Skin lesion classification methodologies and workflow

Problems in the dataset of ISIC 2024 challenge: its binary classification task on a multimodal data.

They contain a lot of null values, have a huge class imbalance ratio in the training set.

The images are all in irregular shape.

EDA:

1. Target Class Distribution
The dataset is heavily imbalanced, with:

- Class 0 (Benign samples): 400,666 samples.
- Class 1 (Malignant samples): 393 samples.

This imbalance suggests the need for techniques like class weighting, oversampling, or SMOTE to improve model performance.

2. Missing Values Analysis

Several columns contain high proportions of missing values:

- iddx_5, mel_mitotic_index, and mel_thick_mm are missing in 99% or more of the dataset.
- The sex feature has ~2.87% missing values.
- age_approx has ~0.69% missing values.

Among malignant samples:
iddx_5 is completely missing.
mel_mitotic_index and mel_thick_mm are missing in ~86% and 84% of cases, respectively.

Inference: Columns with extreme missing rates might be dropped or require imputation strategies like mean/median imputation or predictive modeling.

3. Summary Statistics of Numerical Features

The dataset contains 401,059 samples and 37 numerical features.
Mean Age: ~58 years.
Clinical Size (Long Diameter): Mean = 3.93mm, with a range from 1mm to 28.4mm.
The dataset contains features related to lesion characteristics, including tbp_lv_A, tbp_lv_B, and mel_thick_mm.

Inference: The variability in lesion characteristics can provide valuable predictive signals. However, extreme outliers may require normalization or log transformation.

4. Data Visualization Findings

- Gender Bias: The dataset is male-biased, with significantly more samples from males than females.

- Skewed Distributions: Many numerical features exhibit positive or negative skew, with very few showing normal distribution.
- Many numerical variables contain a large number of zeros, indicating possible sparsity.

Inference: Feature transformation techniques (log scaling, normalization) may be required to handle skewed distributions.

5. Pairwise Scatter Plot Analysis

A scatter matrix was generated for numerical columns, color-coded by target (malignancy status). Visual inspection suggests potential correlations among lesion features, but some attributes show high variance.

Inference: Feature selection or dimensionality reduction (PCA, feature importance analysis) can help identify the most predictive attributes.

Methods and workflow :

Basic model – version 4

Date processing:

1. Imputing null values:

   Using random forest classifier to predict the missing values by building a random forest model on the missing column as the target and training it on the non-null values to predict the target column then using the trained model to predict the missing values.

   This is done primarily for three columns: Sex, age approx., anatom site general

2. Feature selection:

   Then we remove some columns without any useful data such as,

   'patient_id','image_type','tbp_tile_type','attribution','copyright_license'

3. Handling Class Imabalance:

   To address class imbalance, we perform random downsampling of the negative or benign class to 0.01%, and upsampling of the malignant or positive class by a factor of 10 through repetition, resulting in an equi-proportional sample.

4. One hot encoding:

   We encode the following categorical variables as one-hot encoded columns

   'sex', 'anatom_site_general', 'tbp_lv_location', 'tbp_lv_location_simple'

5. Min-Max scaling:
   We normalize the dataset through min-max scaler such that all features are in the same data scale.
6. Image resizing: We resize the images using bilinear interpol technique to a shape of 128x128x3. As most of the images are in the range between 100-150.

We get a resultant data as a 66 dimensional vectors for the metadata.

Image Augmentation techniques:

i. Random Horizontal Flip: This technique randomly flips the image horizontally with a 50% probability. By doing so, it helps the model become invariant to the direction of an object within the image, improving generalization.

ii. Random Brightness Adjustment: This function adjusts the brightness of the image by a random factor. The max_delta parameter of 0.2 means that the brightness can be varied within the range of -20% to +20%. This augmentation simulates different lighting conditions.

iii. Random Contrast Adjustment: Here, the image contrast is adjusted by a random factor. The lower and upper parameters of 0.8 and 1.2, respectively, indicate that the contrast of the image can be increased by up to 20% or decreased by up to 20%. This augmentation helps in making the model robust to variations in contrast.

Model architecture: This model is a multi-input neural network using Convolutional Neural Networks (CNNs) for image processing and Dense layers for tabular data processing. The two branches are concatenated before passing through fully connected layers for binary classification.

Input Layers:

img_input: Processes images of shape (IMAGE_HEIGHT, IMAGE_WIDTH, 3).
tab_feat: Processes tabular features with 66 input values.

Image Processing Branch (CNN):

4 convolutional blocks:

- Conv2D layers with ReLU activation and batch normalization.
- MaxPooling2D layers to reduce spatial dimensions.
- Fully connected layers:
- Flatten() layer to convert feature maps into a vector.
- Dense(512, activation='relu') followed by Dropout(0.5).

Tabular Data Processing Branch:

- Two fully connected layers:
- Dense(64, activation='relu') → BatchNorm → Dropout(0.3)
- Dense(128, activation='relu') → BatchNorm → Dropout(0.3)

Fusion & Classification:

- Concatenation of both branches
- Fully connected layers after merging:
- Dense(640, activation='relu') → Dropout(0.4)
- Dense(128, activation='relu') → Dropout(0.3)

Final Output Layer: Dense(1, activation='sigmoid') for binary classification.

Model Summary:

Total Parameters: 26,586,753
Trainable Parameters: 26,585,409
Non-trainable Parameters: 1,344

2. Compilation & Training

Loss Function: binary_crossentropy (since it is a binary classification problem).
Optimizer: Adam with learning_rate=1e-3.

Evaluation Metrics:

AUC (Area Under the Curve) with ROC (pAUC)
Accuracy

Callbacks:

EarlyStopping: Stops training if validation loss does not improve for 10 epochs.
ReduceLROnPlateau: Reduces learning rate by a factor of 0.2 if validation loss stagnates.
ModelCheckpoint: Saves the best model based on validation loss.

Training Process:

Epochs: 50
Dataset: Uses train_dataset and val_dataset.

Final Metrics:
Training Accuracy: 0.0172, Training AUC: 0.9855, Training Loss: 0.1453
Validation Accuracy: 0.0063, Validation AUC: 0.9898, Validation Loss: 0.1188
Public test score: 0.11640
Private test score: 0.10998

Densenet – model:

Preprocessing – same as the above model,

- **Handling Class Imbalance:**
  - **Before Sampling:**
    - Class 0: **99.90%**
    - Class 1: **0.10%** (Highly imbalanced)
  - **After Random Undersampling & Oversampling:**
    - Class 0: **91.07%**
    - Class 1: **8.93%**
  - **Techniques Used:**
    - Random undersampling of class 0 (5% sample)
    - Random oversampling of class 1 (500% with replacement)
    - **Final dataset is shuffled**
- **Feature Scaling:**
  - **Min-Max Normalization:**
    - MinMaxScaler() is applied to numerical columns (num_col).
- **Train-Validation Split:**
  - **Stratified Split:**
    - Ensures class distribution is preserved across training and validation sets.
    - **Test Size:** 20%
    - **Random State:** 42 (for reproducibility)
- **Data Shapes:**
  - **Image batch:** (64, 224, 224, 3)
  - **Tabular data:** (64, 66)
  - **Labels:** (64, 1)
  - **Value Ranges:**
    - **Images:** (-0.04 to 1.04)
    - **Tabular:** (0.00 to 1.00)

---

2. Class Imbalance Handling

- **Class Weights Calculation:**
  - **Using compute_class_weight('balanced')**
  - **Computed Weights:**
    - Class 0: 0.549
    - Class 1: 5.597 (to address minority class underrepresentation)
  - **Applied in model.fit()** via class_weight parameter.

---

3. Model Parameters

- **Multimodal Architecture:**
  - **Input:**
    - **Image Branch:** (224, 224, 3) (Processed via DenseNet-169)
    - **Tabular Branch:** 66 features
  - **Image Branch (CNN-based using DenseNet-169):**
    - GlobalAveragePooling2D
    - Dropout (0.2)
  - **Tabular Branch (MLP layers):**
    - Dense(64, activation='selu') → BatchNorm → Dropout(0.3)
    - Dense(128, activation='selu') → BatchNorm → Dropout(0.3)
    - Dense(256, activation='selu')
  - **Concatenation & Fully Connected Layers:**
    - Dense(1024, activation='selu') → Dropout(0.2)
    - Dense(256, activation='selu')
    - Dense(64, activation='selu') → Dropout(0.2)
    - Dense(16, activation='selu')
  - **Output Layer:**
    - Dense(1, activation='sigmoid') (for binary classification)
- **Model Summary:**
  - **Total Parameters:** 14,936,289 (56.98 MB)
  - **Trainable Parameters:** 14,777,505 (56.37 MB)
  - **Non-trainable Parameters:** 158,784 (620.25 KB)

---

4. Model Compilation & Metrics

- **Optimizer:** Adam(learning_rate=1e-4)
- **Loss Function:** Binary Crossentropy
- **Evaluation Metric:** AUC (Precision-Recall Curve)

---

5. Callbacks Used

1. **Early Stopping:**
   - Monitors val_loss, patience 5, restores best weights.
2. **ReduceLROnPlateau:**
   - Monitors val_loss, reduces learning rate by 0.2 if no improvement after 2 epochs.
3. **Model Checkpointing:**
   - Saves best model based on val_loss.

---

6. Training Progress

- **Total Epochs:** 150
- **Current Training Status (Epoch 14/150):**
  - **Training Loss:** 0.0074
  - **Training pAUC:** 0.9955
  - **Validation Loss:** 0.0428

- **Validation pAUC:** 0.9695
- **Current Learning Rate:** 1.0000e-06 (reduced due to ReduceLROnPlateau)

Resultant scores:

Dataset – eda - version 2:

1. Preprocessing Techniques

1. Scaling and general preprocessing steps.
2. Categorical features are processed using one-hot encoding for small categories and feature hashing for high-cardinality features.
3. Iterative Imputer is used for missing values.
4. MinMaxScaler is applied to numerical features.
5. Age binning is performed to group ages into predefined categories.
6. Resulting metadata feature count: 52
7. Startified splitting of the train and val dataset to ensure no overlap between the patient id's thus making sure of no data leakage.
8. No data imbalance handling

2. XGBoost Model Configuration

- Objective Function: binary:logistic (for binary classification with logistic regression output).
- Evaluation Metric: AUC (Area Under the Curve, a common metric for classification tasks).
- Class Balancing: scale_pos_weight is computed as the ratio of the positive class to the negative class.
- Hyperparameters:
  - max_depth = 6: Limits tree depth to prevent overfitting.
  - eta = 0.1: Learning rate for boosting iterations.
  - subsample = 0.8: Uses 80% of the training data per boosting round.
  - colsample_bytree = 0.8: Uses 80% of features per tree.
  - random_state = 42: Ensures reproducibility.

3. resultant scores:

Private score: 0.07441
Public score: 0.10538

Basic model – version 5 :

Preprocessing Methodology:

- Column Removal:
  - Dropped train-only columns (lesion_id, iddx_full, etc.) to avoid data leakage.
  - Removed irrelevant columns (image_type, attribution, etc.).

- Feature Engineering:
  - Separated numerical and categorical features.
  - Age Processing: Binned into discrete intervals.

- Categorical Encoding:
  - One-hot encoding for low-cardinality features.
  - Feature hashing for high-cardinality categorical features.

- Scaling & Imputation:
  - Applied MinMaxScaler to numerical features.
  - Iterative Imputer for handling missing values.

- Final Preprocessing Output:
  - Generated a structured dataset with transformed numerical and categorical features.

- Class Imbalance Handling:
  - Applied SMOTETomek (SMOTE + Tomek Links):
  - Undersampled the majority class before applying SMOTE.
  - Used nearest neighbors to ensure consistency of categorical features.
  - Created a balanced dataset for training.

- Stratified Split:

  - Ensured training and validation sets do not share patient IDs.
  - Maintained class distribution while splitting the dataset.

Model Architecture:

Dual-Input Model (Image + Tabular Data)

Image Processing Branch (ResNet Backbone):

- Used resnet_vd_18_imagenet as a feature extractor.
- Applied GlobalAveragePooling2D and Dropout (0.2).

Tabular Data Processing Branch:

- Dense layers with SELU activation and Batch Normalization.
- Layers: Dense(64) → BatchNorm → Dropout(0.3) → Dense(128) → BatchNorm → Dropout(0.3) → Dense(256)

Fusion & Classification:

- Concatenated image and tabular features.
- Fully connected layers: Dense(1024) → Dropout(0.2) → Dense(256) → Dense(64) → Dropout(0.2) → Dense(16)
- Output Layer: Dense(1, activation='sigmoid') for binary classification.

Model Parameters & Training:

- Loss Function: SquaredHinge
- Optimizer: Adam (learning_rate=1e-4)

- Evaluation Metric: AUC (PR curve)

Callbacks Used:
- EarlyStopping (patience=5, restores best weights)
- ReduceLROnPlateau (factor=0.2, patience=2, min_lr=1e-6)
- ModelCheckpoint (saves the best model based on validation loss)

Training Configuration:
- Epochs: 150
- Class Weights: Applied to handle imbalance
- Final Training Metrics:
- Training Accuracy, AUC, Loss
- Validation Accuracy, AUC, L

resultant scores:
- Private score: 0.03494
- Public score: 0.02920

Basic model – version 9:

Preprocessing Methodology:

1. **Column Removal:**
   - Columns present only in the training dataset (lesion_id, iddx_full, iddx_1, etc.) are removed.
   - Irrelevant columns (image_type, tbp_tile_type, etc.) are removed from both train and test datasets.
2. **Feature Engineering:**
   - Separation of features (X_train) and target variable (y_train).
   - The PreprocessingPipeline class is implemented for structured preprocessing:
     - Missing values in the age_approx column are imputed with the median and categorized into bins.
     - Categorical features with fewer than six unique values are one-hot encoded; others are hashed.
     - Numeric features undergo missing value imputation using IterativeImputer and scaling using MinMaxScaler.
3. **Feature Consistency Check:**
   - Ensures that train and test datasets have the same features after processing.
   - The final dataset contains 54 features.

Class Imbalance Handling:

1. **Resampling Strategy:**
   - A pipeline is used to address class imbalance:
     - Random undersampling of the majority class (strategy: 0.05).
     - SMOTETomek resampling to balance classes.
   - The numerical and categorical data are separately processed to maintain coherence in feature representation.
   - The final dataset after resampling consists of 15,716 samples with 54 features.
2. **Stratified Splitting:**
   - The dataset is split into training and validation sets, ensuring that patient_id does not overlap.
   - Class distribution:
     - Training: 6,270 samples (class 0), 6,228 samples (class 1).

- Validation: 1,590 samples (class 0), 1,628 samples (class 1).

Model Architecture:

1. **Input Layers:**
   - Image input (224x224x3 RGB images).
   - Tabular input with 52 features.
2. **Image Branch:**
   - Uses ResNet_VD_18 pretrained backbone.
   - Global average pooling followed by a dropout layer (0.2).
3. **Tabular Branch:**
   - Sequential dense layers with SELU activation and batch normalization.
   - Dropout applied at different stages (0.3, 0.3, etc.).
4. **Fusion & Classification:**
   - The image and tabular branches are concatenated.
   - Fully connected layers with SELU activation:
     - Layers: 1024, 512, 256, 64, 32, 16, 8, 5.
     - Dropout applied (0.2 at multiple layers).
   - Output layer: single neuron with sigmoid activation for binary classification.

Model Parameters & Training Metrics:

- **Dataset Specification:**
  - Image batch shape: (128, 224, 224, 3).
  - Tabular data shape: (128, 52).
  - Labels shape: (128, 1), dtype: int32.
  - Image value range: (-0.04, 1.03).
  - Tabular value range: (-1.00, 4.00).
- **Model Parameters**
  - Total Parameters: 11,799,627 (45.01 MB)
  - Trainable Parameters: 11,789,387 (44.97 MB)
  - Non-trainable Parameters: 10,240 (40.00 KB)
- **Training Parameters**
  - Optimizer: Nadam(learning_rate=0.001, weight_decay=0.01)
  - Loss Function: BinaryCrossentropy()
  - Metric: AUC(curve='PR', name='pAUC')
  - Epochs: 250
  - Batch Size: *(Not explicitly mentioned, inferred from dataset)*
  - Validation Dataset: val_dataset
  - Class Weights: {0: 0.9967, 1: 1.0034} (Computed for imbalance handling)
- **Callbacks:**
  - **EarlyStopping**
    - Monitor: 'loss'
    - Patience: 7
    - Restore Best Weights: True
  - **ReduceLROnPlateau**
    - Monitor: 'loss'
    - Factor: 0.2
    - Patience: 3
    - Minimum Learning Rate: 1e-6
  - **ModelCheckpoint**
    - Filepath: 'best_model.keras'
    - Monitor: 'val_loss'
    - Save Best Only: True
- **Current Training Progress**
  - **Epochs Completed:** 8 / 250
  - **Last Epoch Performance:**
    - **Loss:** 0.6417
    - **pAUC:** 0.1681

- **Validation Loss:** 0.6823
- **Validation pAUC:** 0.6758
- **Learning Rate:** 4.0000e-4 (Reduced from initial due to ReduceLROnPlateau)

resultant scores:

- Private score: 0.04173
- Public score: 0.03305

Basic model – version 11:

Preprocessing: Same as the base model version 9

1. **Handling Class Imbalance:**
   - Downsampled the majority class to 5% using RandomUnderSampler.
   - Applied SMOTETomek to generate synthetic samples and remove Tomek links.
   - Used a nearest neighbors approach to recover categorical data after resampling.
   - Stratified train-test split ensuring patient_id doesn't overlap.
2. **Image Preprocessing:**
   - Decoded JPEG images.
   - Resized images to (224, 224).
   - Applied data augmentation:
     - Random horizontal flip.
     - Random brightness adjustment.
     - Random contrast adjustment.
   - Normalized pixel values to [0,1].
3. **Tabular Data Processing:**
   - Dropped isic_id and patient_id columns.
   - Converted numerical data to float32.
   - Encapsulated images and tabular data into TensorFlow Dataset.

Class Imbalance Handling:

- Downsampling (reducing class 0 instances before SMOTE).
- SMOTETomek applied for resampling.
- Computed class weights for training:
  {0: 0.9967, 1: 1.0034} (almost balanced).

Model Parameters:

- Architecture:
  - Image Branch: ResNet-152 backbone (pretrained on ImageNet).
  - Tabular Branch:
    - 3 dense layers: 64 → 128 → 256 neurons, selu activation.
    - Batch normalization and dropout (0.3).
  - Fully Connected Layers:
    - Concatenation of image and tabular features.
    - Dense layers: 1024 → 512 → 256 → 64 → 32 → 16 → 8 → 5.
    - selu activation and dropout (0.2).
  - Output Layer:
    - Single neuron with sigmoid activation for binary classification.
- Compilation:
  - Optimizer: Adam (learning_rate=1e-2, weight_decay=1e-3).
  - Loss: Binary Crossentropy.
  - Metric: AUC (Precision-Recall curve) pAUC.

- Callbacks:
    - Early Stopping (patience=10, restores best weights).
    - Reduce Learning Rate on Plateau (factor=0.2, patience=5, min_lr=1e-6).
    - Model Checkpoint (best_model.keras).
- Params:The model has a total of 59,671,467 parameters, out of which:
    - Trainable parameters: 59,519,659 (227.05 MB)
    - Non-trainable parameters: 151,808 (593.00 KB)

Model Performance (Epoch 16/55):

- Training Loss: 0.8886
- Training pAUC: 0.2277
- Validation Loss: 1.0475
- Validation pAUC: 0.5059
- Learning Rate: 0.0100

resultant scores:

- Private score: 0.02000
- Public score: 0.02000

Basic model – version 15:

Preprocessing Steps

1. Column Removal
    a. Dropped columns present only in the train dataset.
    b. Removed irrelevant columns from both train and test datasets.
2. Feature Engineering
    a. Separated features (X_train) and target (y_train).
    b. Implemented a preprocessing pipeline:
        i. Age Transformation: Median imputation followed by binning into five age groups.
        ii. Categorical Encoding:
            1. One-hot encoding for low-cardinality categorical features.
            2. Feature hashing (5 features) for high-cardinality categorical features.
        iii. Numerical Features:
            1. Imputation using IterativeImputer.
            2. Scaling with MinMaxScaler.
3. Feature Consistency Check
    a. Ensured features match between train and test datasets.
    b. Total number of processed features: 54.

Class Imbalance Handling

1. Undersampling with NearMiss
    a. Used NearMiss(version=2, sampling_strategy=0.005, n_neighbors=3, n_jobs=-1) to undersample the majority class based on nearest neighbors.
2. Oversampling with RandomOverSampler
    a. Applied RandomOverSampler(sampling_strategy=0.1, shrinkage=0.4) to rebalance the minority class.
3. Final Resampled Dataset Sizes
    a. Total samples after resampling: 86,460.
    b. Class distribution after resampling:
        i. Class 0: 78,600
        ii. Class 1: 7,860
4. Stratified Split for Training and Validation
    a. Ensured patient-wise stratification.
    b. Final train set:
        i. Class 0: 60,859
        ii. Class 1: 6,455
    c. Final validation set:
        i. Class 0: 17,741
        ii. Class 1: 1,405

Model Parameters

1. Architecture
    a. Multimodal model using ResNetV2-50 as image backbone.
    b. Tabular data processed through a fully connected network.
    c. Concatenated both branches before passing through dense layers.
2. Layer Details
    a. Image Branch:
        i. ResNetV2-50 pretrained on ImageNet.
        ii. Global Average Pooling layer.
    b. Tabular Branch:
        i. Dense layers: 64 → 128 → 512 (Batch Normalization & Dropout).
    c. Concatenation Layer.
    d. Fully Connected Layers:
        i. 1024 → 512 → 256 → 64 → 32.

        ii. Dropout (0.2) applied at several layers.
    e. Output Layer:
        i. Single neuron with sigmoid activation.
3. Total Parameters
    a. 25,085,441 (95.69 MB).
    b. Trainable Parameters: 25,039,617.
    c. Non-trainable Parameters: 45,824.
4. Optimizer & Loss Function
    a. Optimizer: AdamW (learning_rate=1e-3, amsgrad=True).
    b. Loss function: Binary Focal Crossentropy (gamma=3.0, class balancing applied).

Model Metrics

- Training Metrics:
    - PR_pAUC: 0.9985
    - ROC_pAUC: 0.9999
    - Loss: 0.3304
- Validation Metrics:
    - val_PR_pAUC: 0.9381
    - val_ROC_pAUC: 0.9737
    - val_Loss: 4.1662
- Learning Rate: 0.0010

resultant scores:

- Private score: 0.04972
- Public score: 0.04408

Dataset – EDA version 4:

Preprocessing Steps:

1. Drop Irrelevant Columns – Removes columns like patient_id, image_type, tbp_tile_type, attribution, and copyright_license.
2. Drop Train-Only Columns – Excludes columns that exist only in the training set (e.g., lesion_id, iddx_full, mel_mitotic_index, etc.).
3. Identify Column Types:
    a. Categorical columns (object type).
    b. Numerical columns (int64, float64).
4. Preprocessing Pipelines:
    a. Numerical Pipeline: Imputation (median) → Standard Scaling.
    b. Categorical Pipeline: One-Hot Encoding (handle unknowns).
5. Apply Column Transformer – Transforms data using the pipelines and aligns train-test columns.

6. Train-Test Split – Splits data into training (80%) and validation (20%) while maintaining stratification.

Class Imbalance Handling:

1. Random Under-Sampling – Reduces majority class instances to 10% of their original count.
2. Random Over-Sampling – Increases minority class instances to 30% of the majority class count.
3. Class Weights Computation – Uses inverse class frequencies:
    a. Class 0: Weight = 1.3
    b. Class 1: Weight = 4.33
4. Scale Positive Weight – Adjusted in XGBoost using scale_pos_weight = 4.33 / 1.3 ≈ 3.33.

Model Architecture:

- Model Type: XGBoost (Extreme Gradient Boosting)
- Objective: binary:logistic (Binary classification)
- Loss Function: logloss
- Feature Importance: Uses colsample_bytree and subsample for feature selection.

Model Parameters:

Parameter	Value
objective	binary:logistic
eval_metric	logloss
scale_pos_weight	3.33
max_depth	6
eta (learning rate)	0.05
subsample	0.8
colsample_bytree	0.8
random_state	63
num_boost_rounds	282

Model Performance Metrics:

- Cross-validation results (5-fold CV):
    - Minimum test-logloss = 0.07705
    - Train-logloss improves from 0.7771 → 0.0094
    - Test-logloss improves from 0.7799 → 0.0770

- Standard deviation for test-logloss ≈ 0.0145

Resultant scores:

- Private score: 0.13656
- Public score: 0.16416

Dataset EDA – version 6

Preprocessing Steps

1. Dropping Irrelevant Columns
    a. Removed: 'patient_id', 'image_type', 'tbp_tile_type', 'attribution', 'copyright_license'
2. Dropping Train-Only Columns
    a. Removed: 'lesion_id', 'iddx_full', 'iddx_1', 'iddx_2', 'iddx_3', 'iddx_4', 'iddx_5', 'mel_mitotic_index', 'mel_thick_mm', 'tbp_lv_dnn_lesion_confidence'
3. Identifying Column Types
    a. Categorical Columns: Detected using select_dtypes(include=['object']), except 'isic_id'.
    b. Numerical Columns: Detected using select_dtypes(include=['int64', 'float64']), except 'target'.
4. Preprocessing Pipelines
    a. Numerical Data:
        i. Imputation: KNNImputer()
        ii. Scaling: StandardScaler()
    b. Categorical Data:
        i. Encoding: OneHotEncoder(handle_unknown="ignore")
    c. Applied transformations using ColumnTransformer.
5. Alignment of Train & Test Data
    a. Ensured the test dataset contained the same feature columns as the training dataset.
    b. Missing columns in test data were filled with zeros.

Class Imbalance Handling

1. Random Undersampling
    a. Applied using RandomUnderSampler(sampling_strategy=0.1) to balance majority class.
2. Random Oversampling
    a. Applied using RandomOverSampler(sampling_strategy=0.3) to increase minority class instances.
3. Final Data Shapes After Resampling:

    a. Training Set: (4082, 73)
    b. Validation Set: (1027, 73)
4. Class Weights Calculation:
    a. Class 0: 1.3
    b. Class 1: 4.33
    c. Used in XGBoost as scale_pos_weight = class_weights[1] / class_weights[0].

Model Architecture

- Model Type: XGBoost Binary Classifier
- Input: Processed features excluding 'isic_id'
- Output: Binary classification for 'target'

Model Parameters

- Objective: 'binary:logistic'
- Evaluation Metric: 'logloss'
- Class Weight Scaling: scale_pos_weight = 4.33 / 1.3
- Max Depth: 6
- Learning Rate (eta): 0.05
- Subsample Ratio: 0.8
- Colsample_bytree: 0.8
- Random Seed: 63
- Boosting Rounds: 300 (Selected from cross-validation)

Model Metrics (Cross-Validation)

Boosting Round	Train LogLoss (Mean)	Test LogLoss (Mean)
0	0.7741	0.7770
50	~0.2000	~0.2200
100	~0.1000	~0.1200
200	~0.0090	~0.1015
300 (Best)	0.0084	0.1011

Resulatant scores:

- Private score: 0.14480
- Public score: 0.16030

Basic model buildding version 20:

Preprocessing Steps:

1. Dropping irrelevant columns: patient_id, image_type, tbp_tile_type, attribution, copyright_license.
2. Dropping train-only columns: lesion_id, iddx_full, iddx_1, iddx_2, iddx_3, iddx_4, iddx_5, mel_mitotic_index, mel_thick_mm, tbp_lv_dnn_lesion_confidence.
3. Identifying categorical and numerical columns.
4. Applying transformations:
    a. Numerical columns: KNN Imputation followed by Standard Scaling.
    b. Categorical columns: One-Hot Encoding.
5. Storing train columns for consistency between train and test data.
6. Aligning test dataset with train dataset by adding missing columns with value 0.

Class Imbalance Handling:

1. Random undersampling to reduce the majority class to 5%.
2. Random oversampling to increase the minority class to 40%.
3. Final dataset shape: (8792, 73) features with balanced target labels.

Model Architecture:

- Inputs:
    - Image Input: (224, 224, 3)
    - Tabular Input: (72,)
- Image Branch:
    - ResNet_V2_50 as backbone.
    - Flatten layer.
- Tabular Branch:
    - Dense Layer (128, ELU, HeNormal) -> BatchNorm -> Dropout (0.2)
    - Dense Layer (256, ELU, HeNormal) -> BatchNorm -> Dropout (0.2)
    - Dense Layer (512, ELU, HeNormal)
- Concatenation of both branches.
- Fully Connected Layers:
    - Dense Layer (1024, ELU) -> Dropout (0.2)
    - Dense Layer (512, ELU) -> Dropout (0.2)
    - Dense Layer (256, ELU)
    - Dense Layer (64, ELU) -> Dropout (0.2)
    - Dense Layer (32, ELU)

- Output Layer: Dense (1, Sigmoid) for binary classification.

Model Parameters:

- Total Parameters: 75,514,049
- Trainable Parameters: 75,467,841
- Non-trainable Parameters: 46,208
- Optimizer: Adam with Exponential Decay Schedule:
  - Initial Learning Rate: 1e-2
  - Decay Steps: 5000
  - Decay Rate: 0.9
- Loss Function: Binary Crossentropy.
- Metrics:
  - Precision-Recall AUC (PR_pAUC)
  - ROC AUC (ROC_pAUC)
- Callbacks:
  - Early Stopping (patience=5, restore best weights, start after epoch 5)
  - Model Checkpoint (saving best model)

Model Metrics:

- After 11 epochs:
  - PR_pAUC: 0.9198
  - ROC_pAUC: 0.9709
  - Loss: 0.1976
  - Training Time per Epoch: ~39s per epoch (138 steps).

Resultant scores:

- Private score: 0.12413
- Public score: 0.14620

Basic model building version – 21

Model Analysis

Preprocessing Steps

1. Dropping irrelevant columns: 'patient_id', 'image_type', 'tbp_tile_type', 'attribution', 'copyright_license'.
2. Dropping train-only columns: 'lesion_id', 'iddx_full', 'iddx_1', 'iddx_2', etc.
3. Identifying categorical and numerical columns.
4. Applying transformations:
   - Numerical: KNNImputer for missing values, StandardScaler for normalization.
   - Categorical: OneHotEncoder for categorical encoding.

5. Aligning test data with training data features.

Class Imbalance Handling

1. Random undersampling applied to reduce majority class (sampling_strategy=0.005).
2. Random oversampling applied to increase minority class (sampling_strategy=0.4).
3. Final dataset size: 87,920 samples.

Model Architecture

Input Layers:

- Image Input: (224, 224, 3)
- Tabular Data Input: (72,)

Image Processing Branch:

- Backbone: ResNet-101 from TensorFlow Hub.
- Global Average Pooling followed by Flatten layer.
- Dense layers: 2048 units (Leaky ReLU).

Tabular Processing Branch:
- Dense layers: 128, 256, 512 units (Leaky ReLU) with dropout.

Fusion Layer:
- Concatenation of image and tabular branches.
- Fully connected layers: 1024 → 512 → 256 → 64 → 32 units (Leaky ReLU).
- Dropout applied at multiple stages.

Output Layer:
- Binary classification with a single neuron (Sigmoid activation).

Total Parameters: 50,272,961 (191.78 MB).
Trainable Parameters: 50,167,617 (191.37 MB).
Non-trainable Parameters: 105,344 (411.50 KB).

Model Parameters

Optimizer: Adam with PiecewiseConstantDecay learning rate schedule.
Learning Rate Schedule:
- Boundaries: [5000, 10000, 15000]
- Values: [1e-3, 5e-4, 1e-4, 1e-5]
- Loss Function: Binary Cross-Entropy.
- Evaluation Metrics:
- Precision-Recall AUC (PR_pAUC).
- ROC AUC (ROC_pAUC).
- Callbacks:
- EarlyStopping (patience=5, restore_best_weights=True, start_from_epoch=5).
- ModelCheckpoint (saving best model based on validation loss).

Model Metrics:

- After 12 epochs:

- PR_pAUC: 1.00
- ROC_pAUC: 1.00
- Loss: 1.3180e-06
- Training Time per Epoch: ~**658s** s per epoch (1374 steps).

Resultant scores:

- Private score: 0.12706
- Public score: 0.14285

Dataset EDA version 8:

Preprocessing Steps:

1. Dropping Irrelevant Columns: Removes ['patient_id', 'image_type', 'tbp_tile_type', 'attribution', 'copyright_license'].
2. Dropping Train-Only Columns: Removes ['lesion_id', 'iddx_full', 'iddx_1', 'iddx_2', 'iddx_3', 'iddx_4', 'iddx_5', 'mel_mitotic_index', 'mel_thick_mm', 'tbp_lv_dnn_lesion_confidence'].
3. Identifying Column Types:
    a. Categorical: object type columns (excluding isic_id).
    b. Numerical: int64 and float64 type columns (excluding target).
4. Preprocessing Pipelines:
    a. Numerical Features:
        i. KNNImputer (for missing values).
        ii. StandardScaler (for feature scaling).
    b. Categorical Features:
        i. OneHotEncoder (ignores unknown categories).
5. Column Transformation: Uses ColumnTransformer to apply numerical and categorical pipelines.
6. Test Set Processing: Aligns test dataset with train columns to maintain consistency.

Class Imbalance Handling:

1. Under-Sampling: Reduces the majority class to 5% of its original count using RandomUnderSampler(sampling_strategy=0.05).
2. Over-Sampling: Increases the minority class count to 10% using RandomOverSampler(sampling_strategy=0.1).
3. Final Data Shape:
    a. X_final shape: (8646, 73)
    b. Y_final shape: (8646,)
4. Class Weights Calculation:

a. Majority class (0): Weight = 1.1
b. Minority class (1): Weight = 11.0

Model Architecture (XGBoost):

- Objective: binary:logistic (Binary classification).
- Evaluation Metric: logloss.
- Scale Positive Weight: class_weights[1] / class_weights[0].
- Hyperparameters:
  - max_depth = 10
  - eta = 0.05 (Learning rate).
  - subsample = 0.8 (Row subsampling).
  - colsample_bytree = 0.8 (Feature subsampling).
  - random_state = 63
- Cross-validation:
  - 5-fold cross-validation.
  - Early stopping after 10 rounds.
  - num_boost_rounds = 86 (Best number of boosting rounds based on log-loss).

Model Performance (Log Loss):

- Train Set (Best Log Loss): 0.039354
- Test Set (Best Log Loss): 0.165273
- Cross-validation Results:
  - Train Log Loss decreased steadily.
  - Test Log Loss stabilized around 0.165 at num_boost_rounds = 86.

Resultant scores:

- Private score: 0.13121
- Public score: 0.15839

Basic model version 24:

Preprocessing Steps:

1. Dropping Irrelevant Columns
   a. Columns removed: ['patient_id', 'image_type', 'tbp_tile_type', 'attribution', 'copyright_license']
2. Dropping Train-Only Columns

    a. Columns removed: ['lesion_id', 'iddx_full', 'iddx_1', 'iddx_2', 'iddx_3', 'iddx_4', 'iddx_5', 'mel_mitotic_index', 'mel_thick_mm', 'tbp_lv_dnn_lesion_confidence']

3. Identifying Feature Types
    a. Categorical columns: Identified from object dtype
    b. Numerical columns: Identified from int64 & float64 dtypes
4. Pipeline Construction
    a. Numerical Pipeline: KNNImputer → StandardScaler
    b. Categorical Pipeline: OneHotEncoder(handle_unknown="ignore")
5. Dataset Alignment
    a. Train-test column alignment ensures consistency in features

Class Imbalance Handling:

1. Random Undersampling:
    a. RandomUnderSampler(sampling_strategy=0.05) to reduce majority class
2. Random Oversampling:
    a. RandomOverSampler(sampling_strategy=0.2) to increase minority class
3. Final dataset shape after resampling:
    a. X_train_final shape: (9432, 73)
    b. Y_train_final shape: (9432,)
4. Class Weights Computed for Training:
    a. {0: 0.6000, 1: 2.9988} (to balance class representation)

Model Architecture (Functional API):

Inputs:

- Image Input: (224, 224, 3) (Processed via ResNet-50 backbone)
- Tabular Input: (72,) features

Image Processing Branch:

1. ResNet-50 Pretrained Backbone
2. Flatten() layer
3. Dense(4096, activation='leaky_relu')

Tabular Data Processing Branch:

1. Dense(128, activation='leaky_relu') → Dropout(0.2)
2. Dense(256, activation='leaky_relu') → Dropout(0.2)

Fusion and Fully Connected Layers:

1. Concatenation of Image & Tabular Features
2. Dense(1024, activation='leaky_relu') → Dropout(0.2)
3. Dense(512, activation='leaky_relu') → Dropout(0.2)

4. Dense(256, activation='leaky_relu')
5. Dense(64, activation='leaky_relu') → Dropout(0.2)
6. Dense(32, activation='leaky_relu')
7. Output Layer: Dense(1, activation='sigmoid') (Binary classification)

Model Parameters:

- Total Parameters: 439,781,569 (1.64 GB)
- Trainable Parameters: 439,728,449
- Non-trainable Parameters: 53,120

Model Compilation & Training:

Optimizer:

- Adam with Piecewise Constant Decay:
    - Learning Rate Schedule: [1e-3 → 1e-4 → 1e-5] (based on training step boundaries [300, 750])

Loss Function:

- Binary Crossentropy Loss

Metrics:

- Precision-Recall AUC (PR_pAUC)
- ROC AUC (ROC_pAUC)

Training Setup:

- Epochs: 40
- Early Stopping: Patience = 3, Restore Best Weights = True
- Checkpoint Saving: Best model saved as 'best_model.keras'
- Class Weights Applied: {0: 0.6000, 1: 2.9988}

Model Performance (Epoch 10 Snapshot):

- Loss: 0.0079
- PR AUC (PR_pAUC): 0.9987
- ROC AUC (ROC_pAUC): 0.9997

Resultant scores:

- Private score: 0.12247
- Public score: 0.14666

Basic model version 26:

Dataset Information

- Dataset Size:

    - Train: 33,012 samples
    - Validation: 8,253 samples (20% split from 41,265 resampled dataset)
- Feature Dimensions:
    - Image Features: (224, 224, 3)
    - Tabular Features: 72

Preprocessing Steps & Techniques

- Tabular Data Processing:
    - Missing values handled using KNNImputer
    - StandardScaler applied to numerical features
    - OneHotEncoder applied to categorical features
    - Columns removed: 'patient_id', 'image_type', 'tbp_tile_type', 'attribution', 'copyright_license'
    - Train-only columns removed: 'lesion_id', 'iddx_full', 'mel_mitotic_index', etc.
- Image Processing:
    - Images loaded from HDF5 byte strings
    - Resized to 224x224
    - Augmentation: random flip, brightness adjustment, contrast adjustment
    - Normalized to range [0,1]

Class Imbalance Handling

- Resampling Strategy:
    - Random UnderSampling: Majority class reduced to 1%
    - Random OverSampling: Minority class increased to 5%
    - Final Balanced Distribution:
        - Class 0: 31,440 samples
        - Class 1: 1,572 samples

Model Architecture:

- **Multimodal Model using Functional API**
- **Image Input:** (224, 224, 3), processed using DenseNet-169 backbone, followed by:
    - GlobalAveragePooling2D
    - Flatten
    - Fully connected layers: 2048 → 1024 (with SELU activation, BatchNormalization, Dropout)
- **Tabular Input:** (72,), processed through:

    - Dense layers: 128 → 256 → 512 (with SELU activation, BatchNormalization, Dropout)

- **Fusion & Classification:**
    - Concatenation of image and tabular features
    - Fully connected layers: 1024 → 512 → 256 → 64 → 32
    - Output: 1 neuron with sigmoid activation (for binary classification)

Model Parameters Count:

- **Total Parameters:** 20,594,881
- **Trainable Parameters:** 20,425,793
- **Non-Trainable Parameters:** 169,088

Training Techniques:

- **Optimizer:** AdamW with PiecewiseConstantDecay learning rate schedule (1e-3 → 1e-4 → 1e-5)
- **Loss Function:** BinaryCrossentropy
- **Metrics:** AUC-PR (Precision-Recall AUC), AUC-ROC
- **Class Weighting:** {0: 0.8, 1: 2.5} (to address class imbalance)
- **Early Stopping:**
    - Monitors val_loss
    - patience=5
    - restore_best_weights=True
- **Model Checkpointing:** Saves best model based on val_loss

Training Metrics and Results (Final Epoch 30/30):

- **Train:**
    - PR_pAUC: 0.9152
    - ROC_pAUC: 0.9943
    - Loss: 0.0651
- **Validation:**
    - PR_pAUC: 0.8306
    - ROC_pAUC: 0.9853
    - Loss: 0.0848

Final Inference:

- The model achieved strong **ROC-AUC (0.9943 on train, 0.9853 on validation)** and **PR-AUC (0.9152 on train, 0.8306 on validation)**, indicating robust classification performance.
- Low validation loss (0.0848) suggests good generalization.

- The use of **DenseNet-169 as the image feature extractor** and **SELU activation** in fully connected layers contributes to the model's strong performance.

Resultant scores:

- Private score: 0.05751
- Public score: 0.07654

Basic model version 27:

Dataset Size

- **Training set (after resampling and splitting):**
  - Before splitting: **5502 samples**
  - After train-validation split (80-20): **4401 training samples, 1101 validation samples**
- **Feature dimensions:**
  - **Tabular features:** 72
  - **Image features:** 224 × 224 × 3 (RGB images)
  - **Target variable:** Binary classification (0 or 1)

Preprocessing Steps & Techniques

- **Handling missing values:**
  - **Numerical features:** KNNImputer()
  - **Categorical features:** OneHotEncoder(handle_unknown="ignore")
- **Feature scaling:**
  - **Numerical:** StandardScaler()
- **Feature encoding:**
  - **Categorical:** OneHotEncoder
- **Dropped Irrelevant Columns:**
  - 'patient_id', 'image_type', 'tbp_tile_type', 'attribution', 'copyright_license'
- **Dropped Train-Only Columns:**
  - 'lesion_id', 'iddx_full', 'iddx_1', 'iddx_2', 'iddx_3', 'iddx_4', 'iddx_5', 'mel_mitotic_index', 'mel_thick_mm', 'tbp_lv_dnn_lesion_confidence'
- **Image Preprocessing:**

- Images loaded from HDF5 byte strings.
- Resized to **224×224**.
- Augmentations applied:
  - Random horizontal flip
  - Random brightness adjustment (max delta = **0.2**)
  - Random contrast adjustment (range: **0.8 - 1.2**)
  - Normalization (scaled to **[0,1]**)

Class Imbalance Handling

- **Undersampling:** RandomUnderSampler(sampling_strategy=0.1)
- **Oversampling:** RandomOverSampler(sampling_strategy=0.4)

Final Dataset Structure

- **Batch size:** 64
- **Training batch sample dimensions:**
  - **Images:** (64, 224, 224, 3)
  - **Tabular features:** (64, 72)
  - **Labels:** (64, 1)

Model Architecture

- **Type:** Multimodal deep learning model combining image and tabular data.
- **Backbone:** DenseNet-169 (pretrained on ImageNet).
- **Input:**
  - **Image Branch:** 224x224x3 images.
  - **Tabular Branch:** 72 tabular features.
- **Image Branch Processing:**
  - DenseNet-169 backbone.
  - Global Average Pooling.
  - Dense layers: 2048 → 1024 (SELU activation, Dropout, BatchNorm).
- **Tabular Branch Processing:**
  - Dense layers: 128 → 256 → 512 (SELU activation, Dropout, BatchNorm).
- **Fusion & Final Layers:**
  - Concatenation of both branches.
  - Fully connected layers: 1024 → 512 → 256 → 64 → 32.
  - Output: 1 neuron (sigmoid activation for binary classification).

Model Parameters Count

- **Total Parameters:** 20,580,801 (~78.51 MB).
- **Trainable Parameters:** 20,418,753 (~77.89 MB).
- **Non-trainable Parameters:** 162,048 (~633 KB).

Training Techniques

- **Optimizer:** AdamW with a piecewise constant learning rate schedule.
- **Loss Function:** Binary Crossentropy.
- **Metrics:**
    - Precision-Recall AUC (PR_pAUC).
    - ROC AUC (ROC_pAUC).
- **Class Weights for Imbalance Handling:**
    - Class 0: 0.6999
    - Class 1: 1.7506
- **Callbacks:**
    - Early stopping (patience: 5, starts from epoch 3).
    - Model checkpointing (saving the best model).

Training Metrics & Results

- **Epoch 23:**
    - **Training:**
        - PR_pAUC: **0.9944**
        - ROC_pAUC: **0.9970**
        - Loss: **0.0524**
    - **Validation:**
        - PR_pAUC: **0.9676**
        - ROC_pAUC: **0.9892**
        - Loss: **0.1327**

Final Inference

- The model exhibits strong AUC scores on both training and validation data.
- Some overfitting may be present (train loss: 0.0524, val loss: 0.1327).

Resultant scores:

- Private score: 0.11759
- Public score: 0.14307

Basic model version 30:

Dataset Size

- **Training Dataset (After Resampling and Splitting)**:
    - **Before splitting**: 11,004 samples (from class balancing step)
    - **After splitting**:
        - **Training set**: 8,803 samples
        - **Validation set**: 2,201 samples
- **Feature Dimensions**:
    - **Tabular Features**: 72
    - **Image Features**: (224, 224, 3)
    - **Target (Labels)**: Binary (0 or 1)

Preprocessing Steps and Techniques

1. **Column Selection**:
    a. Dropped irrelevant columns: patient_id, image_type, tbp_tile_type, attribution, copyright_license
    b. Dropped train-only columns: lesion_id, iddx_full, iddx_1, iddx_2, iddx_3, iddx_4, iddx_5, mel_mitotic_index, mel_thick_mm, tbp_lv_dnn_lesion_confidence
2. **Data Transformation**:
    a. **Numerical Features**:
        i. **Imputation**: KNNImputer
        ii. **Scaling**: StandardScaler
    b. **Categorical Features**:
        i. **Encoding**: OneHotEncoder(handle_unknown="ignore")
3. **Image Processing**:
    a. Decoding JPEG images
    b. Resizing to (224, 224)
    c. Augmentation:
        i. Random horizontal flip
        ii. Random brightness adjustment (max_delta=0.2)
        iii. Random contrast adjustment (range 0.8 - 1.2)
    d. Normalization: Scaling pixel values to [0,1]

Class Imbalance Handling

- **Undersampling**: RandomUnderSampler(sampling_strategy=0.05)
    - Reduces the majority class to 5% of the original dataset.

- **Oversampling**: RandomOverSampler(sampling_strategy=0.4)
    - Increases the minority class to 40% of the majority class.

Final Class Distribution After Resampling:

- Training: **Class 0 → 6,288 samples, Class 1 → 2,515 samples**
- Validation: **Class 0 → 1,572 samples, Class 1 → 629 samples**

Dataset Pipeline Output

- **Batch Size**: 64
- **Image Data Shape**: (64, 224, 224, 3)
- **Tabular Data Shape**: (64, 72)
- **Label Shape**: (64, 1)

Model Architecture

- **Input Layers:**
    - **Images**: (224, 224, 3) (Processed through a DenseNet-169 backbone)
    - **Tabular Data**: (72,)
- **Image Processing Branch:**
    - **Backbone Model**: DenseNet-169 (pretrained on ImageNet)
    - **Global Average Pooling**
    - **Flatten Layer**
    - **Fully Connected Layers**:
        - Dense (2048) → ReLU → Dropout(0.3)
        - Dense (1024) → ReLU → Dropout(0.3) → BatchNormalization
- **Tabular Data Processing Branch:**
    - Dense (128) → ReLU → Dropout(0.3)
    - Dense (256) → ReLU → Dropout(0.3)
    - Dense (512) → ReLU → BatchNormalization
- **Concatenation & Fully Connected Layers:**
    - Concatenation of Image and Tabular feature branches
    - Dense (1024) → ReLU → Dropout(0.3)
    - Dense (512) → ReLU → Dropout(0.3)
    - Dense (256) → ReLU → BatchNormalization
    - Dense (64) → ReLU → Dropout(0.3)
    - Dense (32) → ReLU → BatchNormalization

- **Output Layer**: Dense (1) → Sigmoid (for binary classification)

Model Parameters Count

- **Total Parameters**: 20,580,801
- **Trainable Parameters**: 20,418,753
- **Non-trainable Parameters**: 162,048

Training Techniques

- **Activation Functions**: ReLU (for intermediate layers), Sigmoid (for final output)
- **Kernel Initializer**: HeNormal()
- **Regularization Methods**:
  - Dropout (0.3) in multiple layers
  - BatchNormalization in key layers
- **Pretrained Backbone**: DenseNet-169 (Imagenet)

Training metrics and loss:

- **Training PR pAUC:** 0.9269
- **Training ROC pAUC:** 0.9690
- **Validation PR pAUC:** 0.9215
- **Validation ROC pAUC:** 0.9678
- **Training loss:** 0.2055
- **Validation loss:** 0.2557

Observations:

1. **Performance:**
   a. The PR and ROC pAUC scores are high, suggesting strong classification performance.
   b. The validation scores are slightly lower than the training scores, but the difference is small, indicating the model is not overfitting significantly.
2. **Loss:**
   a. The training loss (0.2055) is lower than the validation loss (0.2557).

Resultant scores:

- Private score: 0.07012
- Public score: 0.11236

CNN based models version 31:

Dataset Information:

- **Dataset Size**:
  - Training data (X_train_final): 49,518 samples with 73 features.
  - Training labels (y_train_final): 49,518 samples.
  - Test data (X_test): Not explicitly mentioned, but the test_metadata was processed similarly to the training data.
- **Feature Dimensions**:
  - 73 features (after preprocessing) in the training dataset.
  - The features consist of both numerical and categorical data, which are processed separately using pipelines.
- **Preprocessing Steps**:
  - **Dropping Irrelevant Columns**: The code drops irrelevant columns like patient_id, image_type, attribution, and others.
  - **Imputation**:
    - Numerical columns are imputed using KNNImputer (nearest neighbors method).
    - Categorical columns are handled using OneHotEncoder.
  - **Scaling**: Numerical columns are scaled using StandardScaler.
  - **One-Hot Encoding**: Categorical columns are encoded using OneHotEncoder.
  - **Column Alignment**: Ensures that test data has the same columns as the train data.
- **Class Imbalance Handling Techniques**:
  - **Undersampling**: The code uses RandomUnderSampler from imblearn to undersample the majority class.
  - **Oversampling**: After undersampling, RandomOverSampler from imblearn is applied to oversample the minority class.
  - **Stratified Splitting**: In the train-test split, the stratify parameter is used to ensure the distribution of classes in both training and validation datasets remains similar.
- **Class Imbalance Parameters**:
  - **Undersampling**: sampling_strategy=0.01 (undersample to keep only 1% of the majority class).
  - **Oversampling**: sampling_strategy=0.4 (oversample the minority class to reach 40% of the majority class size).

Additional Information:

- **Image Preprocessing**:
  - Images are decoded from byte strings and resized to a fixed height (224) and width (224).

- Augmentation is performed with random left-right flips, brightness, and contrast adjustments.
- The images are normalized to a [0, 1] range after the preprocessing steps.

- **TensorFlow Dataset**:
  - The dataset is converted into a TensorFlow Dataset using tf.data.Dataset.from_tensor_slices().
  - A preprocessing function is applied to the images using the .map() method with parallel processing (tf.data.AUTOTUNE).

Model Architecture:

The model is a **multimodal model** combining image data (processed with DenseNet backbone) and tabular data.

1. **Input Layers:**
   a. images: Shape (None, 224, 224, 3) (64 images of size 224x224 with 3 color channels).
   b. tabular: Shape (None, 72) (64 tabular features).
2. **Image Branch (DenseNet Backbone):**
   a. Backbone model: DenseNet-169 pretrained on ImageNet.
   b. After the backbone, the model uses:
      i. GlobalAveragePooling2D layer.
      ii. Flatten layer.
      iii. A series of fully connected layers (Dense with ReLU activations) and Dropout for regularization.
3. **Tabular Data Branch:**
   a. Several Dense layers for tabular data processing, followed by Dropout for regularization.
4. **Concatenation:**
   a. The features from the image and tabular branches are concatenated.
5. **Fully Connected Layers (after concatenation):**
   a. Multiple Dense layers (with ReLU activations) and Dropout for regularization.
   b. The output layer is a **single neuron** with a **sigmoid activation** (for binary classification).

Total Parameters:

- **Total params**: 20,580,801 (78.51 MB)
- **Trainable params**: 20,418,753 (77.89 MB)
- **Non-trainable params**: 162,048 (633.00 KB)

Training Techniques:

1. **Optimizer:**

a. Adam optimizer with a learning rate schedule defined by PiecewiseConstantDecay. The learning rate changes at epoch boundaries: [400, 1500] with values [1e-3, 1e-4, 1e-5].

2. **Loss Function:**
    a. Binary Crossentropy for binary classification.
3. **Metrics:**
    a. **PR AUC (Precision-Recall AUC)**.
    b. **ROC AUC (Receiver Operating Characteristic AUC)**.
4. **Callbacks:**
    a. **EarlyStopping**: Monitors val_loss, restores the best weights after 4 epochs of no improvement.
    b. **ModelCheckpoint**: Saves the best model based on val_loss.
5. **Class Weights:**
    a. The class weights are computed using compute_class_weight to handle class imbalance, with weights {0: 0.7, 1: 1.75}.

Training Results (Epoch 21/75):

- **Training Metrics:**
    - PR pAUC: 0.9956
    - ROC pAUC: 0.9982
    - Loss: 0.0427
- **Validation Metrics:**
    - PR pAUC: 0.9939
    - ROC pAUC: 0.9986
    - Loss: 0.0345

Final Inference:

- The model is performing exceptionally well, with very high **PR AUC** and **ROC AUC** values, suggesting excellent classification performance.
- The loss values are low, and the model has been regularized effectively using **Dropout** layers.
- The **early stopping** and **checkpointing** mechanisms are in place to prevent overfitting and save the best model.

Resultant scores:

- Private score: 0.00947
- Public score: 0.00944

CNN based models version 32:

Dataset Size

- **Training data**: 5109 samples (after class imbalance handling and resampling).
- **Testing data**: Not explicitly stated, but it can be inferred from the test metadata loaded during the submission.

Feature Dimensions

- **Image features**: Each image is resized to (224, 224, 3) (Height x Width x Channels).
- **Tabular data features**: 72 features after preprocessing (numerical and categorical combined).
- **Labels**: 1-dimensional array, binary labels (0 or 1), representing the target.

Preprocessing Steps and Techniques

1. **Data Preprocessing for Tabular Data**:
    a. Drop irrelevant and train-only columns.
    b. Identify categorical and numerical columns.
    c. **Numerical Pipeline**:
        i. Impute missing values using KNN imputation.
        ii. Standardize numerical values using StandardScaler.
    d. **Categorical Pipeline**:
        i. One-hot encode categorical columns.
    e. Combine numerical and categorical transformations using ColumnTransformer.
2. **Image Preprocessing**:
    a. Decode the byte string into an image.
    b. Resize images to (224, 224).
    c. Apply random image augmentations:
        i. Random horizontal flip.
        ii. Random brightness and contrast adjustments.
    d. Normalize pixel values to the [0, 1] range.
3. **Class Imbalance Handling**:
    a. **Undersampling**: Applied to the majority class with a target sampling strategy of 0.1.
    b. **Oversampling**: Applied to the minority class with a target sampling strategy of 0.3.

Class Imbalance Handling Techniques and Parameters

- **Undersampling (RandomUnderSampler)**:
    - Sampling strategy: 0.1 (reduces majority class to 10% of the total dataset size).
- **Oversampling (RandomOverSampler)**:
    - Sampling strategy: 0.3 (increases minority class to 30% of the total dataset size).

Additional Details

- **Dataset structure after preprocessing**:
    - **Image batch shape**: (64, 224, 224, 3)
    - **Tabular data shape**: (64, 72)
    - **Labels shape**: (64, 1)
- **Data types**:
    - **Images**: float32
    - **Tabular data**: float32
    - **Labels**: int32

Model Architecture:

This is a **multimodal neural network** combining image and tabular data using a DenseNet backbone for the image branch and dense layers for the tabular data branch. The model uses a **Functional API** approach for building the architecture.

1. **Image Branch:**
    a. **Input:** Image with shape (224, 224, 3)
    b. **Backbone:** DenseNet-169 (from TensorFlow Hub)
    c. **GlobalAveragePooling2D:** Reduces the feature map from the backbone to a 1D vector.
    d. **Flatten:** Converts the pooled 2D features into a 1D vector.
    e. **Dense Layers:** Fully connected layers with ReLU activations, followed by batch normalization, dropout, and a final output layer.
2. **Tabular Data Branch:**
    a. **Input:** Tabular data with shape (72,)
    b. **Dense Layers:** A sequence of dense layers with ReLU activations, dropout for regularization, and batch normalization.
3. **Concatenation:**
    a. The outputs of the image and tabular branches are concatenated and passed through additional fully connected layers.
4. **Output Layer:**
    a. A single neuron with a **sigmoid activation** for binary classification.

Model Parameters Count:

- **Total Parameters:** 15,712,513 (approximately 59.94 MB)
- **Trainable Parameters:** 15,553,089 (approximately 59.33 MB)
- **Non-Trainable Parameters:** 159,424 (approximately 622.75 KB)

Training Techniques:

- **Optimizer:** Adam optimizer with a **PiecewiseConstantDecay** learning rate schedule.

- **Loss Function:** Binary Crossentropy (for binary classification).
- **Metrics:**
    - **PR_pAUC (Precision-Recall AUC)**
    - **ROC_pAUC (ROC AUC)**
- **Callbacks:**
    - **EarlyStopping:** Monitors the validation loss and stops training after 4 epochs without improvement, restoring the best weights.
    - **ModelCheckpoint:** Saves the best model based on validation loss.
- **Class Weights:** Used to handle imbalanced classes, with a class weight dictionary {0: 0.8, 1: 2.5}.

Training Metrics and Results (Epoch 24 to 28):

- **Epoch 24:**
    - **PR_pAUC:** 0.9824
    - **ROC_pAUC:** 0.9956
    - **Loss:** 0.0964
    - **Validation PR_pAUC:** 0.9571
    - **Validation ROC_pAUC:** 0.9913
    - **Validation Loss:** 0.0996

Final Inference:

- The model appears to perform well, with **PR_AUC** and **ROC_AUC** scores consistently high on both the training and validation sets. The **loss** fluctuates slightly, but the validation metrics show the model generalizing well across epochs.
- The training also uses **class weights** to address class imbalance, and **early stopping** ensures the model doesn't overfit. The best model is saved based on the lowest validation loss.

Resultant scores:

- Private score: 0.13614
- Public score: 0.15238

Tree based models version – 9

1. Dataset Size:

- Training dataset (X_train): 10218 samples, 73 features (after resampling).
- Training labels (y_train): 10218 samples.

The exact size of the test dataset (X_test) is not mentioned in the output, but it is processed similarly.

2. Feature Dimensions:

- After preprocessing, the training data has 73 features.
- These include both numerical and categorical features.

3. Preprocessing Steps and Techniques Used:

- Dropping irrelevant columns: Columns like patient_id, image_type, etc., are dropped.
- Dropping columns specific to the training set: Columns that only appear in the training dataset, such as lesion_id and iddx_*, are dropped.
- Identifying column types:
    - Categorical columns are identified using object data type.
    - Numerical columns are identified using int64 or float64 data types.
- Imputation of missing values:
    - Numerical data is imputed using KNNImputer.
- Scaling of numerical data: The numerical features are scaled using StandardScaler.
- One-hot encoding for categorical features: Categorical features are transformed using OneHotEncoder.
- Column transformation: ColumnTransformer is used to apply the above transformations selectively to the numerical and categorical columns.

4. Class Imbalance Handling Techniques:

- Undersampling: Applied to reduce the number of samples in the majority class, using a sampling strategy of 0.05 (5% of the original number of majority class samples).
- Oversampling: Applied to increase the number of samples in the minority class, using a sampling strategy of 0.3 (increasing the minority class to 30% of the total samples).

The following resampling methods are used:

- RandomUnderSampler (for undersampling the majority class).
- RandomOverSampler (for oversampling the minority class).

5. Resampling Parameters:

- Undersampling strategy: sampling_strategy=0.05
- Oversampling strategy: sampling_strategy=0.3

6. Other Information:

- Train Columns: After preprocessing, the training dataset has columns based on both numerical and categorical features, along with isic_id and target.
- Final shapes after resampling:
    - X_train_final.shape: (10218, 73)
    - y_train_final.shape: (10218,)

7. Model Architecture:

The code trains three different models:

- XGBoost (binary classification with logistic regression objective)
- LightGBM (binary classification)
- CatBoost (binary classification)

XGBoost:

- Objective: binary:logistic
- Evaluation metric: logloss
- Parameters:
    - max_depth=10
    - eta=0.01
    - subsample=0.9
    - colsample_bytree=0.9
    - scale_pos_weight=class_weights[1] / class_weights[0]
    - random_state=63

LightGBM:

- Objective: binary
- Evaluation metric: binary_logloss
- Parameters:
    - eta=0.01
    - num_iterations=10000
    - num_leaves=63
    - random_state=42

CatBoost:

- Objective: binary classification with Logloss loss function
- Parameters:
    - iterations=10000
    - depth=6
    - learning_rate=0.01
    - random_seed=42

8. Model Parameters Count:

- XGBoost model uses hyperparameters like max_depth=10, eta=0.01, subsample=0.9, and others, but the exact parameter count is not provided in the code.
- LightGBM model uses parameters like num_iterations=10000, num_leaves=63, but the exact parameter count is not provided.
- CatBoost model uses parameters like iterations=10000, depth=6, and learning_rate=0.01.

9. Training Techniques:

- Cross-validation for XGBoost:
    - The model is trained using 5-fold cross-validation with early stopping at 10 rounds to prevent overfitting.
    - Logloss is used as the evaluation metric during training.
- LightGBM is trained with the train function for a specified number of iterations (10,000), and the evaluation metric is also binary_logloss.
- CatBoost is trained using the fit function, specifying a loss function of Logloss and applying sample weights for class imbalance handling.

10. Training Metrics and Results:

- XGBoost Cross-validation Results:
    - Logloss results over the training rounds:
        - Train Logloss starts at 0.8178 and decreases to around 0.0021 after 1271 boosting rounds.
        - Test Logloss starts at 0.8186 and decreases to around 0.0344 by the final boosting round.

11. Final Inference:

- Final Prediction: The final prediction is calculated by averaging the predictions from the three models:
    - final_pred = np.mean([xgb_pred(dtest), lgb_pred(X_test), cb_pred(X_test)], axis=0)
    - Resulting final predictions: [0.00030571, 0.00017759, 0.00169694]

Resultant scores:

- Private score: 0.14565
- Public score: 0.16540

Tree based models version – 10

1. Dataset Size:

- Training dataset (X_train):
    - After preprocessing, the final shape of the training data (X_train_final) is (79386, 73) with 79,386 samples and 73 features.
- Training labels (y_train):
    - The shape of the final labels (y_train_final) is (79386,) with 79,386 target values.
- Test dataset (X_test):
    - The shape of the test dataset (X_test) is not explicitly mentioned in the output, but it is transformed to match the feature columns of the training dataset.

2. Feature Dimensions:

- After preprocessing, the features are transformed, and the final dataset X_train_final has 73 features.
- The categorical features undergo OneHotEncoding, which increases the feature space by creating new binary columns for each unique value in the categorical features.

3. Preprocessing Steps and Techniques Used:

- Irrelevant Columns Dropped:
    - Columns like 'patient_id', 'image_type', 'tbp_tile_type', 'attribution', and 'copyright_license' are dropped.
- Train-only Columns Dropped:
    - Columns present only in the training set (e.g., 'lesion_id', 'iddx_full', 'mel_mitotic_index', etc.) are removed.
- Feature Identification:
    - Categorical columns are identified as columns with data type object.
    - Numerical columns are identified as columns with data types int64 or float64, excluding the 'target' column.
- Data Transformation Pipelines:
    - Numerical Pipeline: Imputes missing values using KNNImputer and then scales the features using StandardScaler.
    - Categorical Pipeline: Encodes categorical features using OneHotEncoder (ignoring unknown categories).
    - The processed data is stored in a ColumnTransformer which applies these transformations to the respective columns.
- Feature Alignment:
    - The test data is aligned with the training data columns to ensure consistency between train and test datasets.

4. Class Imbalance Handling Techniques and Parameters:

- Resampling Strategy:
    - Random Undersampling is applied to the numerical data with a sampling strategy of 0.005, meaning the majority class is undersampled to only 0.5% of its original size.
    - Random Oversampling is then applied to the minority class with a sampling strategy of 0.01, meaning the minority class is oversampled to 1% of its original size.
- Resampling Function (resampler_data):
    - The function applies undersampling first and then oversampling on the dataset to handle class imbalance.

5. Model Architecture:

Three models are used for prediction:

- **XGBoost Model:**

- Objective: 'binary:logistic'
- Evaluation Metric: 'logloss'
- Key Parameters:
  - max_depth: 10
  - eta: 0.01
  - subsample: 0.9
  - colsample_bytree: 0.9
  - scale_pos_weight: class imbalance adjustment
- The model is trained with **cross-validation** (xgb.cv) using **5-fold** cross-validation and **early stopping** after 10 rounds.

- **LightGBM Model:**
  - Objective: 'binary'
  - Evaluation Metric: 'binary_logloss'
  - Key Parameters:
    - num_iterations: 10000
    - num_leaves: 63
    - eta: 0.01
  - The model is trained using **LightGBM's train()** function with the specified parameters.

- **CatBoost Model:**
  - Loss Function: 'Logloss'
  - Key Parameters:
    - iterations: 10000
    - depth: 6
    - learning_rate: 0.01
  - The model is trained using **CatBoostClassifier** with **sample weights** based on class weights.

6. Training Techniques:

- **Cross-validation** is used to evaluate the performance of the **XGBoost model** with early stopping after 10 rounds.
- **Early Stopping:** The XGBoost model uses early_stopping_rounds=10 in cross-validation to halt training if the validation metric does not improve for 10 rounds.
- **Class Weights:** Class imbalance is handled by assigning higher weights to the minority class in all models using sample_weight or weight.

7. Training Metrics and Results:

- **Cross-validation Results (XGBoost):**

- Training Log Loss (mean) decreased from **1.10** to **0.08** over 502 rounds of training.
- Test Log Loss (mean) decreased from **1.10** to **0.22**.

- **Log Loss Metrics:**
  - The training metrics show that the XGBoost model improved significantly with the number of boosting rounds.
- **Best Boosting Rounds:**
  - The best number of boosting rounds (num_boost_rounds) is determined by the minimum **test-logloss-mean**, which is **501**.
- **LightGBM and CatBoost:**
  - LightGBM and CatBoost models are trained with similar strategies, and class imbalance is handled via **sample weights**.

8. Final Inference:

- **Ensemble of Predictions:**
  - The final prediction is an average of the predictions from **XGBoost**, **LightGBM**, and **CatBoost** models.

Resultant scores:

- Private score: 0.11055
- Public score: 0.12443

Tree based models version – 11

1**. Dataset Size:**

- Training Dataset Size (X_train): 102,180 samples with 73 features.
- Target (y_train): 102,180 labels.
- Test Dataset Size (X_test): Same number of features as the training dataset, but the exact sample size isn't explicitly mentioned here.

2. Feature Dimensions:

- After preprocessing, the X_train dataset has 73 features.
- The target (y_train) contains a single column, which is the target label.
- The features include both categorical and numerical types, processed separately using different techniques.

3. Preprocessing Steps and Techniques:

- Dropping Irrelevant Columns:

    - Columns like patient_id, image_type, tbp_tile_type, attribution, and copyright_license are removed from the dataset using _drop_irrelevant_columns().

- Dropping Train-only Columns:
    - Columns present only in the training dataset (such as lesion_id, iddx_full, etc.) are removed using _drop_train_only_columns().
- Categorical and Numerical Columns Identification:
    - The _identify_column_types() method is used to segregate columns into categorical and numerical types, excluding isic_id and target.
- Numerical Data Preprocessing:
    - Imputation: Missing values in numerical columns are handled using KNNImputer().
    - Scaling: Data is scaled using StandardScaler().
- Categorical Data Preprocessing:
    - One-Hot Encoding: Categorical columns are transformed using OneHotEncoder with handle_unknown='ignore' to handle unseen categories during prediction.
- Column Transformation:
    - A ColumnTransformer is used to apply the respective preprocessing pipelines to the numerical and categorical columns.
- Train/Test Column Alignment:
    - The columns of the test data are aligned with those of the training data using _align_train_test_columns() to ensure that both datasets have the same features.

4. Class Imbalance Handling Techniques:

- Undersampling and Oversampling:
    - Undersampling: The majority class is undersampled using RandomUnderSampler with a sampling strategy of 0.005 (i.e., retaining 0.5% of the minority class size).
    - Oversampling: After undersampling, oversampling is applied to the minority class using RandomOverSampler with a sampling strategy of 0.3 (i.e., increasing the minority class size to 30% of the original dataset).
- Sampling Strategies:
    - Undersampling Strategy: The majority class is downsampled to 0.5% of the dataset size.
    - Oversampling Strategy: The minority class is upsampled to 30% of the dataset size.
- Final Dataset Shape (After Resampling):
    - X_train_final: The final training features have 102,180 samples and 73 features.
    - y_train_final: The final target labels contain 102,180 samples.

5. Model Architecture:

- Ensemble Model: The architecture used is a Stacking Classifier with the following base models:

    - XGBoost: xgb.XGBClassifier with the parameters defined for XGBoost.
    - LightGBM: lgb.LGBMClassifier with the parameters defined for LightGBM.
    - CatBoost: cb.CatBoostClassifier with specified parameters.
- Final Estimator (Meta-model): A RandomForestClassifier is used as the final estimator to combine predictions from the base models.

6. Model Parameters Count:

- The parameters for the base models are as follows:
    - XGBoost (XGBClassifier):
        - objective: 'binary:logistic'
        - eval_metric: 'logloss'
        - scale_pos_weight: class weight ratio (calculated as class_weights[1] / class_weights[0])
        - max_depth: 10
        - eta: 0.01
        - subsample: 0.9
        - colsample_bytree: 0.9
        - random_state: 63
    - LightGBM (LGBMClassifier):
        - objective: 'binary'
        - metric: 'binary_logloss'
        - eta: 0.01
        - num_iterations: 10,000
        - num_leaves: 63
        - random_state: 42
        - verbose: -1
    - CatBoost (CatBoostClassifier):
        - iterations: 10,000
        - depth: 6
        - learning_rate: 0.01
        - loss_function: 'Logloss'
        - random_seed: 42
- Meta-model (RandomForestClassifier): No explicit parameters are defined in the code snippet, assuming default values.

8. Training Techniques:

- Stacking Classifier:

    - Base Models: XGBoost, LightGBM, and CatBoost.
    - Final Estimator: RandomForestClassifier.
- Stacking Method: The stack_method is set to 'predict_proba', meaning it uses the predicted probabilities from the base models to train the final estimator.
- Multithreading: The n_jobs=-1 parameter is used to run the training in parallel on all available processors.

9. Training Metrics and Results:

- Class Weights:
    - The computed class weights are {0: 1.3, 1: 4.33}, used to handle class imbalance during training.
- Model Training:
    - The base models (XGBoost, LightGBM, and CatBoost) are fitted using the respective parameters.
    - The final model is trained using a stacking classifier with these base models.
- Warning: There is a runtime warning due to potential conflicts with multithreading (os.fork() being called).

Resultant scores:

- Private score: 0.02104
- Public score: 0.02155

Tree based models version – 12

1. Dataset Size:

- Training dataset (after resampling):
    - Samples: 5,109
    - Features: 73

2. Feature Dimensions:

- Total Features: 73
- The features include both numerical and categorical variables (after preprocessing).

3. Preprocessing Steps and Techniques Used:

- Dropping Irrelevant Columns:
    - The following columns are removed: patient_id, image_type, tbp_tile_type, attribution, copyright_license
- Dropping Train-Only Columns:

- Features that exist only in the training dataset are removed:
  lesion_id, iddx_full, iddx_1, iddx_2, iddx_3, iddx_4, iddx_5, mel_mitotic_index, mel_thick_mm, tbp_lv_dnn_lesion_confidence

- Feature Engineering:
  - Identifying Column Types:
    - Categorical columns: Identified based on object data type.
    - Numerical columns: Identified based on int64 and float64 types.
  - Numerical Feature Processing:
    - Missing Value Imputation: KNNImputer() (neighbors-based imputation instead of the standard mean/median)
    - Feature Scaling: StandardScaler()
  - Categorical Feature Processing:
    - One-Hot Encoding: OneHotEncoder(handle_unknown="ignore")
- Column Transformer:
  - Combines numerical and categorical preprocessing steps into a unified transformation.
- Ensuring Consistency Between Train and Test Sets:
  - Missing columns in the test set are filled with 0 to align with the train set.

4. Class Imbalance Handling Techniques and Parameters:

- Random Undersampling (Applied First):
  - Technique: RandomUnderSampler(sampling_strategy=0.1)
  - Purpose: Reduces the majority class so that the minority class constitutes at least 10% of the total dataset.
- Random Oversampling (Applied Next):
  - Technique: RandomOverSampler(sampling_strategy=0.3)
  - Purpose: Increases the size of the minority class so that it makes up 30% of the dataset after undersampling.

5. Model Architecture:

The notebook implements an ensemble stacking model consisting of three base models:

1. XGBoost Classifier (XGBClassifier)
2. LightGBM Classifier (LGBMClassifier)
3. CatBoost Classifier (CatBoostClassifier)
4. Final Estimator: Logistic Regression (LogisticRegression) to combine predictions.

Each of the base models predicts probabilities, and the final stacking classifier uses logistic regression to make the final prediction.

6. Model Parameters Count:

The total number of parameters is not explicitly mentioned, but based on the architecture:

- XGBoost (xgb.XGBClassifier) Parameters:
    - max_depth=10
    - eta=0.01
    - subsample=0.9
    - colsample_bytree=0.9
    - num_boost_round (selected based on cross-validation)
    - scale_pos_weight=4.333/1.3 ≈ 3.33
- LightGBM (lgb.LGBMClassifier) Parameters:
    - num_leaves=63
    - eta=0.01
    - num_iterations=10,000
    - random_state=42
- CatBoost (cb.CatBoostClassifier) Parameters:
    - iterations=10,000
    - depth=10
    - learning_rate=0.01
    - loss_function='Logloss'
    - random_seed=42

Since all models use gradient boosting techniques, the parameter count will be dependent on the number of boosting rounds and tree depth.

7. Training Techniques:

- Class Weighting:
    - Applied to handle class imbalance using scale_pos_weight for XGBoost and sample_weight for LightGBM and CatBoost.
    - Class weights calculated as:
      {0: 1.3, 1: 4.33}
- Gradient Boosting Training:
    - XGBoost Training:
        - Cross-validation: 5-fold (nfold=5)
        - Early Stopping: early_stopping_rounds=10
        - Evaluation Metric: Log loss (logloss)
    - LightGBM Training:

        - 10000 boosting iterations
    - CatBoost Training:
        - 10000 iterations
        - Depth = 10
- Ensemble Learning:
    - Stacking Classifier is used to combine predictions from XGBoost, LightGBM, and CatBoost, with logistic regression as the meta-classifier.
    - Parallel Processing using n_jobs=-1

8. Training Metrics and Results:

- Evaluation Metric: Binary Log Loss (logloss)
- Cross-Validation Results for XGBoost:
    - The model was trained using xgb.cv, and the optimal number of boosting rounds was determined based on the lowest log loss.
    - The exact log loss values are not shown in the output.

Resultant scores:

- Private score: 0.13445
- Public score: 0.15628

Vision transformers version – 2

Dataset Size

- Training dataset before resampling: (355,468, 73)
- Training dataset after resampling: (394,965, 73)
- Validation dataset (split from resampled data): 10% of the final training data
- Batch size: 256

Feature Dimensions

- Image features: (72, 72, 3)
- Tabular features: 72 numerical/categorical transformed columns
- Labels: (1) (binary classification)

Preprocessing Steps and Techniques

- Tabular Data:
    - Dropped irrelevant columns (patient_id, image_type, etc.)
    - Dropped train-only columns (lesion_id, mel_mitotic_index, etc.)
    - Identified categorical and numerical columns
    - Numerical processing:
        - Missing values imputed using KNN Imputer

        - Standardized using StandardScaler
    - Categorical processing:
        - One-hot encoding (ignoring unknown categories)
    - Data alignment between train and test sets
- Image Data:
    - Decoded from JPEG format
    - Resized to (72, 72)
    - Applied data augmentation:
        - Random horizontal flip
        - Random brightness adjustment
        - Random contrast adjustment
    - Normalized to [0,1] range

Class Imbalance Handling Techniques

1. Random Undersampling (with sampling_strategy=0.001)
    a. Reduces majority class samples to 0.1% of their original count.
2. Random Oversampling (with sampling_strategy=0.005)
    a. Increases minority class samples to 0.5% of the majority class.

Final Dataset Characteristics

- Image batch shape: (256, 72, 72, 3)
- Tabular data shape: (256, 72)
- Labels shape: (256, 1

Model Architecture

- Inputs:
    - img_input: Image input with shape (IMAGE_HEIGHT, IMAGE_WIDTH, 3).
    - tab_feat: Tabular input with shape (72,).
- Feature Extraction:
    - ShiftedPatchTokenization: Implements shifted patch tokenization by shifting images diagonally and extracting patches.
    - PatchEncoder: Adds positional encoding to extracted patches.
- Transformer Block:
    - Multiple layers of Transformer blocks.
    - Each block consists of:
        - Multi-Head Attention with Locality Self Attention (LSA)
        - Layer Normalization

        - Skip connections
        - MLP (Feedforward network)
- Final Layers:
    - Flattened representation of encoded patches
    - Dropout
    - MLP for classification
    - Final Dense Layer with sigmoid activation for classification

Model Parameters Count:

- Total Trainable Parameters: Sum of attention, dense, and patch embedding layers.
- Breakdown of Key Parameter Counts:
    - Shifted Patch Tokenization: 35,704
    - Patch Encoder: 9,216
    - Multi-Head Attention Layers: Each 66,369
    - Dense Layers: Each 8,320 (128 neurons), 8,256 (64 neurons)

Training Techniques:

- Optimizer: Likely Adam or AdamW
- Regularization:
    - Dropout Layers (prevent overfitting)
    - Layer Normalization (stabilize training)
- Augmentation Techniques: Data augmentation (if mentioned in preprocessing)
- Learning Rate Strategy: Possibly warmup schedule with cosine decay.

Training Metrics & Results:

- PR_pAUC (Precision-Recall partial AUC): 0.9534 (train) and 0.9796 (validation)
- ROC_pAUC (Receiver Operating Characteristic partial AUC): 0.9963 (train) and 0.9923 (validation)
- Loss: 0.0107 (train) and 0.0018 (validation)

Key Observations:

1. High PR_pAUC & ROC_pAUC: Your model is achieving high precision-recall and ROC scores, which suggests strong performance in distinguishing between classes.
2. Low Loss Values: The loss is significantly lower on validation data compared to training, which may indicate that the model generalizes well.

Resultant scores:

- Private score: 0.10680

- Public score: 0.10106

Vision transformers version – 4

Dataset Size

- Training Data Shape: (4951, 73) before resampling.
- Resampled Data Shape: (5502, 73) after handling class imbalance.
- Image Dimensions: 72 × 72 pixels.
- Batch Size: 256.

Preprocessing Methodologies

1. **Tabular Data Processing:**
    a. Feature Selection:
        i. Drops irrelevant columns: ['patient_id', 'image_type', 'tbp_tile_type', 'attribution', 'copyright_license'].
        ii. Removes train-only columns: ['lesion_id', 'iddx_full', 'iddx_1', ..., 'tbp_lv_dnn_lesion_confidence'].
    b. Feature Engineering:
        i. Identifies categorical and numerical columns.
        ii. One-hot encoding for categorical features.
        iii. KNN imputation and standard scaling for numerical features.
2. **Image Processing:**
    a. Reads images from HDF5 files.
    b. Resizes images to 72×72 pixels.
    c. Augmentations applied:
        i. Random horizontal flip
        ii. Random brightness adjustment
        iii. Random contrast adjustment
    d. Normalization to the [0, 1] range.
3. **Dataset Construction:**
    a. Combines image and tabular data into a structured dictionary for training.
    b. Uses TensorFlow datasets (tf.data.Dataset).
    c. Parallelized image loading for efficiency.

Class Imbalance Handling

- Undersampling:
    - Reduces majority class instances to 10% of the dataset.
- Oversampling:

- Increases minority class instances to 40% of the dataset.

- Final Resampled Shape:
  - X_train_final: (5502, 73)
  - y_train_final: (5502,)

Model architecture:

This model combines a Vision Transformer (ViT) for image processing with an additional tabular input.

Inputs:

1. Image Input: (72, 72, 3) - RGB image of size 72x72.
2. Tabular Input: (72,) - A separate tabular feature input.

Architecture Components:

1. Shifted Patch Tokenization (SPT):
   a. Extracts patches from the image with a shifting mechanism to capture local information.
   b. Uses Layer Normalization and Dense Projection.
2. Patch Encoding:
   a. Adds positional embeddings to the extracted patches.
3. Transformer Blocks (6 layers):
   a. Multi-Head Attention with Locality Self Attention (LSA) (5 heads, key dimension 64).
   b. Feed-Forward MLP with LeakyReLU activations and dropout.
   c. Skip Connections and Layer Normalization.
4. Feature Aggregation:
   a. Flattens the transformer output.
   b. Uses a dropout layer.
5. MLP Head:
   a. Fully connected layers with sizes [1024, 256].
   b. Dropout applied for regularization.
6. Final Output Layer:
   a. Dense Layer with Sigmoid Activation (NUM_CLASSES = 1).

Model Parameters:

- Total Parameters: 10,344,575 (39.46 MB)
  - Trainable Parameters: 10,344,575 (39.46 MB)
  - Non-trainable Parameters: 0 (0.00 B)

Training Methodologies:

1. Learning Rate Schedule:
    a. Piecewise Constant Decay:
        i. Boundaries: [100, 500]
        ii. Values: [1e-3, 1e-4, 1e-5]
    b. Optimizer: Adam with learning rate scheduler and weight decay of 1e-5.
2. Optimizer:
    a. Adam optimizer with Piecewise Constant Learning Rate.
    b. Learning rate values adjust based on the defined boundaries and values schedule.
3. Loss Function:
    a. Binary Cross-Entropy: keras.losses.BinaryCrossentropy()
4. Metrics:
    a. AUC-PR (Precision-Recall Area Under Curve): keras.metrics.AUC(curve='PR', name='PR_pAUC')
    b. AUC-ROC (Receiver Operating Characteristic Area Under Curve): keras.metrics.AUC(curve='ROC', name='ROC_pAUC')
5. Class Weights:
    a. Computed using the balanced method from compute_class_weight.
    b. Class Weights: {0: 0.7001, 1: 1.7495}

Callbacks:

1. Early Stopping:
    a. Monitor: val_loss
    b. Mode: min
    c. Patience: 5 (Stop training after 5 epochs with no improvement in validation loss).
    d. Restore best weights.
    e. Start from epoch 3.
2. Model Checkpoint:
    a. Saves the best model based on val_loss.

Model performance:

- Epoch 17/102:
    - Training:
        - PR_pAUC: 0.7841
        - ROC_pAUC: 0.8798
        - Loss: 0.4374

- Validation:
    - val_PR_pAUC: 0.8093
    - val_ROC_pAUC: 0.8928
    - val_loss: 0.4329

Model Inference on Training Results:

- The model is performing well on both the training and validation sets. The **PR_AUC** and **ROC_AUC** scores indicate that it has a strong ability to differentiate between the classes, and the relatively low **loss** values confirm that the model is learning effectively.
- **Validation performance** surpasses **training performance** in both AUC and loss metrics, which is a good sign of generalization and indicates that the model is likely not overfitting.

Resultant scores:

- Private score: 0.09413
- Public score: 0.08403

Vision transformers version – 5

Dataset Size

- Total dataset size before resampling: 41,265 samples
- Train-validation split: 90%-10%
- Final training set size: 37,138 samples

Image and Tabular Data Details

- Image shape: (72, 72, 3) (RGB images)
- Tabular feature count: 72 columns
- Labels shape: (batch_size, 1) (Binary classification)

Preprocessing Methodologies

- Data Cleaning:
    - Dropping irrelevant columns: patient_id, image_type, tbp_tile_type, attribution, copyright_license
    - Dropping train-only columns: lesion_id, iddx_full, mel_mitotic_index, etc.
- Handling Missing Values:
    - Numerical Data: Imputation using KNNImputer()
    - Categorical Data: Encoding using OneHotEncoder(handle_unknown="ignore")
- Standardization:
    - Numerical features are scaled using StandardScaler()

Class Imbalance Handling

- Undersampling: Majority class is undersampled to 1% of the dataset using RandomUnderSampler(sampling_strategy=0.01).
- Oversampling: Minority class is oversampled to 5% using RandomOverSampler(sampling_strategy=0.05).
- Final balanced dataset size: 41,265 samples

Data Augmentation for Images

- Random Horizontal Flip
- Random Brightness Adjustment (max delta=0.2)
- Random Contrast Adjustment (range 0.8 - 1.2)
- Normalization: Pixel values scaled to [0,1]

Dataset Pipeline & Batching

- Batch Size: 256
- Shuffling Buffer: 512
- Parallel Processing: Uses tf.data.AUTOTUNE for efficient mapping.

Model Architecture

Input Layers

- Image Input: (72, 72, 3)
- Tabular Input: (72,) (determined dynamically from the dataset)

Patch Tokenization and Encoding

- Shifted Patch Tokenization:
  - Extracts image patches (size 6x6)
  - Uses LayerNormalization and Dense projection layer (64 units)
- Patch Encoder:
  - Adds positional encoding (Embedding layer)

Transformer Encoder (ViT Block)

- 6 Transformer Layers:
  - Multi-Head Self Attention (LSA-based)
    - 5 attention heads
    - Projection dimension: 64
    - Uses dropout=0.1
  - Feedforward MLP
    - Two fully connected layers: [128, 64]
    - LeakyReLU activation
    - Dropout (0.1)

Feature Extraction & Classification

- Feature Representation
    - LayerNormalization
    - Flatten
    - Dropout (0.3)
- MLP Head
    - Two Dense layers: [2048, 512]
    - Dropout (0.3)
- Output Layer
    - Dense (1 neuron, sigmoid activation)

Model Parameters

- Total Parameters: 20,569,727 (~78.47 MB)
- Trainable Parameters: 20,569,727
- Non-trainable Parameters: 0

Training Methodologies

1. Learning Rate Schedule:
    a. Piecewise Constant Decay with boundaries [100, 500] and values [1e-3, 1e-4, 1e-5]
    b. The learning rate changes as training progresses:
        i. 1e-3 initially
        ii. 1e-4 after 100 steps
        iii. 1e-5 after 500 steps
2. Optimizer:
    a. Adam optimizer with the defined learning rate schedule
    b. Alternative commented options:
        i. RMSprop(learning_rate=0.001, rho=0.98, momentum=0.1)
        ii. Nadam(learning_rate=0.001, weight_decay=0.01)
3. Loss Function:
    a. Binary Crossentropy
    b. Alternative commented options:
        i. Focal Loss (gamma=2.0, alpha=0.25)
        ii. Binary Focal Crossentropy (gamma=3.0, apply_class_balancing=True)
4. Class Weight Balancing:
    a. Used to handle imbalanced data
    b. Computed dynamically based on training data

c. Class Weights: {0: 0.5249, 1: 10.5028}

d. This means the minority class (1) is given significantly higher importance.

5. Callbacks:

    a. Early Stopping:

        i. Monitors val_loss

        ii. Stops training if val_loss doesn't improve for 7 epochs

        iii. Restores best weights

        iv. Starts monitoring from epoch 5

    b. Model Checkpointing:

        i. Saves the best model (best_model.keras) based on val_loss

    c. Reduce LR on Plateau (commented out):

        i. Reduces learning rate if loss does not improve for 3 epoch**s**

Training Metrics

- Primary Metrics:
    - PR AUC (Precision-Recall Area Under Curve) → PR_pAUC
    - ROC AUC (Receiver Operating Characteristic Area Under Curve) → ROC_pAUC
- Epoch 25 Training Results:
    - PR_pAUC: 0.4293
    - ROC_pAUC: 0.8806
    - loss: 0.4402
    - Validation PR_pAUC: 0.3875
    - Validation ROC_pAUC: 0.8890
    - Validation loss: 0.4069

Resultant scores:

- Private score: 0.09158
- Public score: 0.08228

Vision transformers version – 6

Dataset Size:

- Training dataset:
    - Preprocessed shape: (4951, 73) (before resampling)
    - After resampling: (5502, 73) (includes both undersampling & oversampling)
- Batch size: 256

- Image size: 72x72x3 (RGB)
- Tabular data features: 72 numerical/categorical features

Preprocessing Methodologies:

- Tabular Data:
    - Handling missing values:
        - Numerical features: KNN Imputer
        - Categorical features: OneHot Encoding
    - Feature Scaling: StandardScaler (for numerical data)
    - Column selection:
        - Dropped irrelevant columns (e.g., patient_id, image_type, tbp_tile_type)
        - Dropped train-only columns (e.g., lesion_id, mel_mitotic_index)
- Image Data:
    - Decoding: JPEG images read from HDF5 files
    - Resizing: 72x72
    - Augmentation:
        - Random horizontal flip
        - Random brightness adjustment (max_delta=0.2)
        - Random contrast adjustment (lower=0.8, upper=1.2)
    - Normalization: [0,1] range by dividing by 255.0

Class Imbalance Handling Techniques:

1. Undersampling:
    a. RandomUnderSampler (sampling_strategy=0.1) to reduce majority class
2. Oversampling:
    a. RandomOverSampler (sampling_strategy=0.4) to increase minority class
3. Stratified Train-Validation Split:
    a. Ensures class distribution remains balanced in training & validation sets

Model Architecture:

Inputs:

1. **Image Input (img_input)**: (72, 72, 3)
2. **Tabular Input (tab_feat)**: (72,) (or based on dataset shape)

Architecture Components:

1. **Shifted Patch Tokenization**:
    a. Splits images into patches (size 6x6).

    b. Applies diagonal shifts.
    c. Flattens and projects patches into 64-dimensional tokens.
2. **Patch Encoding**:
    a. Adds positional embeddings to patches.
3. **Transformer Blocks (5 Layers)**:
    a. **Multi-Head Self-Attention with Locality Self Attention (LSA)**:
        i. 3 attention heads.
        ii. 64-dimensional projection.
        iii. Diagonal attention masking.
    b. **MLP Layer** (applied after each attention block):
        i. Two fully connected layers (128, 64 units).
        ii. Leaky ReLU activation.
        iii. Dropout (0.1).
4. **Final Representation**:
    a. **Flatten layer**.
    b. **Dropout (0.3)**.
    c. **MLP Head**:
        i. Two dense layers (2048, 512 units).
        ii. Leaky ReLU activation.
        iii. Dropout (0.3).
5. **Output Layer**:
    a. **Single Neuron** with **Sigmoid Activation** (for binary classification).

Model Parameters

- **Total Parameters**: 20,304,190 (77.45 MB)
- **Trainable Parameters**: 20,304,190 (All parameters are trainable)
- **Non-trainable Parameters**: 0

Training Methodologies and Components Used

Optimizer:

- Adam optimizer with a **piecewise constant learning rate decay**:
    - Learning rate schedule changes at epochs 100 and 500:
        - Initial LR: 1e-3
        - After 100 steps: 1e-4

        - After 500 steps: 1e-5

Loss Function:

- BinaryCrossentropy (standard binary classification loss)
- Previously commented-out options:
    - BinaryFocalCrossentropy (helps with imbalanced classes)
    - FocalLoss with gamma=2, alpha=0.25

Metrics Tracked:

- PR_pAUC (Precision-Recall AUC)
- ROC_pAUC (Receiver Operating Characteristic AUC)

Callbacks Used:

1. **EarlyStopping**
    a. Monitors val_loss, stops training if it doesn't improve for **7 epochs**
    b. **Restores best weights** and starts monitoring from **epoch 5**
2. **ModelCheckpoint**
    a. Saves the **best model** based on val_loss to "best_model.keras"

Class Weights for Imbalanced Data Handling:

- Computed dynamically using compute_class_weight('balanced', ...)
- **Class Weights Applied:**
    - **Class 0**: 0.7001
    - **Class 1**: 1.7495
- (Manually set weights were {0:0.8, 1:10.0}, but they weren't used)

Training Results and Performance Inference

- **Epochs Trained:** 56 out of 102
- **Best Performance (Epoch 56):**
    - **Training:**
        - PR AUC: **0.8267**
        - ROC AUC: **0.9035**
        - Loss: **0.3372**
    - **Validation:**
        - PR AUC: **0.8487**
        - ROC AUC: **0.9159**
        - Loss: **0.3232**

Inference:

- High ROC AUC (0.91) and PR AUC (0.85) suggest strong classification performance.
- The model generalizes well, as validation metrics are slightly better than training metrics.
- Binary cross-entropy loss is stable (0.32), indicating good convergence.
- Early stopping likely triggered before 102 epochs, preventing overfitting.

Resultant scores:

- Private score: 0.09617
- Public score: 0.07985

Vision transformers version – 9

Dataset Size

- Initial Training Set Size: 51,090 samples
- After Train-Validation Split:
    - Train Set: 45,981 samples
    - Validation Set: 10% split
- Image Input Shape: (72, 72, 3)
- Tabular Input Shape: (72,)
- Class Labels: Binary (0/1)

Preprocessing Methodologies

- Numerical Data:
    - KNN Imputer (for missing values)
    - StandardScaler (for normalization)
- Categorical Data:
    - OneHotEncoder (handling unknown categories)
- Feature Selection:
    - Dropped Columns: Unnecessary metadata (patient_id, image_type, lesion_id, etc.)

Class Imbalance Handling

- Under-sampling: RandomUnderSampler (reducing majority class to 1% ratio)
- Over-sampling: RandomOverSampler (increasing minority class to 30% ratio)
- Final Resampled Size: 51,090 samples

MODEL ARCHITECTURE

Input:

- img_input: Image input of shape (72, 72, 3)
- tab_feat: Tabular input of shape (tab_features_len,)

Patch Tokenization & Encoding:

1. **Shifted Patch Tokenization (SPT):**
    a. Image is shifted in diagonal directions.
    b. Patches of size 6x6 are extracted and flattened.
    c. Layer normalization and dense projection are applied.
2. **Patch Encoding:**
    a. Position embeddings are added to patch tokens.

Transformer Block (Repeated 8 Times):

Each transformer block consists of:

1. **Layer Normalization**
2. **Multi-Head Attention (LSA-based or Standard)**
    a. 4 heads (NUM_HEADS = 4)
    b. Key dimension = 64
    c. Dropout = 0.1
    d. Uses diagonal attention masking
3. **Skip Connection**
4. **Layer Normalization**
5. **Feedforward MLP**
    a. Hidden units: [128, 64] (TRANSFORMER_UNITS)
    b. Activation: Leaky ReLU
    c. Dropout: 0.1
6. **Skip Connection**

Final Layers:

1. **Layer Normalization & Flattening**
2. **Dropout (0.3)**
3. **MLP Head**
    a. Hidden units: [1024, 512]
    b. Activation: Leaky ReLU
    c. Dropout: 0.2
4. **Classification Layer**
    a. Dense layer with 1 output unit
    b. Activation: sigmoid

Model Parameters:

- **Total Parameters:** 10,674,177 (~40.72 MB)
- **Trainable Parameters:** 10,674,177
- **Non-Trainable Parameters:** 0

Training Methodology:

- **Optimizer:** Adam with:
    - **Learning Rate Schedule:** PiecewiseConstantDecay
        - Learning rates change at different steps:
            - 1e-3 for first 300 steps
            - 1e-4 for steps 301-1000
            - 1e-5 after step 1000
    - **Weight Decay:** 1e-6
- **Loss Function:** BinaryCrossentropy()
    - (Commented options include Focal Loss and Binary Focal Crossentropy)
- **Metrics Monitored:**
    - **Precision-Recall AUC (PR_pAUC)**
    - **ROC AUC (ROC_pAUC)**
- **Class Weights Used:** Adjusted to handle class imbalance
    - Class 0: 0.65
    - Class 1: 2.17
    - (Computed dynamically using compute_class_weight)

Training Components:

- **Callbacks Used:**
    - **Early Stopping:**
        - Monitors val_loss
        - Stops training if no improvement for 10 epochs
        - Restores best weights
        - Starts monitoring after epoch 5
    - **Model Checkpointing:**
        - Saves the best model based on val_loss
        - File saved as 'best_model.keras'

(Commented alternative: ReduceLROnPlateau for reducing LR on plateau)

Training Results (Final Epoch - 102/102):

- **Training Performance:**
    - PR_pAUC: 0.9989
    - ROC_pAUC: 0.9998
    - Loss: 0.0119
- **Validation Performance:**
    - val_PR_pAUC: 0.9966
    - val_ROC_pAUC: 0.9995
    - val_loss: 0.0303

Model Performance & Inference:

- **Extremely High AUC Scores** (~0.999 for both PR and ROC curves) indicate:
    - Model is **highly effective** at distinguishing between classes.
    - Minimal False Positives and False Negatives.
- **Low Loss (0.0119 training, 0.0303 validation)** suggests:
    - The model has learned **well** without significant overfitting.
- **Validation Performance is Close to Training Performance**, meaning:
    - The model **generalizes well** to unseen data.

Resultant scores:

- Private score: 0.11688
- Public score: 0.12120

Vision transformers version – 11

Dataset Size

- Total Training Data After Resampling: (38,907, 73)
- Total Training Data Before Splitting: (43,230, 73)
- Validation Split: 10%

Preprocessing Steps

1. Column Dropping
    a. Irrelevant Columns Removed:
    ['patient_id', 'image_type', 'tbp_tile_type', 'attribution', 'copyright_license']

b. Train-Only Columns Removed:
['lesion_id', 'iddx_full', 'iddx_1', 'iddx_2', ..., 'mel_thick_mm', 'tbp_lv_dnn_lesion_confidence']

2. Feature Engineering
    a. Numerical Features:
        i. KNNImputer for missing values
        ii. StandardScaler for scaling
    b. Categorical Features:
        i. OneHotEncoder(handle_unknown="ignore")
    c. Final Columns Kept: [72 tabular features + image data]

Class Imbalance Handling

- Resampling Strategy:
    - RandomUnderSampler (Strategy: 0.01) → Reduce majority class
    - RandomOverSampler (Strategy: 0.1) → Boost minority class
- Final Data After Resampling:
    - X_final shape: (43,230, 73)
    - Y_final shape: (43,230,)

Image Processing & Augmentation

- Resizing: 72 x 72
- Augmentations Applied:
    - Random Flip (Left-Right)
    - Random Brightness Adjustment (Max Delta: 0.2)
    - Random Contrast Adjustment (Range: 0.8 - 1.2)
    - Normalization: [0,1] range

Model Architecture:

Model Inputs:

1. **Image Input (img_input)**: Shape = (72, 72, 3)
2. **Tabular Input (tab_feat)**: Shape = (72,)

Shifted Patch Tokenization (ViT Encoder Input)

- The image is processed using ShiftedPatchTokenization, which:
    - Applies diagonal shifting augmentation.
    - Extracts patches of size (6,6).
    - Flattens patches and applies layer normalization and a dense projection.

Patch Encoding

- The extracted patches are passed through PatchEncoder to add positional embeddings.

Transformer Encoder (Vision Transformer)

- 8 Transformer layers, each containing:
    - Layer Normalization
    - **Multi-Head Self-Attention (LSA)**
    - Skip Connection
    - Layer Normalization
    - **MLP Block** (2 fully connected layers with Leaky ReLU activation)
    - Skip Connection
- After Transformer layers:
    - Layer Normalization
    - Flattening
    - Dropout (0.3)

MLP Head for Image Features

- Dense Layer (2048 units, Leaky ReLU, He Normal Initialization)

MLP Head for Tabular Features

- Dense Layer (256 units, Leaky ReLU, He Normal Initialization) → Dropout (0.2)
- Dense Layer (512 units, Leaky ReLU, He Normal Initialization) → Dropout (0.2)

Fusion and Classification Head

- **Concatenation** of image and tabular feature vectors.
- Dense Layer (1024 units, Leaky ReLU) → Dropout (0.2)
- Dense Layer (512 units, Leaky ReLU) → Dropout (0.2)
- Dense Layer (256 units, Leaky ReLU) → Dropout (0.2)
- **Final Classification Layer**: Dense (1 unit, Sigmoid activation)

Model Parameters

- **Total Parameters:** 23,016,193 (~87.8 MB)
- **Trainable Parameters:** 23,016,193 (~87.8 MB)
- **Non-Trainable Parameters:** 0

Training Methodology & Components Used

1. Learning Rate Schedule

- **Piecewise Constant Decay**:
    - Learning rate changes at predefined boundaries:
        - **First 300 steps:** 1e-3

        - **Next 700 steps (until 1000 steps):** 1e-4
        - **After 1000 steps:** 1e-5

2. Optimizer

- **Adam Optimizer** with:
    - **Piecewise Constant Learning Rate Decay**
    - **Weight Decay:** 1e-6

3. Loss Function

- **Binary Focal Crossentropy**:
    - **Class Balancing Applied**
    - **Alpha = 0.25, Gamma = 3.0**
    - **Label Smoothing = 0.1**

4. Evaluation Metrics

- **Precision-Recall AUC (PR_pAUC)**
- **ROC AUC (ROC_pAUC)**

5. Callbacks Used

- **Early Stopping**:
    - Monitors val_loss
    - **Patience:** 10 epochs
    - **Restores best weights**
    - Starts from **epoch 5**
- **Model Checkpointing**:
    - Saves the best model based on **validation loss** (val_loss)
    - **Filepath:** 'best_model.keras'

6. Class Weights for Imbalanced Data

- **Computed using compute_class_weight**:
    - **Class 0:** Weight = **0.55**
    - **Class 1:** Weight = **5.5**
    - Balances the dataset to handle class imbalance.

Training Results

Final Epoch: 102

Metric	Training	Validation
PR AUC	**0.7799**	**0.8353**
ROC AUC	**0.9697**	**0.9800**

Loss	**0.0071**	**0.0061**

Performance Inference

- **High Validation PR AUC (0.8353)**: Indicates strong precision-recall balance, useful for imbalanced datasets.
- **High Validation ROC AUC (0.9800)**: Suggests excellent discrimination between classes.
- **Low Validation Loss (0.0061)**: Shows strong convergence with effective generalization.
- **Early Stopping Applied:** The model likely stopped training at an optimal point before overfitting.

Resultant scores:

- Private score: 0.11722
- Public score: 0.13676

Mobile ViT version – 2

Dataset Size:

- **Final training dataset size**: (24759, 73)
- **Resampled dataset size**: (27510, 73) (after class balancing)
- **Batch Size**: 64
- **Image dimensions**: 128 x 128 x 3

Preprocessing Methodologies:

1. **Tabular Data Processing:**
    a. Dropping irrelevant columns: ['patient_id', 'image_type', 'tbp_tile_type', 'attribution', 'copyright_license']
    b. Dropping training-only columns: ['lesion_id', 'iddx_full', 'iddx_1', 'iddx_2', 'iddx_3', 'iddx_4', 'iddx_5', 'mel_mitotic_index', 'mel_thick_mm', 'tbp_lv_dnn_lesion_confidence']
    c. Identifying categorical and numerical columns.
    d. **Numerical Data Pipeline:**
        i. Imputation using KNNImputer
        ii. Standardization using StandardScaler
    e. **Categorical Data Pipeline:**
        i. Encoding using OneHotEncoder (with handle_unknown="ignore")
    f. Column alignment between train and test data to ensure consistency.
2. **Image Preprocessing:**

a. Decoding images from bytes.
b. Resizing to 128x128.
c. Augmentation:
    i. **Random horizontal flip**
    ii. **Random brightness adjustment** (max delta 0.2)
    iii. **Random contrast adjustment** (range 0.8 - 1.2)
d. **Normalization**: Pixel values scaled to [0,1].

3. **Dataset Creation:**
    a. Images are loaded from an HDF5 file.
    b. Multiprocessing (ProcessPoolExecutor) is used to speed up image loading.
    c. TensorFlow dataset is created with both tabular and image data.
    d. Dataset is shuffled before batching.

Class Imbalance Handling:

- **Undersampling**:
    - RandomUnderSampler(sampling_strategy=0.02) to reduce the majority class.
- **Oversampling**:
    - RandomOverSampler(sampling_strategy=0.4) to increase the minority class.
- **Final dataset size after resampling**: (27510, 73)

Model Architecture (MobileViT-based Hybrid Model)

Inputs:

1. **Image Input**: (128, 128, 3)
2. **Tabular Input**: (72,) (Based on output)

Feature Extraction:

1. **Initial Convolution Block**:
    a. Conv2D(filters=16, kernel_size=3, strides=2, activation=swish)
2. **MobileNetV2 Blocks**:
    a. Inverted Residual Block (16 → 16)
    b. Inverted Residual Block (16 → 24, strides=2)
    c. Inverted Residual Block (24 → 24)
    d. Inverted Residual Block (24 → 24)
3. **MobileViT Blocks**:
    a. **Stage 1**: Inverted Residual Block (24 → 48, strides=2) → MobileViT Block (2 Transformer layers, 64 projection dim)

    b. **Stage 2**: Inverted Residual Block (64 → 64, strides=2) → MobileViT Block (4 Transformer layers, 80 projection dim)
    c. **Stage 3**: Inverted Residual Block (80 → 80, strides=2) → MobileViT Block (3 Transformer layers, 96 projection dim)
4. **Final Convolution Layer**:
    a. Conv2D(filters=320, kernel_size=1, strides=1)
5. **Global Pooling**:
    a. GlobalAvgPool2D()

Tabular Feature Processing:

1. Dense(256, activation='silu') → Dropout(0.2)
2. Dense(512, activation='silu') → Dropout(0.2)

Fusion & Classification Head:

1. **Feature Concatenation**: Image Features + Tabular Features
2. **Fully Connected Layers**:
    a. Dense(1024, activation='silu') → Dropout(0.2)
    b. Dense(512, activation='silu') → Dropout(0.2)
    c. Dense(256, activation='silu') → Dropout(0.2)
3. **Final Classification Layer**:
    a. Dense(1, activation='sigmoid') (Binary Classification Output)

Model Metrics

- **Primary Loss Function**: BinaryFocalCrossentropy (with class balancing: alpha=0.25, gamma=3.0, label_smoothing=0.1)
- **Performance Metrics**:
    - **Precision-Recall AUC (PR_pAUC)**
    - **ROC AUC (ROC_pAUC)**

Training Results (Epoch 23 Snapshot)

- **Training Performance**:
    - **PR AUC**: 0.9957
    - **ROC AUC**: 0.9984
    - **Loss**: 0.0029
- **Validation Performance**:
    - **PR AUC**: 0.9911

- **ROC AUC**: 0.9972
- **Loss**: 0.0049

Performance Inference

- **High AUC scores** (>0.99 in both training and validation) suggest the model is performing well in distinguishing between classes.
- **Low training loss (0.0029) and low validation loss (0.0049)** indicate the model generalizes well.
- **Validation AUC remains high**, meaning the model retains predictive power on unseen data.
- **Possible Overfitting**:
  - The training loss is significantly lower than the validation loss.
  - However, AUC scores are still high, so overfitting is minimal.

Model Parameters

- **Optimizer**: Adam with PiecewiseConstantDecay learning rate schedule
  - **Initial LR**: 1e-3
  - **Decay at Steps**: [300, 1000]
  - **Decay Values**: [1e-3 → 1e-4 → 1e-5]
  - **Weight Decay**: 1e-6
- **Loss Function**: BinaryFocalCrossentropy (adjusted for class imbalance)
- **Total Epochs**: 152
- **Batch Size**: 64
- **Dropout in Model**: 0.2 at various stages
- **Activation Functions**:
  - Swish for intermediate layers
  - Sigmoid for final classification

Training Methodologies & Components

Learning Rate Schedule

- **Piecewise Constant Decay**:
  - Starts at 1e-3
  - Drops to 1e-4 at 300 steps
  - Further drops to 1e-5 at 1000 steps

Callbacks Used

- **Early Stopping** (monitor=val_loss, patience=10, restore_best_weights=True, start_from_epoch=5)
    - Prevents excessive overfitting by stopping training early.
- **ModelCheckpoint** (filepath='best_model.keras', save_best_only=True, monitor='val_loss')
    - Saves the best model during training.

Resultant scores:

- Private score: 0.11469
- Public score: 0.13332

Mobile ViT version – 5

Dataset Size:

1. **Train Data**: **37,138 samples** (final after preprocessing).
2. **Validation Data**: **4,127 samples** (10% of the training set).
3. **Total Processed Data**: **41,265 samples** (before train-validation split).
4. **Features**: **73 tabular features**.

Preprocessing Methodologies:

5. **Image Processing:**
    a. Resizing images to **128x128**.
    b. Data augmentation: **Random horizontal flip**.
    c. Normalization: Scaling pixel values to **[0,1]**.
6. **Tabular Data Processing:**
    a. **Missing Value Handling**:
        i. Numerical columns: **KNN Imputer + StandardScaler**
        ii. Categorical columns: **OneHotEncoder**
    b. **Feature Selection**:
        i. Dropped irrelevant columns (patient_id, image_type, etc.).
        ii. Dropped train-only columns (lesion_id, mel_mitotic_index, etc.).
    c. **Final Data Alignment**:
        i. Ensuring test dataset aligns with train columns.

Class Imbalance Handling Techniques:

7. **Under-sampling**:
    a. **RandomUnderSampler** applied with sampling_strategy=0.01 to reduce majority class.
8. **Over-sampling**:

a. **RandomOverSampler** applied with sampling_strategy=0.05 to boost minority class.

Model Architecture: MobileViT with Tabular Data

Inputs:

- img_input: Image input of shape (IMAGE_HEIGHT, IMAGE_WIDTH, 3)
- tab_feat: Tabular feature input of shape (tab_features_len,)

Feature Extraction Blocks:

1. **Conv Stem & Initial Inverted Residual Block:**
    a. Conv2D(16, kernel_size=3, strides=2, activation='leaky_relu')
    b. Inverted Residual Block (16 -> 16 channels)
2. **Downsampling with Inverted Residual Blocks:**
    a. Inverted Residual Block (16 -> 24 channels, strides=2)
    b. Inverted Residual Block (24 -> 24 channels)
    c. Inverted Residual Block (24 -> 24 channels)
3. **First MobileViT Block:**
    a. Inverted Residual Block (24 -> 48 channels, strides=2)
    b. MobileViT Block with 2 Transformer Layers, Projection Dim=64
4. **Second MobileViT Block:**
    a. Inverted Residual Block (64 -> 64 channels, strides=2)
    b. MobileViT Block with 4 Transformer Layers, Projection Dim=80
5. **Third MobileViT Block:**
    a. Inverted Residual Block (80 -> 80 channels, strides=2)
    b. MobileViT Block with 4 Transformer Layers, Projection Dim=96
6. **Final Convolution & Global Pooling:**
    a. Conv2D(320, kernel_size=1, strides=1, activation='leaky_relu')
    b. Global Average Pooling

Tabular Feature Processing:

1. Dense(256, activation='leaky_relu', kernel_initializer=HeNormal)
2. Dropout(0.2)

Fusion & Classification Head:

1. Concatenation of image and tabular features
2. Dense(512, activation='leaky_relu', kernel_initializer=HeNormal)

3. Dropout(0.2)
4. Dense(256, activation='leaky_relu', kernel_initializer=HeNormal)
5. Dropout(0.2)
6. Dense(1, activation='sigmoid') (Final Classification Layer)

Model Parameters:

- **Total Parameters:** 1,863,649 (7.11 MB)
- **Trainable Parameters:** 1,861,105 (7.10 MB)
- **Non-trainable Parameters:** 2,544 (9.94 KB)

Training Methodologies & Components Used:

1. **Learning Rate Scheduling:**
    a. **Piecewise Constant Decay:**
        i. Learning rates change at specific epochs:
            1. Epoch ≤ 300 → 1e-3
            2. Epoch ≤ 1000 → 1e-4
            3. After epoch 1000 → 1e-5
2. **Optimizer:**
    a. **Adam** optimizer with:
        i. **Learning rate:** Adjusted dynamically via PiecewiseConstantDecay
        ii. **Weight decay:** 1e-6
3. **Loss Function:**
    a. **Binary Focal Crossentropy** with:
        i. Class balancing applied
        ii. **Alpha:** 0.4
        iii. **Gamma:** 2.0
        iv. **Label smoothing:** 0.1
4. **Metrics Used:**
    a. PR_pAUC (Precision-Recall Partial AUC)
    b. ROC_pAUC (Receiver Operating Characteristic Partial AUC)
5. **Callbacks Used:**
    a. EarlyStopping:
        i. **Monitors:** val_loss
        ii. **Mode:** min (minimization of loss)

iii. **Patience:** 10 epochs (stops training if no improvement)

iv. **Restore best weights:** Enabled

v. **Starts monitoring from epoch 5**

b. ModelCheckpoint:

i. Saves the best model based on val_loss

ii. Saves to 'best_model.keras'

6. Class Weights:

a. Computed dynamically using compute_class_weight for handling imbalanced data.

b. **Weights:**

i. **Class 0:** 0.5249

ii. **Class 1:** 10.5028

Training Results (Epoch 68/152):

- **Train Metrics:**
  - **PR_pAUC:** 0.9986
  - **ROC_pAUC:** 0.9999
  - **Loss:** 0.0012
- **Validation Metrics:**
  - **Val PR_pAUC:** 0.9723
  - **Val ROC_pAUC:** 0.9987
  - **Val Loss:** 0.0048

Performance Inference:

- High Training Performance:
  - PR_pAUC (0.9986) and ROC_pAUC (0.9999) indicate nearly perfect model performance on training data.
  - Extremely low training loss (0.0012), suggesting strong learning ability.
- **Strong Validation Performance:**
  - Validation PR_pAUC (0.9723) and ROC_pAUC (0.9987) show excellent generalization to unseen data.
  - Validation loss (0.0048) is slightly higher than training loss but still very low, indicating minimal overfitting.
- **Class Imbalance Handling:**
  - The **high class weight for minority class (10.5 vs 0.52)** suggests strong adjustments to mitigate class imbalance.

- The performance metrics confirm that the model successfully learns from both classes despite the imbalance.

Resultant scores:

- Private score: 0.10822
- Public score: 0.12237

Mobile ViT version – 7

Dataset Size:

- **Training Data Size**: 394,572 samples after resampling
- **Validation Data Size**: 10% of the training data, obtained by splitting the training data into train and validation sets (X_train_final and y_train_final).
- **Final Shape of Training Data**: (394,572, 73) for the feature set (X_train_final), and (394,572,) for the target set (y_train_final).
- **Final Shape of Images**: (64, 128, 128, 3) for a batch size of 64, indicating 128x128 images with 3 color channels (RGB).

Preprocessing Methodologies:

1. **Tabular Data Preprocessing**:
    a. **Imputation**: Missing values in numerical columns are imputed using KNN imputer.
    b. **Scaling**: Numerical data is standardized using StandardScaler.
    c. **One-Hot Encoding**: Categorical columns are encoded using OneHotEncoder.
    d. **Dropped Columns**: Irrelevant columns like 'patient_id', 'image_type', and columns present only in the training dataset ('lesion_id', 'iddx_full', etc.) are dropped.
2. **Image Data Preprocessing**:
    a. **Decoding and Resizing**: Images are decoded from JPEG byte data and resized to 128x128.
    b. **Augmentation**: Includes random horizontal flips to introduce variability.
    c. **Normalization**: Images are normalized to a range of [0, 1] by dividing by 255.0.
3. **Data Conversion**:
    a. Tabular data is converted to float32 and targets to int32.
    b. Images are loaded from an HDF5 file, with multiprocessing used for efficiency.
4. **Data Transformation**:
    a. The DataPreprocessor class applies the transformations on the training and test datasets, ensuring that both datasets have consistent columns after preprocessing.

Class Imbalance Handling Techniques:

1. **Undersampling**:

    a. Applied to numerical data using RandomUnderSampler to reduce the size of the majority class by a ratio of 0.001.

2. **Oversampling**:

    a. Applied to the minority class using RandomOverSampler with a sampling ratio of 0.004 to balance the classes.

3. **Stratified Split**:

    a. The dataset is split into training and validation sets using train_test_split with stratification to maintain the distribution of classes in both sets.

Model architecture:

1. Input Layer

- **Image Input**: Shape (128, 128, 3) (Height, Width, Channels)
- **Tabular Input**: Shape (tab_features_len,)

2. Shifted Patch Tokenization (Conceptual Step)

This step involves shifting the image diagonally, concatenating the shifted versions, extracting patches, flattening the patches, applying layer normalization, and projecting them.

3. Conv Block

- Convolutional layers with LeakyReLU activation and strides of 2.
- Filters are initialized with 16, and kernel size is 3.

4. Inverted Residual Block

- A series of 1x1 and 3x3 convolutions.
- Batch normalization after each convolution.
- Swish activation function.
- Zero padding and depthwise convolutions for downsampling if required.

5. MobileViT Block

- Combines local convolutions and transformer blocks.
- The image is divided into patches and processed by a transformer.
- The global and local features are fused together using convolutions.

6. MobileViT (MV2 to MobileViT Blocks)

- **First Block**: After the initial convolutions, an inverted residual block with downsampling (strides=2) is followed by the MobileViT block.
- **Second Block**: Another inverted residual block followed by the MobileViT block.
- **Third Block**: Additional inverted residual block and MobileViT block.

7. Final Convolution Block

- After the third MobileViT block, a final convolution with kernel size 1 and 320 filters is applied.

8. Global Average Pooling

- A GlobalAvgPool2D layer is applied to the output of the final convolution.

9. Tabular Input Processing

- Tabular input goes through a series of dense layers with LeakyReLU activation and dropout layers.
- **First Layer**: Dense layer with 256 units.
- **Second Layer**: Dense layer with 512 units.

10. Combining Image and Tabular Features

- The image features and tabular features are concatenated.
- The concatenated features go through several dense layers with LeakyReLU and dropout:
    - **First Layer**: Dense layer with 1024 units.
    - **Second Layer**: Dense layer with 512 units.
    - **Third Layer**: Dense layer with 256 units.

11. Output Layer

- **Logits Layer**: A final dense layer with a single unit and sigmoid activation for binary classification.

Model Parameters

- **Total Parameters**: 2,965,665 (11.31 MB)
- **Trainable Parameters**: 2,963,121 (11.30 MB)
- **Non-trainable Parameters**: 2,544 (9.94 KB)

Learning Rate Schedule

- **Learning Rate Schedule**: PiecewiseConstantDecay
    - **Boundaries**: [300, 1000]
    - **Learning Rates**: [1e-3, 1e-4, 1e-5]
    - **Type**: Piecewise constant decay schedule with a custom learning rate schedule.

Optimizer

- **Optimizer**: Adam
    - **Learning Rate**: Defined by the lr_schedule above.
    - **Weight Decay**: 1e-6

Loss Function

- **Loss Function**: Binary Focal Crossentropy
    - **Parameters**:
        - alpha=0.25: Class balancing factor for positive class.
        - gamma=3.0: Focal loss parameter to down-weight easy examples.
        - label_smoothing=0.1: For smoothing the labels and preventing overfitting on hard-to-classify examples.

Metrics

- **AUC (Area Under the Curve)**:
    - **PR (Precision-Recall) AUC**: Named as PR_pAUC
    - **ROC (Receiver Operating Characteristic) AUC**: Named as ROC_pAUC
    - These metrics are computed for both training and validation during the training process.

Callbacks

- **EarlyStopping**:
    - **Monitor**: val_loss
    - **Mode**: Minimize val_loss
    - **Patience**: 10 epochs before stopping if no improvement is seen.
    - **Restore Best Weights**: Yes, restoring the model with the best weights observed.
    - **Start from Epoch 5**: Begin monitoring from epoch 5.
- **ModelCheckpoint**:
    - **Filepath**: Saves the best model to 'best_model.keras'.
    - **Monitor**: val_loss
    - **Save Best Only**: Saves the model only if val_loss improves.
- **Class Weights**:
    - Calculated using compute_class_weight with balanced class weights based on y_train_final.
    - Class Weights: {0: 0.502, 1: 125.482}
    - These weights help to balance the class distribution, as the model likely deals with imbalanced classes.

Training Methodology

- **Dataset**:
    - **Training Dataset**: train_dataset
    - **Validation Dataset**: val_dataset
- **Epochs**: 152 epochs
- **Verbose**: 1 (to show training progress).
- **Callbacks**: EarlyStopping and ModelCheckpoint as mentioned above.
- **Class Weights**: Not used in the fit() method directly, but class weights were computed and printed.

Training Results (Example Epoch 43)

During training, the following metrics were reported for **Epoch 43/152**:

- **Training Metrics**:

- PR_pAUC: 0.9982
- ROC_pAUC: 1.0000
- Loss: 6.2052e-05

- **Validation Metrics**:
  - val_PR_pAUC: 0.9440
  - val_ROC_pAUC: 0.9926
  - val_loss: 3.0891e-04

Performance Inference

- **Training Performance**:
  - The model achieves a near-perfect training performance with PR AUC close to 1 and ROC AUC of 1. This suggests the model is very well fitted to the training data.
- **Validation Performance**:
  - The validation PR AUC is 0.9440, and the ROC AUC is 0.9926, showing the model performs well on unseen data, though there's a slight drop in performance compared to training.
  - The loss on validation is 3.0891e-04, which is low, indicating good model convergence.

Resultant scores:

- Private score: 0.04613
- Public score: 0.04962

Isic Medmamba version – 1

Dataset Size:

- The dataset consists of **41,265 training samples** (after resampling).
- The validation set contains **4,127 samples** (10% of training data).
- The tabular data has **72 features** after preprocessing.

Preprocessing Methodologies:

1. **Feature Selection:**
   a. Dropped irrelevant columns (patient_id, image_type, tbp_tile_type, etc.).
   b. Dropped train-only columns (lesion_id, iddx_full, etc.).
2. **Feature Engineering:**
   a. Categorical features: Encoded using **One-Hot Encoding**.
   b. Numerical features:
      i. Missing values handled using **KNN Imputer**.
      ii. Scaled using **StandardScaler**.
3. **Data Cleaning & Alignment:**

a. Ensured train and test datasets had the same columns.

b. Missing columns in test data were filled with zeros.

4. **Image Preprocessing:**

    a. **Resized** to (224, 224).

    b. **Random horizontal flip** applied for augmentation.

    c. Converted to **PyTorch tensor**.

Class Imbalance Handling Techniques:

- **Random Undersampling:** Majority class reduced to 1% of its original size.
- **Random Oversampling:** Minority class increased to 5% of the dataset.
- **Stratified Splitting:** Maintained class proportions in training and validation sets.

Model Architecture

1. **Input Layer:**

    a. Accepts images of size **(224, 224, 3)**.

    b. Processes patches of size **4x4** through the PatchEmbed2D module.

2. **Patch Embedding:**

    a. Converts image patches into embeddings using PatchEmbed2D.

    b. Optionally applies Layer Normalization (if patch_norm=True).

3. **Dropout and Positional Embeddings:**

    a. A positional embedding (absolute_pos_embed) is defined but not used (self.ape = False).

    b. A dropout layer (Dropout(p=drop_rate)) is applied.

4. **Backbone:**

    a. **4 main layers (VSSLayer)** process the embeddings sequentially.

    b. Layer dimensions: [96, 192, 384, 768].

    c. Depths per layer: [2, 2, 4, 2].

    d. Uses PatchMerging2D for downsampling between layers.

5. **Stochastic Depth Regularization:**

    a. Implements **drop path (stochastic depth) decay** with torch.linspace(0, drop_path_rate, sum(depths)).

6. **Classification Head:**

    a. **Global Average Pooling (AdaptiveAvgPool2d(1))** reduces feature map size.

    b. Fully Connected (Linear(num_features, num_classes)) maps features to class scores.

7. **Weight Initialization:**

    a. Conv2d: Kaiming Normal.

b. Linear: Truncated Normal.

c. LayerNorm: Bias set to 0, weight set to 1.

8. **Forward Pass:**

a. Extracts features via forward_backbone.

b. Transposes (permute(0,3,1,2)) to match PyTorch conventions.

c. Applies **Global Average Pooling** and **Linear Classification Head**.

Model Metrics

- **Loss Function:** CrossEntropyLoss() (suitable for multi-class classification).
- **Optimizer:** Adam with learning rate 0.0001 and weight decay 0.00001 (helps prevent overfitting).
- **Training Loss (Final Epoch):** 0.003
- **Validation Accuracy (Final Epoch): 57.976%**

Training Methodologies

- **Transfer Learning Approach:**
  - **Pretrained Model:** MedMamba.pth
  - **Feature Extraction:** All layers are frozen except for the last linear classification layer.
  - **Fine-Tuning:** Only the classification head is trained for binary classification.
- **Optimization Strategy:**
  - Uses Adam optimizer for adaptive learning.
  - **Learning rate:** 0.0001 (relatively small for stable updates).
  - **Weight decay:** 0.00001 to prevent overfitting.
- **Training Strategy:**
  - **Mini-batch training** using train_dataloader.
  - **Validation at each epoch** to monitor accuracy.
  - **Best model saving mechanism** (torch.save) based on validation accuracy.

Model Parameters: Toal count: 14454146

Resultant scores:

Private score: 0.10343

Public score: 0.11892

Isic Medmamba version – 2

Dataset Size:

- The dataset consists of **4716 training samples** (after resampling).
- The validation set contains **472 samples** (10% of training data).
- The tabular data has **72 features** after preprocessing.

Preprocessing Methodologies:

5. **Feature Selection:**
    a. Dropped irrelevant columns (patient_id, image_type, tbp_tile_type, etc.).
    b. Dropped train-only columns (lesion_id, iddx_full, etc.).
6. **Feature Engineering:**
    a. Categorical features: Encoded using **One-Hot Encoding**.
    b. Numerical features:
        i. Missing values handled using **KNN Imputer**.
        ii. Scaled using **StandardScaler**.
7. **Data Cleaning & Alignment:**
    a. Ensured train and test datasets had the same columns.
    b. Missing columns in test data were filled with zeros.
8. **Image Preprocessing:**
    a. **Resized** to (224, 224).
    b. **Random horizontal flip** applied for augmentation.
    c. Converted to **PyTorch tensor**.

Class Imbalance Handling Techniques:

- **Random Undersampling:** Majority class reduced to 10% of its original size.
- **Random Oversampling:** Minority class increased to 20% of the dataset.
- **Stratified Splitting:** Maintained class proportions in training and validation sets.

Model Architecture

9. **Input Layer:**
    a. Accepts images of size **(224, 224, 3)**.
    b. Processes patches of size **4x4** through the PatchEmbed2D module.

10. **Patch Embedding:**
    a. Converts image patches into embeddings using PatchEmbed2D.
    b. Optionally applies Layer Normalization (if patch_norm=True).
11. **Dropout and Positional Embeddings:**
    a. A positional embedding (absolute_pos_embed) is defined but not used (self.ape = False).
    b. A dropout layer (Dropout(p=drop_rate)) is applied.
12. **Backbone:**
    a. **4 main layers (VSSLayer)** process the embeddings sequentially.
    b. Layer dimensions: [96, 192, 384, 768].
    c. Depths per layer: [2, 2, 4, 2].
    d. Uses PatchMerging2D for downsampling between layers.
13. **Stochastic Depth Regularization:**
    a. Implements **drop path (stochastic depth) decay** with torch.linspace(0, drop_path_rate, sum(depths)).
14. **Classification Head:**
    a. **Global Average Pooling (AdaptiveAvgPool2d(1))** reduces feature map size.
    b. Fully Connected (Linear(num_features, num_classes)) maps features to class scores.
15. **Weight Initialization:**
    a. Conv2d: Kaiming Normal.
    b. Linear: Truncated Normal.
    c. LayerNorm: Bias set to 0, weight set to 1.
16. **Forward Pass:**
    a. Extracts features via forward_backbone.
    b. Transposes (permute(0,3,1,2)) to match PyTorch conventions.
    c. Applies **Global Average Pooling** and **Linear Classification Head**.

Model Metrics

- **Loss Function:** CrossEntropyLoss() (suitable for multi-class classification).
- **Optimizer:** Adam with learning rate 0.0001 and weight decay 0.00001 (helps prevent overfitting).
- **Training Loss (Final Epoch):** 0.048334
- **Validation Accuracy (Final Epoch):** 44.89 **%**

Training Methodologies

- **Transfer Learning Approach:**
    - **Pretrained Model:** MedMamba.pth

- **Feature Extraction:** All layers are frozen except for the last linear classification layer.
- **Fine-Tuning:** Only the classification head is trained for binary classification.

- **Optimization Strategy:**
  - Uses Adam optimizer for adaptive learning.
  - **Learning rate:** 0.0001 (relatively small for stable updates).
  - **Weight decay:** 0.00001 to prevent overfitting.
- **Training Strategy:**
  - **Mini-batch training** using train_dataloader.
  - **Validation at each epoch** to monitor accuracy.
  - **Best model saving mechanism** (torch.save) based on validation accuracy.

Model Parameters: Toal count: `14454146`

Resultant scores:

Private score: 0.09374

Public score: 0.09298

Isic Medmamba version – 3

Dataset Size:

- The dataset consists of 45981 training samples (after resampling).
- The validation set contains 5109 **samples** (10% of training data).
- The tabular data has **72 features** after preprocessing.

Preprocessing Methodologies:

9. **Feature Selection:**
   a. Dropped irrelevant columns (patient_id, image_type, tbp_tile_type, etc.).
   b. Dropped train-only columns (lesion_id, iddx_full, etc.).
10. **Feature Engineering:**
    a. Categorical features: Encoded using **One-Hot Encoding**.
    b. Numerical features:
       i. Missing values handled using **KNN Imputer**.
       ii. Scaled using **StandardScaler**.
11. **Data Cleaning & Alignment:**

a. Ensured train and test datasets had the same columns.

b. Missing columns in test data were filled with zeros.

12. **Image Preprocessing:**

    a. **Resized** to (224, 224).

    b. **Random horizontal flip** applied for augmentation.

    c. Converted to **PyTorch tensor**.

Class Imbalance Handling Techniques:

- **Random Undersampling:** Majority class reduced to 10% of its original size.
- **Random Oversampling:** Minority class increased to 30% of the dataset.
- **Stratified Splitting:** Maintained class proportions in training and validation sets.

Model Architecture

1. Input Parameters:

- patch_size=4: The size of patches used for the image input.
- in_chans=3: The number of channels in the input image (RGB image).
- num_classes=1000: Number of output classes for classification.
- depths: A list of integers specifying the number of layers at each stage of the network.
- depths_decoder: Similar to depths, but used for decoder layers.
- dims: A list of integers specifying the dimensionality at each stage.
- dims_decoder: Dimensionality for decoder layers.
- d_state=16: The state dimension, used in certain layers (defaults to 16 if not provided).
- drop_rate, attn_drop_rate, drop_path_rate: Dropout rates for various components of the network.
- norm_layer=nn.LayerNorm: The normalization layer used in the network.
- patch_norm=True: Whether to apply normalization to patches.
- use_checkpoint=False: Whether to use checkpointing for saving intermediate results in the forward pass.

2. Model Components:

- patch_embed: A patch embedding layer that divides the image into patches and maps them to the specified embedding dimension.
- ape: Absolute position embedding, which is optional and not enabled in the code (default is False).
- pos_drop: Dropout applied to the position embeddings.
- layers: A list of VSSLayer objects, where each layer consists of multiple operations including attention and feedforward layers. Each layer has its dropout rate and a downsampling mechanism (PatchMerging2D) for intermediate layers.

- avgpool: Adaptive average pooling layer to reduce the spatial dimensions to 1x1 before flattening the features.

3. Fully Connected Layers for Image Features:

- img_fc1: Fully connected layer for transforming the image features from num_features to 512.
- img_fc2: Fully connected layer for transforming from 512 to 256.

4. Fully Connected Layers for Tabular Features:

- tab_fc1: Fully connected layer for tabular data transformation from 72 to 128 features.
- tab_fc2: Fully connected layer for tabular data transformation from 128 to 64 features.

5. Classification Head:

- classifier: A fully connected layer that takes the concatenated image and tabular features (256 + 64) as input and produces the final classification output with num_classes.

6. Forward Pass:

- forward_backbone(x): The backbone performs the patch embedding, position dropout, and passes the data through the layers.
- classify(x, tab): Takes the image features and tabular data, processes them through respective fully connected layers, and concatenates them for classification.
- forward(x, tab): First passes the image through the backbone, applies average pooling, flattens, and then performs classification by combining image and tabular features.

Model Metrics

- **Loss Function:** CrossEntropyLoss() (suitable for multi-class classification).
- **Optimizer:** Adam with learning rate 0.001 and weight decay 0.0001 (helps prevent overfitting).
- **Training Loss (Final Epoch):** 0.006502
- **Validation loss(**Final Epoch)**:** 0.020319
- **Validation Accuracy (Final Epoch):** 41.26%

Model Parameters: Toal count: 14995906

Private score: 0.10833

Public score: 0.12188

Isic Medmamba version – 4

Dataset Size:

- The dataset consists of 5502 **training samples** (after resampling).

- The validation set contains 551 **samples** (10% of training data).
- The tabular data has **72 features** after preprocessing.

Preprocessing Methodologies:

13. **Feature Selection:**
    a. Dropped irrelevant columns (patient_id, image_type, tbp_tile_type, etc.).
    b. Dropped train-only columns (lesion_id, iddx_full, etc.).
14. **Feature Engineering:**
    a. Categorical features: Encoded using **One-Hot Encoding**.
    b. Numerical features:
        i. Missing values handled using **KNN Imputer**.
        ii. Scaled using **StandardScaler**.
15. **Data Cleaning & Alignment:**
    a. Ensured train and test datasets had the same columns.
    b. Missing columns in test data were filled with zeros.
16. **Image Preprocessing:**
    a. **Resized** to (224, 224).
    b. **Random horizontal flip** applied for augmentation.
    c. Converted to **PyTorch tensor**.

Class Imbalance Handling Techniques:

- **Random Undersampling:** Majority class reduced to 10% of its original size.
- **Random Oversampling:** Minority class increased to 40% of the dataset.
- **Stratified Splitting:** Maintained class proportions in training and validation sets.

Model Architecture

1. Input Parameters:

- patch_size=4: The size of patches used for the image input.
- in_chans=3: The number of channels in the input image (RGB image).
- num_classes=1000: Number of output classes for classification.
- depths: A list of integers specifying the number of layers at each stage of the network.
- depths_decoder: Similar to depths, but used for decoder layers.
- dims: A list of integers specifying the dimensionality at each stage.
- dims_decoder: Dimensionality for decoder layers.
- d_state=16: The state dimension, used in certain layers (defaults to 16 if not provided).

- drop_rate, attn_drop_rate, drop_path_rate: Dropout rates for various components of the network.
- norm_layer=nn.LayerNorm: The normalization layer used in the network.
- patch_norm=True: Whether to apply normalization to patches.
- use_checkpoint=False: Whether to use checkpointing for saving intermediate results in the forward pass.

2. Model Components:

- patch_embed: A patch embedding layer that divides the image into patches and maps them to the specified embedding dimension.
- ape: Absolute position embedding, which is optional and not enabled in the code (default is False).
- pos_drop: Dropout applied to the position embeddings.
- layers: A list of VSSLayer objects, where each layer consists of multiple operations including attention and feedforward layers. Each layer has its dropout rate and a downsampling mechanism (PatchMerging2D) for intermediate layers.
- avgpool: Adaptive average pooling layer to reduce the spatial dimensions to 1x1 before flattening the features.

3. Fully Connected Layers for Image Features:

- img_fc1: Fully connected layer for transforming the image features from num_features to 512.
- img_fc2: Fully connected layer for transforming from 512 to 256.

4. Fully Connected Layers for Tabular Features:

- tab_fc1: Fully connected layer for tabular data transformation from 72 to 128 features.
- tab_fc2: Fully connected layer for tabular data transformation from 128 to 64 features.

5. Classification Head:

- classifier: A fully connected layer that takes the concatenated image and tabular features (256 + 64) as input and produces the final classification output with num_classes.

6. Forward Pass:

- forward_backbone(x): The backbone performs the patch embedding, position dropout, and passes the data through the layers.
- classify(x, tab): Takes the image features and tabular data, processes them through respective fully connected layers, and concatenates them for classification.
- forward(x, tab): First passes the image through the backbone, applies average pooling, flattens, and then performs classification by combining image and tabular features.

Model Metrics

- **Loss Function:** CrossEntropyLoss() (suitable for multi-class classification).

- **Optimizer:** Adam with learning rate 0.0001 and weight decay 0.00001 (helps prevent overfitting).
- **Training Loss (Final Epoch):** 0.026236
- **Validation loss(**Final Epoch)**:** 0.261622
- **Validation Accuracy (Final Epoch):** 38.40%

Model Parameters: Toal count: 14995906

Private score: 0.12280

Public score: 0.13613

Isic Medmamba version – 6

Dataset Size:

- The dataset consists of 7781 **training samples** (after resampling).
- The validation set contains 865 samples (10% of training data).
- The tabular data has **72 features** after preprocessing.

Preprocessing Methodologies:

17. **Feature Selection:**
    a. Dropped irrelevant columns (patient_id, image_type, tbp_tile_type, etc.).
    b. Dropped train-only columns (lesion_id, iddx_full, etc.).
18. **Feature Engineering:**
    a. Categorical features: Encoded using **One-Hot Encoding**.
    b. Numerical features:
        i. Missing values handled using **KNN Imputer**.
        ii. Scaled using **StandardScaler**.
19. **Data Cleaning & Alignment:**
    a. Ensured train and test datasets had the same columns.
    b. Missing columns in test data were filled with zeros.
20. **Image Preprocessing:**
    a. **Resized** to (224, 224).
    b. **Random horizontal flip** applied for augmentation.

c. Converted to **PyTorch tensor**.

Class Imbalance Handling Techniques:

- **Random Undersampling:** Majority class reduced to 5% of its original size.
- **Random Oversampling:** Minority class increased to 10% of the dataset.
- **Stratified Splitting:** Maintained class proportions in training and validation sets.

Model Architecture

1.1 Input and Patch Embedding

- The model takes RGB images (3 channels) and tabular data (72 features).
- The image input is processed using a patch embedding layer (PatchEmbed2D), which divides the image into patches and projects them into an embedding space.
- The positional embedding (absolute_pos_embed) is defined but disabled (self.ape = False).

1.2 Backbone (Vision Feature Extractor)

- The model has multiple layers (self.layers), where each layer is an instance of VSSLayer.
- VSSLayer operates as the primary vision feature extractor and contains:
    - A state-space-based architecture (likely using Mamba-style sequence processing).
    - Patch merging (PatchMerging2D) for downsampling in earlier stages.
    - Dropout (drop_rate) and attention dropout (attn_drop_rate) for regularization.
- The output features of the vision backbone are pooled using nn.AdaptiveAvgPool2d(1) and flattened before classification.

1.3 Fully Connected Layers for Image and Tabular Features

- Image Feature Extraction:
    - img_fc1: Maps vision features to 1024 neurons.
    - img_fc2: Maps 1024 → 512 neurons.
- Tabular Feature Extraction:
    - tab_fc1: Maps 72 tabular features to 128 neurons.
    - tab_fc2: Maps 128 → 256 neurons.
- All layers use:
    - SELU activation (F.selu).
    - Dropout (30%) for regularization.

1.4 Feature Fusion and Classification

- The image and tabular features are concatenated (torch.cat).
- Two additional fusion layers (concat1, concat2) refine the merged features:

    - concat1: 512 (image) + 256 (tabular) → 128
    - concat2: 128 → 64
- The final classifier (self.classifier) predicts num_classes outputs.

2. Forward Pass

- Image processing:
    - Passes through patch_embed, self.layers, and AdaptiveAvgPool2d.
    - Flattened into a 1D feature vector.
- Tabular data processing:
    - Passes through tab_fc1 and tab_fc2 layers.
- Feature fusion and classification:
    - Concatenates image and tabular features.
    - Processes through concat1, concat2.
    - Passes through the final classifier.

3. Summary of Model Architecture

Component	Description
Input	Image (3xHxW), Tabular (72 features)
Patch Embedding	Converts image into patches (PatchEmbed2D)
Backbone (VSSLayer)	Mamba-based vision layers (state-space modeling)
Pooling & Flattening	AdaptiveAvgPool2d(1), Flatten
Image Feature Extraction	img_fc1 (1024), img_fc2 (512)
Tabular Feature Extraction	tab_fc1 (128), tab_fc2 (256)
Fusion Layers	concat1 (128), concat2 (64)
Classifier	Fully connected layer (num_classes)

Model Metrics

- **Loss Function:** CrossEntropyLoss() (suitable for multi-class classification).
- **Optimizer:** Adam with learning rate 0.001 and weight decay 0.00001 (helps prevent overfitting).
- **Training Loss (Final Epoch):** 0.253860
- **Validation loss(**Final Epoch)**:** 0.332256
- **Validation Accuracy (Final Epoch):** 46.70%

Model Parameters: Toal count: 15914050

Resultant scores:

Private score: 0.11579

Public score: 0.13825