Prediction of Alzheimer stages for given MRI scans

Aaradhya Verma (2022004) Aarya Khandelwal (2022007)

I. Problem Statement

Alzheimer's disease is a progressive neuro-degenerative disorder that affects millions of people worldwide, leading to cognitive decline and ultimately severe dementia.
Early detection and accurate classification of Alzheimer's disease stages are crucial for timely intervention and treatment. Magnetic Resonance Imaging (MRI) scans offer detailed structural insights into the brain, presenting a valuable resource for diagnosing Alzheimer's disease.
In this study, our objective is to elucidate the data processing methodologies employed on the original dataset and to assess various machine learning models for their efficacy in classifying Alzheimer's disease stages based on MRI scans.

II. Dataset

It comprises images of MRI (Magnetic Resonance Imaging) scans , each resized to 128×128 pixels. It contains approximately 5200 MRI images, with patients classified into four distinct classes:

- Class 1: Mild Demented (896 images)
- Class 2: Moderate Demented (64 images)
- Class 3: Non-Demented (2000 images)
- Class 4: Very Mild Demented (2240 images)

Dataset: https://www.kaggle.com/datasets/sachinkumar413/alzheimer-mri-dataset/data.

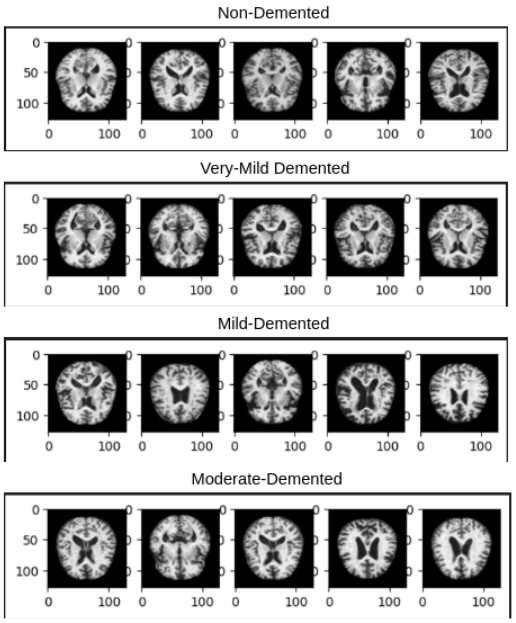

Fig. 1. Samples of every class

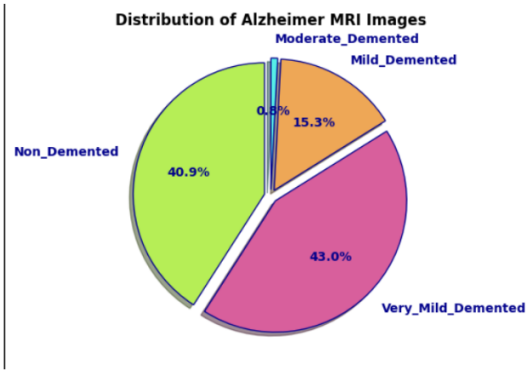

Fig. 2. Dataset1 distribution

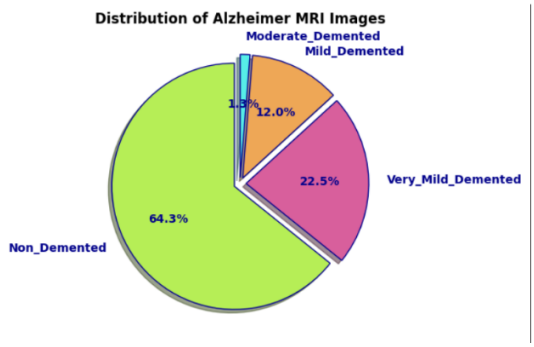

Fig. 3. Dataset2 distribution

III. Data Pre-processing

- **Image Loading:** The dataset was divided into four categories based on disease stage. All of the images were gray-scaled to reduce data-complexity and ensure uniformity in pixel values across all images.
- **Data Splitting:** The dataset was split into training, validation, and test sets. The training set constituted 80% of the data, while the validation and test sets comprised 10% each.
- **Data Augmentation:** To enhance the diversity of the training dataset and improve model generalization, data augmentation techniques were applied. Brightness was adjusted and Gaussian noise was addition to create augmented versions of the original images.
- **Principal Component Analysis (PCA):** PCA was employed to reduce the dimensionality of the feature space, focusing solely on dimensions that have a significant impact on the data. The optimal number of principal components was determined based on the cumulative explained variance, with the objective of retaining the

Fig. 4. Data Augmentation

maximum possible variance in the data.

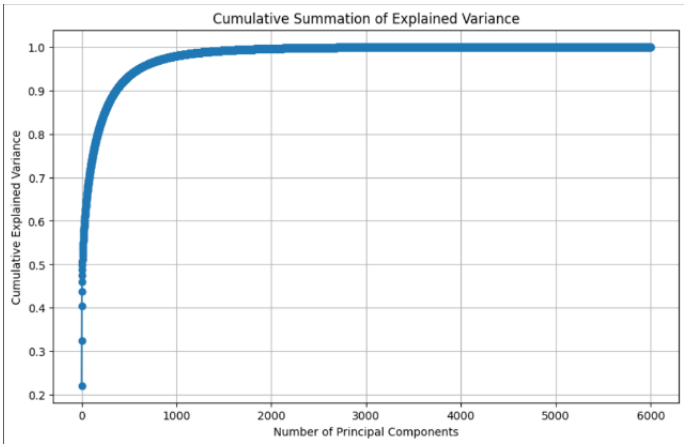

Fig. 5. PCA: dimension-data-explanation plot

IV. Models Applied

- **Decision Tree Classification:** It is a versatile supervised learning algorithm used for both classification and regression tasks. It works by partitioning the data into subsets based on the features' values, creating a tree-like structure of decisions. At each node of the tree, the algorithm selects the feature that best splits the data, optimizing some criterion (like Gini impurity or entropy) to maximize information gain or minimize impurity. The resulting tree can be used to make predictions by following the path from the root to a leaf node, where each leaf corresponds to a class label (in classification) or a numerical value (in regression).

- **Multiclass Logistic Regression:** It is a statistical method used for classification which extends the principles of binary logistic regression to handle multiple classes by employing a set of linear functions, one for each class, combined with the soft-max function to convert raw predictions into probabilities. The class with the highest probability is then assigned as the predicted class for a given input.
  The logistic regression model is based on the logistic function, also known as the sigmoid function:

$$\sigma(x) = \frac{1}{1 + e^{-x}} \tag{1}$$

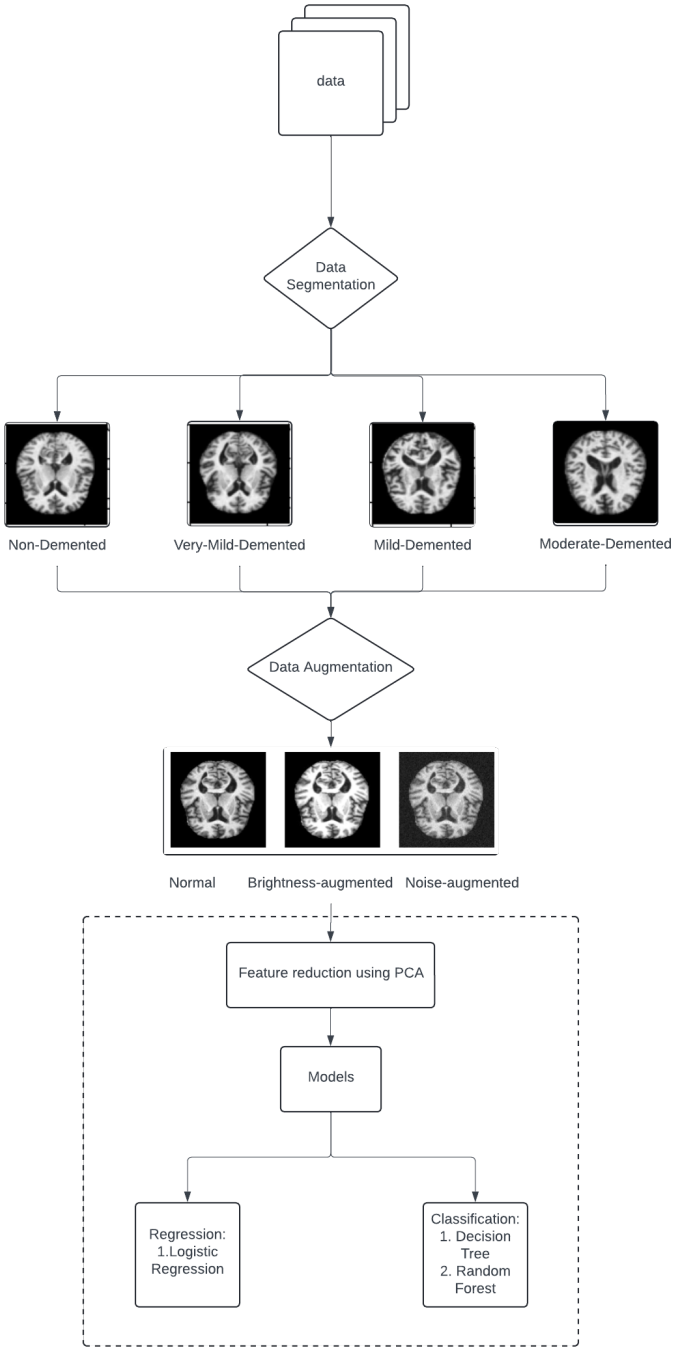

Fig. 6. Project Schema

The output of the logistic function is a value between 0 and 1, which can be interpreted as the probability of the instance belonging to the positive class (e.g., class 1).

- **Random Forest Classification:** Random Forest is an ensemble learning method that utilizes multiple decision trees to make predictions. It operates by constructing a multitude of decision trees during training, where each tree is trained on a random subset of the training data and a random subset of the features. During prediction, each tree in the forest independently predicts the outcome, and the final prediction is determined by a majority vote (for classification) or an average (for regression) of the predictions made by individual trees. Random Forest helps to reduce overfitting compared to a single decision tree by combining the predictions from multiple trees, thereby improving the model's generalization ability.

- **Convolutional Neural Network:** A convolutional neural network (CNN) is a type of deep learning algorithm inspired by the visual cortex of the brain. It excels at processing and recognizing patterns in images by applying convolutional filters that extract progressively more abstract features from the raw pixel data, enabling tasks like image classification and object detection.

V. Methodology

- **Dataset**: The dataset consists of 128*128 .jpg images. Each image is resized to 16348*1 to make it computable and resourceful.
- **Data Augmentation**: This methodology effectively triples the size of the training set by creating two additional versions (brightened and noisy) of the original images for the first 3000 samples. By introducing these variations, the model is exposed to a more diverse set of examples, potentially improving its ability to generalize and handle different image conditions during inference. Brightness of the images are varied by randomly changing the value of pixels. Gaussian noise is also added to pixels to blur the images.
- **Applying PCA**:
  - The `calculate_pca` function is called twice, once for each dataset, with `p=500` to retain 500 principal components.
  - The transformed data with reduced dimensionality is stored in `X_train_1`, `X_test_1`, `X_val_1`, and `X_train_2`, `X_test_2`, `X_val_2`.
  - The shapes of the transformed data are printed using the `print` function for verification.
- **Convolutional Neural Network**: This methodology follows a typical workflow for training and evaluating a CNN model on image data. It includes defining the model architecture, compiling the model, preprocessing the data, training the model with validation monitoring, evaluating the model's performance on test data, making predictions, generating classification reports, and visualizing the training history.
  - It adds a 2D convolutional layer with 32 filters, a 3x3 kernel size, and ReLU activation function. The input shape is (128, 128, 1), which suggests that the input images are grayscale with a size of 128x128 pixels.
  - It adds a 2D max-pooling layer with a 2x2 pool size for downsampling.
  - It adds another 2D convolutional layer with 64 filters and a 3x3 kernel size, followed by a max-pooling layer.
  - It adds a third 2D convolutional layer with 64 filters and a 3x3 kernel size.
  - It flattens the output from the convolutional layers.
  - It adds a dense (fully connected) layer with 64 units and ReLU activation.
  - Finally, it adds an output dense layer with 4 units and a softmax activation function, indicating a multi-class classification problem with 4 classes.
- **Decision Tree**: The algorithm involves $cross - validation$ which results in 5 decision trees. Then, the best decision-tree is selected on basis of $maximum - depth$. This tree is used for valuation, testing and application. The given code snippet demonstrates the use of a Decision Tree model for a multi-class classification problem. Cross-validation is employed to select the optimal maximum depth parameter for the Decision Tree.

A. Parameter Grid and Model Initialization

- A parameter grid `param_grid` is defined, which specifies a range of values for the `max_depth` parameter (25, 50, and 100) to be evaluated during cross-validation.
- An instance of the `DecisionTreeClassifier` class from the `sklearn.tree` module is created and assigned to the `decision_tree` variable.

B. Grid Search with Cross-Validation

- An instance of the `GridSearchCV` class from `sklearn.model_selection` is created with the following parameters:
  - `decision_tree`: The Decision Tree model to be evaluated.
  - `param_grid`: The parameter grid containing the values of `max_depth` to be evaluated.
  - `cv=5`: Specifies 5-fold cross-validation.
  - `scoring='accuracy'`: The evaluation metric is set to accuracy.
- The `fit` method of the `GridSearchCV` object is called with the training data (`X_train`, `Y_train`) to perform the grid search and cross-validation.

C. Best Model Selection

- The best value of `max_depth` is obtained from the `best_params_` attribute of the `GridSearchCV` object and assigned to the `best_depth` variable.
- The best estimator (Decision Tree model with the optimal `max_depth`) is retrieved from the `best_estimator_` attribute of the `GridSearchCV` object and assigned to the `best_decision_tree` variable.

D. Model Evaluation

- The validation accuracy of the best Decision Tree model is calculated using the `score` method on the validation data (`X_val`, `Y_val`) and printed.
- The test accuracy of the best Decision Tree model is calculated using the `score` method on the test data (`X_test`, `Y_test`) and printed.
- Predictions are made on the test data using the `predict` method of the best Decision Tree model: `y_pred = best_decision_tree.predict(X_test)`.

- **Logistic Regression**: Logistic regression is a statistical method used for binary classification problems, where the goal is to predict the probability of an instance belonging to one of two classes (e.g., 0 or 1, true or false, yes or no) based on one or more independent variables or features.
  - A Logistic Regression model is created using `LogisticRegression` from `sklearn.linear_model` with the `multi_class='multinomial'` option for multi-class classification.
  - The model is fitted to the training data (`X_train`, `Y_train`) using the `fit` method.
  - The validation accuracy is calculated using the `score` method on the validation data (`X_val`, `Y_val`) and printed.
  - The test accuracy is calculated using the `score` method on the test data (`X_test`, `Y_test`) and printed.
  - Predictions are made on the test data using the `predict` method: `y_pred = log_reg.predict(X_test)`.
- **Random Forest**: The given code snippet demonstrates the use of a Random Forest model for a multi-class classification problem.

E. Data Combination

- The training and validation data are combined using numpy's `np.concatenate` function:
  - `X_combined = np.concatenate((X_train, X_val))` combines the feature data.
  - `Y_combined = np.concatenate((Y_train, Y_val))` combines the target labels.
- This step is often performed to create a larger dataset for training the Random Forest model.

F. Model Initialization and Training

- An instance of the `RandomForestClassifier` class from `sklearn.ensemble` is created with the `n_estimators` parameter set to 75, which specifies the number of decision trees in the Random Forest.
- The `fit` method of the `random_forest` object is called with the combined training data (`X_combined`, `Y_combined`) to train the Random Forest model.

G. Model Evaluation

- The test accuracy of the Random Forest model is calculated using the `score` method on the test data (`X_test`, `Y_test`) and printed.
- Predictions are made on the test data using the `predict` method of the Random Forest model: `y_pred = random_forest.predict(X_test)`.

H. Confusion Matrix

- The code mentions that the classes "very-mild", "mild", and "moderate" are considered positive, while "non-demented" is considered negative. However, the implementation of this step is not provided in the given code snippet.
- The `confusion_matrix` function from `sklearn.metrics` is used to calculate the confusion matrix, which provides a summary of the correct and incorrect classifications made by the model.
- The confusion matrix is calculated by passing the true labels (`Y_test`) and predicted labels (`Y_pred`) to the `confusion_matrix` function: `conf_matrix = confusion_matrix(Y_test, Y_pred)`.

VI. Results

- **Decision Tree:** It performed with an accuracy of **66.15%** on *validation dataset* and **66.92%** on *test dataset*.

Data	Accuracies (%)
Total	66.923
Non-Demented	68.817
Very-Mild-Demented	73.755
Mild-Demented	53.773
Moderate-Demented	28.57

- **Logistic Regression:** The algorithm performed with an accuracy of **85%** on *validation dataset* and **82.307%** on *test dataset*. And **Confusion Matrix** is used as metric to visualize the true labels against the predicted labels.
- **Random Forest Classifier:** The algorithm performed with an accuracy of **96.7307%** on *test dataset*.

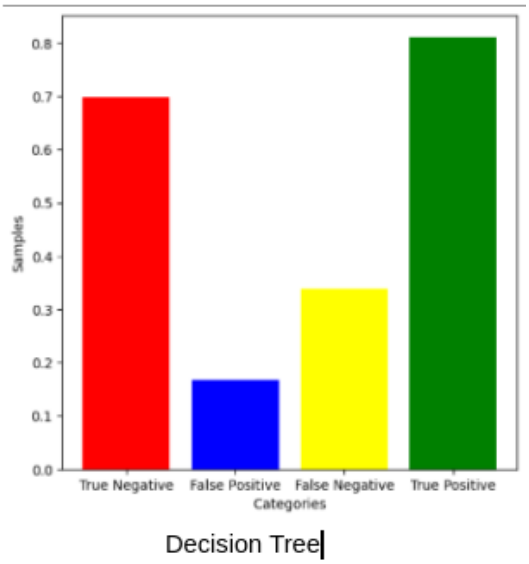

Fig. 7. Decision Tree

Data	**Accuracies (%)**
Total	82.307
Non-Demented	84.44
Very-Mild-Demented	78.733
Mild-Demented	87.7735
Moderate-Demented	100.00

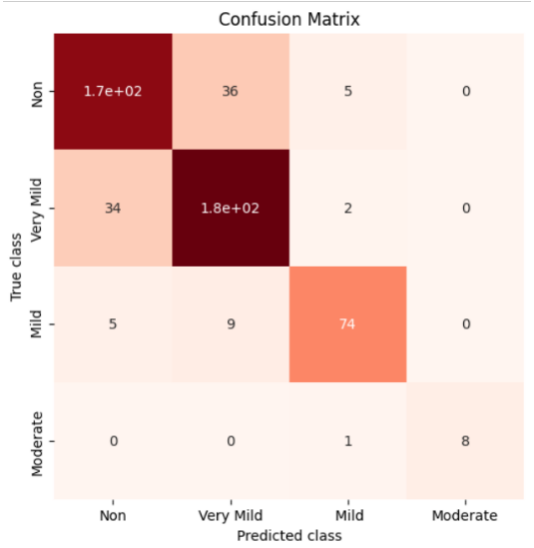

Fig. 8. Logistic Regression: Confusion Matrix

Data	**Accuracies (%)**
Total	96.730
Non-Demented	97.849
Very-Mild-Demented	99.547
Mild-Demented	90.566
Moderate-Demented	71.428

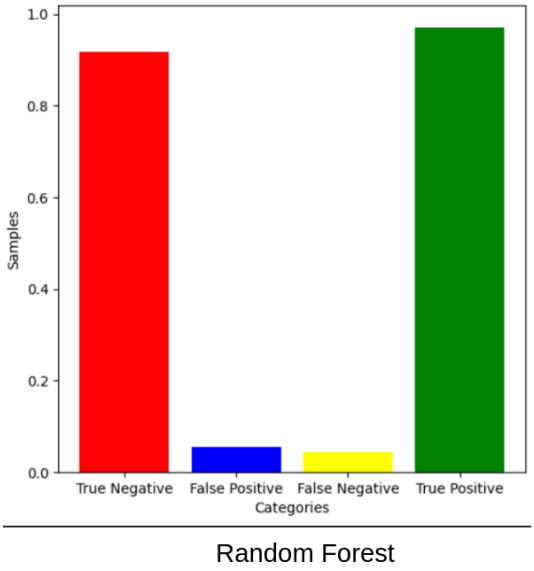

Fig. 9. Random Forest

- **Convolutional Neural Network**: Model is trained on $dataset1$ and the model is used on testing data from itself and $dataset2$. Epoch-wise comparison is also provided.

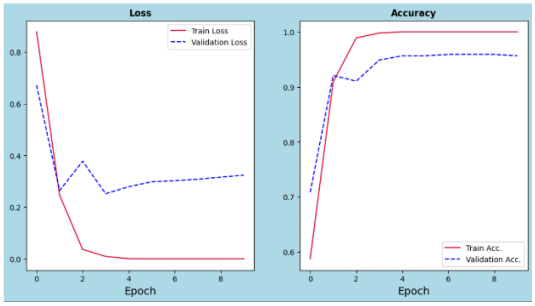

Fig. 10. Comparison between predictions of Dataset 1 and 2

	precision	recall	f1-score	support
0	0.95	0.97	0.96	155
1	0.95	0.95	0.95	174
2	0.95	0.92	0.93	60
3	1.00	1.00	1.00	2
accuracy			0.95	391
macro avg	0.96	0.96	0.96	391
weighted avg	0.95	0.95	0.95	391

Fig. 11. Dataset1 prediction using CNN

	precision	recall	f1-score	support
0	1.00	0.97	0.98	146
1	0.90	1.00	0.95	57
2	1.00	0.98	0.99	44
3	1.00	1.00	1.00	2
accuracy			0.98	249
macro avg	0.98	0.99	0.98	249
weighted avg	0.98	0.98	0.98	249

Fig. 12. Dataset2 prediction using CNN

VII. Resources Referred

- **Dataset**: https://www.kaggle.com/datasets/sachinkumar413/alzheimer-mri-dataset/data
- **Logistic Regression**:
  - https://www.youtube.com/watch?=yIYKR4sgzI8t=135spp=ygUTbG9naXN0aWMgcmVncmVzc2lvbg
  - https://www.ibm.com/topics/logistic-regression
- **Decision Tree Classifier**:
  - https://www.youtube.com/watch?v=LDRbO9a6XPUpp=ygUYZGVjaXNpb24gdHJlZSBjbGFzc2lmaWVy
- https://www.youtube.com/watch?v=$_L39rN6gz7Ypp = ygUYZGVjaXNpb24gdHJlZSBjbGFzc2lmaWVy$
- **Random Forest Classification**:
  - https://scikit-learn.org/stable/modules/generated/sklearn.ensemble.RandomForestClassifier.html
- https://www.youtube.com/watch?v=J4Wdy0Wc$_xQpp = ygUYcmFuZG9tIGZvcmVzdCBjbGFzc2lmaWVy$
- **Convolutional Neural Network**:
  - https://www.geeksforgeeks.org/convolutional-neural-network-cnn-in-machine-learning/
  - https://www.youtube.com/watch?v=QzY57FaENXgpp=ygUGY25uIG1s