Problem Definition

Skin cancer is the most common form of cancer in the United States and ranks 17th globally (WCRF). There are three major types of skin cancer—Basal Cell Carcinoma (BCC), Squamous Cell Carcinoma (SCC), and Melanoma. While BCC and SCC are considered less lethal, melanoma is the deadliest form of skin cancer. It is expected to be diagnosed over 200,000 times in the US in 2024, with nearly 9,000 deaths projected. Automated image analysis tools play a significant role in expediting clinical presentation and diagnosis, positively impacting hundreds of thousands of people each year. For a telehealth app company, addressing the challenge of skin cancer detection in underserved populations or non-clinical settings is particularly significant. Current diagnostic methods rely on high-quality dermatoscope images, which are typically captured in dermatology clinics. These images reveal morphological features not visible to the naked eye. To provide this early detection service on our platform, we need to develop an algorithm capable of accurately classifying lower-quality malignant skin lesions from benign ones. Additionally, this algorithm should assist in diagnosing users based on their type of lesions and personal information.

Data Collection and Ingestion

Dataset

The dataset comprises diagnostically labeled images of skin lesions, provided in JPEG format, accompanied by a .csv file containing metadata. The metadata includes binary diagnostic labels indicating whether each lesion is malignant or benign, as well as additional input variables such as age, sex, and anatomical site. The dataset features standardized cropped lesion images obtained from 3D Total Body Photography (TBP). Each lesion image is a 15x15 mm crop from a high-resolution tomographic image capturing the entire visible skin surface. The images are sourced from thousands of patients seen between 2015 and 2024 across nine institutions and three continents by the International Skin Imaging Collaboration (ISIC) . The institutions involved include Hospital Clínic de Barcelona, Memorial Sloan Kettering Cancer Center, Hospital of Basel, FNQH Cairns, The University of Queensland, Melanoma Institute Australia, Monash University and Alfred Health, University of Athens Medical School, and Medical University of Vienna.The dataset features both "strongly-labeled" tiles, which have been validated through histopathology, and "weak-labeled" tiles, which were deemed benign by a doctor but were not subjected to biopsy. This dataset was found from ISIC 2024 - Skin Cancer Detection with 3D-TBP 2 competition on Kaggle.

The dataset addresses the problem statement by providing a comprehensive collection of images and associated metadata for training and evaluating skin cancer detection algorithms. By differentiating between benign and malignant cases based on diagnostic labels, the dataset enables the development of machine learning models that can predict the likelihood of malignancy in skin lesions. The standardized nature of the images and the inclusion of both strongly and weakly labeled cases allow for robust model training and validation. The use of 3D TBP images mimics real-world scenarios where high-quality dermoscopic images might not always be available, thereby improving the app's capability to handle and accurately assess non-dermoscopic images. This helps in early detection and triaging of skin cancer, potentially improving patient outcomes and enhancing diagnostic efficiency in diverse clinical settings.

Target Variable

The target variable in this dataset is “target,” a binary label indicating whether a lesion is benign (0) or malignant (1). This variable is central to the classification task, as the goal is to predict whether a given lesion is malignant based on the provided image and metadata. Early detection of malignancy is critical for timely intervention in skin cancer, which can significantly improve patient outcomes. As a result, this variable is the primary focus of the predictive model, driving the binary classification task.

Predictors

The predictors in this dataset are divided into two categories: image data and metadata. In the application I am building, users will input an image and some personal information. Any metadata not available to the users will not be used as predictors.

a) **Image Data**

Image: Cropped lesion images (HDF5 format).

Why: Images are essential because skin cancer detection relies on visual patterns such as shape, size, color irregularities, and texture. A deep learning model can process these patterns to identify malignant lesions.

b) **Metadata**

Patient-level data:

- age_approx: Approximate age of the patient at the time of imaging.
- sex: Sex of the patient.

Reason: Age and sex are important factors in predicting skin cancer risk. Incidence rates of invasive melanoma vary across age and gender groups. In men younger than 50, melanoma rates have declined by 1% per year and stabilized in older men. However, women under 50 experience higher rates of melanoma than their male counterparts, possibly due to trends like indoor tanning use. In contrast, men over 50 have a higher rate of melanoma than women of the same age group, potentially due to occupational and recreational UV exposure. This makes age and sex critical features for identifying skin cancer risk.

Lesion Characteristics:

- anatom_site_general: The general location of the lesion on the body.
- clin_size_long_diam_mm: The maximum diameter of the lesion in millimeters

Reason: The anatomical site and size of lesions can be readily assessed by users and are important factors for risk assessment. Lesions in sun-exposed areas, for example, might have a higher likelihood of being malignant.

Unavailable Predictors (Not used in the model):

Features that require medical devices or professional analysis, such as tbp_lv_deltaL, tbp_lv_area_perim_ratio, mel_thick_mm, etc., will not be included as predictors, as these details are unavailable to users.

Exploratory Data Analysis

Variables and Data Types

The dataset includes both structured and unstructured data (images and metadata). Below is an overview of the variables:

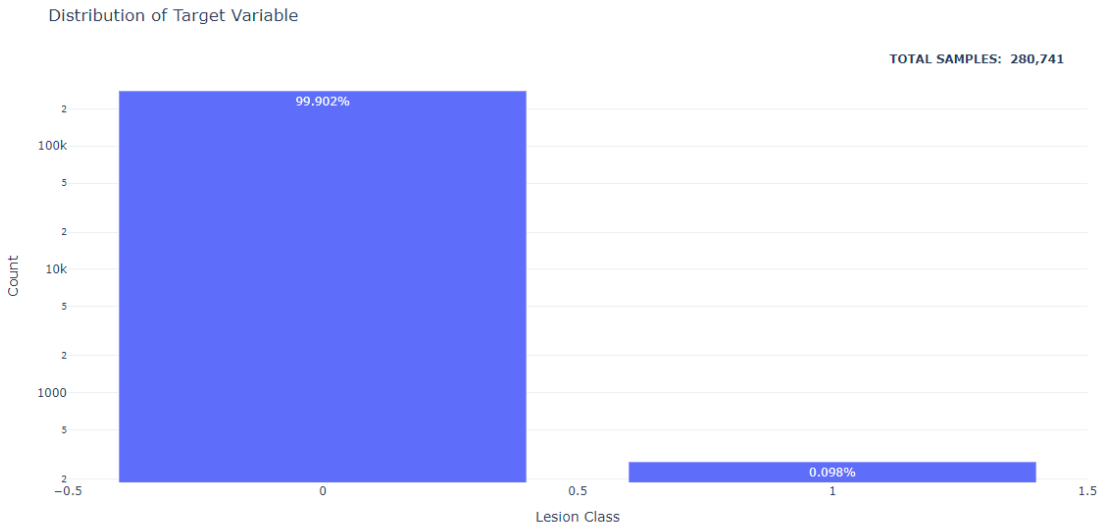
Target Variable:

Columns	Column Description	Data Types
target	Binary, {0: benign, 1: malignant}.	Integer

Predictor Variables:

Columns	Column Description	Data Types
age_approx	Approximate age of the patient at the time of imaging	Integer
sex	Sex of the patient (e.g., Male, Female)	String
anatom_site_general	Anatomical location of the lesion (e.g., upper arm, lower leg)	String
clin_size_long_diam_mm	Maximum diameter of the lesion in millimeters	Float
Image Data	Cropped lesion images (HDF5 format)	Integer

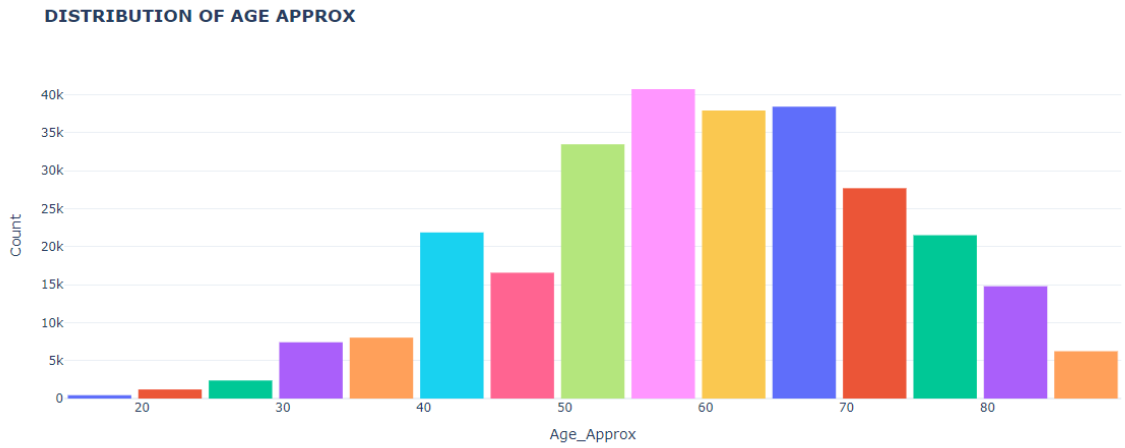
Target Variable Distribution

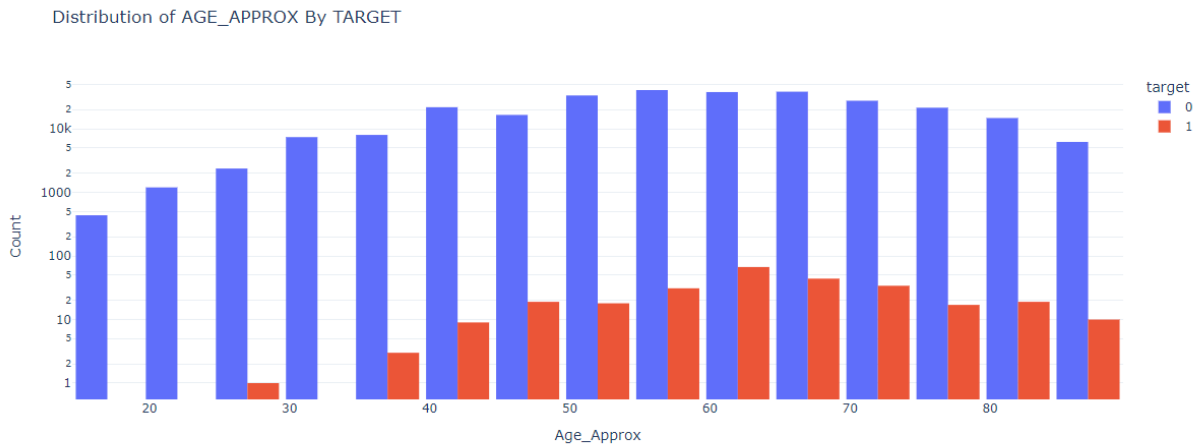


Insight: The target variable (0 for benign and 1 for malignant) is significantly imbalanced, with benign lesions comprising 99.902% of the dataset, while malignant lesions account for only 0.098%. This substantial class imbalance presents a challenge for the classification problem, as models may become biased toward predicting the majority class (benign lesions) and may struggle to accurately identify the minority class (malignant lesions). Addressing this imbalance will be crucial for improving the model's performance and ensuring reliable predictions.

Numerical Feature Analysis

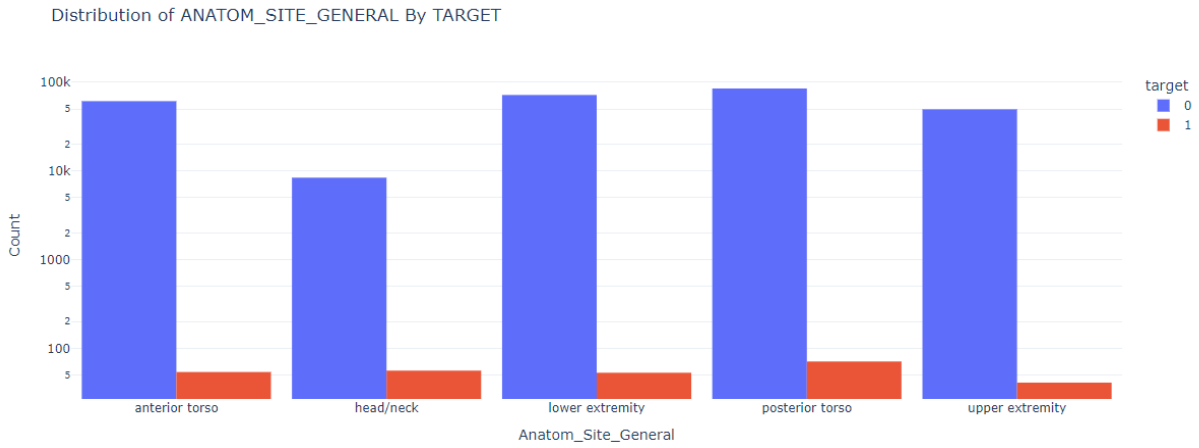
Age





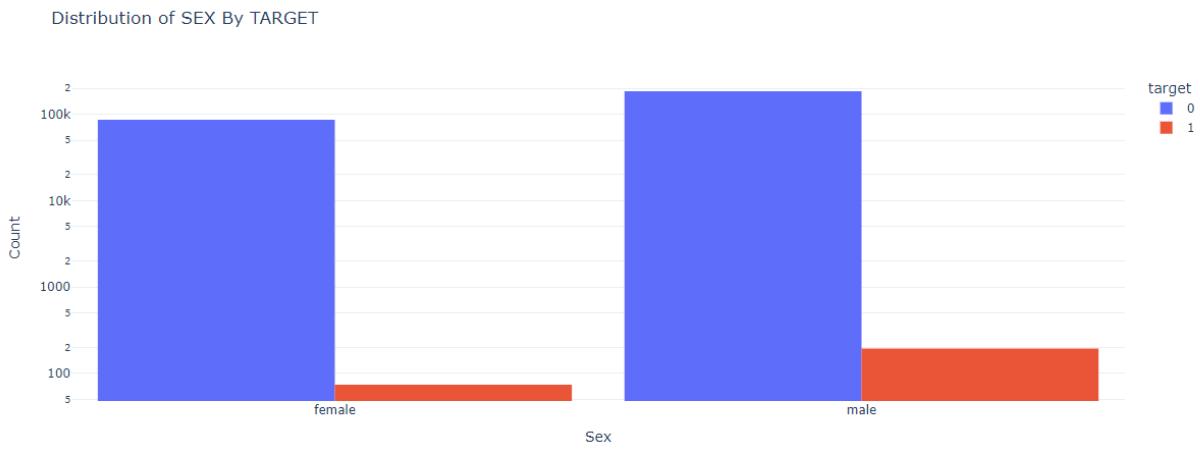
Insight: The majority of patients in the dataset are between the ages of 50 and 70, with patients aged 65 having the highest rate of malignant lesions. There are no malignant lesion cases for patients under the age of 25 and at the age 30.

Anatomical location of the lesion



Insight: The majority of malignant lesions are located on the posterior torso which is the back, indicating that this region may be particularly prone to skin cancers in the dataset.

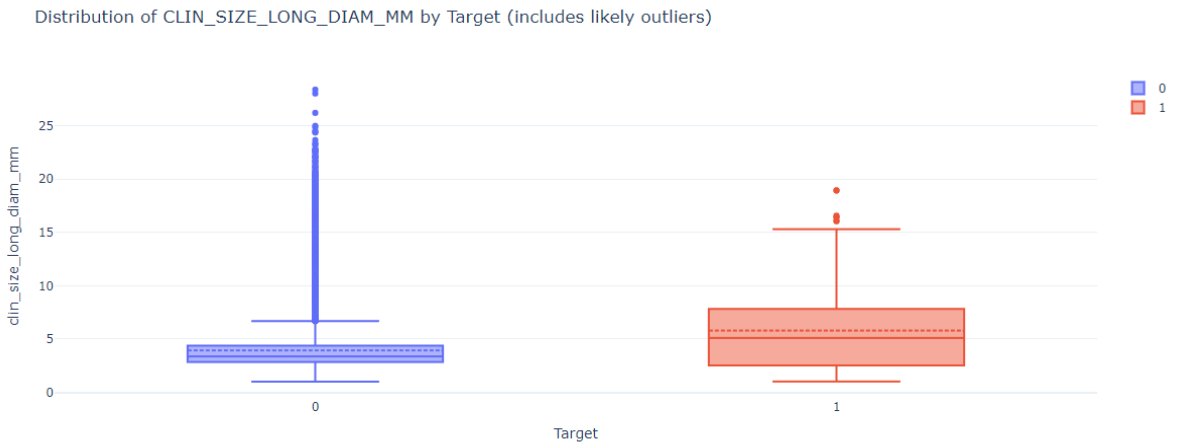
Sex

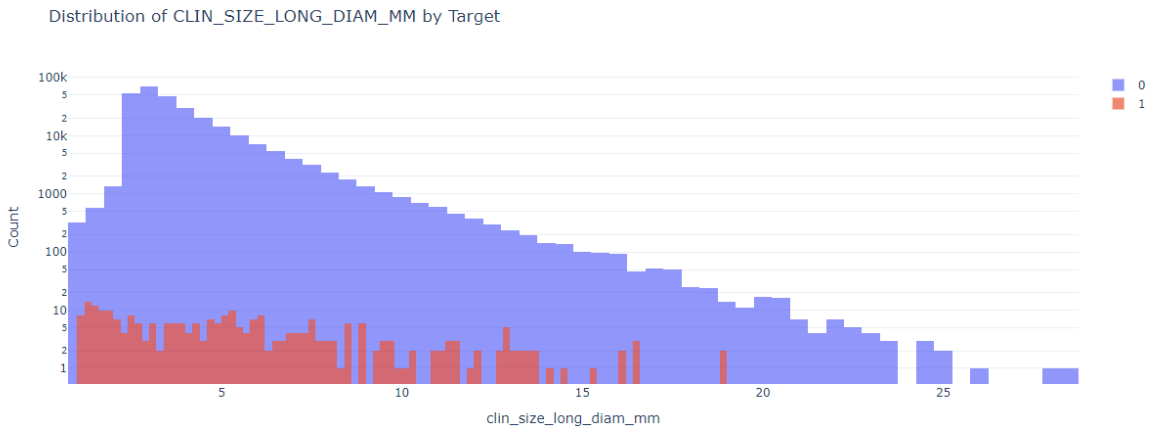


Insight: In this dataset, the majority of patients with malignant lesions are male, indicating a potential gender-based disparity in the occurrence of malignant skin lesions.

Categorical Feature Analysis

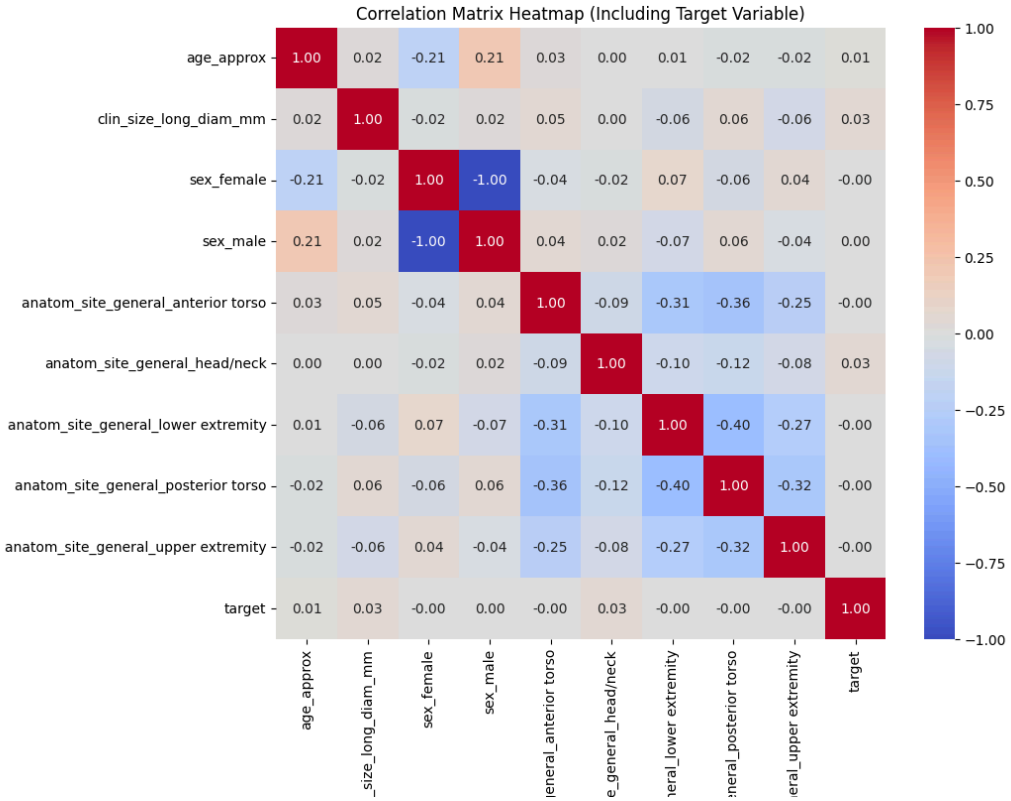
Maximum diameter of the lesion in millimeters





Insight: There are a large number of outliers in this variable, and the second graph of the distribution shows a right-skewed pattern. This indicates that most values are concentrated between 1 mm and 14 mm, with a few extreme values over 25 mm.

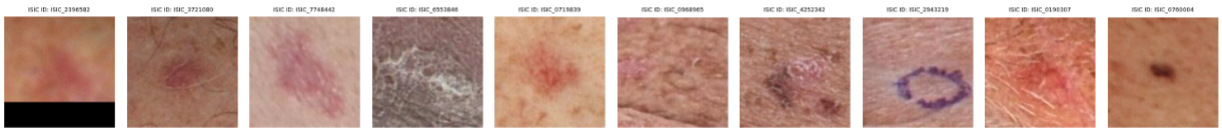
Correlation Analysis



Insight: There’s no strong correlation between the variables in the dataset. We can disregard the male and female variables, as they are encoded and do not provide additional insight into the

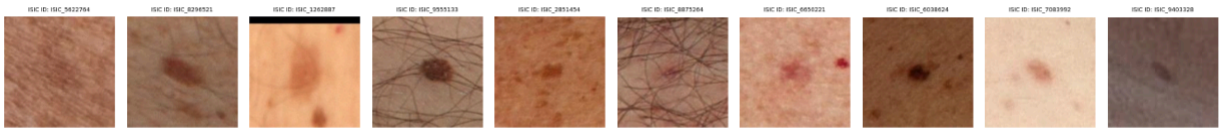
relationships among the other features.

Images of Lesions with Target Value 1



```
plot_images_by_target(processed_df, target_value=0, max_images=10)
```

Images of Lesions with Target Value 0



Data Preprocessing

Upsample and Downsample

The training data is imbalanced, so I employed a common approach of downsampling the majority class and oversampling the minority class. The target distribution for the classes is set at 70% for the majority class and 30% for the minority class. Consequently, using the proportional sample size formula, the fraction for downsampling the majority class is approximately 0.01, while the factor for upsampling the minority class is 5.0.

Set Weights for Training Purpose

After resampling the training data, I set the class weights by computing the class weight using the compute_class_weight function from sklearn. This ensures that the model gives appropriate importance to the minority class during training.

Impute Missing Values for Both Categorical and Numerical Data

To handle missing values, I imputed categorical variables with the most frequent value and numerical variables with the median. Using the median as an imputation method helps avoid the influence of outliers in the data.

One-Hot Encode for Categorical Data

I used one-hot encoding for categorical data such as 'anatom_site_general' because this variable is nominal and does not have an inherent order. One-hot encoding allows each category to be represented as a binary vector, preventing the introduction of any unintended ordinal relationships that could arise with label encoding.

Scale Numerical Variables

I scaled numerical variables to ensure they are on a similar scale, which is crucial for neural networks. Scaling helps improve convergence during training by allowing the optimization algorithm to navigate the loss surface more efficiently. Neural networks are sensitive to the scale of input features; features with larger ranges can disproportionately influence the gradients during backpropagation. By standardizing or normalizing the numerical variables, I ensure that all features contribute equally to the training process, leading to more stable and faster convergence. After week 4, I realized that applying StandardScaler to my dataset may not be the optimal choice for a neural network model. Instead, using MinMaxScaler is more

appropriate, as it scales the data to a range of 0 to 1, which aligns better with the activation functions commonly used in neural networks. This adjustment ensures that the input features are normalized in a way that enhances the model's learning efficiency and stability. Moving forward, this is one of the changes I will implement to improve the overall performance of my model.

Convert Age from Float to Integer

Convert Age from Float to Integer The data type of the age variable is float; therefore, I converted it to integer.

Feature Engineering

Metadata

I did not perform any feature engineering for the metadata as i see that not necessary

Resize Image and Normalized Image Pixels

I resized the image to **128 x 128 pixels** and normalized the pixel values using a general ratio of **255**. This preprocessing step ensures that the image dimensions are consistent for input into the neural network, facilitating effective training and inference. The resizing is done to meet the input size requirements for models like **ResNet (224x224)** and **EfficientNet (224x224)**, which typically expect larger input dimensions. Although I initially resized the images to 128x128, further adjustments will be made to ensure compatibility with these architectures. For your information, I messed up with the size input for ResNet where I resize the images to 225 x 225.

Image Rotation

Each image in the dataset is randomly rotated by up to 10 degrees. This transformation introduces variability into the dataset, allowing the model to become invariant to slight rotations. It helps prevent overfitting by providing a more robust training set that simulates different orientations of the same object.

Random Resized Crop

A random resized crop of size (128, 128) is applied to each image. The cropping scale is set between 0.8 and 1.0 of the original image dimensions.This transformation randomly selects a portion of the image, resizes it to (128, 128), and retains important features while discarding irrelevant background. It aids in focusing the model on different parts of the images and improves generalization by providing multiple views of the same object at varying scales.

Model Development

Modeling

This is a supervised binary classification problem where I will be classifying cases as benign or malignant, with the target variable being 0 for benign and 1 for malignant. My approach is to use a Convolutional Neural Network (CNN) as the base model for image data, then incorporating both metadata and image data to classify the cases. I will develop three multi-input neural network models with slight variations in the image processing component. Each of these models will accept two inputs — image data and metadata — which will be processed independently before being combined for final prediction.

Model 1 - 3

For Model 1 to Model 3, I utilized Convolutional Neural Networks (CNNs) for image processing. Each model follows a consistent CNN architecture that includes three key convolutional layers (conv1, conv2, conv3), each followed by batch normalization (bn1, bn2, bn3) and max pooling.

The rationale behind this architecture is to leverage the power of convolutional layers in capturing spatial hierarchies and local patterns within images, such as edges, textures, and shapes. These patterns are essential for accurately distinguishing between malignant and benign skin lesions. Batch normalization ensures more stable and faster training by normalizing activations, while max pooling reduces the spatial dimensions, enabling the network to focus on the most important features.

Model 4 - 6

For Model 4 to Model 6, I employed the ResNet-18 architecture, which is a more sophisticated and deeper model compared to the 3-layer CNN used in previous models. Unlike the traditional 3-layer CNN that consists of only three convolutional layers followed by batch normalization and max pooling, ResNet-18 features 18 layers structured into residual blocks.

A key innovation in ResNet-18 is the use of shortcut connections, which allow information to bypass certain layers. This design improves the flow of gradients during backpropagation, effectively addressing the vanishing gradient problem that often arises in deeper networks. As a result, ResNet-18 can be trained more efficiently and can learn more complex, hierarchical

feature representations. This is crucial for tasks like skin lesion classification, where fine details in texture, color, and shape are essential for distinguishing between benign and malignant lesions.

Why ResNet-18? While a 3-layer CNN can effectively capture basic patterns such as edges and textures, it may struggle with deeper feature representations required for skin lesion classification. In contrast, ResNet-18 is designed to learn more complex features, thanks to its deeper architecture and shortcut connections. This enables the model to identify subtle differences in skin lesion morphology, which are critical for accurate diagnosis. Moreover, the deeper architecture of ResNet-18 doesn't result in excessive computational cost, making it a more suitable option for image classification tasks with higher complexity, such as malignant **lesion detection**.

Model 7 - 9

For Model 7 to Model 9, I utilized the EfficientNet architecture, which represents a significant advancement over the ResNet-18 model used in Models 4 to 6. Unlike ResNet-18, which relies on residual blocks to improve gradient flow, EfficientNet employs a compound scaling method. This method balances the network's depth, width, and resolution to achieve optimal performance while maintaining computational efficiency. This approach allows EfficientNet to effectively learn more detailed and hierarchical features from images, making it a more suitable option for skin lesion classification, where subtle differences in lesion morphology are crucial for accurate predictions.

EfficientNet’s core innovation lies in its compound scaling strategy, which simultaneously increases model depth (number of layers), width (number of channels), and image resolution. This balanced scaling approach allows the model to learn richer and more intricate feature representations from images without introducing a significant computational burden. Unlike ResNet-18, which relies on residual connections to solve the vanishing gradient problem, EfficientNet scales the network structure more effectively, enabling the model to extract even more refined details from the image.

For skin lesion classification, fine-grained features like subtle color variations, irregular border shapes, and textural differences are essential for distinguishing benign from malignant lesions. EfficientNet's ability to capture such detailed features makes it well-suited for this task. Moreover, while ResNet-18 improves feature extraction compared to a 3-layer CNN, EfficientNet takes it a step further by capturing both low-level and high-level features more effectively, improving the model's ability to generalize on unseen lesion images.

Model Training

The training and validation process for my **skin lesion classification model** follows a structured approach to ensure optimal learning, effective monitoring, and early stopping for better generalization. This process is broken down into several key phases, each of which plays a crucial role in model performance. Below, I detail the major steps of the process, including how training, validation, and evaluation are conducted.

1. Model Initialization

At the start of the process, I initialize key components for training:

- **Model**: The model is instantiated, which could be one of the three architectures (3-layer CNN, ResNet-18, or EfficientNet-B0) I experimented with.
- **Dataloaders**: Data is loaded into batches for training and validation using PyTorch `DataLoader`. These batches are used to iterate through images and metadata for training.
- **Loss Function**: Binary Cross-Entropy Loss (`BCEWithLogitsLoss`) is used to optimize the binary classification task (benign vs. malignant skin lesions).
- **Optimizer**: The Adam optimizer is used for weight updates.
- **Device**: The model is sent to a GPU or CPU, depending on hardware availability.

2. Training Phase

Each epoch consists of the following steps:

1. **Model in Training Mode**: The model is set to `.train()` mode to enable weight updates and gradient tracking.
2. **Batch Iteration**: The model iterates over batches of images, metadata, and labels.
3. **Data Transfer**: Images, metadata, and labels are sent to the device (CPU/GPU).
4. **Forward Pass**: The model predicts the probability for each sample in the batch.
5. **Loss Calculation**: The loss between predictions and ground truth labels is calculated using the BCE loss.
6. **Backward Pass**: The gradients are calculated using `loss.backward()`.
7. **Weights Update**: The optimizer updates the model weights using `optimizer.step()`.
8. **Performance Tracking**: Training loss, accuracy, predictions, and labels are stored for monitoring.

Error Handling: If any exceptions arise (e.g., shape mismatch between predictions and labels), an error message is printed, and training is halted.

3. Validation Phase

Once the training phase for an epoch is complete, the model enters the validation phase.

1. **Model in Evaluation Mode**: The model is set to `.eval()` mode, and gradient tracking is disabled using `torch.no_grad()`. This prevents weight updates during validation.
2. **Batch Iteration**: Validation batches are processed in the same way as training batches, but no gradients are computed.
3. **Data Transfer**: Images, metadata, and labels are sent to the device.
4. **Prediction**: The model generates predictions for each batch.
5. **Loss Calculation**: The loss between predictions and ground truth labels is calculated.
6. **Accuracy Calculation**: The binary classification accuracy is computed as the ratio of correct predictions to the total number of samples.
7. **Performance Metrics Calculation**:
    - **AUROC**: The Area Under the Receiver Operating Characteristic curve is calculated to measure how well the model separates benign from malignant cases.
    - **Partial AUC**: Partial AUC is calculated to focus on the True Positive Rate (TPR) for high-recall ranges, which is crucial for detecting malignant lesions.
    - **Classification Report**: If validation is the final step, a classification report (precision, recall, F1-score) is generated for each class.
8. **Performance Tracking**: Validation loss, accuracy, AUROC, partial AUC, and other metrics are stored for analysis.

Error Handling: If errors arise during validation, such as AUROC calculation issues, warnings are printed to highlight the problem but the process continues.

4. Early Stopping and Best Model Checkpoint

To avoid overfitting, early stopping is implemented as follows:

1. **Monitor Validation Loss**: If the validation loss does not improve for `early_stopping_patience` epochs, training is stopped early.
2. **Save Best Model**: The model with the best validation loss is saved as the "best model." This ensures the best-performing version of the model is used for testing and deployment.
3. **Stopping Criteria**: If early stopping is triggered, the model stops training and moves to final evaluation.

5. Performance Visualization

To track model performance, plots are created for:

- **Training and Validation Loss**: A plot of loss over each epoch for training and validation is created. This allows me to visualize overfitting or underfitting.
- **Training and Validation Accuracy**: A plot of accuracy over each epoch is created. If the training accuracy is significantly higher than the validation accuracy, it may indicate overfitting.

6. Key Metrics Monitored

The following metrics are monitored during training and validation to ensure model quality:

1. **Loss**: Measures how well the model's predictions match the labels.
2. **Accuracy**: Measures the proportion of correctly predicted samples.
3. **AUROC**: Measures the model's ability to distinguish between classes.
4. **Partial AUC**: Provides insight into the True Positive Rate for cases with a higher threshold.
5. **Early Stopping**: Stops training early if no improvement in validation loss is observed.

7. Key Hyperparameters

1. **Epochs**: Number of iterations over the entire training set.
2. **Batch Size**: Number of samples processed at once in each batch.
3. **Learning Rate**: The step size for updating model weights.
4. **Early Stopping Patience**: The number of epochs to wait before stopping training if validation loss does not improve.
5. **Optimizer**: Adam optimizer is used to minimize the loss.

Model Evaluation

1. Validation Loss

Validation loss represents how well the model performs on unseen data, i.e., the model's generalization ability. It is the loss function (Binary Cross-Entropy in this case) calculated on the validation set after each epoch of training. Minimizing validation loss is crucial because the model should generalize well to new, unseen data—especially important for medical applications like skin lesion classification, where the consequences of poor generalization could lead to misdiagnosis. Validation loss helps detect overfitting. If validation loss increases while training loss decreases, the model is likely overfitting to the training data.A lower validation loss indicates that the model is learning well and is able to generalize to new images, ensuring that it can accurately classify skin lesions in real-world scenarios.

2. pAUC-aboveTPR

In the **Model Evaluation phase**, one of the key metrics used to assess the model's performance is the **partial AUC-above-TPR**. This function is designed to compute the area under the ROC curve (AUC) but focuses on a specific segment of the **True Positive Rate (TPR)**. This approach ensures that the evaluation prioritizes regions of the curve where high recall is crucial — an important consideration for **skin lesion detection**, as missing a malignant lesion can have serious health consequences.

The goal of using **partial AUC-above-TPR** is to focus on high-recall regions where the true positive rate (TPR) is above a specified threshold (for example, **TPR ≥ 80%**). Unlike the standard **AUC**, which considers the entire ROC curve, the partial AUC only focuses on a portion of the curve, emphasizing sensitivity (recall) in regions where the stakes are higher.

Rescaling for ROC Calculation:

- The ground truth is **flipped** by computing `v_gt = abs(solution - 1)` to make 1 correspond to the "negative" class and 0 correspond to the "positive" class. This step ensures compatibility with scikit-learn's `roc_curve` function, which expects a specific ordering.
- The predicted probabilities are **flipped** to make higher values represent "positives" for the ROC curve computation. This ensures compatibility with how ROC curves are computed in scikit-learn.

Compute Full ROC Curve:

- The **False Positive Rate (FPR)** and **True Positive Rate (TPR)** are computed using `roc_curve()`. This step calculates the TPR and FPR for various classification thresholds.

Calculate Partial AUC:

- **Maximum FPR** is calculated as `max_fpr = abs(1 - min_tpr)`. For example, if `min_tpr=0.80`, then `max_fpr=0.2`, meaning the partial AUC will only consider the region of the curve where TPR is at least 0.80.
- The point in the curve where **FPR exceeds max_fpr** is located using `np.searchsorted()`. This allows for interpolation of the curve to get a smooth transition.
- The portion of the ROC curve for **FPR < max_fpr** is isolated. The TPR and FPR are updated to only contain points within the desired region. Interpolation ensures a smooth curve for AUC calculation.
- **Partial AUC** is calculated using `auc(fpr, tpr)`, where the FPR and TPR arrays now only span the segment of the ROC curve above the desired **TPR threshold**.

3. Accuracy And Recall

Accuracy represents the percentage of correct predictions out of the total predictions made by the model. It's calculated as the ratio of correctly predicted instances (both benign and malignant) to the total number of instances. While accuracy provides a straightforward measure of how often the model is getting predictions right, it can be misleading in cases of class imbalance (e.g., if benign cases far outnumber malignant ones). As a result, while accuracy is tracked, it is evaluated alongside other metrics.

Given the importance of detecting malignant cases, recall is a primary focus for this classification task. Recall measures the model's ability to correctly identify malignant instances, calculated as the ratio of true positives to the sum of true positives and false negatives. High recall is essential in this context, as it reflects the model's ability to catch as many malignant cases as possible, reducing the chance of missed detections.

By monitoring recall along with accuracy and ROC AUC, I ensure that the model is effectively capturing the malignant cases rather than merely predicting the majority class. This focus on recall aligns the model's performance more closely with the goals of the classification task, where identifying malignant lesions is a top priority.

Hyperparameter Tuning

1. Optimizer (Adam vs. SGD)

The optimizer plays a critical role in determining how the model updates its weights. I evaluated both Adam and SGD to compare their efficiency in converging to an optimal solution.

- Adam: Known for its adaptive learning rate, Adam often converges faster and requires less manual tuning. It is especially helpful when working with image data where feature gradients can vary.
- SGD: While often slower to converge, Stochastic Gradient Descent (SGD) with momentum has been shown to provide better generalization and stability, especially for image classification tasks.

I compared both optimizers to see which led to faster convergence and better generalization, particularly focusing on validation accuracy and loss. This comparison helped determine which optimizer was more suited to the task of skin lesion classification.

2. Learning Rate (0.001 vs. 0.0001)

The learning rate controls how quickly the model updates its weights during training. If set too high, the model risks overshooting the optimal values, while a low learning rate can result in slower convergence. To find the right balance, I evaluated learning rates of 0.001 and 0.0001:

- 0.001: This is a commonly used starting point and typically provides a good balance between learning speed and stability. I tested this to see if it would allow the model to learn at a reasonable pace without being too aggressive.
- 0.0001: A lower learning rate was explored to improve stability, especially when overfitting or erratic training behavior became evident with a higher rate. However, I found that this slower rate required significantly more epochs, making it less practical in my environment.

Ultimately, I evaluated both rates to balance speed and stability, closely monitoring validation loss and accuracy over epochs. While 0.0001 improved stability, I found it less favorable due to the environmental constraints, which demanded longer training times.

3. Batch Size (16 vs. 32 or 64)

The batch size plays a significant role in both the stability of gradient estimates and the memory

requirements during training. A larger batch size (e.g., 64) can speed up training but may lead to overfitting due to smoother gradient updates. In contrast, smaller batch sizes introduce more variability in gradient estimates, which can help the model generalize better by adding noise that allows it to escape local minima.

Due to the constraints of the current environment, I used a batch size of 16 instead of 32 or 64. While this smaller batch size is slower, requiring more iterations to reach comparable performance levels, it introduces beneficial variability into the training process. This variability helps avoid overfitting and improves generalization, which is particularly useful in complex tasks like skin lesion classification.

By comparing the current model with previous versions, I found that using a batch size of 16 resulted in a smoother learning curve with more stable training and validation loss, even though it requires more time per epoch. While larger batch sizes can speed up training, batch size 16 offers more stable progress tracking, making it well-suited for environments with limited computational resources and for achieving robust generalization in challenging classification tasks.

Model Selection

In this phase I will compare each model's performance metrics and select the winning model to perform on the test data.

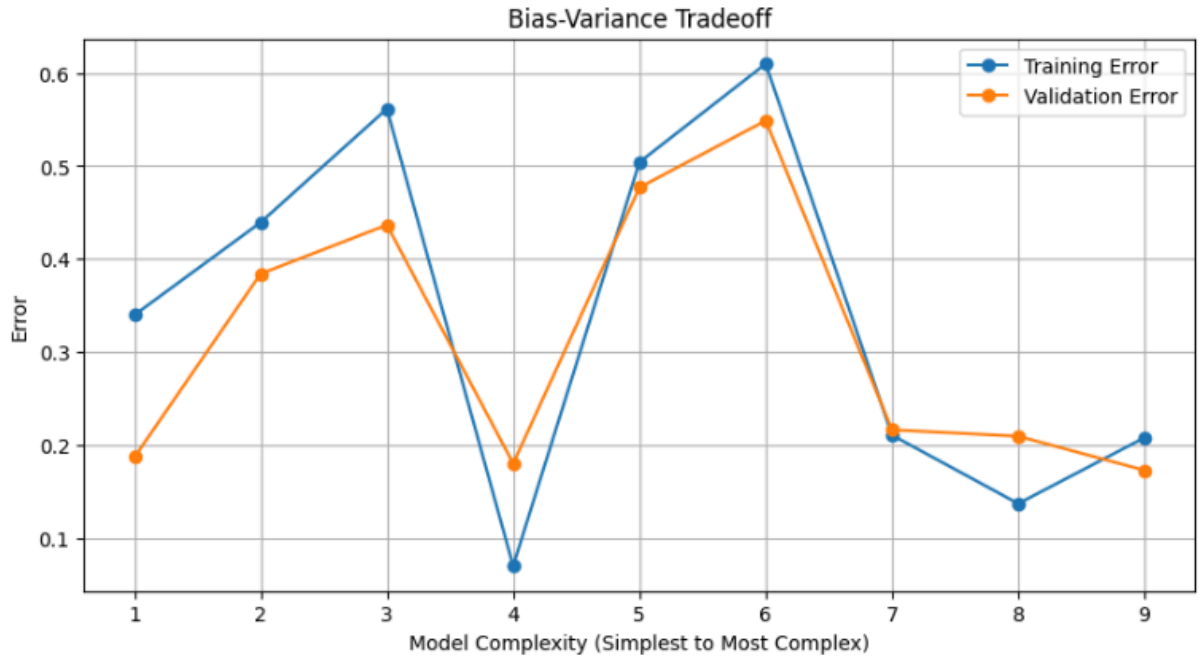
Models Validation Error

Model 1	0.1871
Model 2	0.3842
Model 3	0.4367
Model 4	0.1804
Model 5	0.4769
Model 6	0.5488

Model 7	0.2166
Model 8	0.2097
Model 9	0.1731

Models

Bias Variance Trade Off



The analysis of the bias-variance trade-off using training and validation loss reveals important insights into the model's performance at varying levels of complexity. Models at **complexity levels 4 and 7-9** demonstrate the best balance between **bias and variance**, indicating that they generalize well to unseen data. This suggests that these models are neither underfitting (high bias) nor overfitting (high variance) the data.

However, models at **complexity levels 5 and 6** appear to introduce excessive variance or training instability. This could be due to an increase in the model's capacity (e.g., deeper layers or larger parameter sizes) without sufficient regularization or training time. As a result, the models might overfit the training data while failing to generalize to the validation set.

To fully leverage the potential of more complex models like those at levels **7-9**, it may be necessary to apply techniques such as **hyperparameter tuning**, **dropout**, or **early stopping**. These adjustments can reduce variance and improve generalization. Additionally, using a larger and more diverse training set or employing **data augmentation** may help stabilize training and improve performance on the validation set.

In summary, the models at **complexity levels 4 and 7-9** offer a strong trade-off between bias and variance, while **levels 5 and 6** require further refinement to reduce variance and stabilize training. This insight informs the decision to focus on the more effective models for deployment.

Winning Model

The final winning model, based on the bias-variance trade-off graph, is Model 7, which achieved the lowest validation error among all models. Model 7, an EfficientNet, recorded a partial AUC-above-TPR (pAUC-aboveTPR) of 0.1034 and a recall rate of 0.64 for class 1 (malignant cases). This performance surpasses the other models, making it the optimal choice for deployment.

The superior performance of EfficientNet compared to simpler models like the 3-layer CNN and ResNet can be attributed to its compound scaling strategy, which optimizes the network's depth, width, and resolution. This scaling enables EfficientNet to learn more detailed and hierarchical features from images, enhancing its ability to identify subtle patterns in skin lesions.

However, a notable observation from the training process is that Model 1 (the 3-layer CNN) performed almost as well as Model 7. This could be due to the uniform preprocessing and normalization strategy applied across all models. Complex architectures like EfficientNet and ResNet often require data preprocessing and normalization tailored to their design. EfficientNet, in particular, is sensitive to the size, resolution, and pixel intensity distribution of input images. By applying the same preprocessing steps (like resizing and normalization) to all models, the true potential of ResNet and EfficientNet might not have been fully realized. This oversight could explain why the simpler 3-layer CNN achieved results comparable to more complex models.

Moving forward, optimizing the preprocessing and normalization strategy for each model architecture—especially for EfficientNet and ResNet—could unlock their full potential. This may include image resizing to match the model's expected input size, adjusting normalization to the pre-trained model’s parameters, and possibly employing data augmentation to improve generalization. Despite this, Model 7 remains the most promising option due to its superior performance in key metrics, especially for class 1 recall, which is crucial for detecting malignant skin lesions.

Winning Model on Test Data

```
The partial auroc of the final model on the test image is 0.11427590046074211
              precision    recall  f1-score   support

     Class 0       0.98      0.93      0.96      1431
     Class 1       0.26      0.63      0.37        59

    accuracy                           0.92      1490
   macro avg       0.62      0.78      0.66      1490
weighted avg       0.96      0.92      0.93      1490
```

Based on the test dataset metrics, the model achieved a recall rate of 0.63 for Class 1 and a pAUC-aboveTPR of 0.114, both of which are higher than the validation set metrics. Given the constraints in environment and computing power, these results are satisfactory. However, performance could likely be improved with tailored data preprocessing and normalization specifically suited for EfficieNet. Despite these positive results, I believe the model still isn't optimal for this prediction task, as achieving higher performance would likely require more data for Class 1. This would help the model better capture the patterns necessary to increase recall and overall accuracy in the target class.

Model deployment

Deploy method

The best way to deploy the skin lesion classication model depends on the problem statement and requirements. In this case, real-time deployment is the most suitable approach. This strategy is ideal for clinical settings where instant predictions are needed, such as during a dermatologist's consultation. It is also benecial for scenarios where users upload images and metadata through a web or mobile application and expect immediate feedback. Additionally, real-time deployment supports on-the-y decision-making, such as prioritizing patients based on the malignancy probability of their lesions. Real-time inference is well-suited to the nature of skin lesion classication, which typically involves small-scale inputs, such as a single image and metadata per request. This approach ensures that users, including doctors and patients, can act promptly on the model's predictions, improving the overall user experience by providing immediate and actionable insights about the lesion's likelihood of being malignant. The other reason that I choose real-time inference is users may expect immediate results, making batch processing less practical.

Model Monitoring

As I walk you through my **monitoring plan** for the skin lesion classification model, I want to highlight the key performance, business, and operational metrics I will be tracking. Each of these plays a crucial role in ensuring the model remains effective, reliable, and aligned with clinical, business, and operational goals.

1. Model Performance Metrics

The first step in my monitoring plan is to ensure that the model is able to **classify skin lesions as benign or malignant** accurately. To achieve this, I track key performance metrics that give me a clear view of the model's capabilities and limitations.

- **Accuracy, Precision, Recall, and F1-Score**: I use these metrics to assess the overall and class-specific performance of the model. Since correctly identifying malignant skin lesions (Class 1) is a priority, I place a strong focus on **Recall for Class 1**. This ensures that I am minimizing false negatives, as missing malignant lesions could have serious consequences. Precision is also important to avoid false alarms, but recall is the higher priority.
- **AUROC and Partial AUC**: While AUROC provides insight into how well the model ranks positive (malignant) cases higher than negative (benign) cases, I also track **Partial AUC**, which focuses on the model's performance at high true positive rates (TPR). This ensures that the model is identifying most of the malignant cases, even when set to a more conservative threshold.
- **Log Loss**: Log loss allows me to assess how well the model's predicted probabilities align with reality. If the log loss increases, it indicates that the model might be "overconfident" in its predictions, which could lead to errors. This metric helps me ensure the model produces well-calibrated probabilities.
- **Prediction Latency**: Since this model is intended for use in real-time applications, such as telehealth or mobile apps, I track **prediction latency**. This measures how long it takes for the model to process the image and metadata inputs and return a prediction. If the latency is too high, it could lead to a poor user experience. If I notice latency increases, I would look for ways to reduce model size or use techniques like quantization or on-device inference.

2. Business Metrics

While performance metrics ensure technical accuracy, the business metrics help me align the model's output with clinical impact and operational efficiency. Here's what I plan to track:

- **False Negative Rate for Class 1 (Malignant Lesions)**: Missing a malignant lesion is one of the biggest risks. To avoid this, I track the **false negative rate** to ensure it stays low. If I notice this rate increasing, I would consider retraining the model with a stronger emphasis on sensitivity for Class 1, possibly by adjusting the classification threshold.
- **False Positive Rate for Class 1**: While I want to catch as many malignant lesions as possible, I also don't want to overwhelm the healthcare system with unnecessary referrals. If the false positive rate becomes too high, patients may receive unnecessary follow-ups, which can be stressful for them and resource-intensive for clinicians. I aim to strike a balance here.
- **Resource Utilization**: I also plan to track how the model affects operational workflows. If the model generates too many false positives, it could increase the number of referrals clinicians need to review. This could lead to bottlenecks in healthcare workflows. By tracking the number of referrals and comparing it to normal operational capacity, I can ensure that the system is not overburdened.

3. Operational Metrics

Tracking operational metrics helps me **maintain the long-term health of the model**. It's one thing for the model to perform well on Day 1, but I want it to continue working well over time as data and user behaviors evolve. Here's what I track to keep the system stable:

- **Data Drift**: This happens when the input data starts to differ from what the model was trained on. In my case, it could be due to changes in the demographics of app users (like age or skin tone) or updates to image capture technology. I monitor feature distributions from the incoming data and compare them to the training set. If drift is detected, I would retrain the model with more recent data to realign its understanding of the inputs.
- **Concept Drift**: Concept drift occurs when the relationship between inputs (e.g., images) and the target labels (benign vs. malignant) changes over time. For example, if dermatologists start using new criteria for classifying lesions, the model's predictions may no longer match expert opinions. To catch this, I will monitor changes in key performance metrics like AUROC and recall. If I notice a persistent drop in performance, I would retrain the model on the most recent labeled data.
- **Model Inference Volume**: This metric tracks the number of predictions made by the model each day. If I notice large spikes or drops, it could signal unusual activity, like system misuse, operational failures, or even a shift in how users interact with the system. By tracking this, I can act quickly if something goes wrong.