Diagnosing Alzheimer’s with Machine Learning

An Insight into Understanding the Data Complexity of Algorithm Python Models in Medicine.

Zeeshan Zaidi[†]
Computer Science
UCLA
Los Angeles, CA, United States
zeeshanzaidi@ucla.edu

Arash Dewan
Computer Science
UCLA
Los Angeles, CA, United States
email@email.com

Jason Wan
Dept. Computer Science
UCLA
Los Angeles, CA, United States
jason.wan@ucla.edu

Aneesh Bonthala
Computer Science
UCLA
Los Angeles, CA, United States
aneeshbonthala@g.ucla.edu

William Escobar
Dept. ECE
UCLA
Los Angeles, CA, United States
wescobar96@g.ucla.edu

ABSTRACT

Patients with Alzheimer’s disease experience a gradual decline in memory and cognitive abilities, ultimately losing the capacity to perform basic daily activities. Over 5.8 million Americans are currently affected by Alzheimer’s disease and related dementias [1]. Although the disease is irreversible, early detection and intervention can slow its progression. This study leveraged publicly available diagnostic data on Alzheimer’s to develop predictive models for dementia. Machine learning techniques, including Logistic Regression, Support Vector Machine, Decision Trees, Random Forest, and Neural Networks, were employed. The data were split into training and testing sets to build and evaluate the predictive models. Key risk factors were identified, and the performance of various models were compared to determine the most effective one. The article finishes off with a discussion on findings, limitations, and directions for future research.

1 INTRODUCTION

Alzheimer’s disease (AD) is by far the most common cause of dementia and accounts for up to 80% of all dementia diagnoses [3]. Alzheimer’s disease occurs due to the degeneration and eventual death of numerous neurons in various parts of the brain. It is a slow progressing condition that begins with short-term memory loss. However, as soon as the symptoms begin, and without proper treatment, there is a rapid progression of symptoms which can occur. As such proper methods of diagnosis are needed which aren’t invasive.

Today, definitive diagnosis of Alzheimer’s disease requires either an autopsy or a brain biopsy. Autopsies can only be performed after a patient has died, and brain biopsies are typically considered last due to their high cost, complexity, and lengthy process. Consequently, most Alzheimer's patients are diagnosed based on clinical evaluations. Methods developed consist of cognitive tests, brain imaging, biomarkers, and invasive procedures in order to determine whether a patient has this disorder or not. Most learn when it is too late, and with the delayed administration of treatments, we are always a step behind the quality of life. The consequences of delayed diagnosis are profound.

Given these challenges, the development of innovative diagnostic tools is crucial. Machine learning models offer a promising solution for early and accurate diagnosis of Alzheimers’s disease. By analyzing vast amounts of data, machine learning algorithms can identify patterns and markers associated with the early stages of Alzheimer’s. This approach has the potential to greatly impact the diagnostic process, enabling healthcare providers to detect the disease sooner and initiate treatment earlier, ultimately improving patient outcomes and quality of life. This paper explores the development and implementation of machine learning models to aid the diagnosis process. We analyze diagnostic data paired with various machine learning models to test which model is best suited for the task.

2 Problem Definition and Formalization

The central problem addressed in this research is the ability for machine learning systems to reach the strengths of predicting the likelihood of Alzheimer's using a dataset of clinical features. By identifying relevant features that can serve as indicators of disease, we will teach these to these classification models and formulate accurate predictions.

We are able to mathematically visualize the problem as such: Given a dataset X consisting of n samples with m features each, and corresponding label vector y indicating the presence or absence of Alzheimer's, the objective is to train a classification model $f : X \rightarrow y$. The performance of these models are evaluated using metrics which include accuracy, precision, recall, as well as visualizing it through an operating curve (AUC-ROC).

By gathering advanced data analytics, clinical features, and medical tools, we can pool a dataset that may facilitate early detection. This project envisions a future where early diagnosis is accessible to all, and helps the medical world better understand this disease.

3 Data Preparation, Description, and Preprocessing

Our dataset was a compilation of ten metrics on 373 patients. The target for our model was the group of each patient, which could be "non-demented" if the individual showed no signs of Alzheimer's, "demented" if the individual was showing signs of the disease, and "converted" if the individual began showing symptoms during the trial. The remaining nine metrics served as predictors for the disease state of the individual. They were as follows:

1. Sex (Male or Female)
2. Age (in years)
3. EDUC - Years of Education
4. SES - Socioeconomic Status (Scale of 1-5)
5. MMSE - Mini Mental State Examination
6. CDR - Clinical Dementia Rating
7. eTIV - Estimated total intracranial volume
8. nWBV - Normalized Whole Brain Volume
9. ASF - Atlas Scaling Factor

To prepare our data for experimentation, we first had to account for all missing values in the raw data provided. Nineteen patients had no socioeconomic status listed. To amend this issue without the loss of data, we substituted each missing SES point with the mean of all available ones. Additionally, two patients were missing a MMSE score. Rather than inputting the mean scores across the whole dataset, we substituted the mean score of the twenty-five nearest neighbors of each missing value. This was done using sklearn's KNNImputer function.

With no more missing values, our remaining preprocessing involved normalizing and standardizing our data. The only feature that required normalization was eTIV, as it listed the raw measures of intracranial brain volume, while nWBV provided normalized counts of whole brain volume. To standardize the two measures, we normalized total intracranial volume (eTIV) using a min-max scheme, the same process originally used for whole brain volume (nWBV).

Our last preprocessing step was to assign labels to our binary, categorical data. For sex, we assigned a positive label of 1 to Male, and label 0 to Female. For our target column, we combined all diseased individuals into a single positive category with label 1, and the remaining patients into a negative category with label 0.

Data Metrics

	Age	EDUC	SES	MMSE	CDR	eTIV	nWBV	ASF
count	373.000000	373.000000	354.000000	371.000000	373.000000	373.000000	373.000000	373.000000
mean	77.013405	14.597855	2.460452	27.342318	0.290885	1488.128686	0.729568	1.195461
std	7.640957	2.876339	1.134005	3.683244	0.374557	176.139286	0.037135	0.138092
min	60.000000	6.000000	1.000000	4.000000	0.000000	1106.000000	0.644000	0.876000
25%	71.000000	12.000000	2.000000	27.000000	0.000000	1357.000000	0.700000	1.099000
50%	77.000000	15.000000	2.000000	29.000000	0.000000	1470.000000	0.729000	1.194000
75%	82.000000	16.000000	3.000000	30.000000	0.500000	1597.000000	0.756000	1.293000
max	98.000000	23.000000	5.000000	30.000000	2.000000	2004.000000	0.837000	1.587000

4 Methods Description

4.1 Logistic Regression

4.1.1 Model Architecture & Overview

A logistic regression is a statistical method with a categorical target variable with two possible outcomes. By predicting the probability that a given input belongs to a specific class, it sources through the sigmoid function and maps it graphically. The logistic function is defined as:

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

Where z is the linear combination of input features. For a given set of features X and corresponding coefficients β, the logistic regression model can be expressed as:

$$P(y = 1|X) = \sigma(\beta_0 + \beta_1 X_1 + \beta_2 X_2 + \dots + \beta_n X_n)$$

Where $P(y = 1|X)$ represents the probability of the positive class.

4.2.1 Logistic Implementation and Training

The process of data preprocessing for this type of model is similar to the traditional methods described in Section 3. It begins with handling missing values, encoding categorical variables, and feature scaling.

The following step of model training is oriented around the split of data into training and testing sets to evaluate the model's performance. Once that is formed, the logistic regression model is cross validated through a variety of parameters.

1. *Penalty* - Controls the regularization applied to help prevent overfitting for larger coefficients (e.g. L1: Lasso, L2: Ridge)
2. *Solver* - determines the optimal algorithm to solve the algorithm (i.e. liblinear, lbfgs)
3. *C Value* - controls trade-off between formulating a fitted training data without overfitting

With the most optimal parameters that achieve the best accuracy score, we are able to form the best form of logistic regression classification for the Alzheimer's data.

4.2 Linear and Non-Linear SVM

A Support Vector Machine (SVM) is a supervised machine learning algorithm used for both classification and regression tasks. SVMs work by finding a boundary, known as a hyperplane, that best separates data points of different classes. This boundary is designed to maximize the margin, which is the space between the hyperplane and the closest data points from each class. These closest data points are called support vectors and are crucial in defining the margin.

SVMs can create a hyperplane even when the data is not linearly separable by using different kernels. Kernels are functions that transform the data into a higher-dimensional space, making it possible to find a suitable hyperplane that can separate the classes. The *sklearn* library in Python provides various kernel options, allowing users to determine the best model for their data [4].

In this paper, we utilize sklearn's machine learning library to easily implement an SVM model and determine which kernel is needed to have high accuracy, precision, and recall.

To create an SVM model in code, we used sklearn's *SVC()* classifier function, which implements a SVM model for classification tasks. This function allows us to specify the following parameters to define our model:

1. *Kernel* - allows us to define which kernel function we want to use (linear, poly, rbf, sigmoid).
2. *Probability* - allows for probabilities to be calculated for each class.
3. *Penalty Parameter C* - C is 1 by default and it's a reasonable default choice. If you have a lot of noisy observations you should decrease it: decreasing C corresponds to more regularization.

In order to obtain the best model by using this library we run a program using the most common kernels, and varying the penalty parameter to account for any noise. After creating this model, we calculate the accuracy, precision, recall, and plot the ROC curve for each. This allows us to see which model performs better overall and compare that to all other models that are observed in this paper.

4.3 Decision Tree

4.3.1 Overview of Decision Trees

A Decision tree is a model that is not represented by a single equation; rather, as a computational path. Therefore, they are easy to implement and can model very complex decision boundaries that equations struggle with. At the same time, they are very interpretable to humans. The feature that a decision tree will make its choice split on at any current point is chosen optimally based on what is known as the 'splitting metric'. Other hyperparameters include the maximum number of layers of the tree, the minimum number of data samples that a leaf node must be the destination of, etc.

4.3.2 Implementation

The DecisionTreeClassifier from scikit-learn was utilized to classify the dataset. To optimize the classifier's performance, a GridSearchCV was employed, which performs an exhaustive search over specified hyperparameter values. The parameter grid included options for 'criterion' (gini or entropy), 'splitter' (best or random), 'max_depth' (None, 10, 20, 30), 'min_samples_split' (2, 10, 20), 'min_samples_leaf' (1, 5, 10), and 'max_features' (None, sqrt, log2). A 5-fold cross-validation was conducted to evaluate the combinations of these parameters, using accuracy as the scoring metric. The best parameters and the highest cross-validation accuracy score were identified and reported.

Below is an in-depth description of each of the hyperparameters:

1. *Criterion*: the aforementioned splitting metric.
2. *Splitter*: whether or not to involve randomness in the splitting metric.
3. *Max_depth:* the maximum depth of the tree.
4. *Min_samples_split*: the minimum number of samples required to split an internal node.
5. *Min_samples_leaf*: the minimum number of samples that must be present in a leaf node.
6. *Max_features*: the number of features considered when looking for the best split

4.4 Random Forest

A random forest model is essentially an ensemble of Decision Trees for one set of data. When run on our data, a random forest will consult multiple trees to produce a number of predictions. In this way, each decision tree essentially votes on the classification of a patient, and the majority decision is the model's output. Furthermore, each tree is trained with a random subset of data and variables. This set of differentiated trees helps improve accuracy and overfitting in comparison to a single tree approach.

To create a random forest model in code, we used sklearn's RandomForestClassifier, which implements a random forest ensemble for classification tasks. Nevertheless, to obtain an optimal model, a number of hyperparameters had to be tuned. These included:

1. *N-estimators* - The number of decision trees in the forest.
2. *Criterion* - The function to measure the quality of splits in each tree, where options include Gini impurity, log loss, and entropy calculations.
3. *Max_depth* - The maximum depth of each decision tree.
4. *Min_samples_split* - The minimum number of samples required to split a node in a tree.
5. *Min_samples_leaf* - The minimum number of samples required to be a leaf node.
6. *Max_features* - The amount of features to consider when calculating the "best" split.
7. *Bootstrap* - Determines whether bootstrapped samples are used to construct decision trees, can be set to true or false.

To optimize all of these components for our model, we used sklearn's RandomizedSearchCV function and input our model along with a range for each of the 7 parameters above. Over a number of iterations, the function fit a forest over a random set of parameters, computed its accuracy, then attempted to vary the parameters further to find a better fit. After all iterations were complete, it returned the model with the highest accuracy and its corresponding hyperparameters. For our model, we allowed RandomizedSearchCV to run three sets of 100 iterations, effectively testing 300 models. For categorical inputs, the search was allowed to select from all possibilities. For numerical hyperparameters such as the number of estimators, we provided excessively large ranges to give our optimizer more flexibility.

4.5 Neural Network

4.5.1 Data Preparation and Preprocessing

This model's dataset was initially analyzed using descriptive statistics. The target variable 'Group' was encoded into binary form, with 'Nondemented' mapped to 0, and both 'Demented' and 'Converted' mapped to 1. The dataset was then split into a training+validation set (80%) and a test set (20%) using stratified sampling to maintain class balance.

4.5.2 Model Architecture and Pipeline

A pipeline was constructed to streamline the preprocessing and model training steps. The pipeline consisted of:

1. A preprocessor using ColumnTransformer, which applied:

- StandardScaler for numerical features
- OneHotEncoder for the 'M/F' categorical feature
- OrdinalEncoder for the 'SES' categorical feature

2. SimpleImputer to handle missing values using mean imputation

3. MLPClassifier (Multi-Layer Perceptron) as the core model

4.5.3 Model Hyperparameters

The study explored various hyperparameter combinations:

- *Network sizes*: Small (10), Medium (20, 10), Large (30, 20, 10), Very Large (50, 30, 20, 10), and Extra Large (100, 50, 30, 20, 10)
- *Regularization*: No regularization, L2 (Ridge) weak (0.0001) and strong (0.001), L1 (Lasso) weak (0.0001) and strong (0.001)
- *Initial Learning rates**: 0.001, 0.01, 0.1

All models used ReLU activation functions, Adam optimizer (the initial learnings still have an effect despite this), and were trained for a maximum of 1000 iterations.

4.5.4 Model Selection

K-Fold cross-validation (k=5) was employed to evaluate model performance across different hyperparameter combinations. The mean accuracy and standard deviation were calculated for each model configuration. The best-performing model was selected based on the highest mean cross-validation accuracy.

4.5.5 Model Evaluation

The selected model was then retrained on the entire training+validation set and evaluated on the held-out test set. Performance metrics included accuracy, confusion matrix, classification report (precision, recall, F1-score), and sensitivity (recall for the positive class). The Receiver Operating Characteristic (ROC) curve was plotted, and the Area Under the Curve (AUC) was calculated to assess the model's discriminative ability.

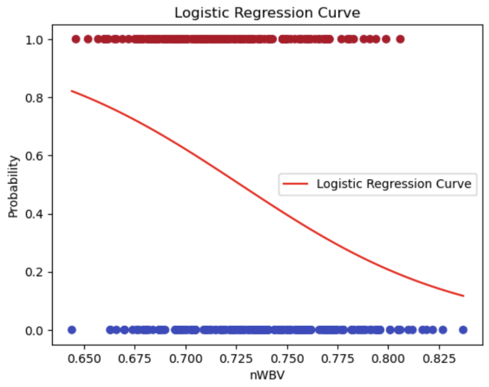
5 Results and Evaluation

5.1 Logistic Regression

The selected cross validation provided us with hyperparameter tuning which resulted in the most optimal parameters being:

1. *C Value: 1*
2. *L1 Regularization*
3. *Liblinear Solver*

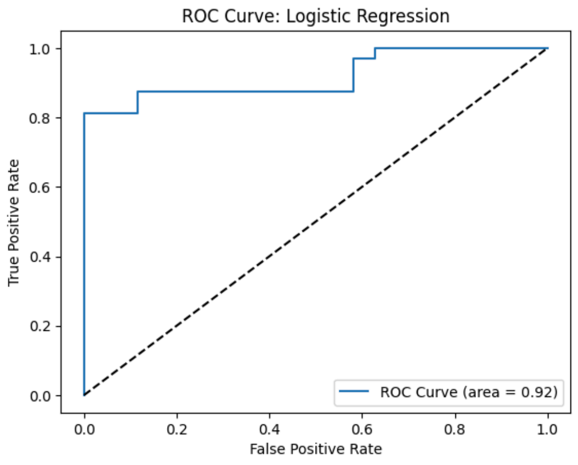
```
Estimated coefficient: [-18.37172347]
Estimated intercept: [13.35679011]
```



With these, we draw an accuracy yielding an impressive accuracy of 95.31%. The robust method for evaluating the model's performance by splitting the data into multiple folds and ensuring the model generalizes well to different subsets of the data.

To better visualize the ability for what logistic regression is capable of, we selected the feature of "nWBV" and it can be seen to have an estimated coefficient of -18.

On the unseen test data, the model achieved an accuracy of 92%. This slight drop from the cross-validation accuracy is expected and indicates that the model generalizes well to new data. Achieving a 92% accuracy on unseen data demonstrates the model's strength in predicting new patients based on clinical features.

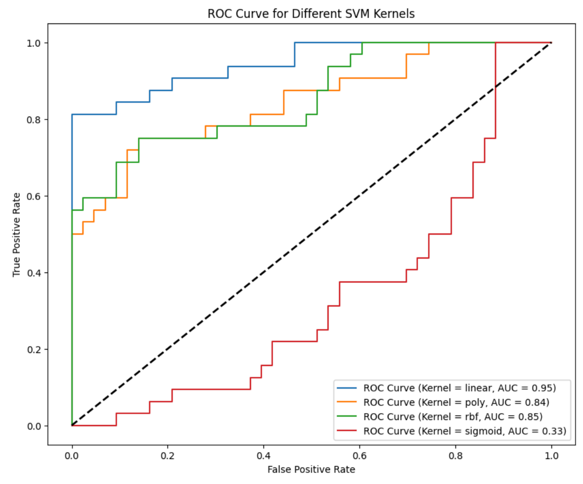


5.2 Linear and Non-Linear SVM

After running our program for different kernels and penalty values, we came up with the following results:

Kernel	Accuracy	Precision	Recall
Linear	0.92	1.0	0.8125
Polynomial (poly)	0.76	0.79	0.59
Gaussian (rbf)	0.79	0.86	0.59
Sigmoid	0.35	0.29	0.375

Additionally, we created the corresponding ROC curves for each of the kernels:



From this output we can see that the best model was for a linear kernel. It offered the highest recall of ~82%. This means that we would only miss 18% of people who were developing Alzheimer's and we would misdiagnose them resulting in a lack of proper treatment.

In spite of these high numbers, it seems that this model is performing *too* well. It seems impossible that this model would be able to predict at an accuracy of 92%. However, after considering the length of the dataset used and how it was cleaned it seems probable. The data set had 3 labels for *demented, converted,* and *non-demented.* Where converted is a classification where a patient shows some minute forms of dementia, but not necessarily having Alzheimer's. However, when we cleaned the data and converted these classifiers to binary values, we grouped both converted and demented under the same value *1* which would mean that that patient suffers from Alzheimer's. This would be one of the reasons why we were more easily able to separate the data using a linear hyperplane and obtaining high accuracy.

In addition, six of the eight classifiers are strongly related to one another. These six classifiers are:

1. *Age (in years)*
2. *MMSE* - Scores typically range from 0-30. Lower scores indicate greater cognitive impairment
3. *CDR* - Clinical Dementia Rating ranges from 0 (no dementia) to 3 (severe dementia)
4. *eTIV* - Estimated total intracranial volume. Lower eTIV often correlates with atrophy to the brian, a shrunken brain which is characteristic of the disease.
5. *nWBV* - Normalized Whole Brain Volume. Lower values indicate smaller brain volume.
6. *ASF* - Atlas Scaling Factor

Considering that the age range for this dataset only covered ages 60-98 years, where patients in this range generally experience some degree of mental degradation (not due to Alzheimer's), we would expect the eTIV, and nWBV to follow. In addition, if there was some degree of dementia that was being detected, then all values would follow in a linear matter. Even though they don't have Alzheimer's the SVM model would classify them as converted which would still be counted as a diagnosis for Alzheimer's.

In any case, to make a more robust model, labels shouldn't be clumped up and be treated separately to properly diagnose patients and provide them with the proper care they need.

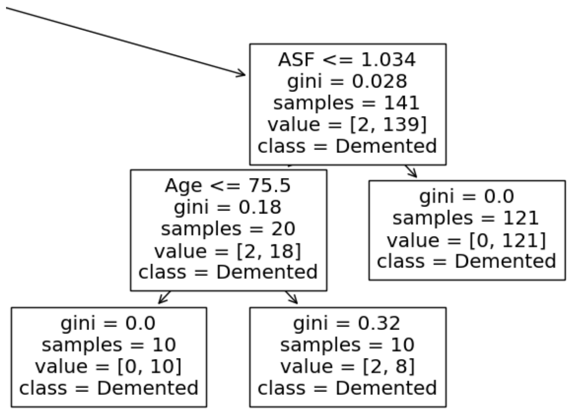
5.3 Decision Tree

Best Model

The best decision tree model (with a cross validation accuracy score of 0.95) was evaluated with the following optimal hyperparameters:

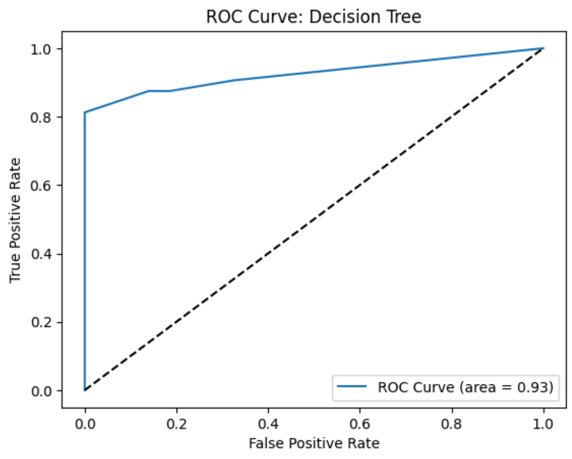
1. *Criterion*: gini.
2. *Splitter*: best (not random).
3. *Max_depth*: None
4. *Min_samples_split*: 2
5. *Min_samples_leaf*: 10
6. *Max_features*: None

The trained model achieved a 92% test set accuracy on our test-train split. To showcase how the tree classifies a test point, shown is a sample (subtree) of the actual model created.



The model also achieved a 93% ROC/AUC score; its confusion matrix and curve shown below.

[[41 2]
[3 29]]



5.4 Random Forest

Best Model

After 300 iterations of our grid search, the best random forest was evaluated using multiple metrics:

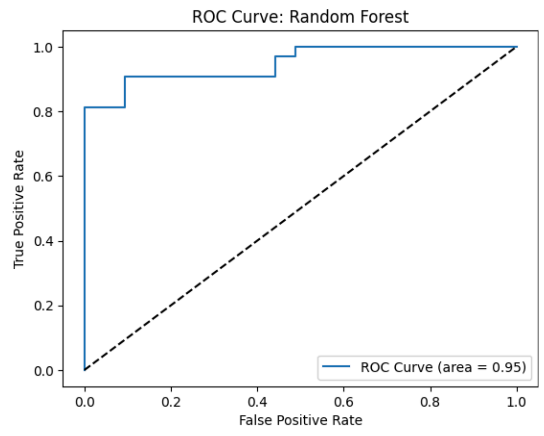
- Cross-Validation Accuracy = 0.96
- Test Set Accuracy = 0.93
- ROC AUC score = 0.95

Furthermore, it produced the confusion matrix and ROC curve plotted below.

Confusion Matrix:

```
[[43  0]
 [ 6 26]]
```

ROC Curve:



5.5 Neural Network

Best Model

After evaluating multiple combinations of model sizes, regularization techniques, and learning rates using 5-fold cross-validation, the best performing model was identified as:

- Architecture: Very Large (50, 30, 20, 10 neurons in hidden layers)
- Regularization: No regularization
- Initial Learning Rate: 0.1

Achieving the highest mean cross-validation accuracy among all tested combinations.

Performance Metrics

The selected model was evaluated on the held-out test set, yielding the following results

- Accuracy: **0.9067** (90.67%)
- Sensitivity (Recall for positive class): 0.8372 (83.72%)
- AUC Score: 0.9426 (94.26%)

The confusion matrix:

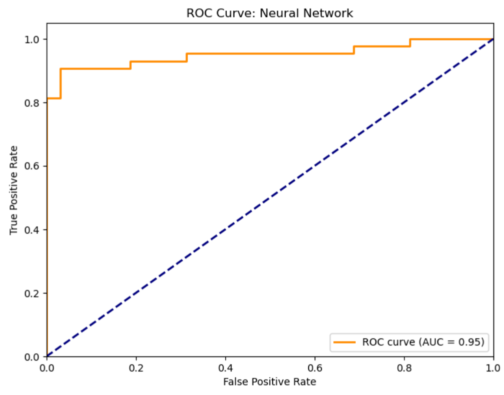
```
[[32  0]
 [ 7 36]]
```

Revealed a larger amount of false negatives to false positives, suggesting a need to lower the threshold to get a more even ratio. For medicine especially, perhaps an even lower threshold would be better to maximize false positives over false negatives.

The classification report provided additional insights:

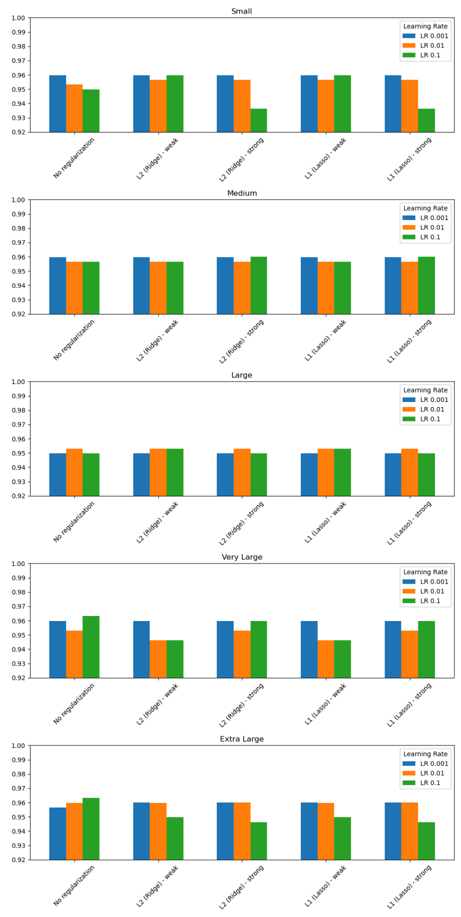
- Precision for class 0 (Non-demented): 0.82
- Recall for class 0: 1.00
- F1-score for class 0: 0.90
- Precision for class 1 (Demented/Converted): 1.00
- Recall for class 1: 0.84
- F1-score for class 1: 0.91

ROC Curve (Image 2):



The ROC curve demonstrates the model's strong discriminative ability, with an AUC of 0.95, indicating great performance in distinguishing between the classes.

Hyperparameter Performance (Image 1):



The bar plots illustrate the performance across different model sizes, regularization techniques, and learning rates. Notably, the best model does not perform the best, as that goes to the 'Extra Large' model, likely due to a greater effect of overfitting.

Overall

These results indicate that the selected neural network model demonstrates high accuracy and robust performance in classifying Alzheimer's disease status, with particularly strong specificity (ability to correctly identify non-demented cases) and good sensitivity in identifying demented or converted cases.

7 Conclusion

Our study compared five machine learning models for early Alzheimer's disease detection using clinical data. The Random Forest model emerged as the top performer, achieving 96% cross-validation accuracy, 93% test accuracy, and a 0.95 ROC AUC score. Its success likely stems from its ability to capture complex feature interactions and resist overfitting.

These results highlight the potential of machine learning in aiding early Alzheimer's diagnosis, which could lead to earlier interventions and improved patient outcomes. However, limitations include the small dataset size and binary classification approach.

Future research should explore larger, more diverse datasets, multi-class classification, image datasets, and real-world clinical validation. Despite promising results, further refinement and integration with clinical expertise are necessary before practical implementation.

Our full code and output, including data processing and implementations of all the models, can be found at this link.

References

[1] "The Truth About Aging and Dementia," *Centers for Disease Control and Prevention*, 2022. https://www.cdc.gov/aging/publications/features/Alz-Greater-Risk.html#:~:text=Alzheimer

[2] Fu, Jay. "PREDICTING THE DIAGNOSIS OF ALZHEIMER'S DISEASE: DEVELOPMENT AND VALIDATION OF MACHINE LEARNING MODELS." *European journal of biomedical and life sciences* 4 (2020): 47-67.

[3] Crous-Bou M, Minguillón C, Gramunt N, et al.: Alzheimer's disease prevention: from risk factors to early intervention. *Alzheimers Res Ther.* 2017;9(1):71. 10.1186/s13195-017-0297-z [PMC free article] [PubMed] [CrossRef] [Google Scholar] F1000 Recommendation

[4] "1.4. Support Vector Machines." *Scikit*, scikit-learn.org/stable/modules/svm.html. Accessed 4 Aug. 2024.