FACULTY OF SCIENCE, TECHNOLOGY AND MEDICINE

Department of Engineering

Master of Science in Sustainable Product Creation

Artificial Intelligence Report

by

Ariel Patricio HERRERA AGUIAR

Matr. Nr. 0230193007

Alzheimer Detection AI Project

Reviewer: Francesco GIOVANNINI

Contents

1. Introduction

The goal of this project is to leverage machine learning techniques to predict the diagnosis of Alzheimer's disease based on patient data [1]. The project involves two main tasks:

- Building and optimizing an Artificial Neural Network (ANN).
- Building and optimizing a Decision Tree (DT).

Through these tasks, we aim to evaluate and compare the performance of these models, highlighting their strengths, weaknesses, and potential applications in real-world scenarios. The final objective is to develop a robust classification system to assist in the early detection of Alzheimer's disease, potentially improving patient outcomes.

2. Objectives

The primary objectives of the project are:

1. To train and evaluate an Artificial Neural Network (ANN) and a Decision Tree (DT) on the Alzheimer's disease dataset.
2. To optimize the parameters of each model, including hyperparameters and architecture.
3. To evaluate the performance of the models using meaningful metrics and visualizations.
4. To compare the two approaches and draw insights regarding their effectiveness for this classification task.

3. Artificial Neural Network (ANN)

This section analyzes the methodology used for the Artificial Neural Network model of the project.

3.1. Task Definition

The objective of the ANN is to classify patients into two categories:

- **Positive**: Alzheimer's disease is detected.
- **Negative**: Alzheimer's disease is not detected.

The classification is based on a dataset that includes demographic information, lifestyle factors, and medical history.

3.2. Data Preprocessing

Before training the ANN, the dataset received the following preprocessing steps:

1. **Column Removal**: Irrelevant columns such as PatientID and DoctorInCharge were dropped.
2. **Scaling**: Numerical features were normalized using StandardScaler to ensure uniformity across inputs.
3. **Data Splitting**: The dataset was divided into:
   - Training set (70%): For model training.
   - Validation set (20%): For hyperparameter tuning and monitoring overfitting.
   - Test set (10%): For final evaluation.

3.3. First Model Design

The ANN was designed with the following architecture:

- **Input Layer**: Matches the number of features in the dataset.
- **Hidden Layers**:
  1. First layer with 64 neurons and ReLU activation, followed by a dropout layer (rate = 0.3).
  2. Second layer with 32 neurons and ReLU activation, followed by another dropout layer (rate = 0.3).
- **Output Layer**: A single neuron with sigmoid activation for binary classification.

The **binary crossentropy loss function** was chosen, as it is suitable for binary classification. The model was optimized using the Adam optimizer with default parameters.

ReLU (Rectified Linear Unit) was selected as the activation function for the hidden layers because of its advantages over traditional functions like sigmoid or tanh. ReLU mitigates the vanishing gradient problem, enabling faster training and better convergence. Additionally, its computational simplicity makes it well-suited for large-scale data processing. The output layer uses a sigmoid activation function to produce a probability score for binary classification, aligning with the task requirements.

3.4. Training

The model was trained for 100 epochs with a batch size of 32 and default learning rate. The training process included monitoring the accuracy and loss on both the training and validation sets.

3.5. Evaluations and Results

The ANN's performance was evaluated using the test set, with the following results:

1. **Accuracy as shown in Fig. 1**:
   - Training accuracy: 92.87%.
   - Validation accuracy: 80.23%.
   - Test accuracy: 82.3%
   - Validation accuracy fluctuated, indicating the need for further regularization or model tuning.
2. **Loss and Accuracy Curves as shown in Fig. 2**:
   - Training loss: 16.98%.
   - Validation loss: 62.72%.
   - Training loss steadily decreased, while validation loss began increasing after some epochs, confirming overfitting.
3. **Confusion Matrix as shown in Fig. 3**:
   - **True Negatives (132)**: Correctly predicted as negative.
   - **True Positives (43)**: Correctly predicted as positive.
   - **False Negatives (24)**: Misclassified as negative but are actually positive.
   - **False Positives (16)**: Misclassified as positive but are actually negative.
   - The confusion matrix revealed a reasonable ability to predict negative cases but a higher error rate in detecting positive cases, which could affect sensitivity.

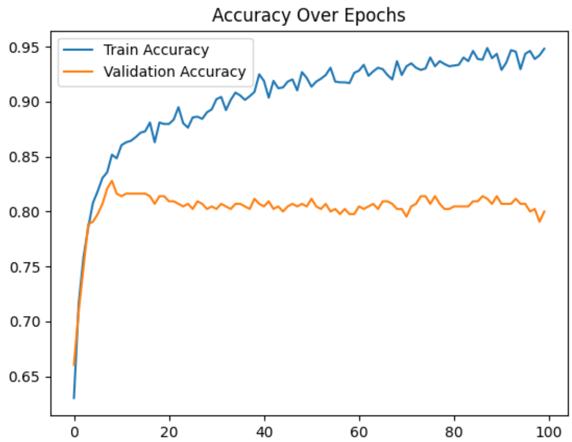


FIGURE 1. Accuracy Over Epochs.

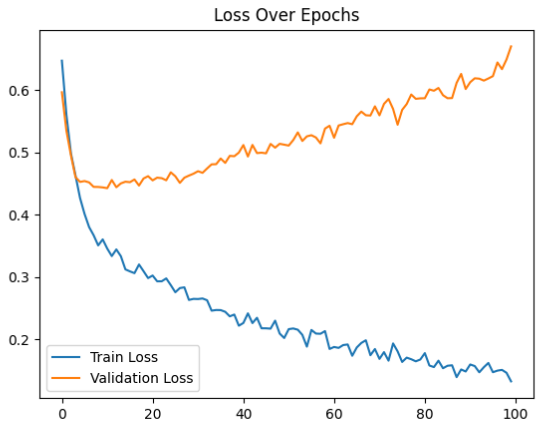


FIGURE 2. Loss Over Epochs.

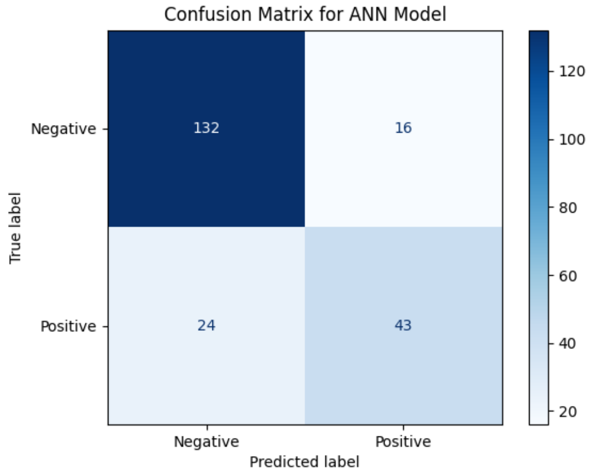


FIGURE 3. Confusion Matrix.

3.6. Parameter Optimisation

The default model exhibited significant overfitting, as evident in the rapid divergence between training and validation losses. Validation performance degraded after approximately 50 epochs, prompting early stopping.

3.6.1. Learning Rate

To identify the optimal learning rate, the values shown in Table 1 were evaluated. The learning rate of 0.001 was selected. The plots of the results over epochs are shown in Fig. 4 and Fig. 5.

TABLE 1. Learning Rate Results.

Learning Rate	Validation Loss	Validation Accuracy
0.1	0.672	0.76
0.01	0.538	0.80
0.001	0.495	0.83
0.0001	0.512	0.81

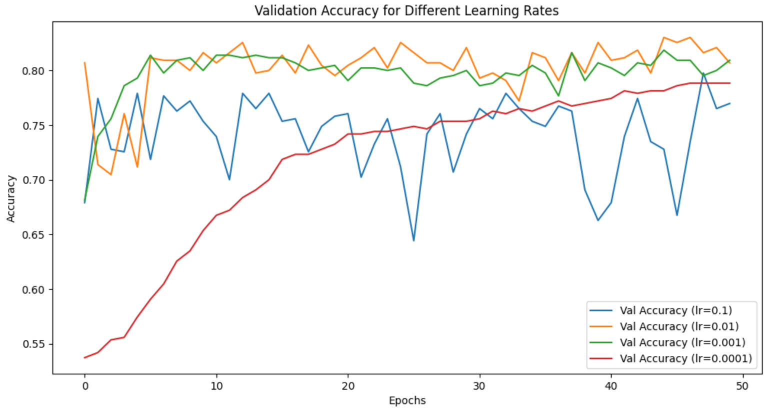


FIGURE 4. Validation Accuracy for Different Learning Rates.

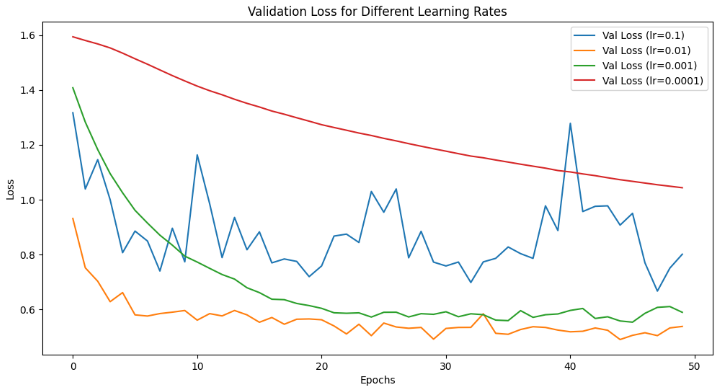


FIGURE 5. Validation Loss for Different Learning Rates

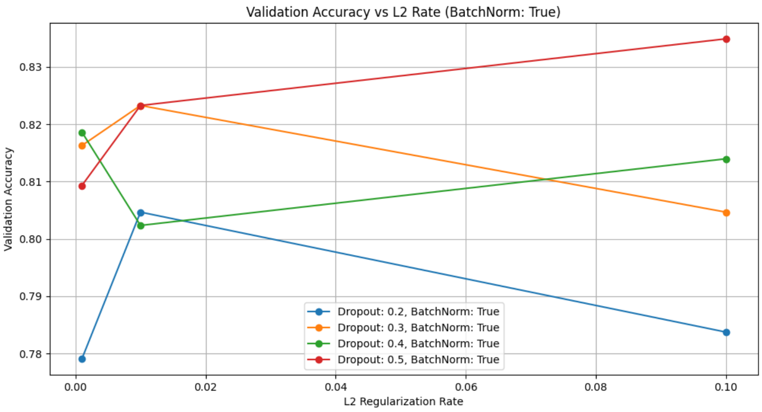
3.6.2. Regularization (L2 and Dropout)

Different L2 regularization rates (l2_rate) and dropout rates (dropout_rate) were explored. The results are shown in Table 2. The model with dropout_rate=0.4, l2_rate=0.01 and without BatchNorm was chosen for its balance between loss and accuracy. The plots of the results are shown in Fig. 6 and Fig. 7.

TABLE 2. Regularization Results.

Dropout	L2 Rate	Validation Loss	Validation Accuracy
BatchNorm = True			
0.2	0.001	0.660	0.78
0.2	0.01	0.685	0.80
0.2	0.1	0.711	0.78

0.3	0.001	0.551	0.81
0.3	0.01	0.581	0.82
0.3	0.1	0.675	0.80
0.4	0.001	0.504	0.82
0.4	0.01	0.558	0.80
0.4	0.1	0.586	0.81
0.5	0.001	0.484	0.81
0.5	0.01	**0.494**	**0.83**
0.5	0.1	0.530	0.83
BatchNorm = False			
0.2	0.001	0.602	0.80
0.2	0.01	0.511	0.83
0.2	0.1	0.516	0.82
0.3	0.001	0.540	0.81
0.3	0.01	0.500	0.82
0.3	0.1	0.527	0.80
0.4	0.001	0.522	0.82
0.4	0.01	**0.479**	**0.83**
0.4	0.1	0.535	0.81
0.5	0.001	0.493	0.83
0.5	0.01	0.487	0.83
0.5	0.1	0.548	0.82



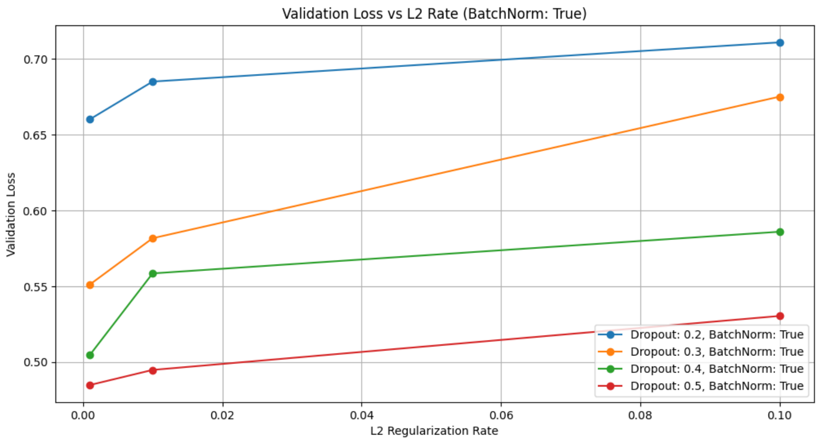


FIGURE 6. Validation Loss and Accuracy vs L2 rate with BatchNorm.

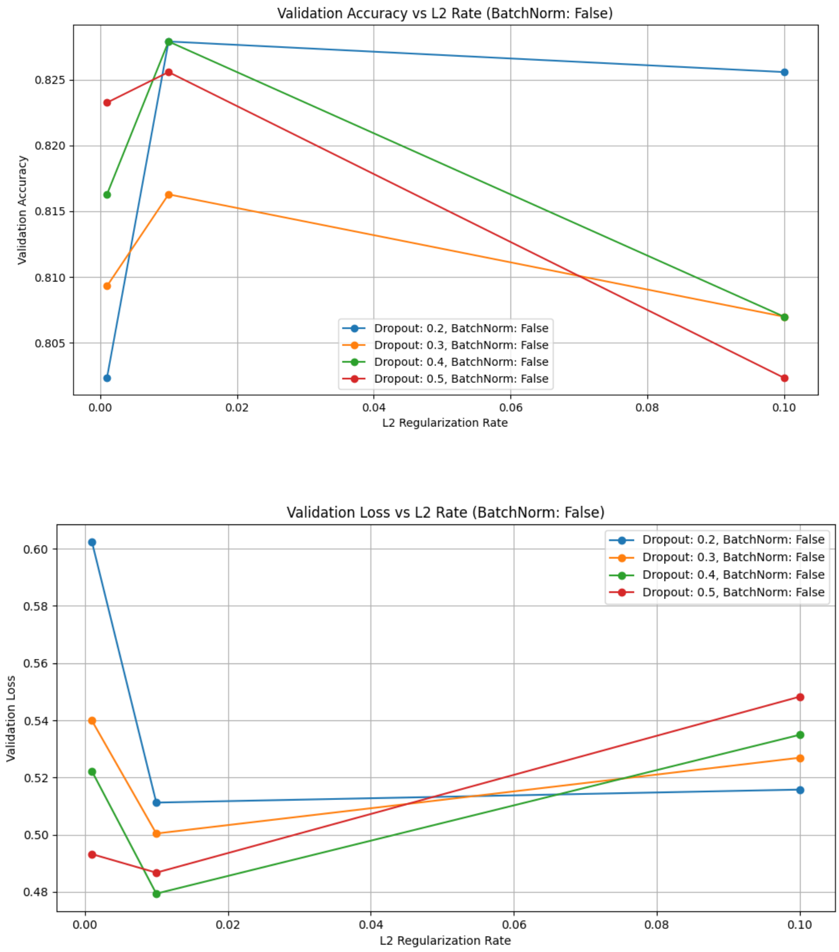


FIGURE 7. Validation Loss and Accuracy vs L2 rate without BatchNorm.

3.6.3. Network Topology

The following topologies were tested, and their results are summarized in Table 3:

- **1_layer_simple:** One hidden layer with 32 neurons.
- **2_layers_basic:** Two hidden layers of 32 neurons each.
- **4_layers_deep:** Four hidden layers of 32 neurons each.
- **2_layers_wide:** Two hidden layers of 64 neurons each.

1_layer_simple was selected for its lower complexity and competitive results. The plots of the results are shown in Fig. 8 and Fig. 9.

TABLE 3. Network Topology Results.

Topology	Validation Loss	Validation Accuracy
1-layer simple	**0.446**	0.83
2-layers basic	0.466	0.83
4-layers deep	0.518	0.82
2-layers wide	0.515	0.80

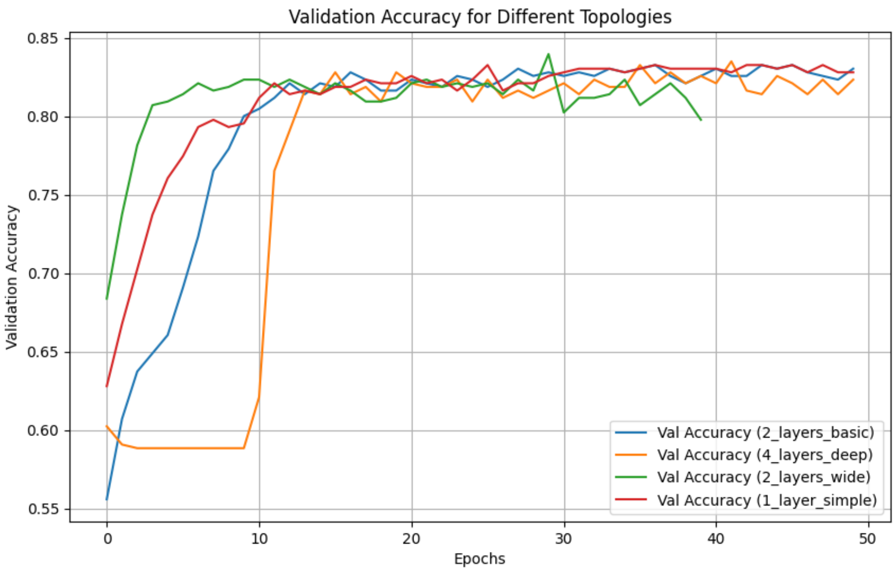


FIGURE 8. Validation Accuracy for Different Topologies.

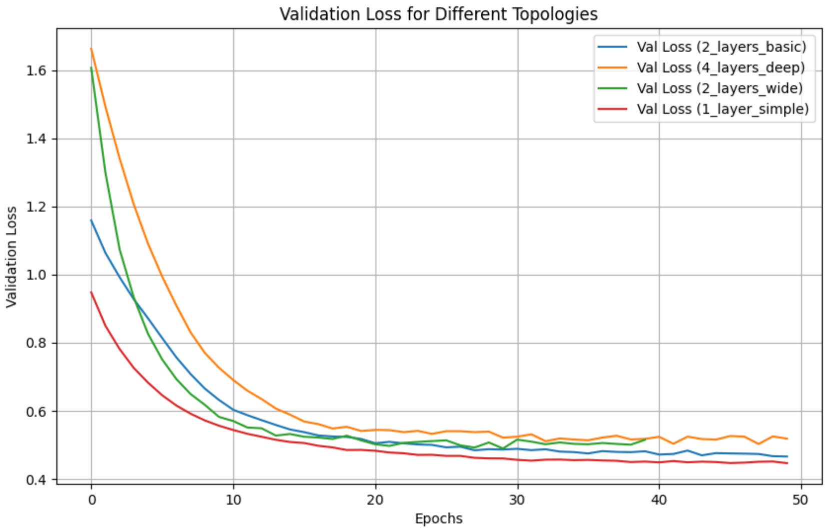


FIGURE 9. Validation Loss for Different Topologies.

3.6.4. Comparison of Activation Functions

The model was trained with tanh and ReLU activation functions. The final results are presented in Table 4 and Fig. 10 and Fig. 11. Using ReLU provided better results on both metrics.

TABLE 4. Activation Function Comparison.

Activation	**Validation Loss**	**Validation Accuracy**
ReLU	**0.452**	**0.83**
Tanh	0.467	0.82

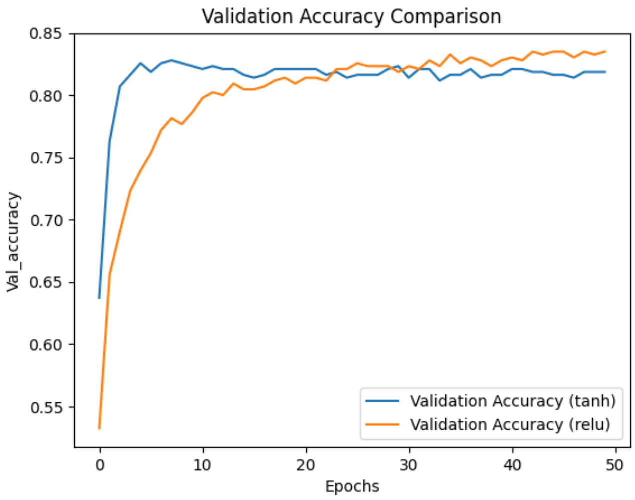


FIGURE 10. Validation Accuracy Comparison between tanh and ReLU activation functions.

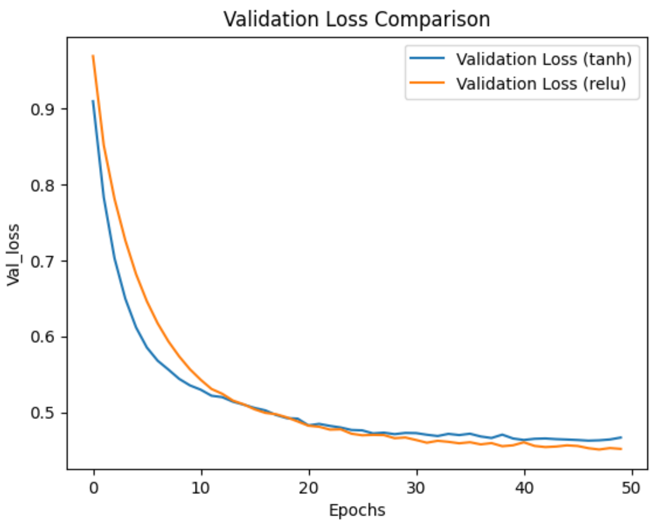


FIGURE 11. Validation Loss Comparison between tanh and ReLU activation functions.

3.7. Final Model Evaluation

The final model was trained with the selected configuration for 100 epochs. Validation loss and accuracy remained stable, indicating minimal overfitting as shown in Table 5. The results obtained for the new model are observed in Fig. 12, Fig. 13, Fig. 14.

TABLE 5. Default Model and Final Model Comparison.

Metric	Default Model	Final Model
Validation Loss	0.672	**0.452**
Validation Accuracy	0.76	**0.83**
Precision	0.70	**0.77**
Recall	0.75	**0.73**
F1-Score	0.72	**0.75**

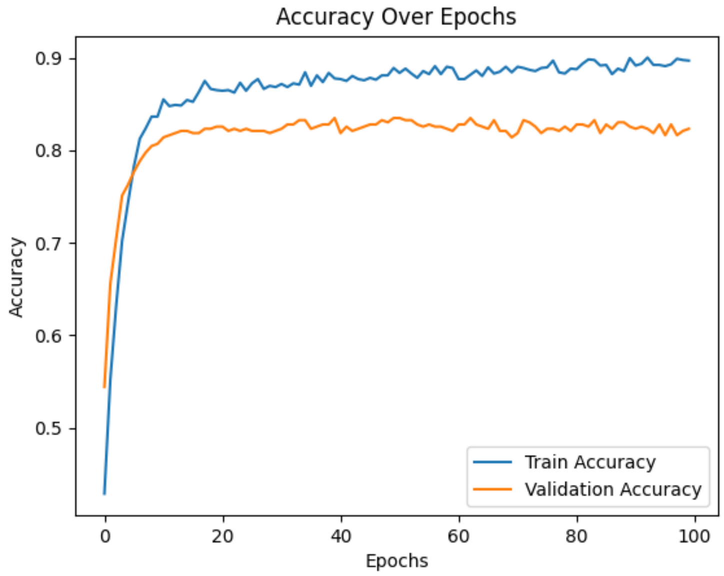


FIGURE 12. Final Model Accuracy Over Epochs.

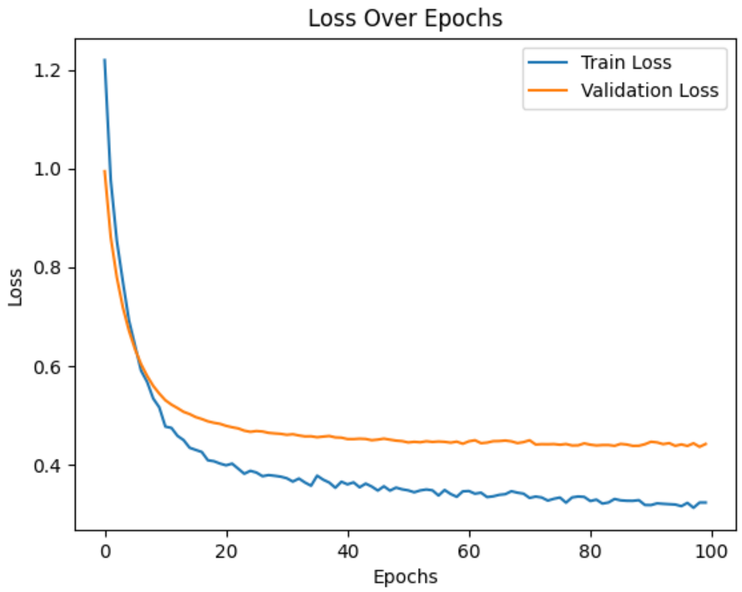


FIGURE 13. Final Model Loss Over Epochs

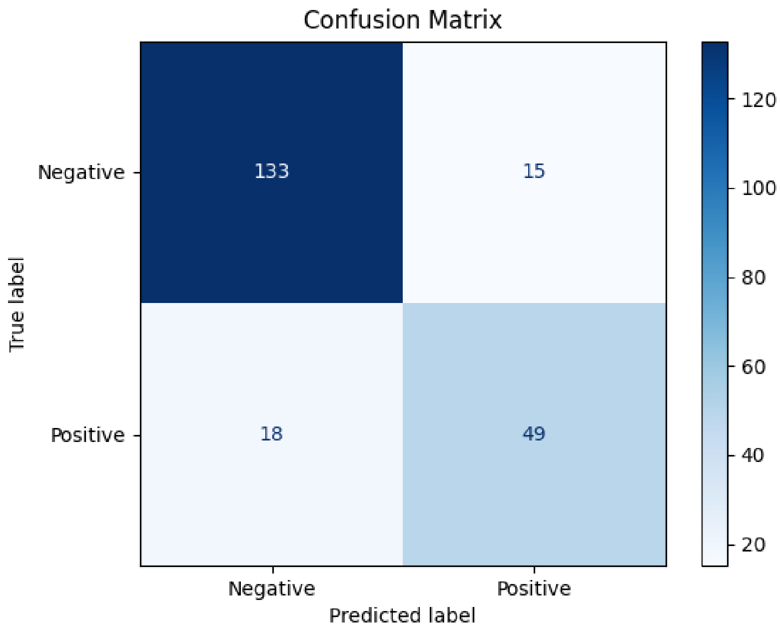


FIGURE 14. Final Model Confusion Matrix

4. Decision Tree

This section analyzes the methodology used for the Decision Tree model of the project.

4.1. Task Definition

The objective of the Decision Tree (DT) is to classify patients into two categories:

- **Positive**: Alzheimer's disease is detected.
- **Negative**: Alzheimer's disease is not detected.

The classification task is based on a dataset containing demographic information, lifestyle factors, and medical history.

4.2. Data Preprocessing

Before training the Decision Tree model, the dataset underwent the following preprocessing steps:

1. **Column Removal**: Irrelevant columns, such as PatientID and DoctorInCharge, were dropped.

2. **Scaling**: Numerical features were normalized using StandardScaler to ensure uniformity across inputs.
3. **Data Splitting**: The dataset was divided into:
   - Training set (70%): For model training.
   - Validation set (20%): For hyperparameter tuning.
   - Test set (10%): For final evaluation.

4.3. Default Model

The Decision Tree model was initially trained with the following default hyperparameters:

- Criterion: Gini
- Maximum Depth: None
- Minimum Samples Split: 2
- Minimum Samples Leaf: 1

The default hyperparameters for the Decision Tree model were selected to establish a baseline performance while allowing the model to explore the full complexity of the dataset. Using gini as the splitting criterion emphasizes minimizing impurity at each split, which is computationally efficient and suitable for balanced datasets like this one. The absence of a maximum depth (max_depth=None) and minimal constraints on splits (min_samples_split=2 and min_samples_leaf=1) allowed the model to grow fully, providing insights into how complex the tree could become without pruning. These initial settings enabled the identification of potential overfitting, which was subsequently addressed through optimization and pruning.

The performance of the default model is summarized in Table 6. The associated Confusion Matrix and Cross-Validation Accuracy are illustrated in Fig. 15 and Fig. 16, respectively.

TABLE 6. Performance metrics for the default Decision Tree model.

Metric	Validation Set	Test Set
Accuracy	91.39%	93.49%
Precision	0.90	0.90
Recall	0.88	0.90
F1-Score	0.89	0.90

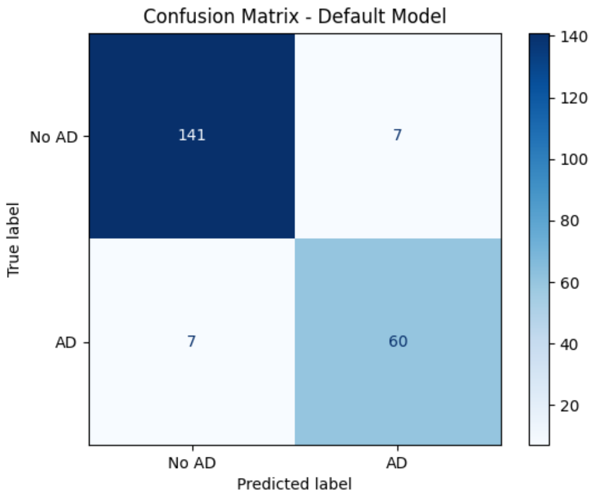


FIGURE 15. Confusion Matrix for the Default Decision Tree model.

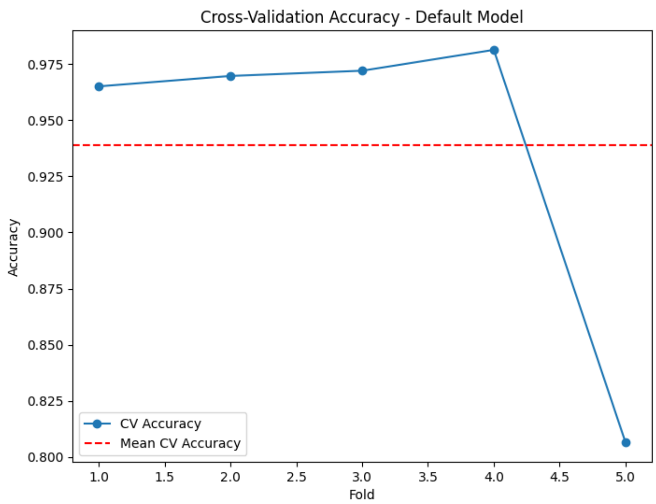


FIGURE 16. Cross-Validation Accuracy for the Default Decision Tree model.

The feature importance plot in Fig. 17 highlights the most influential variables, such as FunctionalAssessment, MMSE, and ADL.

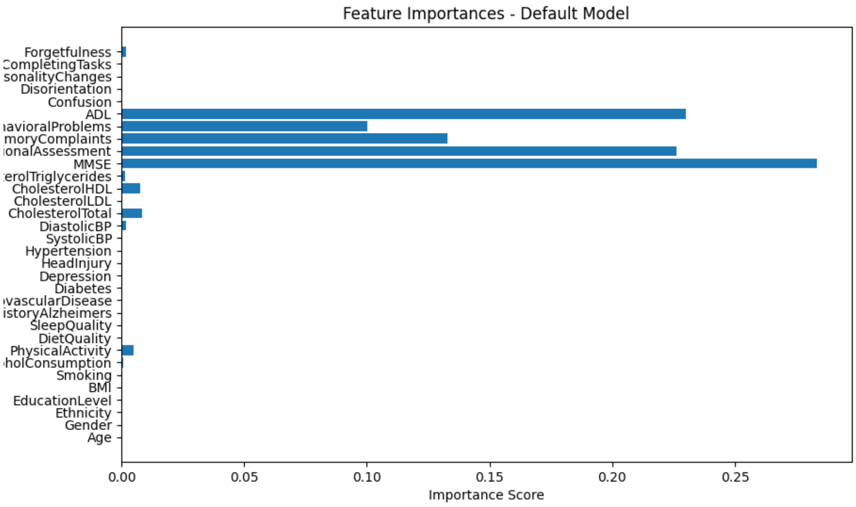


FIGURE 17. Feature Importances for the Default Decision Tree model.

4.4. Optimized Decision Tree Model

4.4.1. Hyperparameter Tuning

The Decision Tree model was optimized using GridSearchCV with the following parameters:

The Decision Tree model was optimized using GridSearchCV with the following parameters:

- Criterion: gini
- Max Depth: [3, 5, 10, None]
- Min Samples Split: [2, 5, 10]
- Min Samples Leaf: [1, 5, 10]

The best parameters were:

- Criterion: gini
- Max Depth: 5
- Min Samples Split: 2
- Min Samples Leaf: 5

The optimized model demonstrated improved accuracy and a balanced performance across precision and recall, as summarized in Table 7. The Confusion Matrix, Cross-Validation Accuracy, and Feature Importances are shown in Fig. 18, Fig. 19, and Fig. 20, respectively.

TABLE 7. Performance metrics for the optimized Decision Tree model.

Metric	Validation Set	Test Set
Accuracy	92.09%	94.42%
Precision	0.92	0.92
Recall	0.89	0.90
F1-Score	0.90	0.91

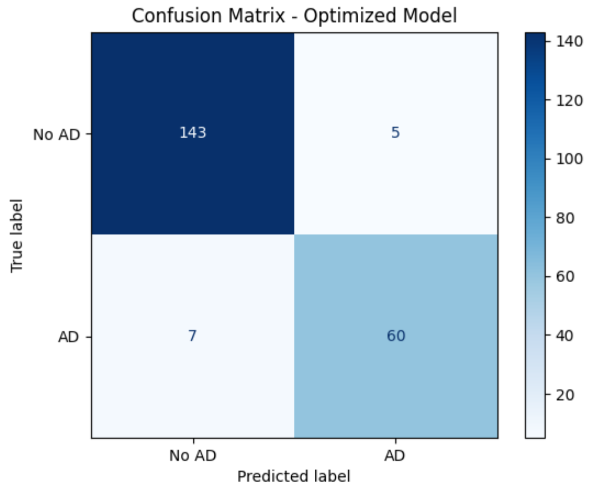


FIGURE 18. Confusion Matrix for the Optimized Decision Tree model.

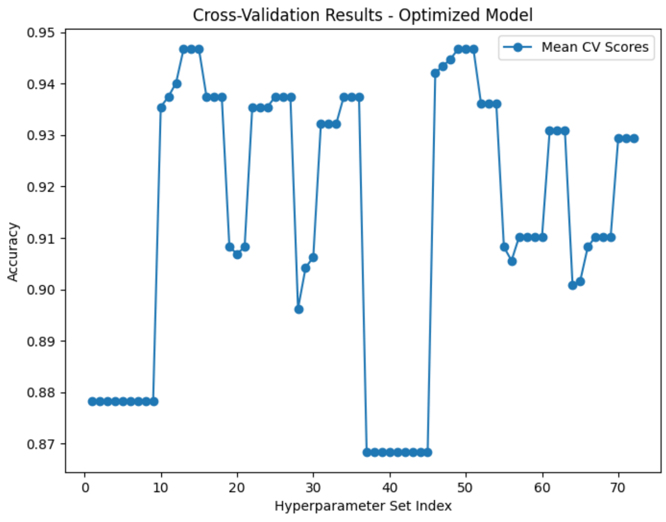


FIGURE 19. Cross-Validation Accuracy for the Optimized Decision Tree model.

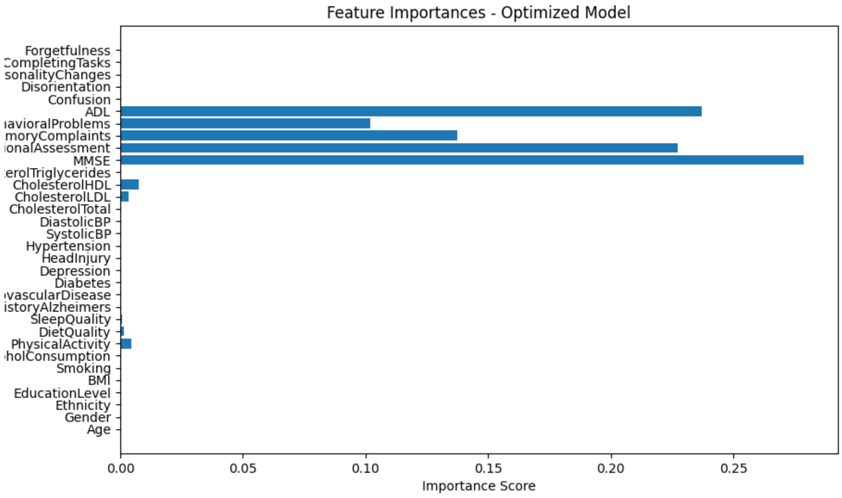


FIGURE 20. Feature Importances for the Optimized Decision Tree model.

4.4.2. Pruned Decision Tree Model

Cost-Complexity Pruning was applied to the optimized model to prevent overfitting. The pruning process evaluated various ccp_alpha values and selected the optimal value based on validation accuracy.

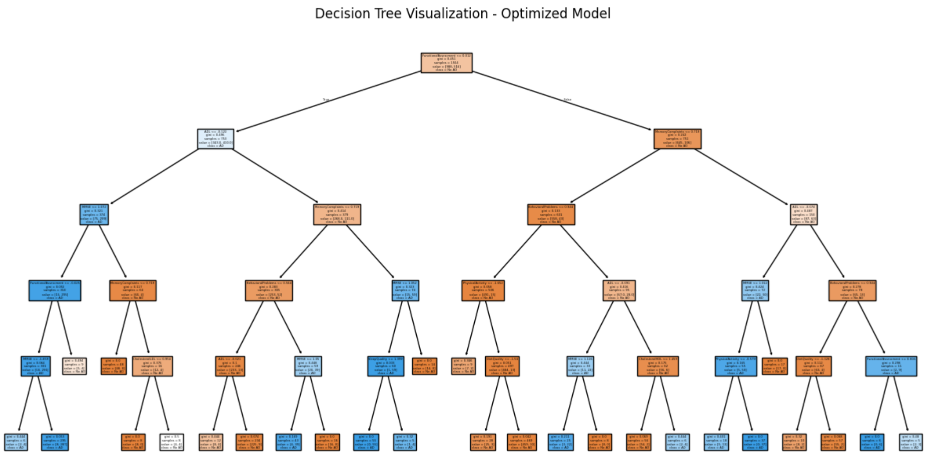
- Best ccp_alpha: 0.0029

The pruned model performed similarly to the optimized model but achieved better interpretability with a reduced number of nodes. Both models yielded equivalent test accuracies. Pruning improved interpretability while maintaining performance. The pruned model is preferable when simplicity is a priority.

The performance of the pruned model is shown in Table 8. The Decision Tree visualization, Validation Accuracy vs. Effective Alpha plot, and Feature Importances are shown in Fig. 21, Fig. 22, and Fig. 23, respectively.

TABLE 8. Performance metrics for the Pruned Decision Tree model.

Metric	Validation Set	Test Set
Accuracy	92.09%	94.42%
Precision	0.92	0.92
Recall	0.89	0.90
F1-Score	0.90	0.91



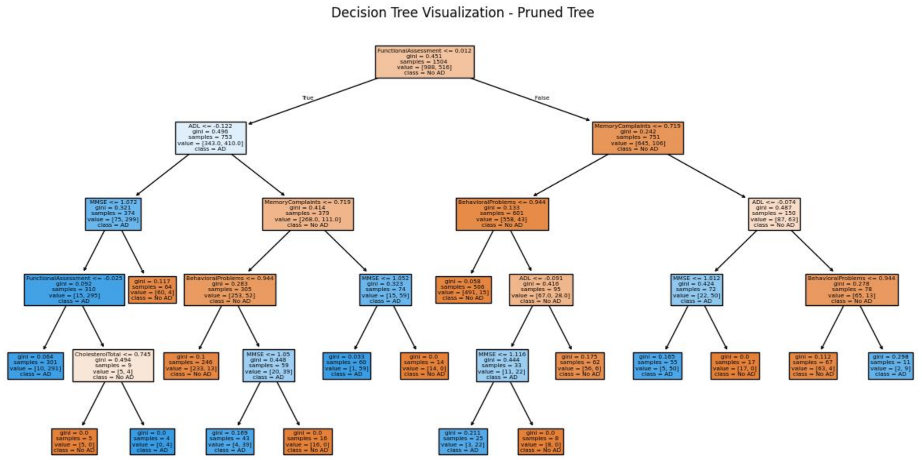


FIGURE 21. Decision Tree and Pruned Decision Tree visualization.

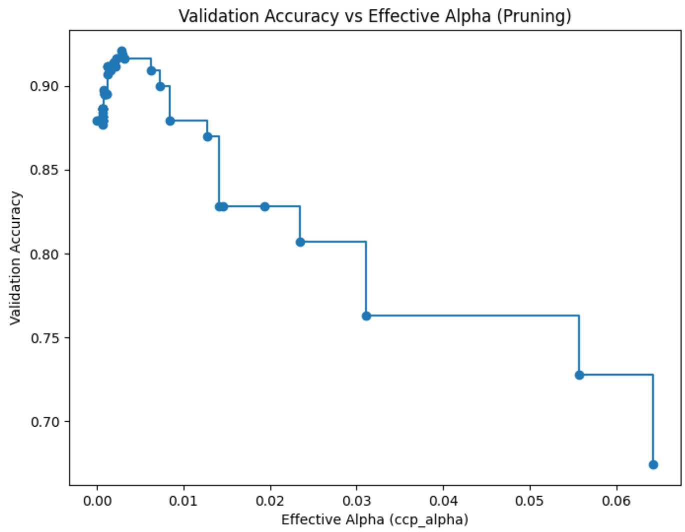


FIGURE 22. Validation Accuracy vs. Effective Alpha for the Pruned Tree.

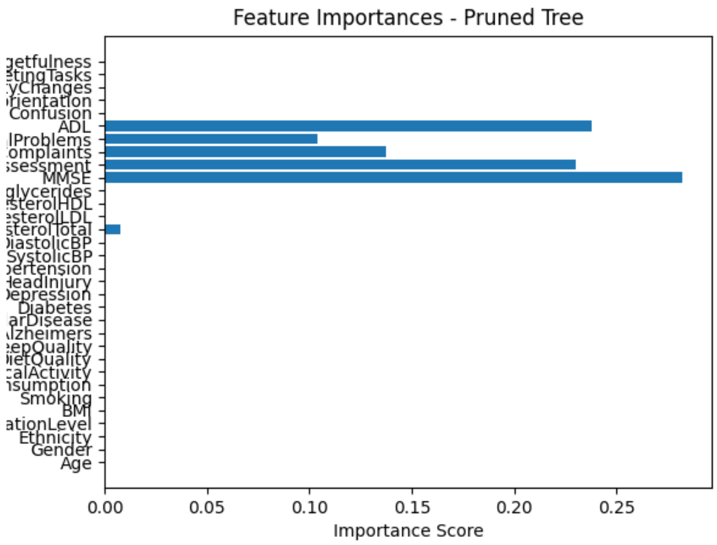


FIGURE 23. Feature Importances for the Pruned Tree.

4.4.3. Random Forest Model

A Random Forest classifier was implemented as an alternative to the Decision Tree. The model used the following hyperparameters:

- Number of Estimators: 100
- Max Features: sqrt
- Criterion: gini

While the Random Forest model demonstrated strong precision, its recall was significantly lower than that of the Decision Tree models. Therefore, the Decision Tree (Optimized or Pruned) remains the preferred choice for this dataset.

The performance of the Random Forest model is summarized in Table 9. The Confusion Matrix and Feature Importances are shown in Fig. 24 and Fig. 25, respectively.

TABLE 9. Performance metrics for the Random Forest model.

Metric	Validation Set	Test Set
Accuracy	83.49%	87.44%
Precision	0.93	0.93
Recall	0.64	0.64
F1-Score	0.76	0.76

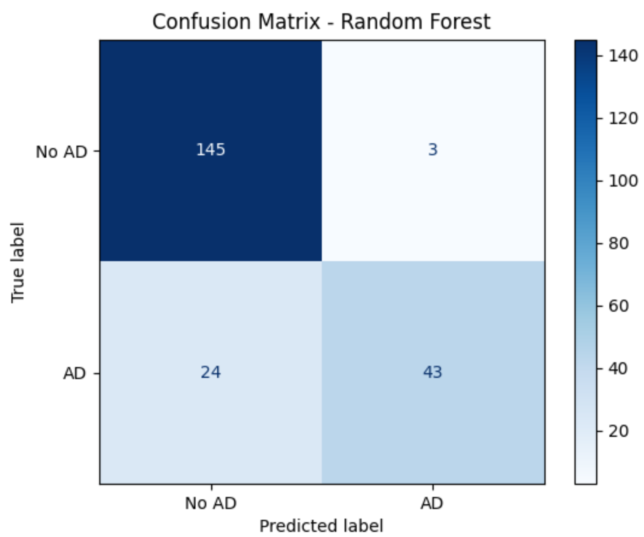


FIGURE 24. Confusion Matrix for the Random Forest model.

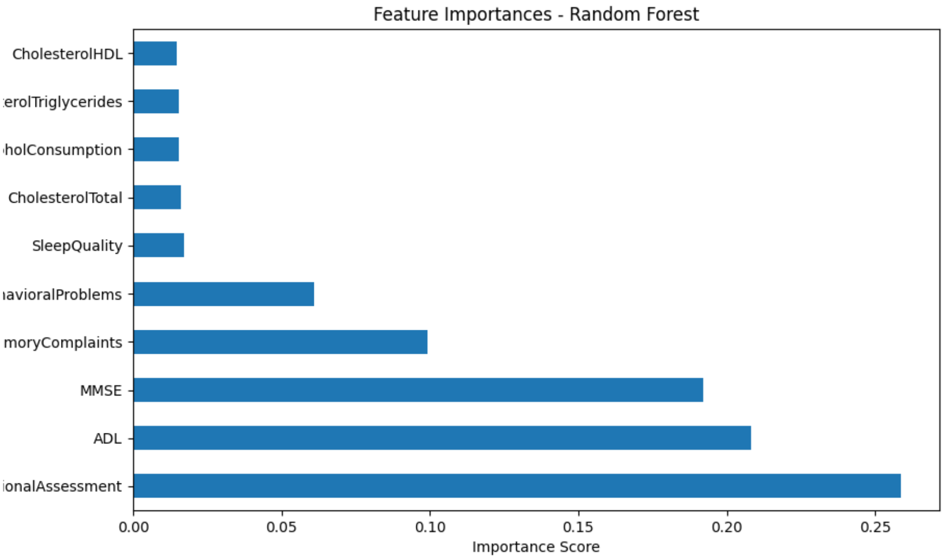


FIGURE 25. Feature Importances for the Random Forest model.

5. Final Comparison: ANN vs Decision Tree

As seen in Table 10, a comparison between the ANN model and the Decision Tree Model is compared.

TABLE 10. Performance metrics for the Random Forest model.

Model	Test Accuracy	Precision	Recall	F1-Score
ANN	83.00%	0.77	0.73	0.75
Decision Tree (Pruned)	94.42%	0.92	0.90	0.91

1. **Which algorithm results in better performance?** The Decision Tree model outperformed the ANN in terms of accuracy and F1-score. However, this conclusion is dataset-specific and does not generalize to all tasks.

2. **Can we claim that this algorithm is a better learning algorithm than the other in general? Why or why not?** No, the performance of a learning algorithm depends on the

dataset and problem context. While the Decision Tree performed well in this study, the ANN may outperform it for larger datasets or more complex tasks requiring higher-dimensional feature interactions.

3. **Suppose that we want to add some new classes to the existing dataset. What changes should be made in Decision Trees and Neural Network classifiers in order to include new classes?**
    - **Decision Trees**: Extend the tree by adding new branches to handle additional classes.
    - **ANNs**: The output layer's size must be adjusted to match the number of classes. The loss function should also change to categorical_crossentropy for multi-class classification. Retraining the model will be necessary.

6. Conclusions

1. **Artificial Neural Networks (ANNs)**:
    - The ANN model underwent extensive hyperparameter tuning, including evaluations of learning rates, regularization techniques (L2, dropout), and batch normalization.
    - After optimization, the ANN achieved a validation accuracy of 0.8348 and maintained stability across epochs, overcoming the initial overfitting issues.
    - The ANN showed better adaptability for handling complex feature interactions, making it well-suited for more intricate datasets or scenarios with large amounts of data.
2. **Decision Tree Classifiers**:
    - The default Decision Tree model provided a solid baseline performance but lacked constraints on depth, leading to overfitting.
    - Optimization with GridSearchCV and pruning improved interpretability and performance, achieving a test accuracy of 0.9441, slightly outperforming the ANN on this dataset.

- Decision Trees demonstrated excellent interpretability, highlighting key features (FunctionalAssessment, MMSE, ADL) as the most significant predictors for Alzheimer's disease.

3. **Random Forest**:
    - As an alternative to Decision Trees, the Random Forest model was robust and provided balanced performance with a test accuracy of 0.8744.
    - However, it underperformed compared to the optimized Decision Tree and required more computational resources, making it less efficient for this specific dataset.
4. **Model Comparisons**:
    - The **optimized Decision Tree** outperformed the ANN in accuracy for this specific dataset, highlighting its suitability for smaller, structured datasets.
    - ANNs showed promise for scalability and adaptability, with the potential to outperform Decision Trees on larger or more complex datasets.
5. **Feature Importance Across Models**:
    - Consistent features (FunctionalAssessment, MMSE, and ADL) emerged as the most critical across Decision Tree models and Random Forest, confirming their relevance for predicting Alzheimer's disease.
    - ANNs do not provide inherent feature importance, but their adaptability to high-dimensional data may reveal complex relationships not easily captured by tree-based models.
6. **Overfitting and Generalization**:
    - The ANN required extensive regularization (dropout, L2) to address overfitting and generalize well to unseen data.
    - Decision Trees benefited significantly from pruning to prevent overfitting, achieving simplicity and interpretability without compromising accuracy.
7. **Algorithm Generalization**:

- While the Decision Tree was the best-performing model in this study, its success cannot generalize across all problems. Neural Networks may outperform Decision Trees in datasets with higher complexity or larger size due to their ability to model complex nonlinear relationships.

8. **Practical Recommendations**:
   - For smaller datasets with clear feature importance and interpretability requirements, Decision Trees (optimized and pruned) are highly effective.
   - For more complex datasets requiring flexibility and scalability, Neural Networks are recommended, provided there is sufficient regularization and computational resources.

9. **Key Insights**:

- This study highlights the trade-offs between interpretability (Decision Trees) and scalability (Neural Networks).
- The importance of optimizing hyperparameters and addressing overfitting was evident in both models.
- Both algorithms have unique strengths that make them suitable for different problem contexts, and neither can be universally declared superior without considering the specifics of the task.

7. References

[1] "🧠 Alzheimer's Disease Dataset 🧠." Accessed: Jan. 20, 2025. [Online]. Available: https://www.kaggle.com/datasets/rabieelkharoua/alzheimers-disease-dataset