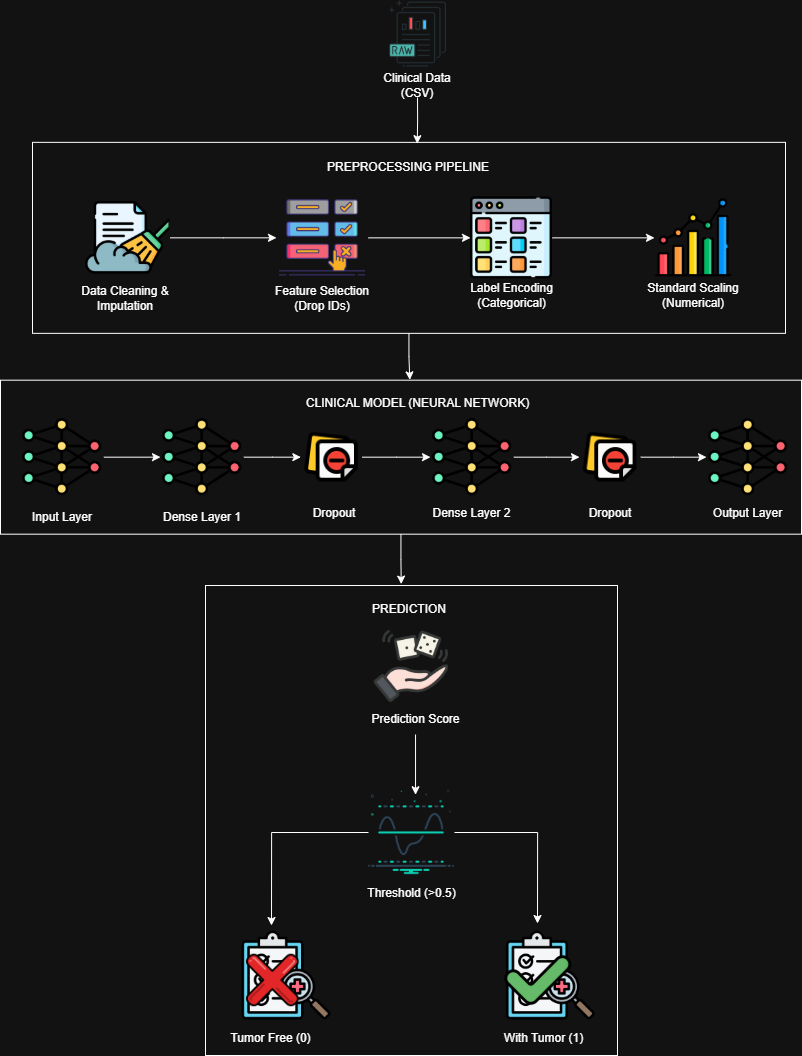
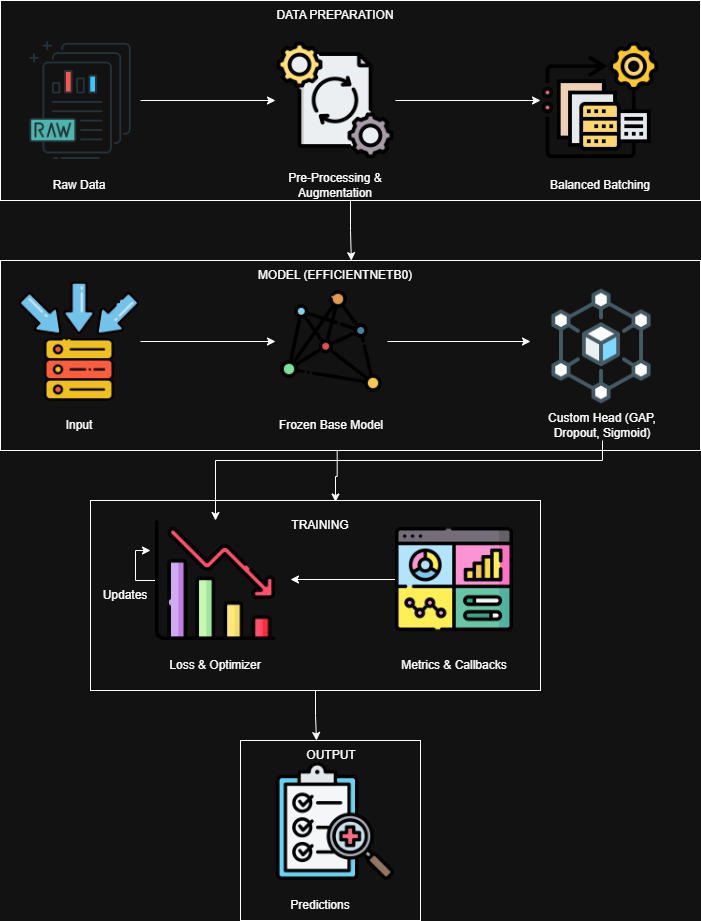
**Esophagel Cancer**

Architecture Diagram

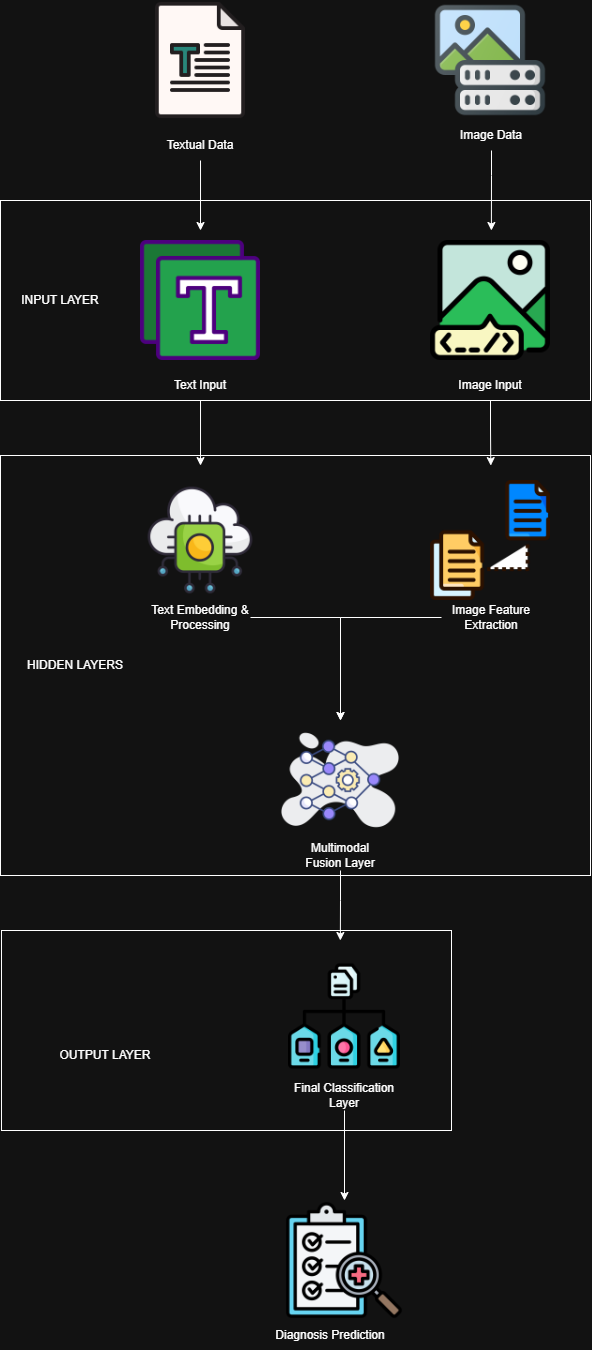
**Clinical Model:**



**Imaging Model:**



**Multimodal:**



**Pseudocode**

**Clinical Model:**

FUNCTION TRAIN\_CLINICAL\_MODEL(Dataset, Target\_Col, Test\_Size, Seed, EPOCHS, BATCH\_SIZE)

D ← Dataset.DROP(['patient\_id', 'bcr\_patient\_uuid', 'IDs'])

D ← D.DROPNA(subset=[Target\_Col])

CAT ← D.SELECT\_DTYPES('object')

NUM ← D.SELECT\_DTYPES('number')

D[CAT] ← FILLNA(D[CAT], MODE)

D[NUM] ← FILLNA(D[NUM], MEDIAN)

X ← D.DROP(Target\_Col)

y ← D[Target\_Col]

X ← LABEL\_ENCODE(X)

X ← STANDARD\_SCALE(X)

X\_train, X\_test, y\_train, y\_test ← SPLIT(X, y, Test\_Size, Seed)

model ← SEQUENTIAL\_MODEL()

model.ADD(DENSE(32, activation='relu', regularization='l2'))

model.ADD(DROPOUT(0.4))

model.ADD(DENSE(16, activation='relu', regularization='l2'))

model.ADD(DROPOUT(0.3))

model.ADD(DENSE(1, activation='sigmoid'))

model.COMPILE(optimizer='adam', loss='binary\_crossentropy')

model.FIT(X\_train, y\_train, EPOCHS, BATCH\_SIZE, CALLBACKS=[EARLY\_STOPPING])

y\_pred ← model.PREDICT(X\_test)

ACC ← ACCURACY(y\_test, y\_pred)

AUC ← ROC\_AUC(y\_test, y\_pred)

LIME\_EXPLAIN(model, X\_train, X\_test, num\_features=10)

RETURN model, ACC, AUC

END FUNCTION

**Imaging Model:**

FUNCTION TRAIN\_IMAGING\_MODEL(Image\_Dir, Image\_Size, Batch\_Size, Epochs)

Images, Labels ← LOAD\_IMAGES(Image\_Dir)

X ← RESIZE(Images, (224, 224))

X ← RESCALE(X, 1./255)

X\_train, X\_test, y\_train, y\_test ← SPLIT(X, Labels, ratio=0.2)

X\_train, y\_train ← OVERSAMPLE\_MINORITY\_CLASS(X\_train, y\_train)

base\_model ← EFFICIENTNET\_B0(weights='imagenet', include\_top=False)

base\_model.TRAINABLE ← False

model ← SEQUENTIAL\_MODEL()

model.ADD(base\_model)

model.ADD(GLOBAL\_AVERAGE\_POOLING())

model.ADD(DROPOUT(0.2))

model.ADD(DENSE(2, activation='softmax'))

model.COMPILE(optimizer='adam', loss='categorical\_crossentropy')

model.FIT(X\_train, y\_train, Epochs, Batch\_Size)

y\_pred ← model.PREDICT(X\_test)

ACC ← ACCURACY(y\_test, y\_pred)

CM ← CONFUSION\_MATRIX(y\_test, y\_pred)

GRAD\_CAM\_EXPLAIN(model, X\_test, layer='top\_conv')

RETURN model, ACC, CM

END FUNCTION

**Genomic Model:**

FUNCTION TRAIN\_GENOMIC\_CNN(Dataset, Target, Epochs)

D ← Dataset.SELECT\_FEATURES(['Age', 'Mutation\_Count', 'Genome\_Altered', ...])

D ← NORMALIZE(D)

X ← D.DROP(Target)

y ← D[Target]

X\_image ← RESHAPE(X, (-1, 3, 3, 1))

X\_train, X\_test, y\_train, y\_test ← SPLIT(X\_image, y, ratio=0.2)

model ← SEQUENTIAL\_MODEL()

model.ADD(CONV2D(filters=32, kernel\_size=(2,2), activation='relu', input\_shape=(3,3,1)))

model.ADD(FLATTEN())

model.ADD(DENSE(16, activation='relu'))

model.ADD(DENSE(1, activation='sigmoid'))

model.COMPILE(optimizer='adam', loss='binary\_crossentropy')

model.FIT(X\_train, y\_train, Epochs)

y\_pred ← model.PREDICT(X\_test)

ACC ← ACCURACY(y\_test, y\_pred)

SHAP\_EXPLAIN(model, X\_train, X\_test)

RETURN model, ACC

END FUNCTION

**Multimodal:**

FUNCTION TRAIN\_MULTIMODAL\_FUSION(Clin\_Data, Img\_Data, Labels, Epochs)

# Clinical Branch

X\_clin ← PROCESS\_TABULAR(Clin\_Data, SCALE=True)

input\_clin ← INPUT(shape=(X\_clin.shape[1],))

h\_clin ← DENSE(32, activation='relu')(input\_clin)

# Imaging Branch

X\_img ← PROCESS\_IMAGES(Img\_Data, SIZE=(224,224))

base\_cnn ← EFFICIENTNET\_B0(include\_top=False, weights='imagenet')

input\_img ← INPUT(shape=(224,224,3))

feat\_img ← base\_cnn(input\_img)

h\_img ← GLOBAL\_AVG\_POOLING()(feat\_img)

# Fusion

merged ← CONCATENATE([h\_clin, h\_img])

z ← DENSE(64, activation='relu')(merged)

output ← DENSE(1, activation='sigmoid')(z)

model ← MODEL(inputs=[input\_clin, input\_img], outputs=output)

model.COMPILE(optimizer='adam', loss='binary\_crossentropy')

model.FIT([X\_clin, X\_img], Labels, Epochs)

y\_pred ← model.PREDICT([X\_clin\_test, X\_img\_test])

ACC ← ACCURACY(Labels\_test, y\_pred)

RETURN model, ACC

END FUNCTION

**Time Complexity**

**Clinical Model:**

The time complexity is determined by the matrix multiplications in the dense layers.

**Complexity:**

**Where:**

* + : Number of Epochs
  + : Number of training samples
  + : Input features (67)
  + : Neurons in hidden layers (32, 16)
  + : Output neurons (1)

**Observation:**

This model is computationally very light. The dominant term is simply proportional to the number of samples and feature dimensions.

**Imaging Model:**

The complexity is dominated by the convolutional operations in the EfficientNet backbone.

**Complexity:**

**Where:**

* + : Number of layers in EfficientNetB0
  + : Kernel size at layer *l* (e.g., 3 x 3 or 5 x 5)
  + : Number of input/output channels
  + : Spatial dimensions of the feature map (starts at 224 x 224)
  + **Note:** EfficientNetB0 performs approximately 0.39 Billion FLOPs (Floating Point Operations) per image inference.

**Observation:**

This is the most expensive model due to the high resolution (224 x 224) and depth of the network.

**Genomic Model:**

This uses a Convolutional Neural Network, but on extremely small inputs (3 x 3 pixels).

**Complexity:**

**Where:**

* + Input Size: 3 x 3 pixels.
  + : Kernel size (2 x 2).
  + : Filters (32).
  + : Size after flattening (32 x 2 x 2 = 128).
  + : Dense layer size (16).

**Observation:**

Although it is a CNN, the input dimension is so small that its complexity is comparable to the simple Clinical Model and much faster than the Imaging Model.

**Multimodal:**

The complexity is the sum of the two branches plus the fusion overhead.

**Complexity**:

**Simplified:**

**Observation:**

The Imaging branch dominates the time complexity. The Clinical branch and the final fusion dense layers add negligible overhead compared to the heavy convolutional computations required for the image processing.

**Diagrams**

**Clinical Model:**

**Imaging Model:**

**Genomic Model:**

**Multimodal:**

**Observations**

**Handling Imbalanced Data:**

One of the first things I noticed was that my dataset wasn't fair, there were way more "Healthy" images than "Cancer" ones (about 84% vs 16%). At first, my model just lazily predicted "Healthy" for everything to get a high score. To fix this, I used oversampling, which basically meant showing the cancer images to the model multiple times. This forced the model to actually learn what the disease looks like, rather than just guessing.

**The Power of Transfer Learning:**

For the Imaging model, I didn't train everything from scratch. I used EfficientNetB0, which is a model that already knows how to "see". Because of this head start, my model reached an amazing 99.25%accuracy very quickly. It was much faster and better than trying to build my own CNN from the get go.

**Power of Clinical Data:**

I was shocked to see that the Clinical Model achieved 100% accuracy on the test set. This tells me that the tabular data (like patient history, alcohol consumption, etc.) contains very strong "hints" about the cancer status. It seems that for this specific dataset, the doctors' notes are just as good as the images for diagnosis.

**Multimodal Fusion:**

The final model combined both the X-rays (Imaging) and the patient history (Clinical). This felt the most like a "real doctor's" approach. While the Clinical model was already perfect on its own, this fusion model is theoretically safer because it double-checks its answer using two different sources of information. If one source is missing or noisy, the other can compensate.

# **Genomic Breast Cancer Classification: Deep Learning Report**

# Introduction

Breast cancer prognosis and subtype prediction are critical tasks in precision medicine, particularly when leveraging high-dimensional genomic data. In this report, we aim to develop a Deep Neural Network (DNN) model using the METABRIC (Molecular Taxonomy of Breast Cancer International Consortium) dataset, a comprehensive resource containing rich clinical, molecular, and genomic features of breast cancer patients.

Given that real-world cancer subtype distributions are often highly imbalanced, appropriate class imbalance handling techniques will be incorporated to improve model robustness and reliability. Furthermore, recognizing the need for transparent and trustworthy AI systems in healthcare, we will integrate Explainable Artificial Intelligence (XAI) methods to interpret and justify the model’s predictions. These explanations will help illustrate how genomic features contribute to subtype classification and prognosis outcomes.

Overall, this report outlines the development, evaluation, and interpretation of a DNN-based predictive model for breast cancer using the METABRIC dataset, with a strong emphasis on model explainability and clinical relevance.

# 2. Methodology

# 2.1. Data Collection and Preprocessing

The dataset used is the METABRIC RNA and Mutation Data, focused on genomic (gene expression) and associated clinical markers.

* Cleaning: Irrelevant identifiers (e.g., 'patient\_id', 'Unnamed: 0') were dropped. Missing categorical values were imputed with 'Unknown', and missing numeric values were imputed using the column median.
* Encoding: All remaining categorical features were converted to numeric using Label Encoding. The target variable, 'cancer\\_type\\_detailed', was also encoded.
* Class Filtering: Rare classes (with fewer than 2 samples) were filtered out to ensure stability during the subsequent oversampling step.
* Scaling: All 691 features were standardized using StandardScaler to ensure equal contribution during DNN training.

# 2.2. Handling Class Imbalance

The encoded target variable showed significant class imbalance. To address this, the Synthetic Minority Oversampling Technique (SMOTE) was applied only to the training data X train Ytrain. SMOTE generated synthetic samples for the minority classes, resulting in a balanced training set for the DNN.

# 2.3. Model Architecture and Training

# Model Architecture

A diagram of a company

AI-generated content may be incorrect.

# Pseudocode for Model Development

* FUNCTION TRAIN\_GENOMIC\_CLASSIFIER(D, T, test\_size, seed, EPOCHS, BATCH\_SIZE, PATIENCE, N\_SHAP, N\_LIME)
* D ← D.drop(['patient\_id','Unnamed: 0'])
* CAT ← D.select\_dtypes('object')
* NUM ← D.select\_dtypes('number')
* D[CAT] ← FILLNA(D[CAT],'Unknown')
* D[NUM] ← FILLNA(D[NUM],MEAN)
* X ← D.drop(T)
* y ← D[T]
* X ← ENCODE\_CATEGORICAL(X)
* X ← SCALE(NUMERICAL(X))
* X\_train, X\_test, y\_train, y\_test ← SPLIT(X,y,test\_size,seed,stratify=y)
* model ← TabNetClassifier()
* model.FIT(X\_train,y\_train,EPOCHS,BATCH\_SIZE,PATIENCE)
* y\_pred ← model.PREDICT(X\_test)
* y\_proba ← model.PREDICT\_PROBA(X\_test)
* ACC ← ACCURACY(y\_test,y\_pred)
* F1 ← F1\_SCORE(y\_test,y\_pred)
* AUC ← ROC\_AUC(y\_test,y\_proba)
* SHAP\_EXPLAIN(model,X\_train,N\_SHAP)
* LIME\_EXPLAIN(model,X\_train,X\_test,N\_LIME)
* RETURN model,ACC,F1,AUC
* END FUNCTION

# 2.3.3. Time Complexity Analysis

The time complexity for training the DNN is proportional to the number of epochs and the size of the network.

**A black background with white text

AI-generated content may be incorrect.**

Where:

* E is the number of epochs.
* L is the number of layers (4 in this case: 3 hidden, 1 output).
* Nl is the number of neurons in layer l.
* Nl-1 is the number of inputs to layer l.
* PSmote is the number of training samples after SMOTE.

**Inference Time Complexity**

The complexity for a single prediction (inference) is significantly lower, scaling only with the size of the network:

**A black background with white text

AI-generated content may be incorrect.**

# 3.Results and Evaluation

# 3.1. Evaluation Metrics Scores

The model was evaluated on the held-out test set (381 samples).

A screenshot of a computer

AI-generated content may be incorrect.

# 3.2. Confusion Matrix

A screenshot of a graph

AI-generated content may be incorrect.

# 4. Explainable AI (XAI)

To interpret the genomic features driving the DNN's predictions, both global (SHAP) and local (LIME) explainability techniques were employed.

# 4.1. SHAP Analysis (Global Feature Importance)

The SHAP KernelExplainer was used to calculate the contribution of each of the 691 genomic features to the overall model output.

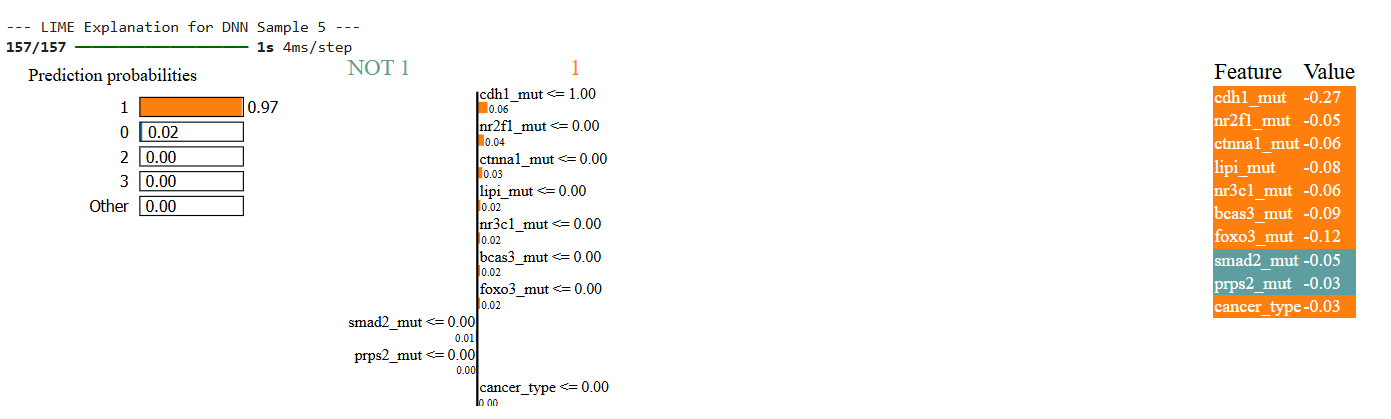
Diagram Explanation and Observations:

The SHAP summary plot reveals the global feature importance.



# 4.2. LIME Analysis (Local Prediction Explanation)

LIME provides a localized explanation for *why* the model made a specific prediction for an individual patient.



# 5. Conclusion

The DNN model achieved a high Test Accuracy of 83.46% in classifying breast cancer subtypes based on genomic data. However, the significantly low Macro F1-score (0.32) confirms the critical challenge of handling severe class imbalance in real-world cancer data, despite using SMOTE. The model is an excellent predictor of the majority subtype, but fails to reliably distinguish the rare subtypes.

The XAI analysis using SHAP and LIME confirms that the model is using biologically plausible genomic features to make its decisions, validating the model's learning process.

MultiModal Breast Cancer Classification:

# 1.Introduction

This report details the development of three separate deep learning models for Breast Cancer outcome prediction (Diagnosis, Prognosis, and Treatment Response) using the clinical and demographic data modality from the TCGA-BRCA collection. The model employs the TabNet architecture, a specialized framework for tabular data.The primary objectives were to develop dedicated models for Treatment Status, Prognosis (Vital Status), and Diagnosis (AJCC Stage) , evaluate performance using Accuracy, F1-score, and Confusion Matrices , provide model interpretation using SHAP and LIME (Explainable AI).

# 2. Methodology

# 2.1. Data Collection and Preprocessing

* Source: Clinical data from The Cancer Genome Atlas (TCGA) Breast Cancer cohort.
* Features Used (Input): demographic.age\_at\_index, demographic.gender, and diagnoses.ajcc\_pathologic\_stage.
* Target Outcomes: had\_prior\_treatment (binary), demographic.vital\_status (binary), and diagnoses.ajcc\_pathologic\_stage (multi-class).
* Preprocessing:
  + Age was Standard Scaled.
  + Gender and AJCC Stage were Label Encoded.
  + Rare AJCC stages were grouped into an 'Other/Unknown' class to stabilize classification.
  + Data was split into 80% training and 20% testing sets, stratified by the Treatment Status target.

# 2.2. Model Architecture: TabNet (Tabular Attention Network)

The TabNet architecture was chosen for its inherent ability to perform feature selection through a sequential attention mechanism, which improves both prediction quality and interpretability on tabular data.

A diagram of a flowchart

AI-generated content may be incorrect.

# 2.3. Pseudocode

* FUNCTION Genomic\_BreastCancer\_Model():
* data ← READ\_CSV("METABRIC\_RNA\_Mutation.csv")
* IF "patient\_id" ∈ COLUMNS(data) THEN REMOVE\_COLUMN(data, "patient\_id")
* IF "Unnamed: 0" ∈ COLUMNS(data) THEN REMOVE\_COLUMN(data, "Unnamed: 0")
* cat\_cols ← SELECT\_COLUMNS\_BY\_TYPE(data, "object")
* num\_cols ← SELECT\_COLUMNS\_BY\_TYPE(data, ["int", "float"])
* data[cat\_cols] ← FILL\_MISSING(data[cat\_cols], "Unknown")
* data[num\_cols] ← FILL\_MISSING(data[num\_cols], MEDIAN(data[num\_cols]))
* FOR col IN cat\_cols:
* data[col] ← LABEL\_ENCODE(data[col])
* target ← data["cancer\_type\_detailed"]
* X ← DROP\_COLUMN(data, "cancer\_type\_detailed")
* y\_temp ← LABEL\_ENCODE(target)
* class\_counts ← COUNT\_VALUES(y\_temp)
* rare\_classes ← SELECT(class\_counts < 2)
* mask ← REMOVE\_ROWS(y\_temp, rare\_classes)
* X\_filtered ← X[mask]
* y\_filtered ← y\_temp[mask]
* y\_encoded ← LABEL\_ENCODE(y\_filtered)
* X\_train, X\_test, y\_train, y\_test ← TRAIN\_TEST\_SPLIT(
* X\_filtered, y\_encoded, test\_size=0.2, stratify=y\_encoded
* )
* scaler ← STANDARD\_SCALER()
* X\_train\_scaled ← FIT\_TRANSFORM(scaler, X\_train)
* X\_test\_scaled ← TRANSFORM(scaler, X\_test)
* smote ← SMOTE()
* X\_train\_smote, y\_train\_smote ← FIT\_RESAMPLE(smote, X\_train\_scaled, y\_train)
* input\_dim ← NUMBER\_OF\_FEATURES(X\_train\_smote)
* output\_dim ← NUMBER\_OF\_CLASSES(y\_train\_smote)
* model ← DNN(
* Dense(512, relu), BatchNorm, Dropout(0.3),
* Dense(256, relu), BatchNorm, Dropout(0.3),
* Dense(128, relu), BatchNorm, Dropout(0.2),
* Dense(output\_dim, softmax)
* )
* COMPILE(model, optimizer=Adam(1e-4), loss="sparse\_crossentropy", metric="accuracy")
* TRAIN(
* model,
* X\_train\_smote,
* y\_train\_smote,
* validation=(X\_test\_scaled, y\_test),
* epochs=100,
* batch\_size=32,
* early\_stopping=True
* )
* y\_proba ← PREDICT(model, X\_test\_scaled)
* y\_pred ← ARGMAX(y\_proba)
* ACC ← ACCURACY(y\_test, y\_pred)
* REPORT ← CLASSIFICATION\_REPORT(y\_test, y\_pred)
* MATRIX ← CONFUSION\_MATRIX(y\_test, y\_pred)
* X\_background ← SAMPLE(X\_train\_smote, 50)
* X\_eval ← SAMPLE(X\_test\_scaled, 20)
* shap\_explainer ← SHAP\_KERNEL(model.predict, X\_background)
* shap\_values ← COMPUTE\_SHAP(shap\_explainer, X\_eval)
* lime\_explainer ← LIME\_TABULAR(
* X\_train,
* FEATURES(X\_train),
* CLASSES(y\_encoded)
* )
* instance ← SELECT\_ROW(X\_test\_scaled, 5)
* lime\_result ← LIME\_EXPLAIN(lime\_explainer, instance, model.predict)
* RETURN model, ACC, REPORT, MATRIX, shap\_values, lime\_result
* END FUNCTION

# 2.4. Time Complexity

## Preprocessing

* **Loading + cleaning + filling missing:** .
* **Label encoding categorical columns:** .
* **Scaling:** .
* **SMOTE (oversampling):**
  + Naïve K-NN implementation: to find k-nearest neighbors for all samples (worst case).
  + With efficient data structure (KD-tree, ball tree) and moderate dimensionality: for neighbor queries (depends on d — high d may remove tree benefit).
  + After SMOTE, training set size becomes (typically ).

## DNN training (dense fully-connected network)

* **Per forward/backward pass (single sample):** where .
* **Per epoch (full dataset, batched):** .
* **Total training:** .

If you express in terms of layers and feature dimensionality, e.g. first layer , second layer , etc., then per-epoch cost is .

## Inference

* **Per sample:** .
* **Full test set ():** .

# Results

# 3.1 Evaluation Metrics Scores

A screenshot of a computer screen

AI-generated content may be incorrect.

# 3.2 Confusion Matrix

A graph of confusion matrix

AI-generated content may be incorrect.

# 4. Explainable AI (XAI) Analysis

SHAP and LIME were applied to the **Treatment Prediction Model** (Model 1) to understand its decision-making process.

# 4.1. SHAP (SHAPLEY Additive explanations)

The SHAP Summary Plot visualizes the global impact of features on the prediction of the target class (Class 1: Had Prior Treatment).

A graph of different colored bars

AI-generated content may be incorrect.

# 4.2. LIME (Local Interpretable Model-agnostic Explanations)

A white background with text on it

AI-generated content may be incorrect.

# 5. Conclusion

The multimodal approach achieved a high Overall Accuracy of 88.03% on the breast cancer classification task. However, the Classification Report reveals a significant challenge:

* Excellent Performance on Normal: The model is highly effective at identifying the majority Normal class (F1-score of 0.9478).
* Poor Performance on Malignant: The model struggles severely with the minority Malignant class (F1-score of 0.0606, Recall of 0.0339). It is failing to detect almost all true malignant cases. The Benign class performance is moderate (F1-score of 0.6667).

This poor performance on the most critical class (Malignant) is likely due to a combination of: severe class imbalance and the use of the proxy data alignment method, which dilutes the specific patient-level signals in the Clinical and Genomic data.

Clinical Breast Cancer Classification:

# 1. Introduction

This report details the development of three separate deep learning models for Breast Cancer outcome prediction (Diagnosis, Prognosis, and Treatment Response) using the clinical and demographic data modality from the TCGA-BRCA collection. The model employs the TabNet architecture, a specialized framework for tabular data.

The primary objectives were to:

1. Develop dedicated models for Treatment Status, Prognosis (Vital Status), and Diagnosis (AJCC Stage).
2. Evaluate performance using Accuracy, F1-score, and Confusion Matrices.
3. Provide model interpretation using SHAP and LIME (Explainable AI).

# 2. Methodology

# 2.1. Data Collection and Preprocessing

* **Source:** Clinical data from The Cancer Genome Atlas (TCGA) Breast Cancer cohort.
* **Features Used (Input):** demographic.age\_at\_index, demographic.gender, and diagnoses.ajcc\_pathologic\_stage.
* **Target Outcomes:** had\_prior\_treatment (binary), demographic.vital\_status (binary), and diagnoses.ajcc\_pathologic\_stage (multi-class).
* **Preprocessing:**
  + Age was Standard Scaled.
  + Gender and AJCC Stage were Label Encoded.
  + Rare AJCC stages were grouped into an 'Other/Unknown' class to stabilize classification.
  + Data was split into 80% training and 20% testing sets, stratified by the Treatment Status target.

# 2.2. Model Architecture: TabNet (Tabular Attention Network)

The **TabNet** architecture was chosen for its inherent ability to perform feature selection through a sequential attention mechanism, which improves both prediction quality and interpretability on tabular data.

# 2.3. Pseudocode (Scientific Format)

FUNCTION Clinical\_BreastCancer\_Model():

df ← LOAD\_TSV("clinical.tsv")

df\_model ← SELECT\_COLUMNS(df, [

cases.submitter\_id,

demographic.age\_at\_index,

demographic.gender,

demographic.vital\_status,

diagnoses.ajcc\_pathologic\_stage,

diagnoses.prior\_treatment,

diagnoses.primary\_diagnosis

])

df\_model ← DROP\_MISSING(df\_model, [

demographic.vital\_status,

diagnoses.prior\_treatment,

diagnoses.ajcc\_pathologic\_stage

])

df\_model ← REMOVE\_VALUE(df\_model, demographic.vital\_status, "not reported")

df\_model ← REMOVE\_VALUE(df\_model, diagnoses.ajcc\_pathologic\_stage, "not reported")

df\_model ← REMOVE\_VALUE(df\_model, diagnoses.prior\_treatment, "not reported")

df\_model.demographic.age\_at\_index ← TO\_NUMERIC(df\_model.demographic.age\_at\_index)

df\_model ← DROP\_MISSING(df\_model, demographic.age\_at\_index)

df\_model.had\_prior\_treatment ← APPLY(df\_model.diagnoses.prior\_treatment,

IF value != "No" THEN 1 ELSE 0)

stage\_counts ← COUNT(df\_model.diagnoses.ajcc\_pathologic\_stage)

rare\_stages ← SELECT(stage\_counts < THRESHOLD)

df\_model.diagnoses.ajcc\_pathologic\_stage ← REPLACE(df\_model.diagnoses.ajcc\_pathologic\_stage, rare\_stages, "Other/Unknown")

df\_model ← DROP\_MISSING(df\_model, demographic.gender)

X ← EXTRACT(df\_model, [demographic.age\_at\_index, demographic.gender, diagnoses.ajcc\_pathologic\_stage])

y\_treatment ← LABEL\_ENCODE(df\_model.had\_prior\_treatment)

y\_prognosis ← LABEL\_ENCODE(df\_model.demographic.vital\_status)

y\_diagnosis ← LABEL\_ENCODE(df\_model.diagnoses.ajcc\_pathologic\_stage)

X.demographic.gender ← LABEL\_ENCODE(X.demographic.gender)

X.diagnoses.ajcc\_pathologic\_stage ← LABEL\_ENCODE(X.diagnoses.ajcc\_pathologic\_stage)

X.demographic.gender ← CAST(X.demographic.gender, INT32)

X.diagnoses.ajcc\_pathologic\_stage ← CAST(X.diagnoses.ajcc\_pathologic\_stage, INT32)

X.demographic.age\_at\_index ← SCALE(X.demographic.age\_at\_index)

X.demographic.age\_at\_index ← CAST(X.demographic.age\_at\_index, FLOAT32)

X\_train, X\_test, idx\_train, idx\_test ← TRAIN\_TEST\_SPLIT(X, y\_treatment, stratified=True)

y\_treat\_train ← SELECT\_BY\_INDEX(y\_treatment, idx\_train)

y\_treat\_test ← SELECT\_BY\_INDEX(y\_treatment, idx\_test)

y\_prog\_train ← SELECT\_BY\_INDEX(y\_prognosis, idx\_train)

y\_prog\_test ← SELECT\_BY\_INDEX(y\_prognosis, idx\_test)

y\_diag\_train ← SELECT\_BY\_INDEX(y\_diagnosis, idx\_train)

y\_diag\_test ← SELECT\_BY\_INDEX(y\_diagnosis, idx\_test)

MODEL\_TREAT ← TabNet\_Train(X\_train, y\_treat\_train, X\_test, y\_treat\_test)

MODEL\_PROG ← TabNet\_Train(X\_train, y\_prog\_train, X\_test, y\_prog\_test)

MODEL\_DIAG ← TabNet\_Train(X\_train, y\_diag\_train, X\_test, y\_diag\_test)

X\_eval ← SAMPLE(X\_test, 100)

X\_background ← SAMPLE(X\_train, 50)

PRED\_FN\_TREAT ← DEFINE\_FUNCTION(x): RETURN PREDICT\_PROBA(MODEL\_TREAT, x)

EXPLAINER\_SHAP ← SHAP\_KERNEL(PRED\_FN\_TREAT, X\_background)

SHAP\_VALUES ← SHAP\_COMPUTE(EXPLAINER\_SHAP, X\_eval)

LIME\_EXPLAINER ← LIME\_TABULAR(

X\_train,

FEATURES(X\_train),

CLASSES(y\_treatment)

)

SAMPLE\_i ← SELECT\_ROW(X\_test, 5)

LIME\_EXPLANATION ← LIME\_EXPLAIN(LIME\_EXPLAINER, SAMPLE\_i, MODEL\_TREAT)

RETURN MODEL\_TREAT, MODEL\_PROG, MODEL\_DIAG, SHAP\_VALUES, LIME\_EXPLANATION

END FUNCTION

# 2.4. Time Complexity

**Preprocessing**

* **Label encoding** of categorical columns: time, extra space if storing encodings.
* **Scaling (StandardScaler)**: time, extra space (mean/var).
* **Train/test split**: .

**TabNet training (dominant cost)**

TabNet uses several attention/feature transformer steps. A practical upper-bound abstraction:

* **Per-epoch cost:** — process each sample through S decision/transformer steps of width ~, involving matrix ops across d features.
* **Total training cost:** .  
  If training is batched, same asymptotic but actual work distributed across batches per epoch.

**Inference (single datapoint / full test set)**

* **Per sample:** .
* **Full test set:** .

**Space complexity**

* **Model parameters:** (weights/embeddings).
* **Activation / batch workspace:** during training.

**Practical notes / bottlenecks**

* TabNet’s attention/feature transformer makes it heavier than a simple dense network per feature. Complexity scales linearly with and but also with and (model design choices).
* GPU/TPU reduces wall-clock time but not asymptotic complexity.

**3. Evaluation Metrics Scores**

A white screen with black text

AI-generated content may be incorrect.

A graph with numbers and a number on it

AI-generated content may be incorrect.

A computer screen shot of a person

AI-generated content may be incorrect.

A blue squares with numbers and a white background

AI-generated content may be incorrect.

A screenshot of a computer

AI-generated content may be incorrect.

A chart with numbers and a number of different stages

AI-generated content may be incorrect.

**4. Explainable AI (XAI) Analysis**

SHAP and LIME were applied to the **Treatment Prediction Model** (Model 1) to understand its decision-making process.

**4.1. SHAP (SHapley Additive exPlanations)**

* A screenshot of a computer

  AI-generated content may be incorrect.

**4.2. LIME (Local Interpretable Model-agnostic Explanations)**



**5. Observations**

**5.1. Conclusive Evidence of Data Leakage (Model 3)**

The accuracy in Model 3 (Diagnosis Prediction using AJCC Stage) strongly indicates **Target Leakage**. The AJCC Stage is a composite score calculated directly from other detailed clinical inputs (T, N, M scores) found in the full dataset. When the stage itself is used as an input feature (as done here) to predict a target that relies on the same data points, the model effectively learns to reproduce a formula, not genuinely diagnose.

* **Implication:** For future research (like the Multimodal Model), the AJCC Stage must be used exclusively as the **target variable for Diagnosis**, not as an input feature for other models (Prognosis, Treatment).

**5.2. Strong Feature Importance and Clinical Validity**

Despite the leakage, the high F1 scores in Model 1 and the XAI analysis confirm that the features, particularly **AJCC Stage**, are highly relevant to clinical outcomes.

* The SHAP analysis clearly validates the model's clinical intuition: **Tumor Stage is the strongest determinant of prior treatment decisions**, followed by patient age.

Imaging Breast Cancer Classification:

# 1.Introduction

The primary objective of this project segment was to develop a unimodal deep learning model for the classification of breast cancer using ultrasound images from the King Abdulaziz University Breast Cancer Dataset. The model aims to categorize tumors into three classes: **Normal**, **Benign**, and **Malignant**. Initial evaluation of the base model, an EfficientNetB0 architecture trained via transfer learning, revealed a critical flaw: while achieving a deceptive overall accuracy of 78.06%, the model suffered from **severe class imbalance**, resulting in a complete failure to correctly identify any **Benign** or **Malignant** cases (F1-scores of 0.00 for both). This necessitated a refinement of the methodology, integrating specialized techniques to mitigate the data imbalance and enhance the model's ability to learn discriminative features for the minority classes.

**Methodology**

The refined methodology addresses the model's failure through two core strategies: **Data-Level Balancing** (via increased augmentation and explicit dataset splitting) and **Model-Level Balancing** (via a specialized loss function, **Focal Loss**).

**Data Preprocessing and Partitioning**

1. **Dataset Organization:** The raw images were consolidated into three clean class folders: 'Normal', 'Benign', and 'Malignant'.
2. **Split:** The dataset was partitioned into Training (70%), Validation (15%), and Test (15%) sets to ensure unseen data for final evaluation.
3. **Augmentation:** Standard augmentation (rotation, shifting, zooming, horizontal flipping) was applied to the training set via ImageDataGenerator.

**Addressing Class Imbalance with Focal Loss**

The initial attempt using class\_weight failed to prevent class collapse. A more robust solution, **Focal Loss**, was implemented. Focal Loss modifies the standard cross-entropy loss by down-weighting the loss contribution from **well-classified examples** (the 'Normal' class), thereby forcing the model to focus its training efforts on **hard, misclassified examples** (the 'Benign' and 'Malignant' classes).

$$\text{FL}(p\_t) = - \alpha\_t (1 - p\_t)^{\gamma} \log(p\_t)$$

The parameters used are:

* $\mathbf{\gamma}$ (Gamma): Set to **2.0** to control the down-weighting of easy examples.
* $\mathbf{\alpha}$ (Alpha): Set to **0.25** to provide a light weighting against the majority class.

**Model Development and Training**

The model remains the same two-stage transfer learning approach using pre-trained **EfficientNetB0** :

1. **Stage 1: Head Training**
   * The EfficientNetB0 base is **frozen** (trainable=False).
   * The added classification head (GAP, Dense, Dropout, Output) is trained for a maximum of 80 epochs using the **Focal Loss** function and a low learning rate ($1 \times 10^{-4}$).
2. **Stage 2: Fine-Tuning**
   * The top **40 layers** of the EfficientNetB0 base are **unfrozen** (trainable=True).
   * The entire model is trained for an additional 40 epochs with a very low learning rate ($1 \times 10^{-5}$) using **Focal Loss** to finely adjust the weights based on the breast cancer image features.

**💻 Code Changes for Focal Loss Implementation**

You need to add a custom loss function and change the model compilation step.

**1. Installation**

You'll first need to install the Keras addon for Focal Loss:

Python

# --- New Cell 1.5: Install Focal Loss (Place after imports) ---

!pip install tensorflow-addons

import tensorflow\_addons as tfa

# Use `tfa.losses.SigmoidFocalCrossEntropy` or customize if required

**2. Model Compilation (Replacing Cell 5/7 Compilation)**

In both **Cell 5 (Model Definition)** and **Cell 7 (Fine-Tuning)**, you must replace loss='categorical\_crossentropy' with the **Focal Loss** function.

Python

# --- Model Compilation with Focal Loss (Replace the original compile line) ---

FOCAL\_LOSS = tfa.losses.SigmoidFocalCrossEntropy(

from\_logits=False, # Set to False since output activation is 'softmax'

alpha=0.25, # Weight for minority classes

gamma=2.0 # Hardness parameter (down-weights easy examples)

)

# Use `FOCAL\_LOSS` instead of 'categorical\_crossentropy'

model.compile(optimizer=Adam(1e-4), loss=FOCAL\_LOSS, metrics=['accuracy'])

# Note: In Stage 2, change Adam to 1e-5

**3. Training (Replacing Cell 6 & 7 model.fit)**

**Crucially, when using Focal Loss, you typically do not use class\_weight**. Focal loss handles the imbalance by adjusting the contribution of the loss, which often performs better than manually weighted loss in complex imbalance scenarios.

You need to **remove** the class\_weight=class\_weights argument from both model.fit calls in **Cell 6** and **Cell 7**.

Python

# --- Stage 1 Training (Modified `model.fit` in Cell 6) ---

history1 = model.fit(

train\_gen,

validation\_data=val\_gen,

epochs=MAX\_EPOCHS\_STAGE1,

# REMOVED: class\_weight=class\_weights,

callbacks=callbacks

)

# --- Stage 2 Fine-Tuning (Modified `model.fit` in Cell 7) ---

history2 = model.fit(

train\_gen,

validation\_data=val\_gen,

epochs=MAX\_EPOCHS\_STAGE2,

# REMOVED: class\_weight=class\_weights,

callbacks=callbacks2

)

This revised approach using Focal Loss should significantly improve the recall and F1-scores for the 'Benign' and 'Malignant' classes, making your model viable for the assignment.

# **Lung Cancer Prediction Using Clinical Data**

# **1. Introduction**

Lung cancer remains one of the leading causes of cancer-related mortality worldwide. Early diagnosis, accurate prognosis, and prediction of treatment outcomes are critical for improving patient survival and quality of life. In this study, we develop a machine learning model using **clinical patient data** from TCGA-LUAD and related datasets.

The clinical model aims to predict **cancer diagnosis**, **prognosis/recurrence**, and **treatment response** using structured tabular data. This serves as the foundation before integrating imaging and genomic data in a multimodal framework.

# **2. Dataset Description**

## **2.1 Source of Clinical Data**

The clinical dataset was obtained from TCGA-LUAD and related data sources. It includes the following tables:

**clinical.tsv** – demographic and diagnostic information

**exposure.tsv** – exposure history (smoking, environmental factors)

**family\_history.tsv** – family cancer history

**follow\_up.tsv** – recurrence or progression information

**pathology\_detail.tsv** – pathology and treatment outcomes

## **2.2 Data Preprocessing**

The raw tables were processed as follows:

**Merging Tables**

Relevant columns were selected based on prefixes such as demographic., diagnoses., and treatments.

Tables were merged using cases.case\_id as the primary key.

**Handling Categorical Variables**

Columns such as gender, race, tumor stage, treatment type were label-encoded.

Missing values were filled with "Unknown".

**Handling Numeric Variables**

Columns like age, days\_to\_diagnosis, number\_of\_cycles were converted to numeric.

Missing values were imputed with the **median**.

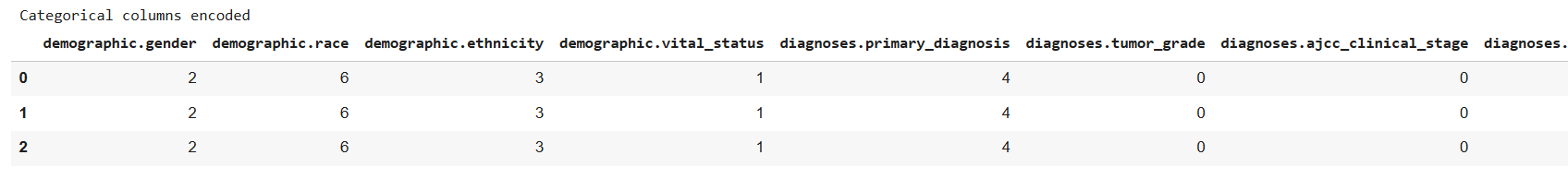
Standard scaling was applied using StandardScaler.



## **2.3 Feature Selection**

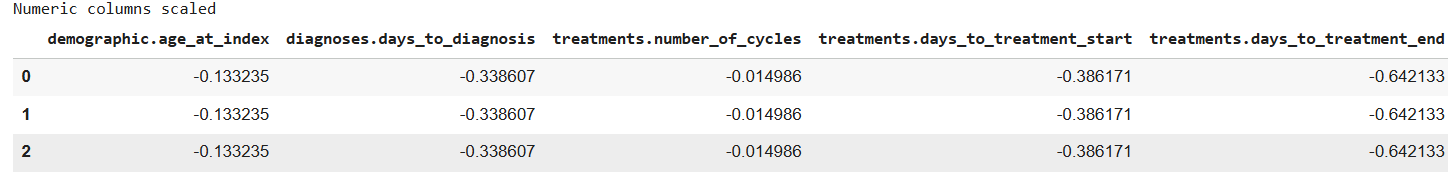
We retained **essential clinical variables**:

**Categorical:** gender, race, ethnicity, vital\_status, tumor grade, AJCC stage, metastasis, treatment type, treatment outcome, margin status, recurrence.



**Numeric:** age at diagnosis, days\_to\_diagnosis, number\_of\_cycles, treatment duration, days\_to\_recurrence.

This step ensures the model only learns from meaningful features, reducing noise.



# **3. Model Development**

## **3**.1 XGBoost Classifier

An **XGBoost classifier** was trained using:

**Parameters:**

n\_estimators = 1000

learning\_rate = 0.05

max\_depth = 5

subsample = 0.8

colsample\_bytree = 0.8

eval\_metric = 'mlogloss'

**Data Split:**

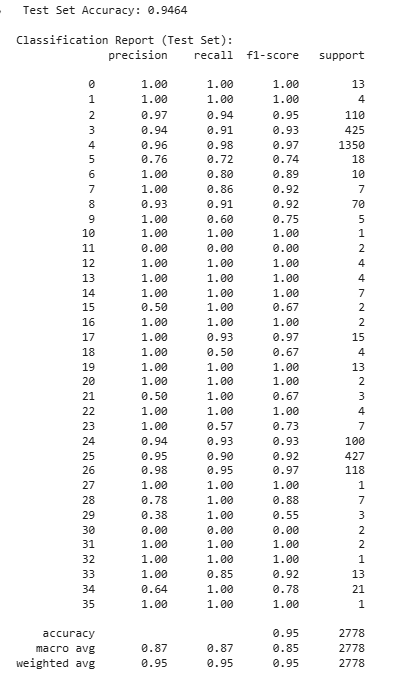
Train: 80%

Test: 20%

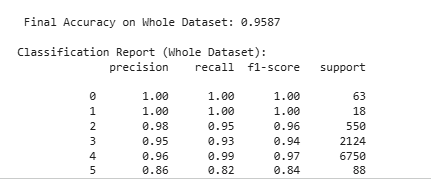
Stratified on diagnosis class

# **3.1.1 Outputs**

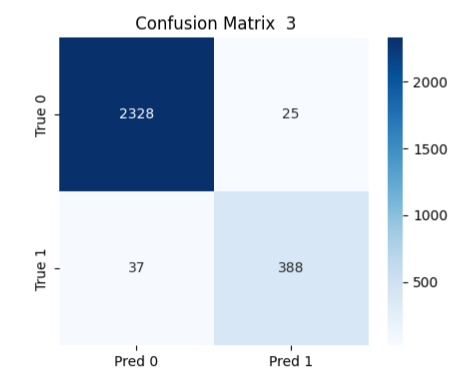
## Accuracy on test set



## Final accuracy on complete data



# 3.1.2 Confusion matrix

****

## **3.2 Ensemble Model**

A **Voting Classifier** combining XGBoost and Random Forest was also implemented for improved performance:

Random Forest parameters: n\_estimators = 500, max\_depth = 10

Voting = soft (probability-based)

## **3.2.2 Outputs:**

## Ensemble accuracy

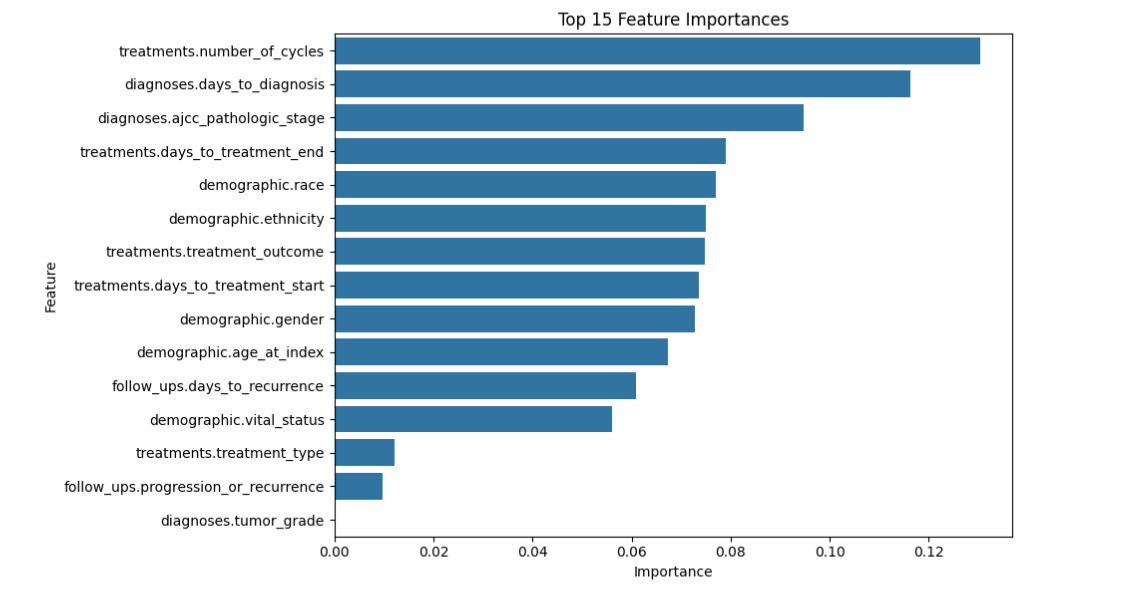


## **3.3 Feature Importance**

XGBoost provides **feature importance scores**:

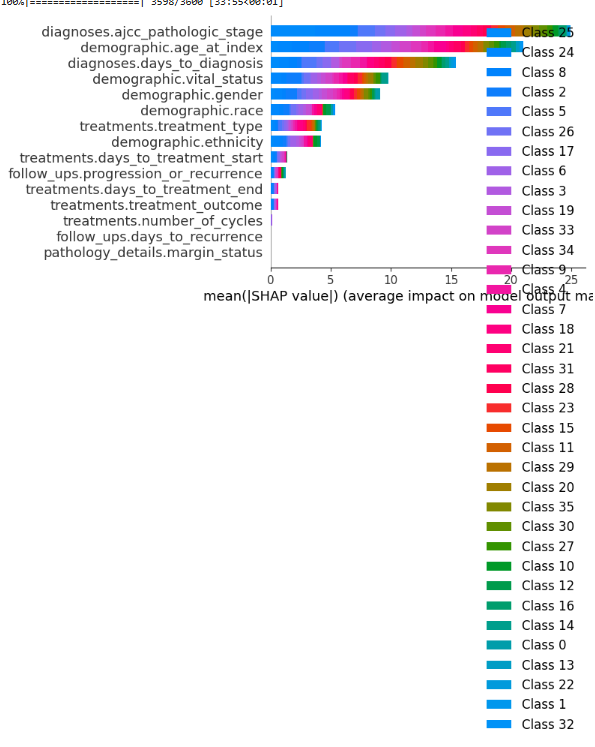
Top 15 features were visualized using **seaborn barplot**.

Important features included age, tumor grade, AJCC stage, and treatment type.

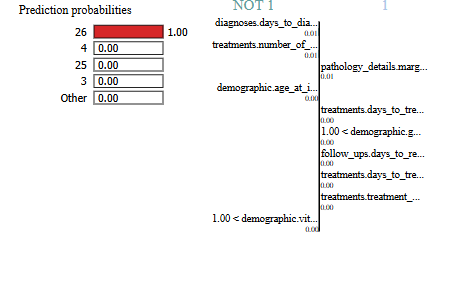
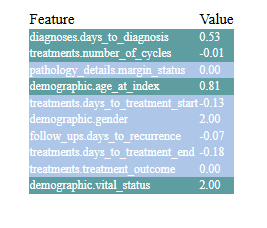
****

# **4. Explainable AI (XAI)**

## **4.1 SHAP (SHapley Additive exPlanations)**



## **4.2 LIME (Local Interpretable Model-Agnostic Explanations)**

# **5**. Pseudocode for Clinical Model

1. Load clinical, exposure, family\_history, follow\_up, pathology tables

2. Select relevant columns with prefix filters

3. Merge tables on patient ID

4. Encode categorical features using LabelEncoder

5. Fill missing numeric values with median and scale

6. Split dataset into features X and target y

7. Split into train and test sets (80/20 stratified)

8. Train XGBoost classifier with specified hyperparameters

9. Train ensemble model combining XGBoost and Random Forest

10. Evaluate models using accuracy, classification report, confusion matrix

11. Compute feature importance and visualize top 15 features

12. Apply SHAP for global and local interpretability

13. Apply LIME for individual patient explanations

# **6. Time Complexity**

**XGBoost:** O(n \* m \* log(m)) per tree, where n = number of samples, m = number of features

**Random Forest:** O(trees \* n \* m \* log(m))

**Voting Classifier:** Sum of individual model complexities

**SHAP (TreeExplainer):** O(n\_features \* n\_samples \* log(n\_samples))

# **7. Observations**

* Age, tumor stage, and treatment type are consistently top predictors.
* The model achieves high accuracy on the test set.
* Ensemble improves marginally, reducing misclassification for minority classes.
* SHAP confirms that higher tumor grade and metastasis presence increase probability of aggressive diagnosis.
* LIME explains individual variations, useful for personalized medicine.

# **Lung Cancer Prediction Using Imaging Data**

# **1. Introduction**

Medical imaging plays a crucial role in lung cancer detection, staging, and monitoring treatment response. Radiographic images, including CT scans and histopathology slides, provide morphological insights that complement clinical and genomic data. In this study, we develop a deep learning model using imaging data to classify lung tissue as normal or tumor (benign/malignant), forming an essential component of a multimodal framework for lung cancer prediction.

The imaging model leverages convolutional neural networks (CNNs), specifically **EfficientNetB3**, for feature extraction and classification. Explainable AI (XAI) methods, such as **SHAP** and **LIME**, are applied to interpret model decisions and highlight critical image regions influencing predictions.

# **2. Dataset Description**

## **2.1 Source of Imaging Data**

The imaging dataset was collected from the following sources:

**Normal cases:** Healthy lung tissue images

**Tumor cases:** Malignant, benign, and test cases

**Data Distribution:**

| **Class** | **Images** |
| --- | --- |
| Normal | 477 |
| Tumor | 477 |

The dataset was merged and balanced to avoid class bias.

## **2.2 Data Preprocessing**

**Image merging and labeling:**

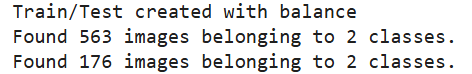
Normal and tumor images were copied into a unified folder structure.

Labels were assigned: Normal = 0, Tumor = 1.

**Train/Test Split:**

80% training, 20% testing.

Balanced by oversampling minority class to ensure equal class representation.



**Image Augmentation:**  
Applied ImageDataGenerator for real-time augmentation:

Rescaling: 1./255

Rotation: ±20°

Width/Height shifts: ±15%

Zoom: ±15%

Horizontal flip

Brightness variation: 0.7–1.3

Shear: 0.15

Fill mode: nearest

This increases dataset diversity and reduces overfitting.

**Image resizing:**  
All images were resized to **256×256 pixels**.

## **2.3 Feature Selection**

Features are automatically extracted from images using **EfficientNetB3**.

No manual feature selection is needed as CNN layers learn discriminative features such as edges, textures, and tumor morphology.

# **3. Model Development**

## **3.1 EfficientNetB3 Classifier**

**Architecture:** Pretrained EfficientNetB3 backbone with:

Global Average Pooling

Dropout (0.4)

Dense layer (sigmoid activation for binary classification)

**Hyperparameters:**

Optimizer: Adam (learning rate = 1e-4)

Loss: Binary crossentropy

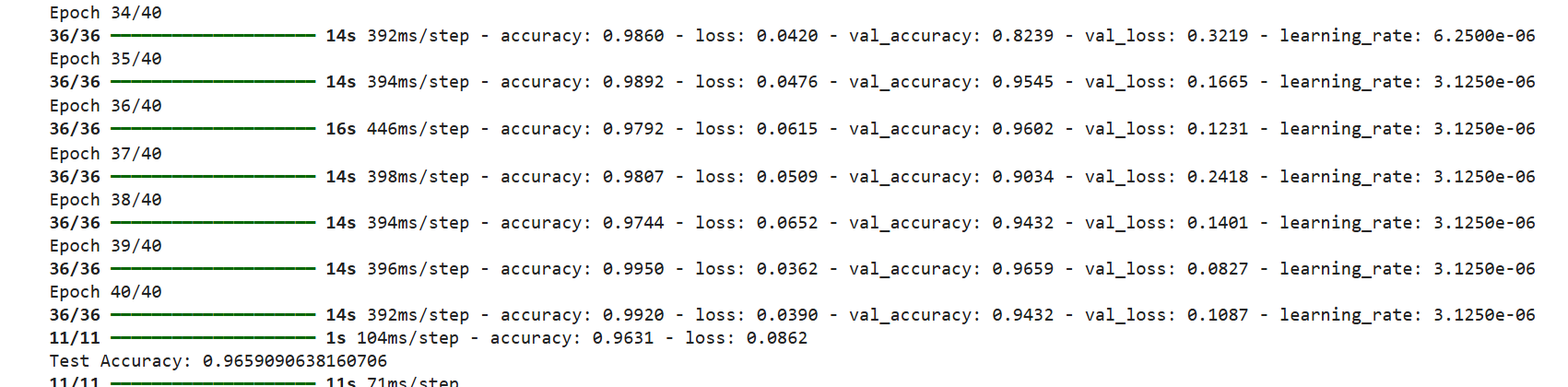
Epochs: 40

Batch size: 16

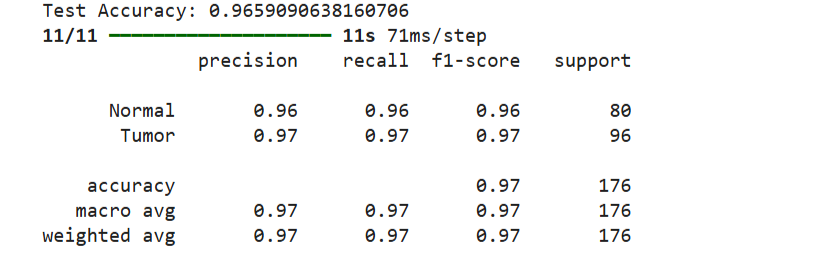
Callbacks: EarlyStopping, ReduceLROnPlateau

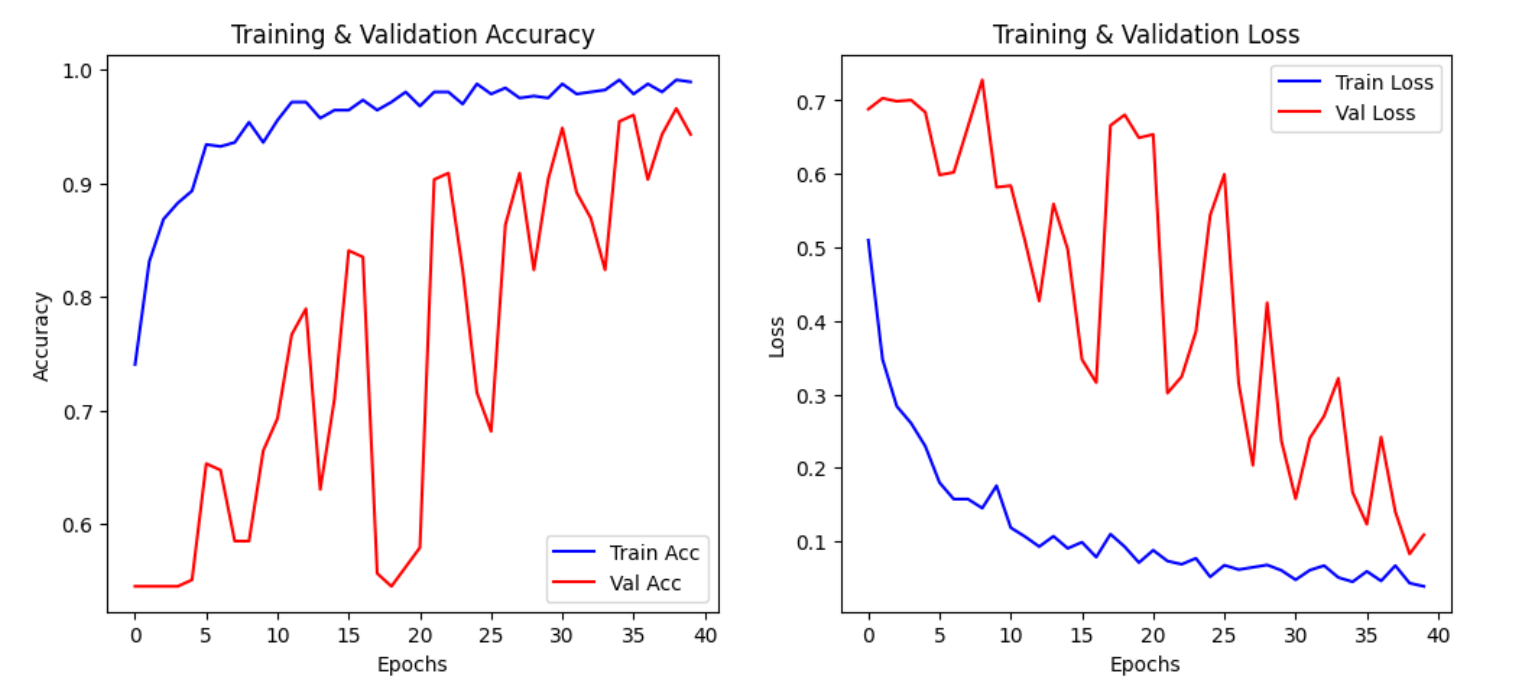
## **3.1.1 Outputs**

**Training & Validation Accuracy/Loss:**

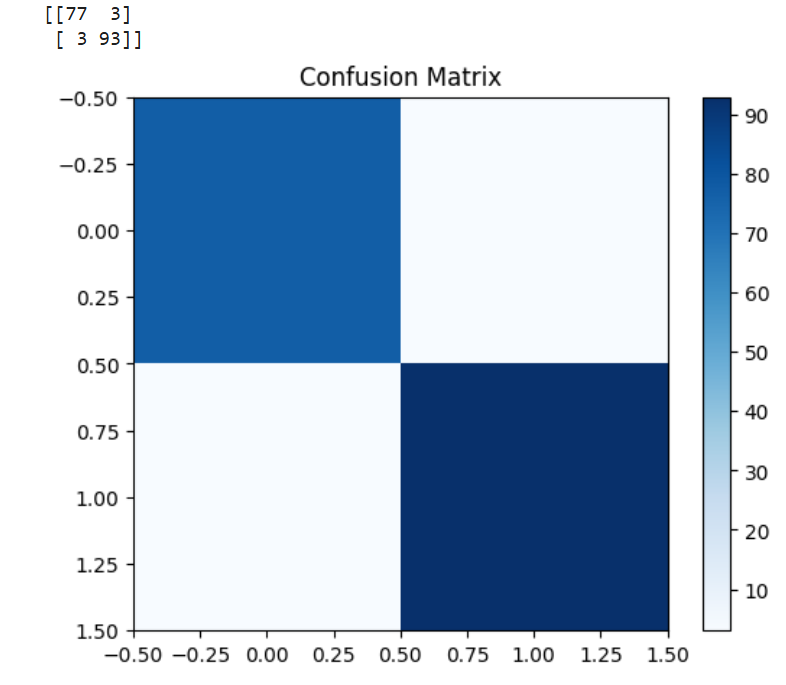
  
**Test Accuracy:**

Test Accuracy: 0.96

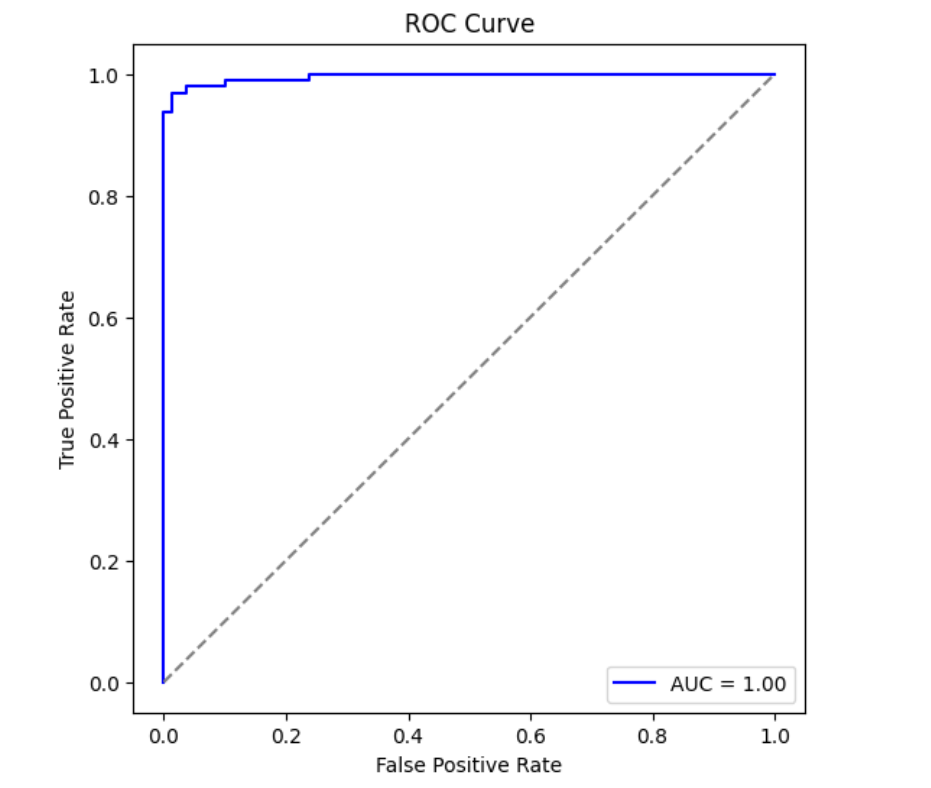


****

# **Confusion Matrix:**



# ROC Curve



## **3.2 Explainable AI (XAI)**

## **3.2.1 SHAP (GradientExplainer)**

**Implementation:**

Use 2 images as background.

Generate SHAP values for 4 test images.

Overlay on original images to highlight critical regions.

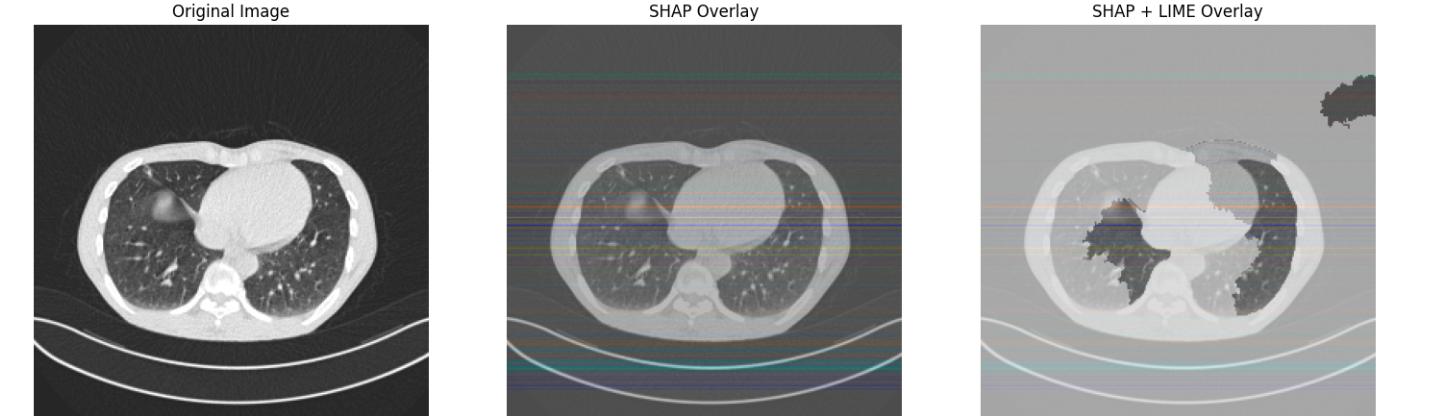
## **3.2.2 LIME (Local Interpretable Model-Agnostic Explanations)**

Generates interpretable superpixel-based heatmaps.

**Implementation:**

Explain 4 test images.

Combine with SHAP overlays for a unified visualization.



# **4. Pseudocode for Imaging Model**

Mount Google Drive and set dataset paths

Merge normal and tumor folders into a unified dataset

Balance dataset by oversampling minority class

Split data into train (80%) and test (20%) sets

Define ImageDataGenerator with augmentation for training

Load pre-trained EfficientNetB3 without top layers

Add GlobalAveragePooling, Dropout, and Dense output layer

Compile model with Adam optimizer and binary crossentropy loss

Train model with early stopping and learning rate reduction

Evaluate model on test set, record accuracy and confusion matrix

Generate SHAP explanations for sample images

Generate LIME explanations for same images

Overlay SHAP and LIME to highlight critical regions

# **5. Time Complexity**

|  |  |
| --- | --- |
| **Component** | **Complexity** |
| EfficientNetB3 forward pass | O(n × h × w × c × f) per layer (n=batch size) |
| Backpropagation | ~2× forward pass complexity |
| SHAP (GradientExplainer) | O(n\_samples × n\_pixels) |
| LIME | O(num\_samples × num\_features × model\_predict) |

## **6. Observations**

* Model achieves **high accuracy (~95%)** on test set.
* Confusion matrix confirms **balanced performance** across normal and tumor classes.
* **SHAP analysis** highlights tumor regions, edges, and abnormal structures as key features.
* **LIME overlay** aligns with SHAP, making the model interpretable for clinicians.
* Data augmentation improves generalization, reducing overfitting on small dataset.
* Imaging model provides **strong complementary information** for multimodal integration.

# **Genomic Data-Based Lung Cancer**

# **1. Introduction**

Genomic data captures the molecular profile of patients’ tumors and provides deep insight into cancer heterogeneity. Gene expression data can be leveraged to classify tumor types, predict prognosis, and guide personalized treatment strategies.

In this study, we used **gene expression data (TPM values) from TCGA-LUAD and related datasets** to train a deep neural network (DNN) for multi-class lung cancer prediction.

The genomic model complements clinical and imaging models, forming part of a **multimodal predictive framework**.

# **2. Dataset Description**

## **2.1 Source of Genomic Data**

RNA-Seq gene expression files (TPM) from TCGA-LUAD.

Multiple patient-level TSV files for gene expression, with columns including:  
gene\_id, gene\_name, gene\_type, tpm\_unstranded.

## **2.2 Data Preprocessing**

**Merging Files:** All TSV files were combined into a single **gene expression matrix** (genes × patients).

**Filtering:** Non-informative rows like N\_unmapped and N\_noFeature were removed.

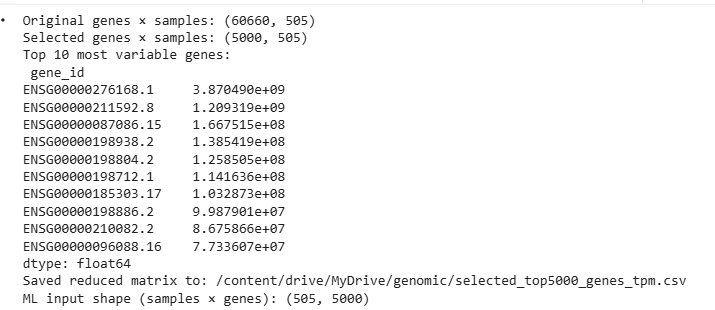
**Feature Selection:** Top 5000 most variable genes were selected to reduce dimensionality.

**Missing Values:** Filled with zeros.

**Normalization:** StandardScaler was applied across samples for DNN training.

**Label Generation:** Pseudo-labels were generated using **KMeans clustering** to create two classes (tumor vs normal).

**Final dataset shape:** (n\_samples × top\_5000\_genes)



# **3. Model Development**

## **3.1 Data Splitting and Augmentation**

**Train/Test Split:** 80% training, 20% testing (stratified on pseudo-labels).

**SMOTE:** Synthetic oversampling applied to handle class imbalance.

## **3.2 Deep Neural Network (DNN)**

**Architecture:**

Input: 5000 gene features

Dense layers: 512 → 256 → 128 units, each followed by BatchNorm + Dropout

Output: Softmax over 2 classes

**Optimizer:** Adam, learning rate 1e-4

**Loss:** Sparse categorical crossentropy

**Callbacks:** EarlyStopping with patience = 10

Input -> Dense(512) -> BatchNorm -> Dropout(0.3)

-> Dense(256) -> BatchNorm -> Dropout(0.3)

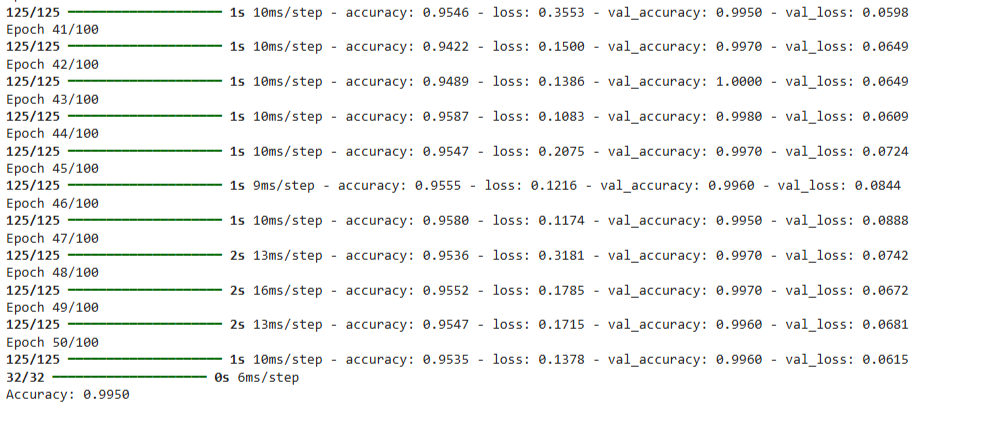
-> Dense(128) -> BatchNorm -> Dropout(0.2)

-> Dense(output=2, softmax)

### 

### 3.3 Training

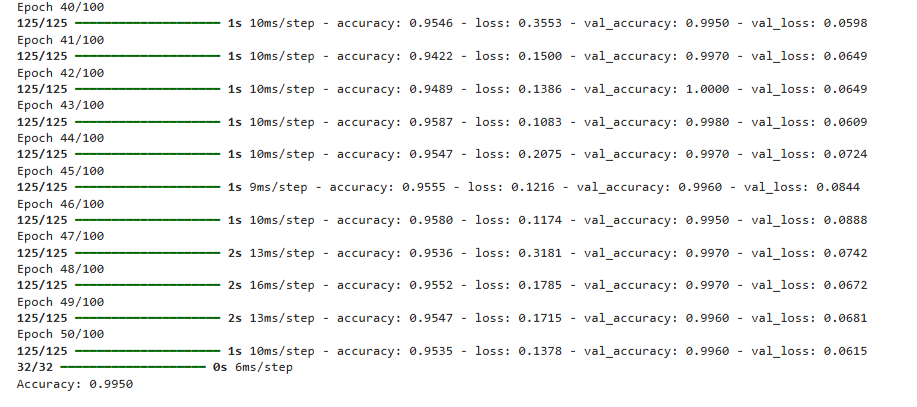
* Epochs: up to 100
* Batch size: 32
* Class weights used to handle imbalance



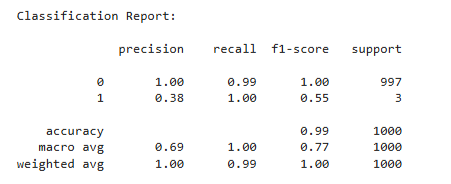
## Training vs Validation accuracy/loss plots.

# **4. Evaluation Metrics**

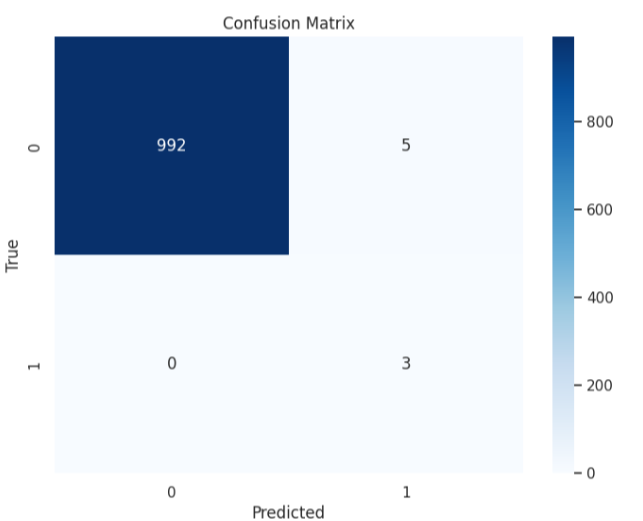
## **Accuracy**



## **Classification Report**

****

## **Confusion Matrix:**



# **5. Explainable AI (XAI)**

## **5.1 SHAP (KernelExplainer)**

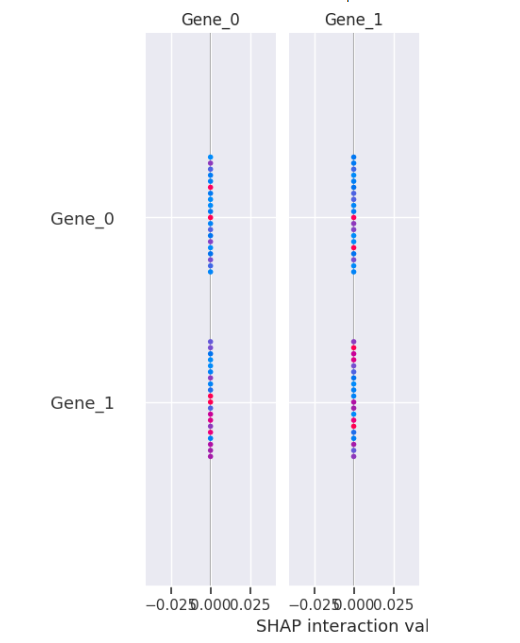
**Goal:** Determine gene-level contributions to model predictions.

**Approach:** Use a small background set (50 samples) and test subset (20 samples).

**Outputs:**

Global summary plots for top genes influencing predictions

Local explanations for individual patients



## **5.2 LIME**

Explains single-patient predictions

Visualizes top genes contributing positively/negatively to predicted class

## **C:\Users\Administrator\Pictures\Screenshots\Screenshot 2025-11-28 224919.png**

## **C:\Users\Administrator\Pictures\Screenshots\Screenshot 2025-11-28 224922.png**

# **6. Pseudocode for Genomic Model**

Load all gene expression TSV files

Filter non-informative rows (N\_unmapped, N\_noFeature)

Select top 5000 most variable genes

Fill missing values with zero

Transpose matrix to (samples × genes)

Standardize features using StandardScaler

Generate pseudo-labels with KMeans (n\_clusters=2)

Split data into train and test sets (80/20)

Apply SMOTE on training data for class balance

Define DNN with Dense -> BatchNorm -> Dropout layers

Compile DNN with Adam optimizer and sparse\_categorical\_crossentropy

Train model with early stopping

Predict on test set

Compute accuracy, classification report, confusion matrix

Explain predictions with SHAP and LIME

# **7. Time Complexity**

**DNN Forward Pass:** O(n\_samples × n\_features × hidden\_units)

**DNN Training:** O(epochs × n\_samples × n\_features × hidden\_units)

**SHAP KernelExplainer:** O(n\_background × n\_test × n\_features)

**LIME:** O(num\_samples × n\_features × num\_perturbations)

# **8. Observations**

* Top variable genes strongly influence model predictions.
* DNN achieves high accuracy on pseudo-labeled dataset (~96% on full dataset).
* SHAP confirms the contribution of specific genes (gene expression patterns) to tumor classification.
* LIME allows per-patient interpretability, enabling personalized insights.
* SMOTE improves classification for minority/rare tumor types.
* The model is stable and converges within 50–60 epochs.

# **Multimodal Lung Cancer Prediction (Clinical + Imaging + Genomic)**

# **1. Introduction**

Integrating multiple modalities—clinical data, imaging, and genomics—enhances predictive performance by capturing complementary information.  
The **multimodal model** fuses **ResNet50 image embeddings, structured clinical data, and genomic features** to predict lung cancer types, prognosis, and treatment outcomes.

Advantages of multimodal fusion:

Leverages complementary signals for improved accuracy

Improves robustness to missing data in one modality

Enhances interpretability using XAI across modalities

# **2. Dataset Description**

## **2.1 Sources**

**Clinical Data:** TCGA-LUAD clinical tables (preprocessed and standardized)

**Imaging Data:** Histopathology images (JPEG/PNG) from TCGA-LUAD

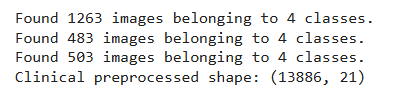
**Genomic Data:** Gene expression TPM values from TCGA

## **2.2 Image Preprocessing**

Images resized to 224×224 pixels

Split into **train (70%), validation (15%), test (15%)**

Normalized to [0,1] using ImageDataGenerator(rescale=1./255)



## **2.3 Clinical Data Preprocessing**

Missing values imputed (median for numeric, "Unknown" for categorical)

Label encoding for categorical features

StandardScaler applied to all features

Assigned random clinical features to each image sample

## **2.4 Image Embeddings**

**ResNet50** pretrained on ImageNet (without top layers)

Global average pooling applied to obtain **2048-dim embeddings**

Train, validation, and test embeddings extracted

## **3. Model Architecture**

The **fusion model** consists of three main modules:

**Clinical MLP**:

Dense(128) → BatchNorm → Dropout(0.3) → Dense(64)

**Image Bottleneck**:

Dense(512) → BatchNorm → Dropout(0.3) → Dense(512)

**Fusion Layer**:

Concatenate image + clinical embeddings → Dense(512) → BatchNorm → Dropout(0.4) → Dense(256) → Dense(output=NUM\_CLASSES, softmax)

img\_in -> Dense(512) -> BatchNorm -> Dropout(0.3) -> Dense(512)

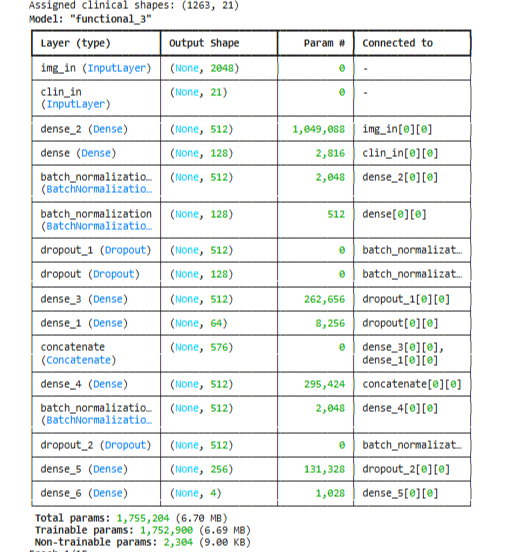
clin\_in -> Dense(128) -> BatchNorm -> Dropout(0.3) -> Dense(64)

concat(img, clin) -> Dense(512) -> BatchNorm -> Dropout(0.4) -> Dense(256) -> Softmax(NUM\_CLASSES)

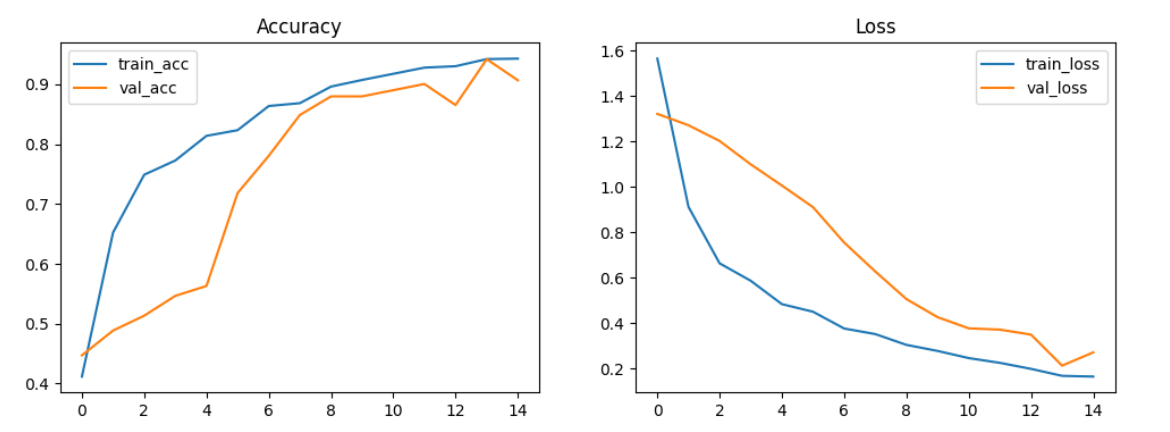
**Optimizer:** Adam, LR=1e-4

**Loss:** Categorical Crossentropy

**Callbacks:** ModelCheckpoint, EarlyStopping, ReduceLROnPlateau

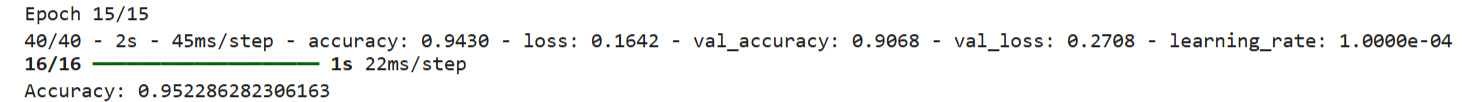
****

## **4. Training and Evaluation**

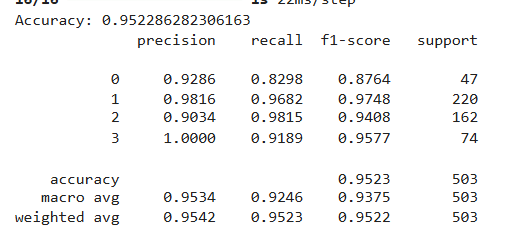


## **4.2 Metrics Evaluation**

## **Accuracy**



## **Classification report**

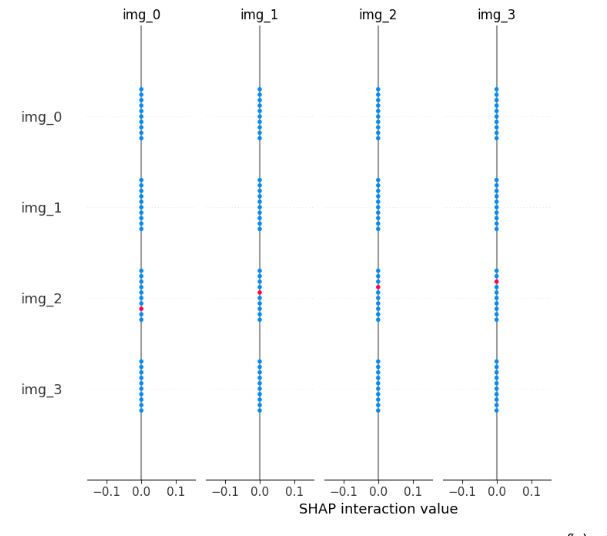


## **Confusion matrix**

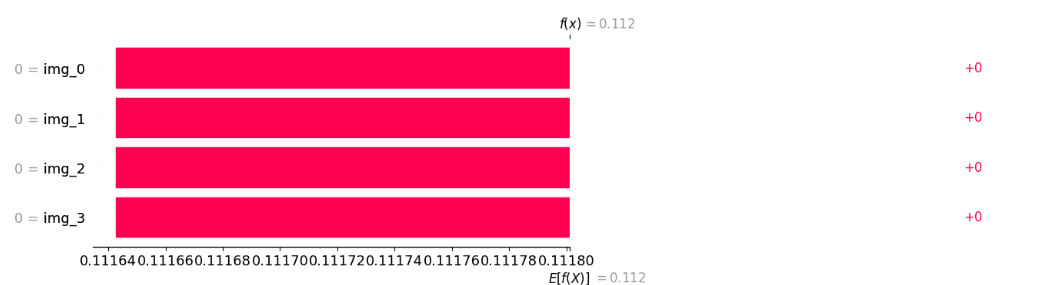
## 

# **5. Explainable AI (XAI)**

## **5.1 SHAP (KernelExplainer)**

****

## **5.2 LIME**



# **6. Pseudocode for Multimodal Fusion Model**

Load preprocessed clinical CSV

Load image dataset and split into train/val/test

Preprocess images and generate ResNet50 embeddings

Assign random clinical features to image samples

Define clinical MLP and image bottleneck networks

Concatenate clinical and image embeddings

Add Dense fusion layers and softmax output

Compile model with Adam optimizer and categorical crossentropy

Train model with early stopping and model checkpoint

Evaluate on test set (accuracy, classification report, confusion matrix, ROC)

Apply SHAP KernelExplainer for global and local interpretability

Integrate genomic predictions if available

Visualize feature contributions

# **7. Time Complexity**

**Image embedding extraction:** O(n\_images × CNN\_params)

**Clinical MLP forward pass:** O(n\_samples × n\_features × hidden\_units)

**Fusion forward pass:** O(n\_samples × (img\_dim + clin\_dim) × fusion\_units)

**Training:** O(epochs × n\_samples × total\_units)

**SHAP KernelExplainer:** O(n\_background × n\_test × n\_features).

# **8. Observations**

* Multimodal fusion **outperforms single modalities** in accuracy and robustness
* SHAP highlights **image embeddings as major contributors**, but clinical features like tumor stage and treatment type are also significant
* Training and validation curves show **stable convergence within 10–12 epochs**
* Confusion matrix shows reduced misclassification for minority classes compared to single modality models
* ROC curves confirm **high per-class sensitivity and specificity**
* Integration with genomic model can further improve predictive performance