
3. **Validation**: After each epoch, validation loss and metrics (accuracy, AUROC, partial AUC) were calculated to monitor generalization.
4. **Early Stopping**: Training stopped if validation loss did not improve for 5 consecutive epochs.

Why I did it:
I used early stopping to prevent overfitting, ensuring the model didn't train for unnecessary epochs. Monitoring partial AUC (above TPR) focuses on the most critical region of the ROC curve, which is essential for catching malignant lesions. Tracking training and validation loss helped identify when the model was overfitting.

Model Selection

What I did:
I selected Model 7 (EfficientNet) as the winning model for its superior performance on validation metrics (recall and partial AUC).

How I did it:
I compared all 9 models based on validation accuracy, partial AUC, and recall for Class 1 (malignant lesions). Model 7 had the lowest validation loss and the highest recall for malignant lesions.

Why I did it:
Selecting a model with the best recall for Class 1 aligns with the project goal of prioritizing malignant lesion detection. While Model 1 (simple CNN) performed well, EfficientNet's architecture enabled it to identify more subtle patterns, leading to better performance.

Model Evaluation

What I did:
I evaluated the final model on unseen test data using AUROC, accuracy, recall, and partial AUC-aboveTPR.

How I did it:
I loaded the saved Model 7, ran predictions on the test set, and computed key metrics. Classification reports were generated to show precision, recall, and F1-score for each class.

Why I did it:
Evaluating on test data provides an unbiased measure of generalization. Since Class 1 (malignant) was the minority class, I paid special attention to its recall. By monitoring partial AUC-aboveTPR, I ensured that the model could maintain high recall at more conservative classification thresholds.

Model Deployment

What I did:
I prepared the model for real-time inference by integrating it into a production-ready pipeline.

How I did it:
I saved the trained model.

Why I did it:
Real-time deployment ensures users receive immediate feedback on lesion malignancy, making it more suitable for telehealth apps where users expect instant results.