Random forest model for early diagnosis of Alzheimer’s Disease using the *Alzheimer’s Disease dataset*

Vicky Wu 1008745351

Yuxuan Du 1008812821

Kaggle Team Name: Group 18 the best!
Final Ranking: 64
Prediction Score: 0.91987

1. Problem Statement

With an increase in the number of aging populations, Alzheimer’s disease has become a major public health concern, affecting millions of people worldwide. As neurons progressively degenerate, people’s cognitive and physiological functions begin to deteriorate, eventually causing dementia (Breijyeh & Karaman, 2020). This poses significant challenges to the economy and society. According to existing research, Alzheimer's disease treatment costs were approximately $305 billion in 2020, and as the population ages, the cost is predicted to rise to above $1 trillion (Wong, 2020). As neurodegeneration progresses, the lifespan and life quality of patients declines exponentially (Tahami Monfared et al., 2024). It is noted that 20 years before symptom onset, asymptotic stages began to develop (Diogo et al., 2022). Therefore, early diagnosis is critical to improve patient health conditions, enhance quality of life, and reduce societal burden. Advances in diagnostic techniques, such as Magnetic Resonance Imaging (MRI) and positron emission tomography (PET), have paved the way for earlier detection. Still, these methods are often expensive and inaccessible globally (Laske et al., 2014). Machine learning techniques have been extensively applied in clinical settings for their advancements. By integrating diverse data sources, such as genetics, lifestyle, and symptoms, machine learning tools can uncover hidden patterns and predict disease onset more effectively, which is essential for performing early diagnosis and optimizing treatment plans.

In this report, we will process the data, employ several machine learning models to analyze the dataset, and predict whether a patient has Alzheimer's disease. By comparing the prediction accuracy, recall, and precision of each model, we aim to identify the most effective features that could potentially be applied in future clinical settings for early diagnosis.

2. Statistical Analyses

2.1 Exploratory Data Analysis (EDA) and Initial Feature Selection

According to the health information of 2149 patients, disease diagnosis can be predicted using machine learning models. Through utilizing data points of demographic details, lifestyle factors, medical history, clinical measurements, cognitive and functional assessments, and symptoms, the diagnosis of Alzheimer’s disease per patient can be done more accurately. However, there are 32 features in the dataset, making it hard to conclude which features impact the prediction the most. Therefore, identifying the most important features that contribute to the diagnosis is a vital part of the report.

The correlational relationships between each predictor are analyzed using a correlation matrix. As displayed below, there is no obvious strong correlation between variables. With no missing data points nor outliers in the dataset, the distribution of each feature is classified into “1: diagnosed with Alzheimer’s disease” and “0: not diagnosed with Alzheimer’s disease”. Then, each feature is separated by its nature and plotted using bar graphs respectively. From Figure 2 below, sixteen categorical features are plotted and compared. Among the categorical features, “Behavioral Problems” and “Memory Complaints” variables exhibit statistically significant differences among the two groups, implying their importance in contributing to the prediction of Alzheimer’s diagnosis.

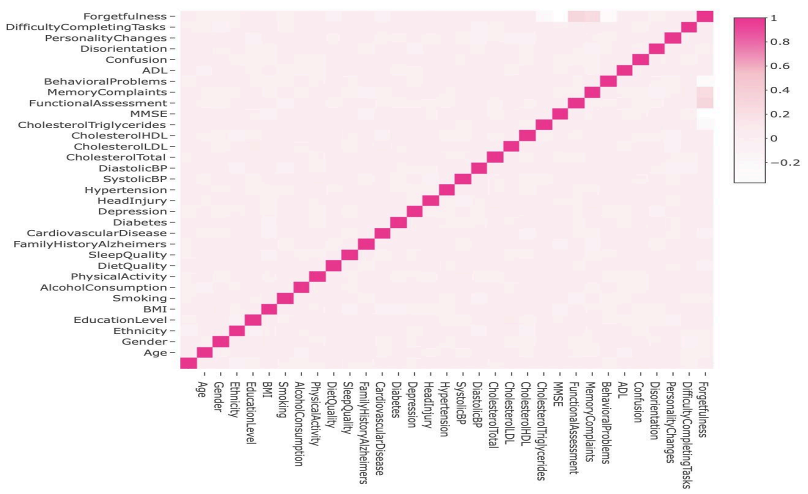

Figure 1. Correlation matrix of 32 explanatory features. The correlation index legend is displayed beside, as the pink color turns darker, so is the correlation magnitude between variables. No obvious correlational relationship was discovered as the plot is mainly covered by light pink color.

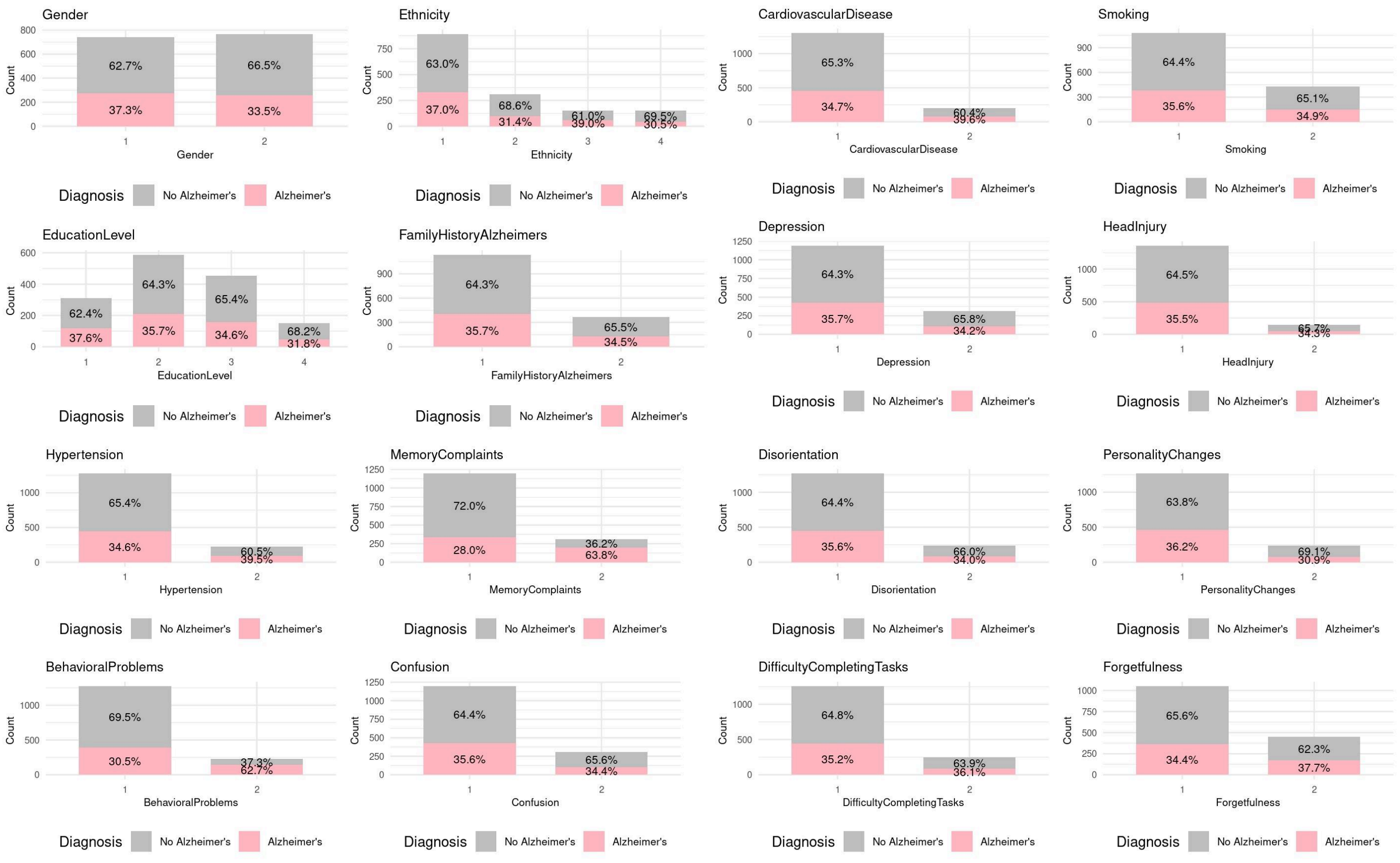

Figure 2. Categorical variables visualization. 16 categorical variables, including "Gender", "Ethnicity", "Cardiovascular Disease", "Smoking", "Education Level", "Family History", "Depression", "Head Injury", "Hypertension", "Memory Complaints", "Disorientation", "Personality Changes", "Behavioral Problems", "Confusion", "Difficulty Completing Tasks", and "Forgetfulness" are classified by Alzheimer's diagnosis. Gray color represents not diagnosed as Alzheimer's, while pink represents diagnosed as Alzheimer's.

Similarly, numerical features are plotted using boxplots as shown in Figure 3. “MMSE”, “Functional Assessment”, and “ADL” show great differences statistically among the two categories, indicating their potential contribution to Alzheimer’s disease early diagnosis.

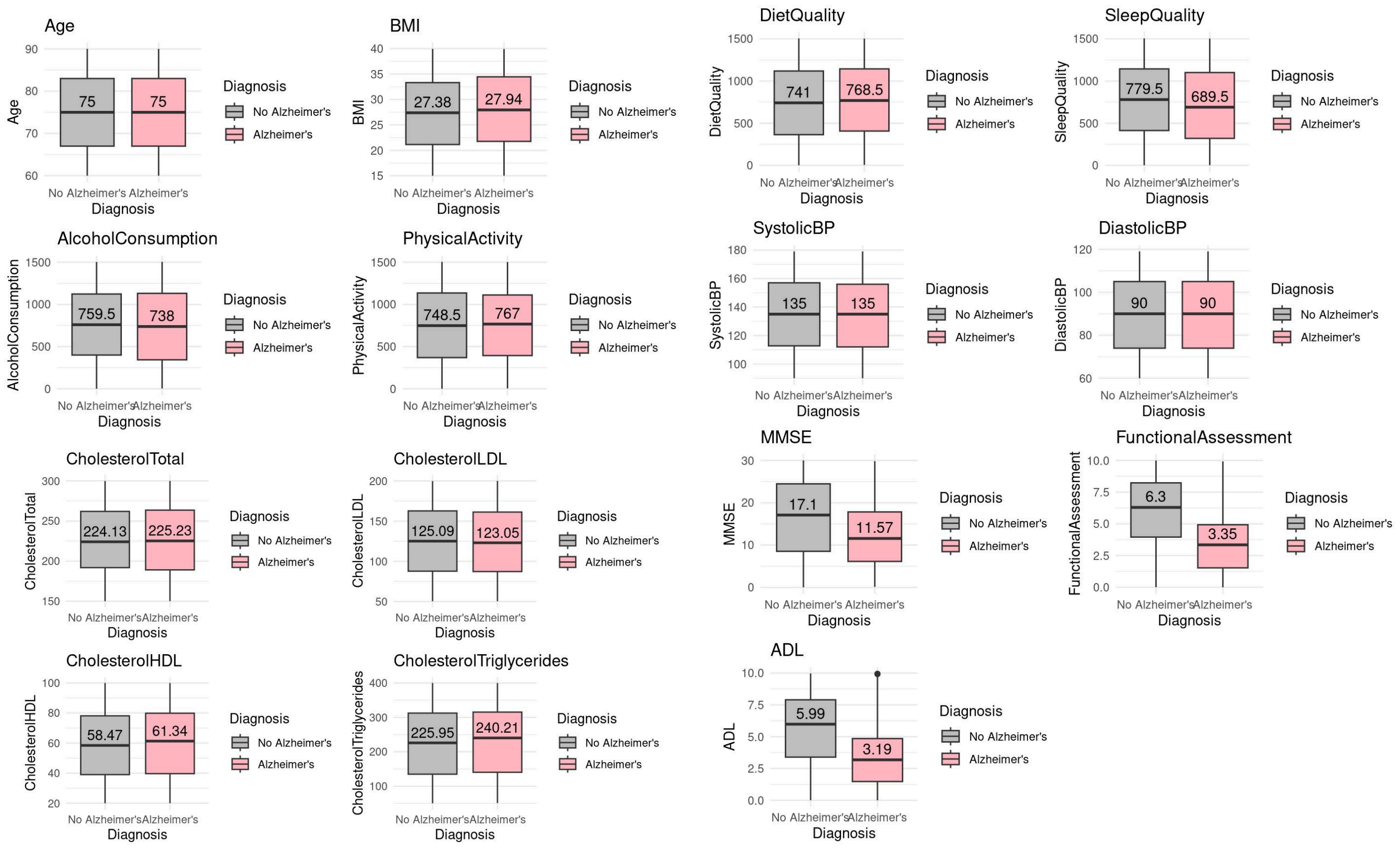

Figure 3. Numerical variables visualization. 15 numerical variables, including “Age”, “BMI”, “Diet Quality”, “Sleep Quality”, “Alcohol Consumption”, “Physical Activity”, “Systolic BP”, “Diastolic BP”, “Cholesterol Total”, “Colesterol LDL”, “Cholesterol HDL”, “Cholesterol Triglycerides”, “MMSE”, “Functional Assessment”, and “ADL” are sorted based on Alzheimer’s diagnosis. Gray color represents not diagnosed as Alzheimer’s, while pink represents diagnosed as Alzheimer’s. The mean values are displayed in the 50th quantile in the boxplots for further comparison.

In conclusion, five predictors: “Behavioral Problems”, “Memory Complaints”, “MMSE”, “Functional Assessment”, and “ADL” are marked as important contributors to Alzheimer’s early diagnosis from EDA.

2.2 Data Processing

Before fitting models, the dataset is pre-processed for better analysis. Firstly, categorical variables are converted into factors, whereas numerical variables are converted to a numeric form, ensuring format compatibility and proper classification. Since "XXXConfid" is present for every patient in the "Doctor In Charge" column, it will not affect the prediction of Alzheimer’s disease. Together with the "Patient ID" column, it is removed from the dataset.

2.3 Best Subset Feature Selection

To further confirm the features selected in EDA contribute statistically significantly to the diagnosis of Alzheimer’s disease, best subset selection is conducted. The selection results are evaluated using two metrics: mallows' Cp and adjusted. As visualized in Figure 4, the smallest train MSE is obtained at seven features, including "Ethnicity," "Sleep Quality," "MMSE," "Functional Assessment," "Memory Complaints," "Behavioral Problems," and "ADL”. These features were then selected.

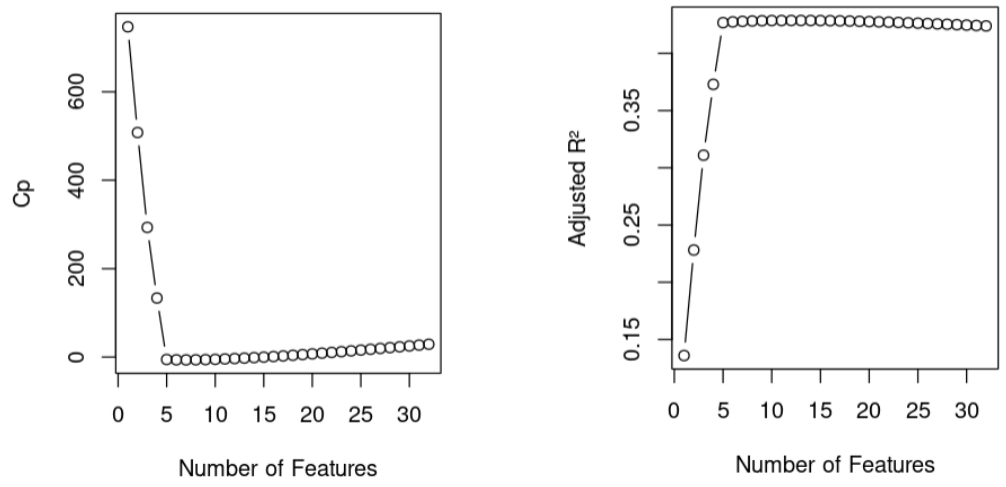

Figure 4. Best subset selection evaluation metrics. The x-axis represents the number of features and the y-axis represents the corresponding metric values. The ideal number of features is achieved with lowest Cp value and highest adjusted R² value.

Subsequently, the dataset is split into predictors (X) and the target variable (y), with an 80-20 train-test split applied to create separate training and testing datasets, ensuring reproducibility with a fixed random seed.

2.4 Model Construction

2.4.1 Decision Tree Model Prediction

The decision tree model is a simple yet effective method for predicting Alzheimer’s disease, especially when datasets contain diverse features across multiple categories. Without the prerequisite of linear relationships between variables, decision trees can capture complex, non-linear interactions. This flexibility makes them well-suited for modeling complicated relationships between various clinical, behavioral, and demographic factors and Alzheimer's diagnosis. In addition, decision trees are highly interpretable, as their structure mirrors human decision-making processes, making the identification of statistically significant features easier. This interpretability makes decision trees an excellent baseline model for prediction.

Table 1. Confusion matrix and evaluation metrics of decision tree model

	0	1	Accuracy	Recall	Precision	Misclassification error rate
0	184	5	0.95	0.9099	0.9528	4.49%
1	10	101				

A decision tree is fitted on the processed dataset using five predictors, “FunctionalAssessment”, “MMSE”, “ADL”, “MemoryComplaints”, and “BehavioralProblems”, to predict the diagnosis. The small misclassification error rate, high accuracy, recall, and precision scores reflects the excellent classification ability of the model.

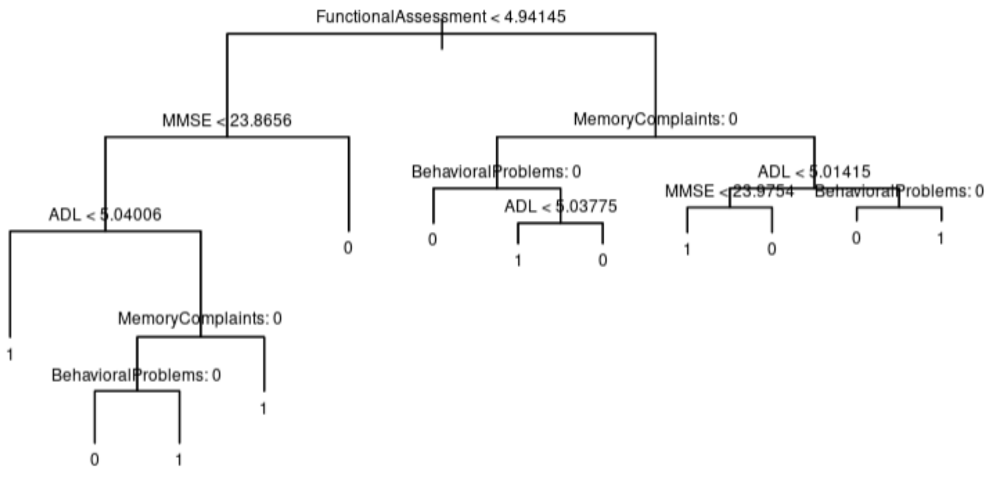

Figure 5. **Decision tree for predicting Alzheimer’s disease.** It includes 12 terminal nodes and 5 predictors.

2.4.2 Random Forest Model Prediction

Random forest is an improvement of decision trees that improves model performance by decorrelating the trees. Through randomly selecting a subset of predictors at each split, individual trees are built differently and less affected by the strong predictors, reducing the variance and improving generalizability. With the advantages of handling high-dimensional data and resisting over-fitting, random forest is effective for predicting the diagnosis of Alzheimer’s disease from a large dataset.

To train the model, 500 trees are built, and 10-fold cross-validation is performed to determine the optimal number of predictors for each split. After cross-validation, the initial model with 32 predictors is refined to an updated model with 16 predictors. This update reduces the out-of-bag (OOB) error rate from 5.32% to 4.49%, indicating improved classification efficiency. Accuracy, recall, and precision scores are also calculated to further evaluate the model. Based on the confusion matrix, the updated model demonstrates slight improvements in both accuracy and recall. Therefore, the updated model with 16 predictors is more effective in diagnosing Alzheimer’s disease.

Table 2. Confusion matrix and evaluation metrics of random forest models

				Accuracy	Recall	Precision
32 Predictors (original model)		0	1	0.96	0.9351	0.9528
	0	187	5			
	1	7	101			
16 predictors (updated model)		0	1	0.9633	0.9439	0.9528
	0	188	5			
	1	6	101			

5 predictors (feature-selection model)		0	1	0.9333	0.9216	0.8868
	0	186	12			
	1	8	94			

Figure 6 visualizes the top seven most important features identified by the updated model. These features-"FunctionalAssessment," "ADL," "MMSE," "MemoryComplaints," "BehavioralProblems," "DietQuality," and "Hypertension"-are ranked based on their contribution to the model, as measured by the mean decrease in accuracy and the mean decrease in the Gini index. A larger decrease in the Gini index signifies a more important predictor, highlighting the significant role these features play in Alzheimer's diagnosis.

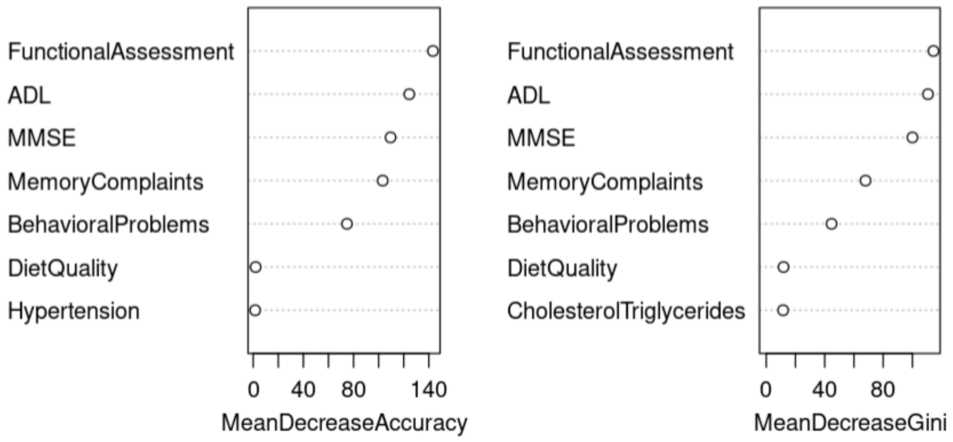

Figure 6. Important features of updated random forest model. Top 7 features are displayed.

Among the important features, five features previously highlighted during EDA: “Behavioral Problems,” “Memory Complaints,” “MMSE,” “Functional Assessment,” and “ADL” emerge as the top contributors to model performance. Consequently, the dataset is updated to include only these five features, and a new random forest analysis is conducted. However, despite the reduced feature set, the updated model's performance, as reflected by the confusion matrix, is worse than the 16 predictors model.

2.4.3 XG Boosting Model Prediction

Unlike random forest, which grows trees independently in parallel, XGBoost constructs trees sequentially. Each new tree attempts to correct the errors of the previously built trees, reducing bias while retaining low variance. Growing slowly and forming smaller trees, the boosting model avoids overfitting, serving as a competing model to the random forest model.

Table 3. Confusion matrix and evaluation metrics of XG Booting model

	0	1	Accuracy	Recall	Precision
0	458	28	0.9415	0.9398	0.8993
1	16	250			

10-fold cross-validation is performed to determine the optimal number of boosting rounds and the best threshold, with 18 rounds identified as the best-performing parameter. Together with a 0.1 learning rate, 2 threads, and 4 maximum depth of each tree, the XGBoost model is constructed. The model is then evaluated using accuracy, recall, and precision scores. The final model achieves a relatively good performance as shown in the table.

3. Results and Conclusion

3.1 Model Performance Comparison

Table 4. Effectiveness comparison among different models

Models	Accuracy	Recall	Precision
Decision Tree	0.95	0.9099	0.9528
Random Forest (updated model)	0.9633	0.9439	0.9528
Random Forest (Feature selection model)	0.93	0.9216	0.8868
XG Boosting	0.9415	0.9398	0.8993

Based on the statistics in Table 4, the random forest model that uses the full dataset and contains 16 predictors has the highest scores for accuracy, recall, and precision. According to the definition, accuracy measures the overall correctness of the model by calculating the proportion of correctly classified instances out of all instances, recall assesses how many of the actual positive instances are correctly predicted, and precision calculates how many of the predicted positive instances are actually positive. For the random forest model with five selected features and the XG Boosting model, they both have relatively high accuracy and recall scores, but relatively low precision scores. This can be attributed to aggressive positive predictions, meaning the model may prioritize capturing all possible positive cases, even at the cost of making more false positive predictions. In contrast, the decision tree model has a relatively high accuracy and precision score, but a relatively low recall score, which can be attributed to conservative positive prediction, meaning the model is cautious about predicting the positive class. It prioritizes being correct when it predicts positive, even at the cost of missing many true positives. In general, the random forest model with 16 features has the best performance among these 4 models, working effectively at distinguishing between classes. In addition, it minimizes both false positives and false negative rates. Therefore, the random forest model containing 16 features is selected as the optimal model and submitted to *Kaggle*.

3.2 Feature Importance

The random forest model achieves the highest accuracy with the updated model, while using only five selected features results in the lowest accuracy, albeit still relatively high at 93%. Despite the drop in accuracy, the latter model offers greater interpretability. The five features ("Behavioral Problems," "Memory Complaints," "MMSE," "Functional Assessment," and "ADL") were chosen based on EDA, best subset selection, and feature importance rankings from the random forest model. Further validated by the decision tree results, these features are crucial to predicting Alzheimer's diagnosis. However, the remaining predictors still contribute to the overall accuracy. For optimal predictive performance, the final random forest model was fitted using all 16 features.

3.3 Conclusion

The random forest model with 16 predictors emerged as the optimal solution, demonstrating superior performance in predicting outcomes. Key features driving the model's success include “Behavioral Problems”, “Memory Complaints”, “MMSE”, “Functional Assessment”, and “ADL”.

4. Discussion

In the early stages of Alzheimer's, individuals may still function independently, continuing to drive, work, and socialize. Nevertheless, they frequently suffer from memory problems, including misplacing objects or forgetting commonly used phrases. The ADL scale can be used to measure these changes and has been reported to have a positive correlation with the advancement of Alzheimer's disease (Reisberg et al., 2001). Measuring orientation, attention, language, and visuospatial skills, the MMSE serves as a great cognitive assessment tool. Lower scores on the MMSE indicate more severe cognitive impairment, which is closely linked to Alzheimer's symptoms (Arevalo-Rodriguez et al., 2021). As early indicators of cognitive degradation and predictors of dementia onset, memory complaints often occur prior to diagnosis (Geerlings et al., 1999). During the preclinical phases, behavioral problems like mood swings and irritability are often observed. In addition to assisting Alzheimer's diagnosis, functional assessments also help to determine support for patients and caregivers (Chaves et al., 2011). By combining these factors, doctors can better forecast Alzheimer's disease and have an improved understanding of the early stages of it.

4.1 Limitations

The dataset lacks physiological data, such as genetic risk factors and biomarkers, which are critical for identifying Alzheimer’s disease. Features like “Memory Complaints” and “ADL” rely on subjective reports of patients or caregivers, making them vulnerable to variability due to differences in perception or reporting accuracy. The demographic composition of the dataset primarily includes Caucasians, African Americans, and Asians, limiting the generalizability of the selected features to other populations. Furthermore, the age range of participants is restricted to 60–90 years, excluding younger individuals who may exhibit early signs of Alzheimer’s and potentially reducing the precision in predicting the disease at its earliest stages.

4.2 Future Directions

This random forest model can be used and verified in clinical situations in the future to evaluate its dependability and practicality. For example, hospitals may carry out examinations on people over the age of 50 to identify early indicators of Alzheimer's disease. Additionally, the model can be improved by incorporating the physiological data mentioned above to make it more comprehensive. By extending its age and demographic range, the model can become more accurate and generalizable.

Appendix

References

Arevalo-Rodriguez, I., Smailagic, N., Roqué-Figuls, M., Ciapponi, A., Sanchez-Perez, E., Giannakou, A., Pedraza, O. L., Bonfill Cosp, X., & Cullum, S. (2021). Mini-mental state examination (MMSE) for the early detection of dementia in people with mild cognitive impairment (MCI). Cochrane Database of Systematic Reviews, 2021(7).

Breijyeh, Z., & Karaman, R. (2020). Comprehensive review on alzheimer’s disease: Causes and treatment. Molecules, 25(24), 5789.

Chaves, M. L. F., Godinho, C. C., Porto, C. S., Mansur, L., Carthery-Goulart, M. T., Yassuda, M. S., & Beato, R. (2011). Cognitive, functional and behavioral assessment: Alzheimer’s disease. Dementia &amp; Neuropsychologia, 5(3), 153–166.

Diogo, V. S., Ferreira, H. A., & Prata, D. (2022). Early diagnosis of alzheimer’s disease using Machine Learning: A multi-diagnostic, generalizable approach. Alzheimer’s Research &amp; Therapy, 14(1).

Economic burden of alzheimer disease and managed care considerations. (2020). The American Journal of Managed Care, 26(Suppl 8).

Geerlings, M. I., Jonker, C., Bouter, L. M., Adèr, H. J., & Schmand, B. (1999). Association between memory complaints and incident alzheimer’s disease in elderly people with normal baseline cognition. American Journal of Psychiatry, 156(4), 531–537.

Laske, C., Sohrabi, H. R., Frost, S. M., López-de-Ipiña, K., Garrard, P., Buscema, M., Dauwels, J., Soekadar, S. R., Mueller, S., Linnemann, C., Bridenbaugh, S. A., Kanagasingam, Y., Martins, R. N., & O'Bryant, S. E. (2015). Innovative diagnostic tools for early detection of Alzheimer's disease. Alzheimer's & dementia : the journal of the Alzheimer's Association, 11(5), 561–578.

Reisberg, B., Finkel, S., Overall, J., Schmidt-Gollas, N., Kanowski, S., Lehfeld, H., Hulla, F., Sclan, S. G., Wilms, H.-U., Heininger, K., Hindmarch, I., Stemmler, M., Poon, L., Kluger, A., Cooler, C., Bergener, M., Hugonot-Diener, L., Robert, P. H., & Erzigkeit, H. (2001). The alzheimer’s disease activities of Daily Living International Scale (ADL-IS). International Psychogeriatrics, 13(2), 163–181.

Tahami Monfared, A. A., Khachatryan, A., Hummel, N., Kopiec, A., Martinez, M., Zhang, R., & Zhang, Q. (2024). Assessing quality of life, economic burden, and independence across the alzheimer’s disease continuum using patient-caregiver dyad surveys. Journal of Alzheimer’s Disease, 99(1), 191–206.

Your home for data science. Kaggle. (n.d.). https://www.kaggle.com/competitions/classification-of-the-alzheimers-disease/data

Codes

```
knitr::opts_chunk$set(echo = TRUE)
library(dplyr)
library(randomForest)
library(caret)
data <- read.csv("train.csv")
install.packages("xgboost")
install.packages("caret")
library(xgboost)
library(caret)

# Data Processing
str(data)
#Pre-processing. Make sure categorical and numerical variables are well classified
train$Diagnosis <- as.factor(train$Diagnosis)
train$Gender <- as.factor(train$Gender)
train$Ethnicity <- as.factor(train$Ethnicity)
train$EducationLevel <- as.factor(train$EducationLevel)
train$Smoking <- as.factor(train$Smoking)
train$FamilyHistoryAlzheimers <- as.factor(train$FamilyHistoryAlzheimers)
train$CardiovascularDisease <- as.factor(train$CardiovascularDisease)
train$Diabetes <- as.factor(train$Diabetes)
train$Depression <- as.factor(train$Depression)
train$HeadInjury <- as.factor(train$HeadInjury)
train$Hypertension <- as.factor(train$Hypertension)
train$MemoryComplaints <- as.factor(train$MemoryComplaints)
train$BehavioralProblems <- as.factor(train$BehavioralProblems)
train$Confusion <- as.factor(train$Confusion)
train$Disorientation <- as.factor(train$Disorientation)
train$PersonalityChanges <- as.factor(train$PersonalityChanges)
train$DifficultyCompletingTasks <- as.factor(train$DifficultyCompletingTasks)
train$Forgetfulness <- as.factor(train$Forgetfulness)
train$AlcoholConsumption <- as.numeric(train$AlcoholConsumption)
train$PhysicalActivity <- as.numeric(train$PhysicalActivity)
train$DietQuality <- as.numeric(train$DietQuality)
train$SleepQuality <- as.numeric(train$SleepQuality)
data <- train %>% select(-c(PatientID, DoctorInCharge))
# Split data into predictors (X) and target (y)
X <- data %>% select(-Diagnosis)
y <- data$Diagnosis

#EDA
#Correlation Matrix
install.packages("plotly")
library(plotly)
```

```r
# Compute the correlation matrix
cor_matrix <- cor(data, use = "complete.obs")
# Create the heatmap
cor_matrix <- cor(data, use = "complete.obs")
fig <- plot_ly(
  z = cor_matrix,
  type = "heatmap",
  colors = colorRamp(c("white", "#FFEBF0", "pink", "deeppink")),
  text = round(cor_matrix, 2),
  hoverinfo = "text",
  showscale = TRUE,
  width = 800,
  height = 800
) %>%
  layout(
    title = "Correlation Matrix",
    xaxis = list(tickvals = seq_along(colnames(cor_matrix)), ticktext = colnames(cor_matrix)),
    yaxis = list(tickvals = seq_along(rownames(cor_matrix)), ticktext = rownames(cor_matrix)),
    margin = list(l = 100, r = 100, b = 100, t = 100, pad = 4))
print(fig)
#Visualizing Categorical Variables
library(purrr)
library(gridExtra)
categorical_vars <- c("Gender", "Ethnicity", "EducationLevel", "FamilyHistoryAlzheimers",
                      "CardiovascularDisease", "Smoking", "Depression", "HeadInjury",
                      "Hypertension", "MemoryComplaints", "BehavioralProblems",
                      "Confusion", "Disorientation", "PersonalityChanges",
                      "DifficultyCompletingTasks", "Forgetfulness")
subset_vars <- categorical_vars[1:4]
plot_categorical_distribution <- function(variable) {
  data %>%
    group_by(.data[[variable]], Diagnosis) %>%
    summarise(count = n(), .groups = "drop") %>%
    group_by(.data[[variable]]) %>%
    mutate(proportion = count / sum(count)) %>%
    ggplot(aes(x = factor(.data[[variable]]), y = count, fill = factor(Diagnosis))) +
    geom_bar(stat = "identity", width = 0.7) +
    geom_text(aes(label = scales::percent(proportion, accuracy = 0.1)),
              position = position_stack(vjust = 0.5), color = "black", size = 3) +
    scale_fill_manual(values = c("1" = "grey", "2" = "#FFB6C1"),
                      labels = c("No Alzheimer's", "Alzheimer's")) +
    labs(title = paste(variable),
         x = variable,
         y = "Count",
         fill = "Diagnosis") +
    theme_minimal() +
    theme(
      plot.title = element_text(size = 10),
      axis.title = element_text(size = 8),
      axis.text = element_text(size = 7),
      legend.position = "bottom",
      legend.text = element_text(size = 8))}
plots <- map(subset_vars, plot_categorical_distribution)
do.call(grid.arrange, c(plots, ncol = 2))
subset_vars <- categorical_vars[5:8]
plot_categorical_distribution <- function(variable) {
  data %>%
    group_by(.data[[variable]], Diagnosis) %>%
    summarise(count = n(), .groups = "drop") %>%
    group_by(.data[[variable]]) %>%
    mutate(proportion = count / sum(count)) %>%
    ggplot(aes(x = factor(.data[[variable]]), y = count, fill = factor(Diagnosis))) +
    geom_bar(stat = "identity", width = 0.7) +
    geom_text(aes(label = scales::percent(proportion, accuracy = 0.1)),
              position = position_stack(vjust = 0.5), color = "black", size = 3) +
    scale_fill_manual(values = c("1" = "grey", "2" = "#FFB6C1"),
                      labels = c("No Alzheimer's", "Alzheimer's")) +
    labs(title = paste(variable),
         x = variable,
         y = "Count",
         fill = "Diagnosis") +
    theme_minimal() +
    theme(
      plot.title = element_text(size = 10),
      axis.title = element_text(size = 8),
      axis.text = element_text(size = 7),
      legend.position = "bottom",
      legend.text = element_text(size = 8))}
plots <- map(subset_vars, plot_categorical_distribution)
do.call(grid.arrange, c(plots, ncol = 2))
subset_vars <- categorical_vars[9:12]
plot_categorical_distribution <- function(variable) {
  data %>%
    group_by(.data[[variable]], Diagnosis) %>%
    summarise(count = n(), .groups = "drop") %>%
    group_by(.data[[variable]]) %>%
    mutate(proportion = count / sum(count)) %>%
    ggplot(aes(x = factor(.data[[variable]]), y = count, fill = factor(Diagnosis))) +
    geom_bar(stat = "identity", width = 0.7) +
    geom_text(aes(label = scales::percent(proportion, accuracy = 0.1)),
              position = position_stack(vjust = 0.5), color = "black", size = 3) +
    scale_fill_manual(values = c("1" = "grey", "2" = "#FFB6C1"),
                  labels = c("No Alzheimer's", "Alzheimer's")) +
    labs(title = paste(variable),
         x = variable,
         y = "Count",
         fill = "Diagnosis") +
    theme_minimal() +
    theme(
      plot.title = element_text(size = 10),
      axis.title = element_text(size = 8),
      axis.text = element_text(size = 7),
      legend.position = "bottom",
      legend.text = element_text(size = 8))}
plots <- map(subset_vars, plot_categorical_distribution)
do.call(grid.arrange, c(plots, ncol = 2))
subset_vars <- categorical_vars[13:16]
plot_categorical_distribution <- function(variable) {
  data %>%
    group_by(.data[[variable]], Diagnosis) %>%
    summarise(count = n(), .groups = "drop") %>%
    group_by(.data[[variable]]) %>%
    mutate(proportion = count / sum(count)) %>%
    ggplot(aes(x = factor(.data[[variable]]), y = count, fill = factor(Diagnosis))) +
    geom_bar(stat = "identity", width = 0.7) +
    geom_text(aes(label = scales::percent(proportion, accuracy = 0.1)),
              position = position_stack(vjust = 0.5), color = "black", size = 3) +
    scale_fill_manual(values = c("1" = "grey", "2" = "#FFB6C1"),
                  labels = c("No Alzheimer's", "Alzheimer's")) +
    labs(title = paste(variable),
         x = variable,
         y = "Count",
         fill = "Diagnosis") +
    theme_minimal() +
    theme(
      plot.title = element_text(size = 10),
      axis.title = element_text(size = 8),
      axis.text = element_text(size = 7),
      legend.position = "bottom",
      legend.text = element_text(size = 8))}
plots <- map(subset_vars, plot_categorical_distribution)
do.call(grid.arrange, c(plots, ncol = 2))
#Visualizing Numerical Variables
numerical_vars <- c("Age", "BMI", "AlcoholConsumption", "PhysicalActivity",
                  "DietQuality", "SleepQuality", "SystolicBP", "DiastolicBP",
                  "CholesterolTotal", "CholesterolLDL", "CholesterolHDL",
                  "CholesterolTriglycerides", "MMSE", "FunctionalAssessment",
                  "ADL")
```

```
subset_vars_n <- categorical_vars[1:4]
plot_numerical_distribution <- function(variable) {
  # Calculate median values for each Diagnosis
  median_values <- data %>%
    group_by(Diagnosis) %>%
    summarise(median_value = median(.data[[variable]], na.rm = TRUE), .groups = "drop")
   data %>%
    ggplot(aes(x = factor(Diagnosis), y = .data[[variable]], fill = factor(Diagnosis))) +
    geom_boxplot() +
    scale_fill_manual(values = c("1" = "grey", "2" = "#FFB6C1"),
                      labels = c("No Alzheimer's", "Alzheimer's")) +
    scale_x_discrete(labels = c("1" = "No Alzheimer's", "2" = "Alzheimer's")) +
    labs(title = paste(variable),
         x = "Diagnosis",
         y = variable,
         fill = "Diagnosis") +
    theme_minimal() +
    geom_text(data = median_values,
              aes(x = factor(Diagnosis), y = median_value,
                  label = round(median_value, 2)),
              color = "black", size = 4, vjust = -0.5)}
plots <- map(subset_vars_n, plot_numerical_distribution)
do.call(grid.arrange, c(plots, ncol = 2))
subset_vars_n <- categorical_vars[5:8]
plot_numerical_distribution <- function(variable) {
  # Calculate median values for each Diagnosis
  median_values <- data %>%
    group_by(Diagnosis) %>%
    summarise(median_value = median(.data[[variable]], na.rm = TRUE), .groups = "drop")
     data %>%
    ggplot(aes(x = factor(Diagnosis), y = .data[[variable]], fill = factor(Diagnosis))) +
    geom_boxplot() +
    scale_fill_manual(values = c("1" = "grey", "2" = "#FFB6C1"),
                      labels = c("No Alzheimer's", "Alzheimer's")) +
    scale_x_discrete(labels = c("1" = "No Alzheimer's", "2" = "Alzheimer's")) +
    labs(title = paste(variable),
         x = "Diagnosis",
         y = variable,
         fill = "Diagnosis") +
    theme_minimal() +
    geom_text(data = median_values,
              aes(x = factor(Diagnosis), y = median_value,
                  label = round(median_value, 2)),
              color = "black", size = 4, vjust = -0.5)}
plots <- map(subset_vars_n, plot_numerical_distribution)
do.call(grid.arrange, c(plots, ncol = 2))
```

```
subset_vars_n <- categorical_vars[9:12]
plot_numerical_distribution <- function(variable) {
  # Calculate median values for each Diagnosis
  median_values <- data %>%
    group_by(Diagnosis) %>%
    summarise(median_value = median(.data[[variable]], na.rm = TRUE), .groups = "drop")
     data %>%
    ggplot(aes(x = factor(Diagnosis), y = .data[[variable]], fill = factor(Diagnosis))) +
    geom_boxplot() +
    scale_fill_manual(values = c("1" = "grey", "2" = "#FFB6C1"),
                      labels = c("No Alzheimer's", "Alzheimer's")) +
    scale_x_discrete(labels = c("1" = "No Alzheimer's", "2" = "Alzheimer's")) +
    labs(title = paste(variable),
         x = "Diagnosis",
         y = variable,
         fill = "Diagnosis") +
    theme_minimal() +
    geom_text(data = median_values,
              aes(x = factor(Diagnosis), y = median_value,
                  label = round(median_value, 2)),
              color = "black", size = 4, vjust = -0.5)}
plots <- map(subset_vars_n, plot_numerical_distribution)
do.call(grid.arrange, c(plots, ncol = 2))
subset_vars_n <- categorical_vars[13:15]
plot_numerical_distribution <- function(variable) {
  # Calculate median values for each Diagnosis
  median_values <- data %>%
    group_by(Diagnosis) %>%
    summarise(median_value = median(.data[[variable]], na.rm = TRUE), .groups = "drop")
     data %>%
    ggplot(aes(x = factor(Diagnosis), y = .data[[variable]], fill = factor(Diagnosis))) +
    geom_boxplot() +
    scale_fill_manual(values = c("1" = "grey", "2" = "#FFB6C1"),
                      labels = c("No Alzheimer's", "Alzheimer's")) +
    scale_x_discrete(labels = c("1" = "No Alzheimer's", "2" = "Alzheimer's")) +
    labs(title = paste(variable),
         x = "Diagnosis",
         y = variable,
         fill = "Diagnosis") +
    theme_minimal() +
    geom_text(data = median_values,
              aes(x = factor(Diagnosis), y = median_value,
                  label = round(median_value, 2)),
              color = "black", size = 4, vjust = -0.5)}
plots <- map(subset_vars_n, plot_numerical_distribution)
do.call(grid.arrange, c(plots, ncol = 2))
```

```
# Best Subset Feature Selection
library(leaps)
full_data <- data.frame(y, X)
# Perform best subset selection
best_subset <- regsubsets(y ~ ., data = full_data, nvmax = ncol(X))
subset_summary <-summary(best_subset)
# Plot metrics to determine optimal number of features
par(mfrow = c(1, 2))
plot(subset_summary$cp, xlab = "Number of Features", ylab = "Cp", type = "b")
plot(subset_summary$adjr2, xlab = "Number of Features", ylab = "Adjusted R²", type = "b")
# Get the number of features with the lowest Cp
optimal_features <- which.min(subset_summary$cp)
print(paste("Optimal number of features:", optimal_features))
optimal_vars <- names(coef(best_subset, optimal_features))
print(optimal_vars)

# 80-20 Train-Test Split
set.seed(135) # For reproducibility
train_index <- createDataPartition(data$Diagnosis, p = 0.8, list = FALSE)
train_data <- data[train_index, ]
test_data <- data[-train_index, ]

#Full random forest model using 32 predictors
rf_model <- randomForest(Diagnosis ~ ., data = train_data, ntree = 500, mtry = 32, importance = TRUE)
rf_model
yhat.bag <- predict(rf_model,test_data)
# Convert both predictions and actual values to numeric
yhat_numeric <- as.numeric(as.character(yhat.bag))
actual_numeric <- as.numeric(as.character(test_data$Diagnosis))
# Calculate Mean Squared Error (MSE)
test_mse <- mean((yhat_numeric - actual_numeric)^2)
print(paste("Test MSE:", test_mse))
yhat.bag <- factor(yhat.bag, levels = c(0, 1))
actual <- factor(test_data$Diagnosis, levels = c(0, 1))
# Evaluate model performance
confusion <- confusionMatrix(yhat.bag, test_data$Diagnosis)
print(confusion)
# Feature importance
importance <- importance(rf_model)
varImpPlot(rf_model)

# Cross-Validation of Predictor Numbers
control <- trainControl(method = "cv", number = 10)
# Define the grid of mtry values
tuneGrid <- expand.grid(mtry = seq(2, ncol(X), by = 2))
```

```
# Train the model
rf_cv <- train(
  Diagnosis ~ .,
  data = train_data,
  method = "rf",
  metric = "Accuracy",
  tuneGrid = tuneGrid,
  trControl = control,
  ntree = 500)

# Updated random forest model with 16 predictors
rf_model_optimal <- randomForest(Diagnosis ~ ., data = train_data, ntree = 500, mtry = 16, importance =
TRUE)
rf_model_optimal
yhat_optimal <- predict(rf_model_optimal, test_data)
# Convert both predictions and actual values to numeric
yhat_new <- as.numeric(as.character(yhat_optimal))
actual_numeric <- as.numeric(as.character(test_data$Diagnosis))
# Calculate Mean Squared Error (MSE)
test_mse1 <- mean((yhat_new - actual_numeric)^2)
print(paste("Test MSE:", test_mse1))
confusionMatrix(yhat_optimal, test_data$Diagnosis)
print(confusionMatrix)
# Feature importance
importance <- importance(rf_model_optimal)
# Extract variable importance as a data frame
importance_df <- data.frame(
  Variable = rownames(importance),
  MeanDecreaseAccuracy = importance[, "MeanDecreaseAccuracy"],
  MeanDecreaseGini = importance[, "MeanDecreaseGini"])
# Sort
top_features <- importance_df %>%
  arrange(desc(MeanDecreaseAccuracy)) %>%
  head(7)
# Visualize only the top 7 features
varImpPlot(rf_model_optimal, sort = TRUE, n.var = 7, main = "Top 7 Feature Importance")

# Random Forest Model with five selected features
data1 <- read.csv("train_new.csv")
str(data1)
train_new$MemoryComplaints <- as.factor(train_new$MemoryComplaints)
train_new$BehavioralProblems <- as.factor(train_new$BehavioralProblems)
train_new$Diagnosis <- as.factor(train_new$Diagnosis)
data1 <- train_new %>% select(-c(PatientID, DoctorInCharge))
# Split data into predictors (X) and target (y)
X1 <- data1 %>% select(-Diagnosis)
```

```
y1 <- data1$Diagnosis
set.seed(4324)
train_new_index <- createDataPartition(data1$Diagnosis, p = 0.8, list = FALSE)
train_new_data <- data1[train_new_index, ]
test_new_data <- data1[-train_new_index, ]
rf_model1 <- randomForest(Diagnosis ~ ., data = train_new_data, ntree = 500, mtry = 5, importance =
TRUE)
rf_model1
yhat.new <- predict(rf_model1,test_new_data)
# Convert both predictions and actual values to numeric
yhat_numeric1 <- as.numeric(as.character(yhat.new))
actual_numeric1 <- as.numeric(as.character(test_new_data$Diagnosis))
# Calculate Mean Squared Error (MSE)
test_mse1 <- mean((yhat_numeric1 - actual_numeric1)^2)
print(paste("Test MSE:", test_mse1))
yhat.new <- factor(yhat.new, levels = c(0, 1))
actual1 <- factor(test_new_data$Diagnosis, levels = c(0, 1))
# Evaluate model performance
confusion1 <- confusionMatrix(yhat.new, test_new_data$Diagnosis)
print(confusion1)
# Feature importance
importance1 <- importance(rf_model1)
varImpPlot(rf_model1)

#Pre-processing the test data
test$Gender <- as.factor(test$Gender)
test$Ethnicity <- as.factor(test$Ethnicity)
test$EducationLevel <- as.factor(test$EducationLevel)
test$Smoking <- as.factor(test$Smoking)
test$FamilyHistoryAlzheimers <- as.factor(test$FamilyHistoryAlzheimers)
test$CardiovascularDisease <- as.factor(test$CardiovascularDisease)
test$Diabetes <- as.factor(test$Diabetes)
test$Depression <- as.factor(test$Depression)
test$HeadInjury <- as.factor(test$HeadInjury)
test$Hypertension <- as.factor(test$Hypertension)
test$MemoryComplaints <- as.factor(test$MemoryComplaints)
test$BehavioralProblems <- as.factor(test$BehavioralProblems)
test$Confusion <- as.factor(test$Confusion)
test$Disorientation <- as.factor(test$Disorientation)
test$PersonalityChanges <- as.factor(test$PersonalityChanges)
test$DifficultyCompletingTasks <- as.factor(test$DifficultyCompletingTasks)
test$Forgetfulness <- as.factor(test$Forgetfulness)
test$AlcoholConsumption <- as.numeric(test$AlcoholConsumption)
test$PhysicalActivity <- as.numeric(test$PhysicalActivity)
test$DietQuality <- as.numeric(test$DietQuality)
test$SleepQuality <- as.numeric(test$SleepQuality)
```

```
test_predictions <- predict(rf_model_optimal, newdata = test)
print(test_predictions)


# Decision Tree Model
library(tree)
tree.AD <- tree(Diagnosis ~ ., data = train_data)
summary(tree.AD)
# Use the fitted tree to predict on the test data
tree_pred <- predict(tree.AD, test_data, type = "class")
# Create a contingency table
table(tree_pred, test_data$Diagnosis)
# Calculate the accuracy
accuracy <- sum(tree_pred == test_data$Diagnosis) / length(tree_pred)
cat("Accuracy: ", accuracy, "\n")
# Plot the tree
plot(tree.AD, cex = 1.2)
text(tree.AD1, pretty = 0, cex = 0.6)


# XG Boosting Model
# For XG boosting, all the variables need to be numeric variable
train$Diagnosis <- as.numeric(train$Diagnosis)
train$Gender <- as.numeric(train$Gender)
train$Ethnicity <- as.numeric(train$Ethnicity)
train$EducationLevel <- as.numeric(train$EducationLevel)
train$Smoking <- as.numeric(train$Smoking)
train$FamilyHistoryAlzheimers <- as.numeric(train$FamilyHistoryAlzheimers)
train$CardiovascularDisease <- as.numeric(train$CardiovascularDisease)
train$Diabetes <- as.numeric(train$Diabetes)
train$Depression <- as.numeric(train$Depression)
train$HeadInjury <- as.numeric(train$HeadInjury)
train$Hypertension <- as.numeric(train$Hypertension)
train$MemoryComplaints <- as.numeric(train$MemoryComplaints)
train$BehavioralProblems <- as.numeric(train$BehavioralProblems)
train$Confusion <- as.numeric(train$Confusion)
train$Disorientation <- as.numeric(train$Disorientation)
train$PersonalityChanges <- as.numeric(train$PersonalityChanges)
train$DifficultyCompletingTasks <- as.numeric(train$DifficultyCompletingTasks)
train$Forgetfulness <- as.numeric(train$Forgetfulness)
train$AlcoholConsumption <- as.numeric(train$AlcoholConsumption)
train$PhysicalActivity <- as.numeric(train$PhysicalActivity)
train$DietQuality <- as.numeric(train$DietQuality)
train$SleepQuality <- as.numeric(train$SleepQuality)
train$Age <- as.numeric(train$Age)
train$DiastolicBP <- as.numeric(train$DiastolicBP)
```

```
train$SystolicBP <- as.numeric(train$SystolicBP)
# Remove unnecessary variable
data <- train %>% dplyr::select(-c(PatientID, DoctorInCharge))
# Create train and test group, each 50 percent
train_index <- createDataPartition(data$Diagnosis, p = 0.5, list = FALSE)
train_data <- data[train_index, ]
test_data <- data[-train_index, ]
# Ensure the target variable is numeric variable in both test and train dataset
train_data$Diagnosis <- as.numeric(as.factor(train_data$Diagnosis)) - 1
test_data$Diagnosis <- as.numeric(as.factor(test_data$Diagnosis)) - 1
# Split the data into features (X) and target (y)
X_train <- as.matrix(train_data[, -ncol(train_data)])
y_train <- train_data$Diagnosis
X_test <- as.matrix(test_data[, -ncol(test_data)])
y_test <- test_data$Diagnosis
# Create the DMatrix (XGBoost format)
dtrain <- xgb.DMatrix(data = X_train, label = y_train)
# Specify the model's parameters
params <- list(
  objective = "binary:logistic",  \
  eval_metric = "error",
  max_depth = 4,
  eta = 0.1,
  nthread = 2)
# 10-fold cross-validation to choose the best number of trees
cv_result <- xgb.cv(
  params = params,
  data = dtrain,
  nrounds = 100,
  nfold = 10,
  verbose = 1,
  early_stopping_rounds = 10,
  maximize = FALSE)
# Train the model using the best number of rounds
xgb_model <- xgboost(
  params = params,
  data = dtrain,
  nrounds = cv_result$best_iteration,
  verbose = 1)
# Make predictions on the training data
preds <- predict(xgb_model, X_test)
# Final predictions
final_preds_binary <- ifelse(preds > 0.5, 1, 0)
# Calculate final accuracy
final_accuracy <- mean(final_preds_binary == y_test)
# Confusion Matrix
```

```
confusion_matrix <- confusionMatrix(factor(final_preds_binary), factor(y_test))
print(confusion_matrix)
# Ensure test dataset' only contains the predictor columns
data_test <- test %>% dplyr::select(-c(PatientID, DoctorInCharge))
X_test_new <- as.matrix(data_test)
# Make predictions
preds_new <- predict(xgb_model, X_test_new)
```