

# Modeling Report

Students Names:

- Alam Bebar - 207415407
- Ryan Thawkho - 305070567

## Executive Summary

This report presents a comparative analysis of machine learning , and deep learning models applied to two datasets related to hair fall: a tabular dataset from Mendeley and an image-based dataset from Roboflow. The [Selecting Modelling Techniques](#) part explains the rationale behind choosing specific models for each dataset. The [Test Design](#) outlines how data was partitioned and how models were evaluated. In the [Model Description](#), the report documents the parameter settings and performance summaries for each model. Tabular models include **Logistic Regression, Logistic Regression with GridSearchCV, K-Nearest Neighbors (KNN), KNN with GridSearchCV, Weighted KNN, Random Forest, Random Forest with GridSearchCV**, and **XGBoost**. Image-based models include **ResNet, MobileNet, VGG16**, and **a custom CNN**. The [Model Assessment](#) section compares these models based on accuracy and practical insights. The Appendix contains [result visualizations](#) and a [glossary](#) of key terms.

Table of Content

1. Selecting Modelling Techniques.....3
1.1 Choosing the Right Modeling Techniques.....3
Hair Fall Causes Dataset (Mendeley):..... 3
Scalp Image Dataset (Roboflow):.....3
1.2 Model Assumptions.....4
2. Test Design..... 4
3. Model Description..... 5
3.1 Parameter Settings.....5
Hair Fall Causes Dataset (Mendeley)..... 5
Scalp Image Dataset (Roboflow).....11
3.2 Model Description..... 15
Hair Fall Causes Dataset (Mendeley)..... 15
Scalp Image Dataset (Roboflow).....19
4. Model Assessment..... 21
Hair Fall Causes Dataset (Mendeley)..... 22
Scalp Image Dataset (Roboflow).....23
Appendix: Models Results Graphs..... 23
Hair Fall Causes Dataset (Mendeley)..... 23
Scalp Image Dataset (Roboflow).....27
Appendix: Terminology..... 29

# 1. Selecting Modelling Techniques

We are working with two different types of data: a tabular dataset from **a hair fall causes survey (Mendeley)** and a **visual dataset of scalp images (Roboflow)**. For each dataset, we selected suitable models based on data format, size, and goals.

## 1.1 Choosing the Right Modeling Techniques

### Hair Fall Causes Dataset (Mendeley):

After preprocessing the dataset and adding new entries, the final dataset now consists of 484 male respondents, with cleaned and encoded features, along with additional records collected through the Google Forms questionnaire. This dataset includes categorical features that were one-hot encoded. Based on the structure and size, we plan to use the following models:

- **Logistic Regression:** Simple baseline model, suitable for binary classification.
- **K-Nearest Neighbors (KNN):** Good for small to medium datasets, but sensitive to feature scaling.
- **Random Forest:** Handles categorical and numerical features well, robust to missing values.
- **XGBoost:** Boosting model known for high accuracy and good handling of tabular data.

### Scalp Image Dataset (Roboflow):

This dataset contains 2,075 scalp images categorized into 6 male pattern baldness stages from multiple angles (front, back, side, top) and are labeled based on **Norwood hair loss stages** (See [Appendix: Terminology](#)). For these images, we plan to use deep learning models:

- **ResNet50:** Strong baseline CNN with pre-trained ImageNet weights. Requires images resized to 224×224 pixels and normalized pixel values
- **MobileNet:** Lightweight CNN for faster training and lower resource usage.
- **VGG16:** Simple and consistent CNN architecture using stacked 3×3 convolutions.
- **Custom CNN:** Designed for flexibility and to experiment with architecture adjustments.

All CNNs require input in shape (224, 224, 3) (RGB), with normalized pixel values.

## 1.2 Model Assumptions

We ensured the following for both datasets:

- Categorical data is encoded numerically using one-hot encoding.
- Irrelevant or inconsistent columns (like timestamps) were removed.
- Missing or incorrect values were fixed.
- For CNNs, images will be resized, normalized, and augmented using flipping, rotation, etc.

## 2. Test Design

*What data will be used to test the models? Have you partitioned into train/test sets?*

- Our scalp image dataset is already organized into separate train, test, and validation folders, while we plan to apply an 80/20 train-test split to the tabular dataset. For the scalp images, we will first evaluate the model's accuracy, and based on the results, we will consider balancing the number of images across the folders and Norwood baldness stages to improve accuracy for each individual stage.

*How might you measure the success of supervised models?*

- We are planning to use **accuracy, precision, recall, F1-score, and confusion matrix to evaluate the models**. For the image CNNs and the survey classifiers we'll track overall accuracy, per-class precision/recall, and F1-score. Those tell us not just how often we're right, but whether we're confusing adjacent Norwood stages (or stable vs. progressing) too much.

*How many times are you willing to rerun a model with adjusted settings before switching to another approach?*

- We'll give each model type about 3–5 tuning rounds—say, different learning rates or number of frozen layers for ResNet, or different  $k$  values for KNN—before concluding its limits. If performance still stalls, we'll move on (e.g., from ResNet50 to MobileNet or from Logistic Regression to Random Forest).

### 3. Model Description

#### 3.1 Parameter Settings

Hair Fall Causes Dataset (Mendeley)

Model Name	Parameters Settings
Logistic Regression	<p><b>Parameters Used:</b></p> <p>In this run, we used <b>default settings</b> of Logistic Regression from Scikit-learn.</p> <p>That means no special parameters were manually adjusted.</p> <p><b>Important default parameters:</b></p> <ul style="list-style-type: none"> <li>• <code>penalty='l2'</code> This means we apply <b>L2 regularization</b>, which helps prevent overfitting by penalizing large coefficients.</li> <li>• <code>solver='lbfgs'</code> This is the optimization algorithm used to find the best model weights. It's good for small-to-medium datasets.</li> <li>• <code>max_iter=100</code> The model will run up to <b>100 iterations</b> to find the best solution. If it doesn't converge in that time, it stops.</li> <li>• <code>random_state</code> Not set inside the model directly, but we <b>set <code>random_state=42</code></b> when splitting the data, to make</li> </ul>

	<p>sure the results are reproducible.</p>
<p><b>Logistic Regression with GridSearchCV</b></p>	<p><b>Purpose of GridSearchCV:</b></p> <p>GridSearchCV systematically tries all combinations of the specified parameters, evaluates them using cross-validation, and selects the model with the best performance.</p> <p>We also used GridSearchCV to optimize Logistic Regression by testing different combinations of C, <b>penalty</b>, and <b>solver</b> values. This helped us improve the accuracy compared to the default model. The best parameters found were selected automatically, and the tuned model achieved higher accuracy than the original one.</p> <p>Below is the output for the best parameters:</p> <p>Best Parameters: {'C': 10, 'penalty': 'l2', 'solver': 'liblinear'}</p>
<p><b>K-Nearest Neighbors (KNN)</b></p>	<p><b>Parameters used:</b></p> <p>We tested different k values first using the <b>Elbow Method</b>, and then selected <b>k=5</b> based on the graph results.</p> <p><b>Important Parameters:</b></p> <ul style="list-style-type: none"> <li><b>n_neighbors=5:</b> This means the model looks at the <b>5 nearest neighbors</b> to classify each new sample.</li> <li><b>weights='uniform'</b> (default): All neighbors contribute <b>equally</b> to the prediction. (In future experiments, we could try 'distance' weighting so closer neighbors have more influence.)</li> <li><b>algorithm='auto'</b> (default): Scikit-learn automatically chooses the best algorithm to find neighbors (<b>ball_tree</b>, <b>kd_tree</b>, or <b>brute</b>).</li> <li><b>metric='minkowski'</b> (default): It uses <b>Euclidean distance</b> by default to measure how close neighbors are.</li> </ul>

# Modeling Report

	<ul style="list-style-type: none"> <li>We applied the <b>Elbow Method</b> first. We trained KNN models with different <b>k</b> values (1 to 20) and plotted the <b>error rate</b>. Based on the curve, we selected <b>k=5</b> where the error rate started to stabilize (best trade-off between bias and variance).</li> </ul>
KNN with GridSearchCV	<p>We used GridSearchCV to tune the number of neighbors (<b>k</b>) for the KNN model. This helped us choose the most accurate value of <b>k</b> by testing all options from 1 to 20. The best model found by the grid search achieved improved accuracy compared to the default KNN, as confirmed by the updated classification report and confusion matrix.</p> <p>Below is the output for the best parameters:  <b>Best Parameters: {'n_neighbors': 9}</b></p>
Random Forest	<p><b>Parameters Used:</b></p> <p>We mainly used the <b>default settings</b>, except for setting <b>random_state=42</b> to make the results reproducible.</p> <p><b>Important Parameters:</b></p> <ul style="list-style-type: none"> <li><b>n_estimators=100</b> (default): The model builds <b>100 different decision trees</b> and combines their predictions. This helps reduce overfitting compared to a single decision tree.</li> <li><b>criterion='gini'</b> (default): The model uses the <b>Gini impurity</b> to decide how to split nodes inside each tree. (We could also experiment with <b>'entropy'</b> for Information Gain.)</li> <li><b>max_depth=None</b> (default): Trees are grown fully (until all leaves are pure or contain very few samples). Later, we can try setting a limit to avoid very deep trees.</li> <li><b>random_state=42:</b> Ensures that we can <b>reproduce the same results</b></li> </ul>

Modeling Report

	<p>every time we run the code.</p> <ul style="list-style-type: none"> <li><code>bootstrap=True</code> (default): Each tree is trained on a random sample (with replacement) of the data, which adds randomness and improves generalization.</li> </ul>
Random Forest with GridSearchCV	<p>We applied GridSearchCV to test different combinations of hyperparameters for the Random Forest model, including the number of trees (<code>n_estimators</code>), tree depth (<code>max_depth</code>), and minimum samples required to split a node (<code>min_samples_split</code>). This helped us find the best combination for improving prediction accuracy. The optimized model performed better than the default Random Forest, as shown in the evaluation metrics and confusion matrix.</p> <p>Below is the output of the best parameters:  Best Parameters: {'max_depth': 5, 'min_samples_split': 5, 'n_estimators': 200}</p>
XGBoost	<ul style="list-style-type: none"> <li><b>Parameters Used:</b> We used <b>mostly default settings</b>, with <code>random_state=42</code> to make sure we get consistent results when we run the model multiple times.</li> <li><b>Important Parameters:</b> <ul style="list-style-type: none"> <li><code>n_estimators=100</code> (default): The model trains <b>100 small decision trees</b> and combines their results to make a final prediction.</li> <li><code>learning_rate=0.3</code> (default): Controls how much the model adjusts at each step. A lower value means slower but often more accurate learning. (In bigger projects, it's common to reduce it to 0.1 or 0.01.)</li> </ul> </li> </ul>



## Modeling Report

	<ul style="list-style-type: none"> <li>○ <b>max_depth=6</b> (default): Controls how deep each tree can grow. Deeper trees can learn more complex patterns but can also overfit if not careful.</li> <li>○ <b>subsample=1</b> (default): Means that the model uses <b>100% of the training data</b> for each boosting round. Sometimes, using a lower value like 0.8 can prevent overfitting.</li> <li>○ <b>colsample_bytree=1</b> (default): Uses <b>all the features</b> when building each tree. Again, tuning this (for example to 0.8) can sometimes help generalization.</li> <li>○ <b>random_state=42</b>: Guarantees that results are reproducible.</li> </ul> <p><b>We chose not to include the optimized XGBoost model from the GridSearchCV filter because it did not yield higher test accuracy than the XGBoost model with default parameters.</b></p>
Weighted KNN	<p><b>Parameters Used:</b></p> <ul style="list-style-type: none"> <li>● <b>n_neighbors=1</b>: We set the number of neighbors (K) to <b>1</b> based on the Elbow Method graph. That means for every prediction, the model looks at the <b>single closest training point</b> and bases its decision on it.</li> <li>● <b>weights='distance'</b>: Instead of giving all neighbors equal importance, the closer a neighbor is to the point we are predicting, the more influence it has. In other words, <b>closer points are weighted higher</b>, which usually improves accuracy compared to regular (uniform) KNN when the data isn't evenly spread out.</li> </ul>

Modeling Report

	<p><b>Why use Weighted KNN?</b></p> <p>Sometimes neighbors that are far away could hurt the prediction. Weighting by distance helps <b>give more importance to nearby samples</b>, which can make the model smarter, especially when the data has clusters.</p>
--	---

Scalp Image Dataset (Roboflow)

Model Name	Parameters Settings
ResNet50	<p><b>Base Model:</b> Pre-trained ResNet50 with <code>include_top=False</code> to add custom layers.</p> <p><b>Input Size:</b> Images resized to <b>224x224</b> pixels (standard for ResNet50).</p> <p><b>Custom Layers:</b> Added <code>GlobalAveragePooling2D</code>, a dense layer with <b>128 neurons</b> (ReLU activation), and an output layer with softmax matching the number of classes (6 classes: stages 2–7).</p> <p><b>Frozen Layers:</b> ResNet50 layers were frozen to avoid retraining them at first.</p> <p><b>Optimizer:</b> <code>Adam</code> optimizer with a learning rate of <b>0.0001</b>.</p> <p><b>Loss Function:</b> <b>Categorical crossentropy</b> (since it's a multi-class problem).</p> <p><b>Batch Size:</b> <b>32</b>.</p> <p><b>Epochs:</b> <b>5</b> epochs.</p>

	<p><b>Preprocessing:</b> Images normalized using ResNet50's <code>preprocess_input</code>.</p>
MobileNet	<p><b>Base Model:</b> Pre-trained MobileNet with <code>include_top=False</code> to allow for custom classification layers.</p> <p><b>Input Size:</b> Images resized to 224x224 pixels (standard input size for MobileNet).</p> <p><b>Custom Layers:</b> Added <code>GlobalAveragePooling2D</code>, followed by a dense layer with 1024 neurons using ReLU activation, and an output softmax layer with 6 classes (stages 2–7).</p> <p><b>Frozen Layers:</b> All layers of the MobileNet base model were frozen during initial training to preserve pretrained ImageNet features.</p> <p><b>Optimizer:</b> Adam optimizer with a learning rate of 0.0001.</p> <p><b>Loss Function:</b> Categorical crossentropy — appropriate for multi-class classification tasks.</p> <p><b>Batch Size:</b> 32.</p> <p><b>Epochs:</b> 10 epochs.</p> <p><b>Preprocessing:</b> Images were normalized to the [0, 1] range using Keras's</p>

## Modeling Report

	built-in <code>ImageDataGenerator(rescale=1./255)</code> .
<b>VGG16</b>	<ul style="list-style-type: none"> <li>• <b>Base Model:</b> Pre-trained <b>VGG16</b> with <code>include_top=False</code> to add custom layers.</li> <li>• <b>Input Size:</b> Images resized to <b>224x224</b> pixels (standard for VGG16).</li> <li>• <b>Custom Layers:</b> Flattened the output, then added a dense layer with <b>128 neurons</b> (ReLU activation), and a final softmax layer (6 classes for stages 2–7).</li> <li>• <b>Frozen Layers:</b> All <b>VGG16 layers were frozen</b> to keep the pre-learned features.</li> <li>• <b>Optimizer:</b> <b>Adam</b> optimizer with a learning rate of <b>0.0001</b>.</li> <li>• <b>Loss Function:</b> <b>Categorical crossentropy</b> (because it's multi-class classification).</li> <li>• <b>Batch Size:</b> <b>32</b>.</li> <li>• <b>Epochs:</b> <b>10</b> epochs.</li> <li>• <b>Preprocessing:</b> Images normalized using VGG16's <code>preprocess_input</code>.</li> </ul>
<b>Custom CNN</b>	<p><b>Base Model:</b> A manually constructed CNN (not pretrained), built from scratch using Keras's <code>Sequential</code> API.</p> <p><b>Input Size:</b></p>

	<p>Images resized to 224x224 pixels, with 3 color channels (RGB).</p> <p><b>Architecture &amp; Custom Layers:</b></p> <ul style="list-style-type: none"> <li>Conv2D(32, 3x3) + ReLU + MaxPooling2D(2x2)</li> <li>Conv2D(64, 3x3) + ReLU + MaxPooling2D(2x2)</li> <li>Conv2D(128, 3x3) + ReLU + MaxPooling2D(2x2)</li> <li>Flatten layer</li> <li>Dense layer with 128 neurons + ReLU</li> <li>Dropout(0.5) for regularization</li> <li>Output layer: Dense(6) with softmax activation (6 classes: stages 2–7)</li> </ul> <p><b>Frozen Layers:</b> N/A — this model was trained from scratch, so all layers were trainable.</p> <p><b>Optimizer:</b> Adam optimizer with default learning rate.</p> <p><b>Loss Function:</b> Categorical crossentropy, suitable for multi-class classification.</p> <p><b>Batch Size:</b> 32.</p> <p><b>Epochs:</b> 10 epochs.</p> <p><b>Preprocessing:</b></p>
--	--

	<p>Images were normalized to the [0, 1] range using <code>ImageDataGenerator(rescale=1./255)</code>.</p>
--	--

3.2 Model Description

Hair Fall Causes Dataset (Mendeley)

Model Name	Model Description
Logistic Regression	<p><b>Accuracy:</b> ~75%</p> <p><b>Precision/Recall:</b></p> <ul style="list-style-type: none"> <li>Class 0 (No hair fall): precision = 0.65, recall = 0.61</li> <li>Class 1 (Yes hair fall): precision = 0.80, recall = 0.83</li> </ul> <p><b>Meaningful conclusions:</b> The model shows decent separation between classes.</p> <p><b>New insights:</b> The model predicts "Yes" better than "No".</p> <p><b>Execution problems:</b> None, very fast.</p> <p><b>Data quality issues:</b> None (after preprocessing).</p> <p><b>Calculation inconsistencies:</b> None.</p>
Logistic Regression with GridSearchCV	<p><b>Accuracy:</b> ~78%</p> <p><b>Precision/Recall:</b></p> <ul style="list-style-type: none"> <li>Class 0 (No hair fall): precision = 0.67, recall = 0.60</li> <li>Class 1 (Yes hair fall): precision = 0.83, recall = 0.87</li> </ul> <p><b>Meaningful conclusions:</b> The model shows good predictive power, especially for detecting hair fall.</p> <p><b>New insights:</b> Improved performance for Class 1 after tuning hyperparameters (C=10, solver=liblinear).</p> <p><b>Execution problems:</b> None.</p> <p><b>Data quality issues:</b> None after cleaning and encoding.</p>

## Modeling Report

	Calculation inconsistencies: None.
<b>K-Nearest Neighbors (KNN)</b>	<p><b>Meaningful conclusions:</b> The model gave a good accuracy (81.4%), showing it can predict hair fall quite well. It captured "Yes" hair fall better than "No"</p> <p><b>New insights or unusual patterns:</b> It was easier for KNN to predict cases where people <b>had hair fall</b> (recall = 91%) compared to those who <b>did not</b>.</p> <p><b>Execution problems and processing time:</b> No problems. KNN trained very fast since it's a simple algorithm.</p> <p><b>Data quality issues:</b> No issues from missing values, because the data was preprocessed properly.</p> <p><b>Calculation inconsistencies:</b> No inconsistencies found. The Elbow method showed that k=5 was a good choice.</p>
<b>KNN with GridSearchCV</b>	<p><b>Accuracy: ~79%</b></p> <p><b>Precision/Recall:</b></p> <p>Class 0 (No hair fall): precision = 0.73, recall = 0.53</p> <p>Class 1 (Yes hair fall): precision = 0.81, recall = 0.91</p> <p><b>Meaningful conclusions:</b> Yes, especially for identifying Class 1 correctly.</p> <p><b>New insights:</b> The model is much better at detecting hair fall than detecting its absence. The GridSearchCV classifier showed</p>

# Modeling Report

	<p>even higher model accuracy.</p> <p><b>Execution problems:</b> None.</p> <p><b>Data quality issues:</b> None (preprocessed).</p> <p><b>Calculation inconsistencies:</b> None.</p>
Random Forest	<p><b>Meaningful conclusions:</b> The model had solid results (accuracy ~78%), showing it's reliable for predicting hair fall problems.</p> <p><b>New insights or unusual patterns:</b> It balanced well between predicting "Yes" and "No" hair fall cases, but slightly better at catching "Yes" cases.</p> <p><b>Execution problems and processing time:</b> No execution problems. Random Forest took a bit more time than KNN or Logistic Regression but was still reasonable.</p> <p><b>Data quality issues:</b> No data quality problems noticed.</p> <p><b>Calculation inconsistencies:</b> No inconsistencies. Model was stable and gave expected outputs.</p>
RandomForest with GridSearchCV	<p><b>Accuracy:</b> ~80%</p> <p><b>Precision/Recall:</b></p> <p>Class 0 (No hair fall): precision = 0.72, recall = 0.60</p> <p>Class 1 (Yes hair fall): precision = 0.83, recall = 0.90</p> <p><b>Meaningful conclusions:</b> Yes, especially effective at detecting</p>



# Modeling Report

	<p>individuals with hair fall.</p> <p><b>New insights:</b> The model favors predicting “Yes” correctly, but struggles slightly with “No.” The GridSearchCV classifier helped this model achieve the second highest accuracy score among the rest of the tabular models.</p> <p><b>Execution problems:</b> None.</p> <p><b>Data quality issues:</b> None (after cleaning).</p> <p><b>Calculation inconsistencies:</b> None.</p>
XGBoost	<p><b>Meaningful conclusions:</b> XGBoost gave the best accuracy (~82%), making it the strongest model so far for predicting hair fall.</p> <p><b>New insights or unusual patterns:</b> The model predicted "Yes" cases very well with high recall and precision.</p> <p><b>Execution problems and processing time:</b> No problems. It trained a bit slower than Random Forest but was still manageable.</p> <p><b>Data quality issues:</b> No major data issues noticed.</p> <p><b>Calculation inconsistencies:</b> No inconsistencies. Results were clean and strong.</p>
Weighted KNN	<p><b>Meaningful Conclusions:</b> Weighted KNN gave around 77% accuracy. It slightly improved recall compared to basic KNN for "No" cases.</p> <p><b>New Insights:</b> Giving more weight to closer neighbors helped balance predictions.</p> <p><b>Execution Problems:</b> No issues. Training time was fast.</p>

Modeling Report

	<b>Data Quality:</b> Data was clean, no missing values. <b>Calculation Issues:</b> None.
--	---

Scalp Image Dataset (Roboflow)

Model Name	Model Description
ResNet50	<p><b>Meaningful conclusions:</b> Performance was low (~38% accuracy) compared to tabular models.</p> <p><b>New insights:</b> Model struggled with small dataset size and limited samples per class.</p> <p><b>Execution problems:</b> No technical issues, but training time was longer (~7 seconds per batch).</p> <p><b>Data quality issues:</b> Dataset imbalance and small number of images per class caused problems.</p> <p><b>Calculation inconsistencies:</b> None.</p>
MobileNet	<p><b>Meaningful conclusions:</b> The MobileNet model achieved a <b>test accuracy of ~53%</b>, performing notably better on <b>Level 2</b> and <b>Level 4</b>, while struggling with underrepresented classes such as <b>Level 3, 5, 6, and 7</b>.</p> <p><b>New insights:</b> The model's F1-score reveals that performance <b>degraded for rare classes</b>, especially <b>Level 3 and 7</b>, despite being visually distinct. This highlights the importance of class balance and training image diversity in deep learning models.</p> <p><b>Execution problems:</b> No execution issues were encountered. Training and evaluation were <b>computationally efficient</b> due to MobileNet's lightweight architecture, with acceptable batch processing times.</p> <p><b>Data quality issues:</b>The dataset was <b>heavily imbalanced</b>, with <b>Level 6 (8 test / 333 train)</b> and <b>Level 7 (7 test / 159 train)</b> having significantly fewer samples than <b>Level 2 or 3</b>. This</p>

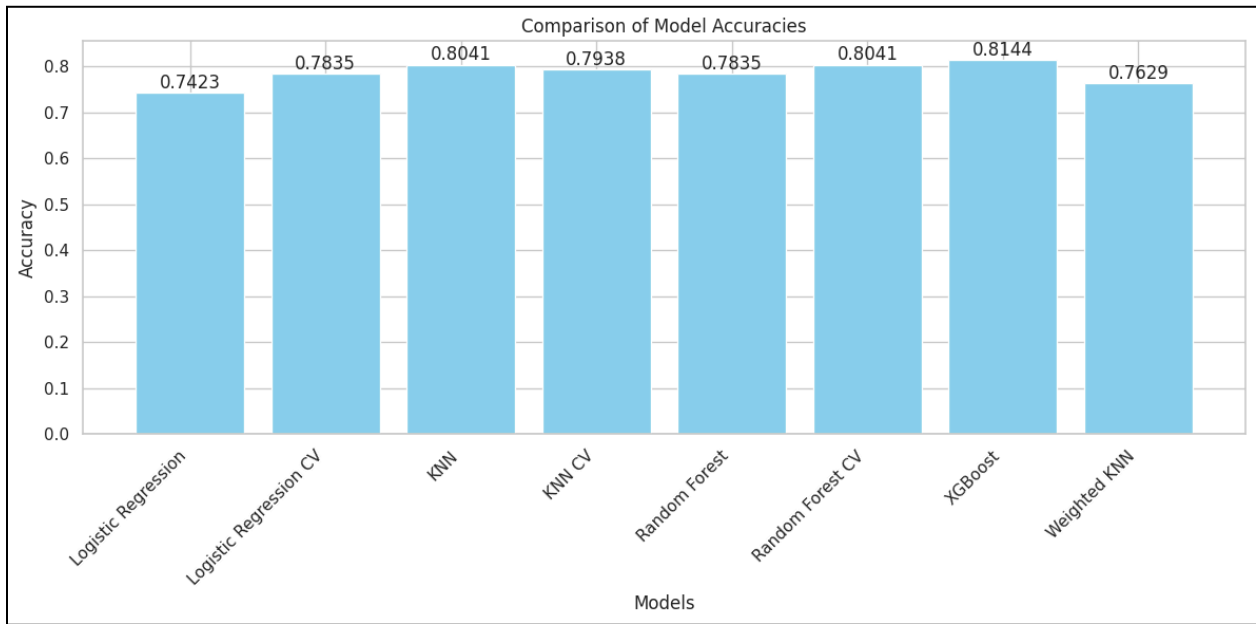
Modeling Report

	<p>imbalance likely contributed to reduced generalization and uneven class-wise accuracy.</p> <p><b>Calculation inconsistencies:</b> None were observed. The model ran smoothly across training, validation, and testing stages.</p>
VGG16	<p>The VGG16 model achieved an overall test accuracy of <b>41%</b>, with a <b>weighted F1-score of 0.40</b>. While the model performed relatively well in predicting <b>Level 3</b> (recall = 0.78) and <b>Level 2</b> (precision = 0.68), it struggled significantly with higher baldness stages like <b>Level 7</b> (precision and recall = 0.14). The confusion matrix reveals frequent misclassifications, particularly between adjacent stages such as Levels 3–4 and 4–5, suggesting the model had difficulty distinguishing between similar visual patterns. Although no technical execution issues occurred, the low scores indicate that the dataset may need better class balance or augmentation to improve performance across all classes.</p>
Custom CNN	<p><b>Meaningful conclusions:</b> The custom CNN model achieved <b>low overall test accuracy (~26%)</b> and <b>low macro and weighted F1-scores (~0.20)</b>. The model performed particularly poorly on the more frequent classes like <b>Level 2 and Level 4</b>, indicating difficulty in learning generalized patterns.</p> <p><b>New insights:</b></p> <p>Interestingly, the model showed <b>high recall (0.89) for Level 3</b> despite modest precision, suggesting that it <b>over-predicted Level 3</b> across various classes. This may be due to its large representation in the training set (529 images). Conversely, <b>Level 5</b> had zero recall and precision, indicating the model completely failed to recognize it.</p>

	<p><b>Execution problems:</b></p> <p>No execution errors were encountered during training or evaluation. The model trained relatively quickly due to its simple architecture, making it suitable for low-resource experimentation — though at the cost of reduced accuracy.</p> <p><b>Data quality issues:</b></p> <p>Similar to MobileNet, the custom CNN struggled with the <b>imbalanced dataset</b>, especially with classes like <b>Level 7 (7 test / 159 train)</b> and <b>Level 5 (14 test / 248 train)</b>. The lack of diverse samples in those categories likely caused poor generalization.</p> <p><b>Calculation inconsistencies:</b></p> <p>No computational issues were noted. However, the <b>zero precision/recall for Level 5</b> could be a sign that either the model didn't learn relevant features, or the features were too weak for the network to capture without deeper layers or augmentation.</p>
--	--

4. Model Assessment

Hair Fall Causes Dataset (Mendeley)



Comparison of model accuracies showing XGBoost as the top performer, followed closely by KNN and Random Forest.

After training and evaluating all the selected models, we compared their performance using accuracy as the main metric. Additionally, we considered how interpretable and easy-to-use each model is for practical use.

Final Model Ranking (based on accuracy & clarity):

- XGBoost (accuracy: 0.8144)**
  - Best performing model in terms of accuracy.
  - Shows strong precision and recall.
  - Suitable as a final model.
- KNN (accuracy: 0.8041)**
  - Performed well with minimal tuning.
  - Simple to understand.
  - Sensitive to feature scaling, but handled well here.
- Random Forest (GridSearchCV, accuracy: 0.8041)**
  - Very good performance and interpretability with feature importance.

## Modeling Report

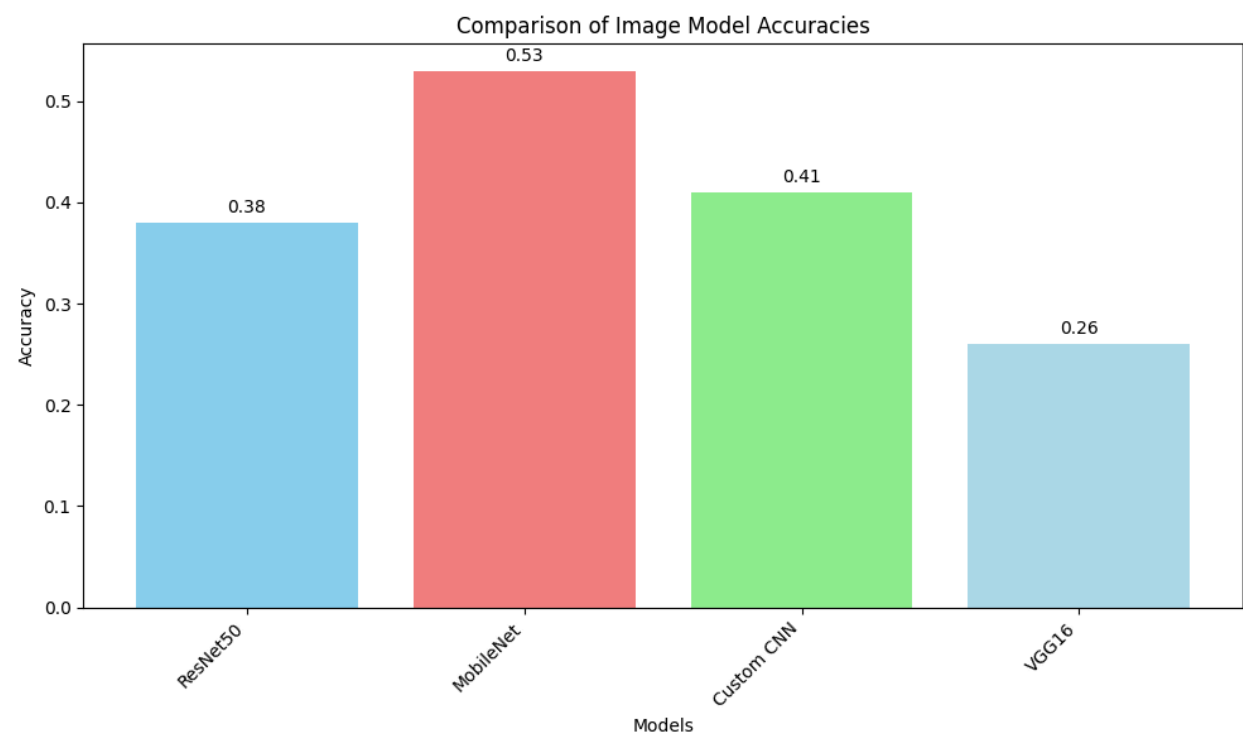
---

- Slightly longer training time.
- 4. **KNN (GridSearchCV, accuracy: 0.7938)**
  - Performance was solid, but not a huge improvement over basic KNN.
  - Easy to run and understand.
- 5. **Logistic Regression (GridSearchCV, accuracy: 0.7835)**
  - Improved over the default Logistic Regression model.
  - Very interpretable and fast.
- 6. **Random Forest (default, accuracy: 0.7835)**
  - Similar to the GridSearchCV version, but without parameter optimization.
- 7. **Weighted KNN (accuracy: 0.7629)**
  - Decent accuracy, but not better than regular KNN.
  - The added complexity didn't result in improved results.
- 8. **Logistic Regression (default, accuracy: 0.7423)**
  - Fast and easy, but performance was the lowest among all models.  
Still useful as a baseline for comparison.

### Summary:

**XGBoost** is the best model in terms of objective performance. However, if interpretability or simplicity is preferred, **KNN** or **Logistic Regression (CV)** could also be good options. All models performed reasonably well after preprocessing, and no major data issues or errors occurred during execution.

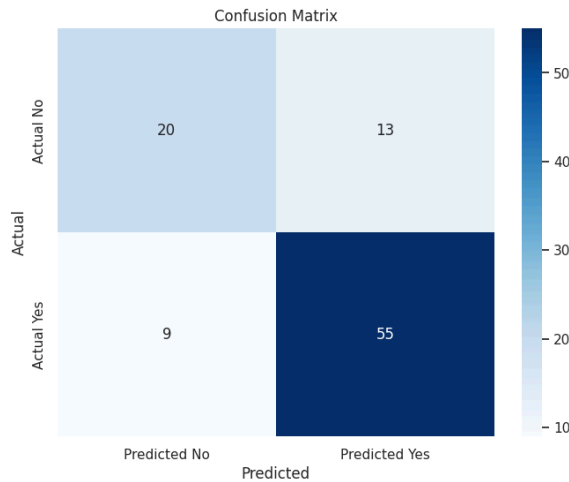
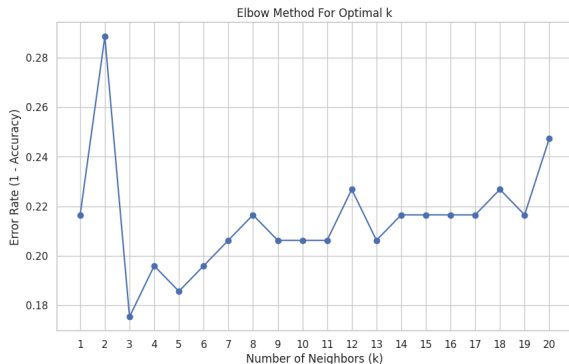
Scalp Image Dataset (Roboflow)



**MobileNet, with a test accuracy of 0.53**, is the most promising model for classifying male pattern baldness stages based on image model accuracy comparison. Following MobileNet, the **Custom CNN achieved 0.41**, **ResNet50 reached 0.38**, and **VGG16 had the lowest accuracy at 0.26**. Both objective and subjective assessments indicate MobileNet as the most effective and potentially deployable model. These results also suggest that deeper networks like VGG16 may not always perform optimally on small or imbalanced datasets, where simpler architectures such as MobileNet can offer better generalization.

## Appendix: Models Results Graphs

### Hair Fall Causes Dataset (Mendeley)

Model Name	Screenshot	Description																																										
Logistic Regression	 <p>Confusion Matrix</p> <table><tr><th></th><th>Predicted No</th><th>Predicted Yes</th></tr><tr><th>Actual No</th><td>20</td><td>13</td></tr><tr><th>Actual Yes</th><td>9</td><td>55</td></tr></table>		Predicted No	Predicted Yes	Actual No	20	13	Actual Yes	9	55	<p>The Logistic Regression model, tuned using GridSearchCV, achieved a test accuracy of approximately 77% and a weighted F1-score of 0.77. It performed particularly well in identifying positive cases of hair fall, with a recall of 0.86 and F1-score of 0.83 for class 1. This indicates the model is effective at minimizing false negatives.</p>																																	
	Predicted No	Predicted Yes																																										
Actual No	20	13																																										
Actual Yes	9	55																																										
K-Nearest Neighbors (KNN)	 <p>Elbow Method For Optimal k</p> <table><tr><th>Number of Neighbors (k)</th><th>Error Rate (1 - Accuracy)</th></tr><tr><td>1</td><td>0.215</td></tr><tr><td>2</td><td>0.285</td></tr><tr><td>3</td><td>0.175</td></tr><tr><td>4</td><td>0.195</td></tr><tr><td>5</td><td>0.185</td></tr><tr><td>6</td><td>0.195</td></tr><tr><td>7</td><td>0.205</td></tr><tr><td>8</td><td>0.215</td></tr><tr><td>9</td><td>0.205</td></tr><tr><td>10</td><td>0.205</td></tr><tr><td>11</td><td>0.205</td></tr><tr><td>12</td><td>0.225</td></tr><tr><td>13</td><td>0.205</td></tr><tr><td>14</td><td>0.215</td></tr><tr><td>15</td><td>0.215</td></tr><tr><td>16</td><td>0.215</td></tr><tr><td>17</td><td>0.215</td></tr><tr><td>18</td><td>0.225</td></tr><tr><td>19</td><td>0.215</td></tr><tr><td>20</td><td>0.245</td></tr></table>	Number of Neighbors (k)	Error Rate (1 - Accuracy)	1	0.215	2	0.285	3	0.175	4	0.195	5	0.185	6	0.195	7	0.205	8	0.215	9	0.205	10	0.205	11	0.205	12	0.225	13	0.205	14	0.215	15	0.215	16	0.215	17	0.215	18	0.225	19	0.215	20	0.245	<p>The KNN model was evaluated both with and without hyperparameter tuning using GridSearchCV. Without tuning, the model achieved a test accuracy of 81.44% and a weighted F1-score of 0.81, with strong performance in identifying individuals with hair fall (Class 1 F1-score: 0.87).</p> <p>After tuning with GridSearchCV, the best</p>
Number of Neighbors (k)	Error Rate (1 - Accuracy)																																											
1	0.215																																											
2	0.285																																											
3	0.175																																											
4	0.195																																											
5	0.185																																											
6	0.195																																											
7	0.205																																											
8	0.215																																											
9	0.205																																											
10	0.205																																											
11	0.205																																											
12	0.225																																											
13	0.205																																											
14	0.215																																											
15	0.215																																											
16	0.215																																											
17	0.215																																											
18	0.225																																											
19	0.215																																											
20	0.245																																											



Modeling Report

	<div><div>KNN Confusion Matrix</div><table><tr><th></th><th>Predicted No</th><th>Predicted Yes</th></tr><tr><th>Actual No</th><td>21</td><td>12</td></tr><tr><th>Actual Yes</th><td>6</td><td>58</td></tr></table></div>		Predicted No	Predicted Yes	Actual No	21	12	Actual Yes	6	58	<p>configuration was found at <code>n_neighbors=9</code>. The optimized model produced a test accuracy of 79.38% and a weighted F1-score of 0.79. While this represented a slight decrease in overall accuracy, the model maintained high performance for Class 1 (F1-score: 0.85) and showed slightly more balanced performance across both classes.</p>
	Predicted No	Predicted Yes									
Actual No	21	12									
Actual Yes	6	58									
Random Forest	<div><div>Random Forest Confusion Matrix</div><table><tr><th></th><th>Predicted No</th><th>Predicted Yes</th></tr><tr><th>Actual No</th><td>21</td><td>12</td></tr><tr><th>Actual Yes</th><td>8</td><td>56</td></tr></table></div>		Predicted No	Predicted Yes	Actual No	21	12	Actual Yes	8	56	<p>The Random Forest model, optimized using GridSearchCV, achieved a test accuracy of approximately 79% and a weighted F1-score of 0.79. The best-performing parameters were found to be <code>n_estimators=200</code>, <code>max_depth=5</code>, and <code>min_samples_split=2</code>. The model demonstrated strong performance in detecting individuals with hair fall, achieving a recall of 0.88 and F1-score of 0.85 for the positive class (Class 1).</p>
	Predicted No	Predicted Yes									
Actual No	21	12									
Actual Yes	8	56									

Modeling Report

		<p>The confusion matrix shows the model predicted 56 out of 64 hair fall cases correctly, with relatively few false negatives. This highlights the model's reliability in identifying the condition, which is important for health-related classification tasks.</p>
<p><b>XGBoost</b></p>	<div> <div> <div>XGBoost Confusion Matrix</div> <div> <div> <div>Actual No</div> <div>Actual Yes</div> </div> <div> <div>20</div> <div>8</div> </div> <div> <div>10</div> <div>59</div> </div> <div> <div>Predicted No</div> <div>Predicted Yes</div> </div> </div> <div> <div>50</div> <div>40</div> <div>30</div> <div>20</div> <div>10</div> </div> </div> </div>	<p>The XGBoost model achieved a test accuracy of approximately 82% with a weighted F1-score of 0.82. The confusion matrix indicates strong performance on both classes: it correctly predicted 59 out of 67 hair fall cases (Class 1) and 20 out of 30 no hair fall cases (Class 0). With only 8 false negatives and 10 false positives, the model demonstrates a good balance between sensitivity and specificity, making it one of the most reliable classifiers in this task.</p>

Modeling Report

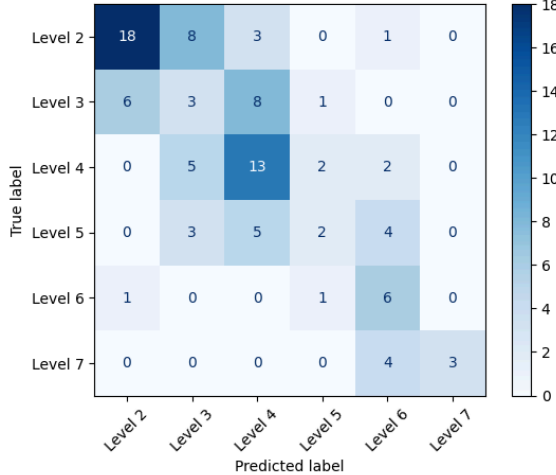
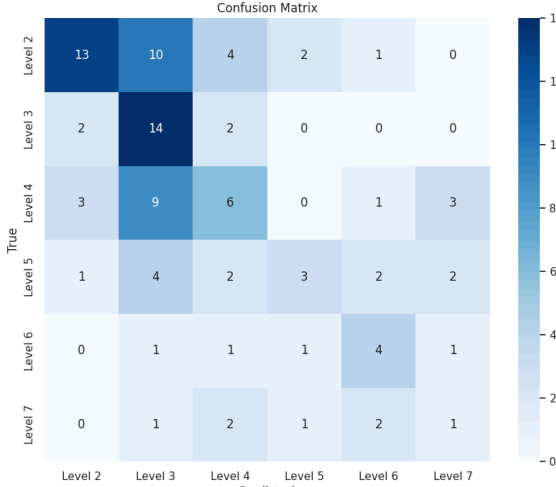
<b>Weighted KNN</b>	<div><p>Weighted KNN Confusion Matrix</p><table><thead><tr><th></th><th>Predicted No</th><th>Predicted Yes</th></tr></thead><tbody><tr><th>Actual No</th><td>20</td><td>10</td></tr><tr><th>Actual Yes</th><td>13</td><td>54</td></tr></tbody></table></div>		Predicted No	Predicted Yes	Actual No	20	10	Actual Yes	13	54	<p>The Weighted KNN model achieved a test accuracy of approximately 77% with a weighted F1-score of 0.78. The confusion matrix shows that the model correctly classified 54 out of 67 individuals with hair fall (Class 1), while 13 were misclassified as not having the condition. For the negative class (Class 0), it predicted 20 out of 30 cases correctly. This model performs slightly better at identifying positive cases than negative ones, making it useful in contexts where detecting hair fall is more critical than avoiding false alarms.</p>
	Predicted No	Predicted Yes									
Actual No	20	10									
Actual Yes	13	54									

Scalp Image Dataset (Roboflow)

Model Name	Screenshot	Description																																																																																														
ResNet50	<div><p>Confusion Matrix</p><table><tr><th></th><th>Level 2</th><th>Level 3</th><th>Level 4</th><th>Level 5</th><th>Level 6</th><th>Level 7</th></tr><tr><th>Level 2</th><td>8</td><td>18</td><td>2</td><td>1</td><td>0</td><td>1</td></tr><tr><th>Level 3</th><td>2</td><td>11</td><td>3</td><td>1</td><td>1</td><td>0</td></tr><tr><th>Level 4</th><td>0</td><td>7</td><td>10</td><td>0</td><td>5</td><td>0</td></tr><tr><th>Level 5</th><td>0</td><td>4</td><td>5</td><td>2</td><td>3</td><td>0</td></tr><tr><th>Level 6</th><td>0</td><td>0</td><td>1</td><td>0</td><td>7</td><td>0</td></tr><tr><th>Level 7</th><td>1</td><td>2</td><td>0</td><td>1</td><td>0</td><td>3</td></tr></table><p>precision    recall    f1-score    support</p><table><tr><td>Level 2</td><td>0.73</td><td>0.27</td><td>0.39</td><td>30</td></tr><tr><td>Level 3</td><td>0.26</td><td>0.61</td><td>0.37</td><td>18</td></tr><tr><td>Level 4</td><td>0.48</td><td>0.45</td><td>0.47</td><td>22</td></tr><tr><td>Level 5</td><td>0.40</td><td>0.14</td><td>0.21</td><td>14</td></tr><tr><td>Level 6</td><td>0.44</td><td>0.88</td><td>0.58</td><td>8</td></tr><tr><td>Level 7</td><td>0.75</td><td>0.43</td><td>0.55</td><td>7</td></tr><tr><td>accuracy</td><td></td><td></td><td>0.41</td><td>99</td></tr><tr><td>macro avg</td><td>0.51</td><td>0.46</td><td>0.43</td><td>99</td></tr><tr><td>weighted avg</td><td>0.52</td><td>0.41</td><td>0.40</td><td>99</td></tr></table></div>		Level 2	Level 3	Level 4	Level 5	Level 6	Level 7	Level 2	8	18	2	1	0	1	Level 3	2	11	3	1	1	0	Level 4	0	7	10	0	5	0	Level 5	0	4	5	2	3	0	Level 6	0	0	1	0	7	0	Level 7	1	2	0	1	0	3	Level 2	0.73	0.27	0.39	30	Level 3	0.26	0.61	0.37	18	Level 4	0.48	0.45	0.47	22	Level 5	0.40	0.14	0.21	14	Level 6	0.44	0.88	0.58	8	Level 7	0.75	0.43	0.55	7	accuracy			0.41	99	macro avg	0.51	0.46	0.43	99	weighted avg	0.52	0.41	0.40	99	<p>The ResNet50 model demonstrated moderate overall performance with an accuracy of 41%, a macro average F1-score of 0.43, and a weighted average F1-score of 0.40. While it excelled at identifying Level 6 (F1-score: 0.58) and Level 7 (F1-score: 0.55) baldness, despite limited data, it struggled with Levels 2 and 3, exhibiting lower recall (0.27 and 0.61) despite larger sample sizes. Analysis of the confusion matrix revealed frequent misclassifications between adjacent baldness levels, indicating difficulty in discerning subtle differences. In conclusion, ResNet50's fine-grained classification capabilities were limited, though it showed promise in recognizing more pronounced patterns of baldness.</p>
	Level 2	Level 3	Level 4	Level 5	Level 6	Level 7																																																																																										
Level 2	8	18	2	1	0	1																																																																																										
Level 3	2	11	3	1	1	0																																																																																										
Level 4	0	7	10	0	5	0																																																																																										
Level 5	0	4	5	2	3	0																																																																																										
Level 6	0	0	1	0	7	0																																																																																										
Level 7	1	2	0	1	0	3																																																																																										
Level 2	0.73	0.27	0.39	30																																																																																												
Level 3	0.26	0.61	0.37	18																																																																																												
Level 4	0.48	0.45	0.47	22																																																																																												
Level 5	0.40	0.14	0.21	14																																																																																												
Level 6	0.44	0.88	0.58	8																																																																																												
Level 7	0.75	0.43	0.55	7																																																																																												
accuracy			0.41	99																																																																																												
macro avg	0.51	0.46	0.43	99																																																																																												
weighted avg	0.52	0.41	0.40	99																																																																																												



## Modeling Report

<div>MobileNet</div>	<div><div>Test Set Confusion Matrix</div><table><thead><tr><th></th><th>precision</th><th>recall</th><th>f1-score</th><th>support</th></tr></thead><tbody><tr><td>Level 2</td><td>0.81</td><td>0.70</td><td>0.75</td><td>30</td></tr><tr><td>Level 3</td><td>0.24</td><td>0.28</td><td>0.26</td><td>18</td></tr><tr><td>Level 4</td><td>0.48</td><td>0.50</td><td>0.49</td><td>22</td></tr><tr><td>Level 5</td><td>0.55</td><td>0.43</td><td>0.48</td><td>14</td></tr><tr><td>Level 6</td><td>0.45</td><td>0.62</td><td>0.53</td><td>8</td></tr><tr><td>Level 7</td><td>0.57</td><td>0.57</td><td>0.57</td><td>7</td></tr><tr><td>accuracy</td><td></td><td></td><td>0.53</td><td>99</td></tr></tbody></table></div>		precision	recall	f1-score	support	Level 2	0.81	0.70	0.75	30	Level 3	0.24	0.28	0.26	18	Level 4	0.48	0.50	0.49	22	Level 5	0.55	0.43	0.48	14	Level 6	0.45	0.62	0.53	8	Level 7	0.57	0.57	0.57	7	accuracy			0.53	99	<div><p>The MobileNet model achieved a test accuracy of approximately 53% with a weighted F1-score of 0.53. The confusion matrix indicates the model correctly classified many Level 2 cases, but struggled with other levels, particularly the underrepresented classes such as Level 6 and Level 7. Some misclassifications occurred between adjacent levels, which may reflect the visual similarity across certain stages of hair loss.</p></div>
	precision	recall	f1-score	support																																						
Level 2	0.81	0.70	0.75	30																																						
Level 3	0.24	0.28	0.26	18																																						
Level 4	0.48	0.50	0.49	22																																						
Level 5	0.55	0.43	0.48	14																																						
Level 6	0.45	0.62	0.53	8																																						
Level 7	0.57	0.57	0.57	7																																						
accuracy			0.53	99																																						
<div>VGG16</div>	<div><div>Confusion Matrix</div></div>	<div><p>The VGG16 model demonstrated poor accuracy (41%) in classifying male pattern baldness stages, struggling to differentiate between the six levels. Although Level 3 showed the highest recall (0.78), its low precision (0.36) indicates a high number of false positives. The model performed particularly poorly on advanced stages (Levels</p></div>																																								

Modeling Report

	<table><thead><tr><th></th><th>precision</th><th>recall</th><th>f1-score</th><th>support</th></tr></thead><tbody><tr><td>Level 2</td><td>0.68</td><td>0.43</td><td>0.53</td><td>30</td></tr><tr><td>Level 3</td><td>0.36</td><td>0.78</td><td>0.49</td><td>18</td></tr><tr><td>Level 4</td><td>0.35</td><td>0.27</td><td>0.31</td><td>22</td></tr><tr><td>Level 5</td><td>0.43</td><td>0.21</td><td>0.29</td><td>14</td></tr><tr><td>Level 6</td><td>0.40</td><td>0.50</td><td>0.44</td><td>8</td></tr><tr><td>Level 7</td><td>0.14</td><td>0.14</td><td>0.14</td><td>7</td></tr><tr><td>accuracy</td><td></td><td></td><td>0.41</td><td>99</td></tr><tr><td>macro avg</td><td>0.39</td><td>0.39</td><td>0.37</td><td>99</td></tr><tr><td>weighted avg</td><td>0.45</td><td>0.41</td><td>0.40</td><td>99</td></tr></tbody></table>		precision	recall	f1-score	support	Level 2	0.68	0.43	0.53	30	Level 3	0.36	0.78	0.49	18	Level 4	0.35	0.27	0.31	22	Level 5	0.43	0.21	0.29	14	Level 6	0.40	0.50	0.44	8	Level 7	0.14	0.14	0.14	7	accuracy			0.41	99	macro avg	0.39	0.39	0.37	99	weighted avg	0.45	0.41	0.40	99	<p>5-7), with Level 7 achieving only 0.14 for both recall and precision. Misclassifications were prevalent across all stages, with significant overlap, especially between adjacent levels. This suggests the VGG16 model struggled to extract relevant features from the scalp images. To improve performance, considerations should include better data balance, augmentation techniques, or fine-tuning of the model.</p>
	precision	recall	f1-score	support																																																
Level 2	0.68	0.43	0.53	30																																																
Level 3	0.36	0.78	0.49	18																																																
Level 4	0.35	0.27	0.31	22																																																
Level 5	0.43	0.21	0.29	14																																																
Level 6	0.40	0.50	0.44	8																																																
Level 7	0.14	0.14	0.14	7																																																
accuracy			0.41	99																																																
macro avg	0.39	0.39	0.37	99																																																
weighted avg	0.45	0.41	0.40	99																																																
Custom CNN	<div><p>Test Set Confusion Matrix</p><table><thead><tr><th></th><th>Level 2</th><th>Level 3</th><th>Level 4</th><th>Level 5</th><th>Level 6</th><th>Level 7</th></tr></thead><tbody><tr><th>Level 2</th><td>7</td><td>22</td><td>0</td><td>0</td><td>1</td><td>0</td></tr><tr><th>Level 3</th><td>4</td><td>12</td><td>0</td><td>1</td><td>1</td><td>0</td></tr><tr><th>Level 4</th><td>1</td><td>12</td><td>2</td><td>1</td><td>6</td><td>0</td></tr><tr><th>Level 5</th><td>0</td><td>7</td><td>0</td><td>3</td><td>4</td><td>0</td></tr><tr><th>Level 6</th><td>0</td><td>3</td><td>1</td><td>1</td><td>3</td><td>0</td></tr><tr><th>Level 7</th><td>0</td><td>2</td><td>0</td><td>1</td><td>4</td><td>0</td></tr></tbody></table></div>		Level 2	Level 3	Level 4	Level 5	Level 6	Level 7	Level 2	7	22	0	0	1	0	Level 3	4	12	0	1	1	0	Level 4	1	12	2	1	6	0	Level 5	0	7	0	3	4	0	Level 6	0	3	1	1	3	0	Level 7	0	2	0	1	4	0	<p>The Custom CNN model reached a test accuracy of approximately 26% and a weighted F1-score of 0.20, indicating limited overall predictive performance. The confusion matrix reveals significant misclassification across all levels, with notably poor recognition of Level 5, which had zero correct predictions. The model showed a tendency to over-predict Level 3, likely due to its high representation in the training set. This behavior reduced precision and contributed to class</p>	
	Level 2	Level 3	Level 4	Level 5	Level 6	Level 7																																														
Level 2	7	22	0	0	1	0																																														
Level 3	4	12	0	1	1	0																																														
Level 4	1	12	2	1	6	0																																														
Level 5	0	7	0	3	4	0																																														
Level 6	0	3	1	1	3	0																																														
Level 7	0	2	0	1	4	0																																														

Modeling Report

	<div> <div>Class</div> <div>Precision</div> <div>Recall</div> <div>F1-score</div> <div>Support</div> </div> <div> <div>Level 2</div> <div>1.00</div> <div>0.10</div> <div>0.18</div> <div>30</div> </div> <div> <div>Level 3</div> <div>0.24</div> <div>0.89</div> <div>0.38</div> <div>18</div> </div> <div> <div>Level 4</div> <div>0.67</div> <div>0.09</div> <div>0.16</div> <div>22</div> </div> <div> <div>Level 5</div> <div>0.00</div> <div>0.00</div> <div>0.00</div> <div>14</div> </div> <div> <div>Level 6</div> <div>0.16</div> <div>0.50</div> <div>0.24</div> <div>8</div> </div> <div> <div>Level 7</div> <div>1.00</div> <div>0.14</div> <div>0.25</div> <div>7</div> </div>	<div> <div>imbalance sensitivity.</div> </div>
--	---	--

Appendix: Terminology

- List of terms or jargon related to the report’s topic:**
  - Multimodal data:** Refers to using two different types of datasets (images and numerical data) together.
  - Deep learning:** A subset of machine learning used for tasks like analyzing scalp images.
  - Machine learning:** Techniques used to analyze the numerical dataset (e.g., blood component levels).
  - Model accuracy:** A performance metric showing how well the model predicts hair loss levels.
  - Male pattern baldness stages (1-7):** This refers to the **Norwood Scale**, which categorizes hair loss in men from Stage 1 (minimal or no hair loss) to Stage 7 (severe hair loss with only a band of hair around the sides and back).
  - Hair Recession:** The process where the hairline moves backward, typically at the temples, resulting in an "M" shape. It is one of the early signs of male pattern baldness.
  - Hair follicle:** The small structure in the scalp that produces and grows hair. Healthy follicles are essential for proper hair growth.
  - Hair loss due to malnutrition:** Hair loss caused by deficiencies in essential nutrients like vitamins (e.g., Vitamin D, B12), minerals (e.g., iron, zinc), or protein. Poor nutrition weakens hair follicles and slows down hair growth.

- **Hair loss due to genetics:** Hereditary hair loss, often referred to as androgenetic alopecia, is caused by genetic factors passed down from either parent, influencing the pattern and severity of hair loss.
- **Hair loss due to hormones (DHT):** Dihydrotestosterone (DHT) is a hormone derived from testosterone. It can shrink hair follicles in genetically predisposed individuals, leading to hair thinning and eventual hair loss.
- **Hair loss preventative methods:** Techniques or treatments aimed at slowing hair loss, such as using **minoxidil**, taking **finasteride**, improving diet, reducing stress, derma rolling, PRP treatments, and avoiding harsh hair treatments.
- **Minoxidil:** A topical medication used to slow hair loss and, in some cases, promote hair regrowth. It is applied directly to the scalp and is commonly used for treating male pattern baldness. It's available over the counter and is most effective when used consistently. Stopping its usage would cause losing any hair regrowth achieved in course of several months.
- **Derma rolling:** This method involves using a handheld device with tiny needles (a derma roller) to create micro-injuries on the scalp. This process stimulates blood circulation, encourages collagen production, and promotes the absorption of topical hair treatments like minoxidil.
- **Finasteride:** An oral medication that helps reduce hair loss by blocking the production of **DHT** (Dihydrotestosterone), the hormone responsible for shrinking hair follicles. It requires a prescription and is effective in treating male pattern baldness, particularly on the crown and mid-scalp areas. Discontinuing finasteride leads to a gradual return of DHT levels to their pre-treatment state. Over several months, hair follicles affected by DHT may shrink again and cause hair loss to resume
- **Hair Transplant Methods (DHI):**
  - i. **Direct Hair Implantation (DHI):** A transplant method where hair follicles are extracted and implanted directly into the scalp using a specialized tool, without pre-creating channels.
  - ii. **Follicular Unit Extraction (FUE):** A technique where individual hair follicles are extracted from the donor area and implanted into the transplant area. It's minimally invasive and leaves tiny scars.



iii. **Hair Transplant Methods (HYBRID):** A combination of DHI and FUE techniques to utilize the strengths of both methods, offering a tailored approach for better results.

- **Hairline:** The outline of hair growth on the forehead. A natural-looking hairline is a key factor in successful hair restoration procedures.
- **Donor area:** The region, usually at the back or sides of the scalp, where hair follicles are taken for transplantation. This area typically has resistant follicles to hair loss (resistant to DHT).
- **Transplant area:** The part of the scalp where hair follicles are implanted during a hair transplant procedure, usually the balding or thinning regions.

