The Classification of the Alzheimer's Disease

Dataset: Alzheimer's Disease Dataset
Team Members: Vincent Wang 1007647539, Hanyu Wang 1006996262, Xinzhu Meng 1007810987, Kaixin Guo 1007052715
Kaggle Team Name: Group 73
Kaggle Ranking: 114
Prediction Score: 0.90705

1. Problem Statement

Alzheimer's disease is a common, irreversible brain disease that causes brain shrinkage and eventual death of brain cells. It affects over 50 million people globally, and a number is projected to rise sharply in the coming decades due to the global population ages. Since the personal self-care and even the ability to perform basic daily tasks would be affected by the disease, the growing number of patients will pose a major challenge to public health. Therefore, in order to slow disease progression and enhance patient outcomes, the diagnostic techniques need to be improved for earlier intervention.

This project leverages data from the **Kaggle Competition on Alzheimer's Disease Classification** to develop a machine-learning model for predicting diagnoses. The dataset includes demographic, lifestyle, medical history, and cognitive assessment data for over 2,000 people, with variables such as age, gender, education level, family history, medical conditions, and cognitive test results like MMSE scores. The project aims to identify critical indicators of early detection by analyzing these diverse features. The ultimate goal is to create a predictive model that supports healthcare professionals in preliminary screenings, enabling timely interventions and reducing the strain on caregivers and healthcare systems. Beyond prediction, this project investigates the relationships between key risk factors and Alzheimer's diagnosis, offering actionable insights for public health strategies focused on prevention and risk reduction.

Machine-learning techniques are used in this study to enhance diagnostic accuracy, and the results demonstrate great potential of data-driven approaches in addressing this complex medical challenge. The outcomes provide a foundation for integrating advanced data types, such as imaging and genetic biomarkers, in future iterations. Ultimately, this work would contribute to global efforts to alleviate the burden of Alzheimer's disease on individuals, families, and healthcare systems.

2. Statistical Analysis

2.1 Data Preprocessing and Feature Engineering

The dataset underwent rigorous preprocessing to ensure its quality and compatibility with machine-learning techniques. Missing data was visualized by the highlighting of patterns and gaps using the Amelia package. Median imputation was employed for numerical features due to its robustness against outliers, while mode imputation preserved class distributions for categorical variables.

Correlation analysis using corrplot was conducted to assess multicollinearity, ensuring that highly correlated variables did not bias the model. Categorical variables such as Gender and Smoking were encoded using one-hot encoding, a step necessary for compatibility with the machine-learning algorithms employed. Continuous variables, including BMI and CholesterolLevels, were normalized using the caret::preProcess function to standardize scales and enhance model convergence.

2.2 Data Loading and Cleaning

The data is divided into training data (train.csv) and testing data (test.csv), they are cleaned and preprocessed during this step. The data were checked for missing values and filled with the median to fill in the missing values for numeric data, and with appropriate transformations for categorical data. Next, we initially removed variables that were not relevant to the study (e.g., PatientID and DoctorInCharge) and ensured that categorical variables were consistently formatted and converted to factor types so that the model could correctly process them correctly.

2.3 Exploratory Data Analysis

Before model training, exploratory analysis was conducted to discover underlying patterns and features through a basic understanding of the data. Both univariate analysis (Figure 1) and bivariate analysis (Figure 2) are done for this study.

Each variable is analyzed individually, with a histogram showing numerical variables (Figure 1) and a bar chart showing the distribution for categories for categorical variables (Figure 2). The visualizations of cognitive scores show the distribution of each variable in the state of diagnosis. For example, MMSE shows clear separations between diagnosed and undiagnosed cases, underscoring their importance as predictive features. The result also indicates that the surveyed individuals are elderly people aged 60 to 90.

For bivariate analysis, the correlation matrices are visualized using corr plots, and the relation between numerical variables and target variables are shown using box-and-line plots.

2.4 Machine Learning Models

In this project, logistic regression, random forest, linear discriminant analysis (LDA), Naive Bayes, and hyper parameter methods are used to fit the models. The following are the explanations for each method.

Logistic regression is a common classification model, it establishes a linear relationship between predictors and the target variable. We fitted a logistic regression model using the training model and evaluated its performance on a validation set via a confusion matrix. The accuracy of the logistic regression model is 86.33%.

Random Forest is an integrated learning method based on decision trees. The training process employed **k-fold cross-validation** (k=5) to mitigate overfitting

and provide reliable estimates of model performance. This strategy leveraged caret for training and hyperparameter tuning. A grid search was performed to optimize the key parameter, mtry, representing the number of variables considered at each split. The final Random Forest model was then built using the optimized mtry value, ensuring strong predictive performance. The accuracy of the model fitted by random forest is 94.33%.

Linear Discriminant Analysis (LDA) is a statistical method used for classification problems by finding linear combinations that maximize inter-class differences and minimize intra-class differences. The accuracy of the model fitted by this method is 85.67%.

Naive Bayes is a probability-based classification method that performs classification by calculating prior probabilities and conditional probabilities of features for each category. The accuracy of the model fitted by this method is 86.67%.

Hyperparameter tuning method defines the range of candidate hyperparameter combinations by constructing a grid. In this study, mtry, which represents the number of features randomly selected at each node split, ranges from 10 to 25. The accuracy of the best model fitted by this method (with mtry = 15) is 94.93%.

2.5 Feature Selection

Random forest method and Recursive Feature Elimination (RFE) are used for feature importance analysis. Both results reveal that Age, Cognitive Assessment Scores, and Family History of Alzheimer's were among the top predictors.

Random forest method evaluates feature importance by randomly shuffling the values of a specific feature and observing the resulting drop in model performance. The greater the performance decline would indicate a higher importance of the feature. A bar plot showing the importance of each feature is involved in Figure 3 in Appendix.

The RFE method retains the features that are most predictive of the target variable by progressively removing features and repeatedly evaluating the performance of the model. Ultimately, the feature selection results from RFE confirmed the results obtained by the random forest method, and it also suggested that a model incorporating only these five features achieves better performance.

2.6 Evaluation Metrics

The model was evaluated using a combination of metrics to assess its reliability and clinical relevance. Accuracy served as the primary metric to measure overall classification performance. Precision and recall were calculated to analyze the trade-offs between false positives and false negatives, with the F1 score capturing the balance between these two metrics. Additionally, the area under the Receiver Operating Characteristic curve (ROC), which is shown in Figure 3 in Appendix, provides a threshold-independent measure of the model's ability to distinguish between diagnosed and undiagnosed cases.

The models were evaluated using several methods to see how well they predicted the diagnosis. Accuracy, which measures the percentage of correct predictions, was the main metric. Logistic Regression had an accuracy of 86.33%, Linear Discriminant Analysis (LDA) achieved 85.67%, and Naive Bayes reached 86.67%. However, Random Forest performed the best with an accuracy of 94.33%.

Precision and recall were used to check how well the models balanced mistakes. Recall, also known as sensitivity, showed how many undiagnosed cases were correctly identified. The Random Forest model did very well, with a recall of 98.45%, meaning it missed very few undiagnosed cases. It also had a specificity of 86.79%, meaning it avoided many false positives.

Another important measure was the Area Under the Receiver Operating Characteristic Curve (ROC-AUC), which shows how well a model can tell the difference between diagnosed and undiagnosed cases. The Random Forest model had the highest score of 0.9616, showing it was better at distinguishing between the two groups than the other models.

The Random Forest model was further improved through hyperparameter tuning. This process adjusted its settings to make it work even better. Using cross-validation and a technique called SMOTE to handle imbalanced data, the best number of variables to split on (mtry) was found to be 15. This tuning reduced the model's error rate and made it even more reliable.

Finally, the confusion matrix confirmed that the Random Forest model was the best overall. It had a balanced accuracy of 92.62%, meaning it was both precise and consistent.

In conclusion, the Random Forest model was the best choice for this task. It had the highest accuracy, excellent recall and specificity, and performed well after tuning, making it a strong and reliable tool for predicting Alzheimer's diagnosis.

3. Results and Conclusion

The analysis began with a thorough exploratory data analysis, including statistical summaries, correlation assessments, and visualizations of distributions and relationships between predictors and the diagnosis outcome. Missing values were handled through median imputation for numeric variables and conversion of relevant fields into factors, ensuring that the modeling phase started with a complete and consistent dataset.

Several classification models were trained and evaluated in order to determine which approach yielded the most reliable predictions. Logistic regression served as the initial baseline model, providing easily interpretable coefficients and odds ratios. Although logistic regression is a standard choice in medical and clinical contexts due to its interpretability, it often has limited flexibility when dealing with complex interactions and non-linearities. Additional models, including Linear Discriminant Analysis (LDA) and Naive Bayes, were also explored, each offering its own theoretical advantages. LDA can be effective when class distributions are

approximately normal and the variance-covariance structure is consistent across classes. Naive Bayes can rapidly provide probabilistic outputs under strong conditional independence assumptions. In practice, both methods served as points of comparison rather than outperforming more flexible machine learning models.

A Random Forest model was then trained using 500 trees and variable importance measures to gauge the contribution of each predictor. This ensemble method is known for handling high-dimensional spaces and detecting non-linear interactions among features without extensive manual feature engineering. After producing predictions for the validation set, the Random Forest approach showed promising improvements over the baseline logistic regression and the other considered classical models. The accuracy and confusion matrices derived from these predictions indicated stronger classification performance. Although the exact validation accuracies were not explicitly included in the provided code segments, the approach suggests that the Random Forest model had the strongest generalization capability among the models tried.

Further improvements were sought through hyperparameter tuning. The number of variables tried at each split (mtry) was optimized using the caret package and five-fold cross-validation. Techniques to address potential class imbalance, such as SMOTE, were employed within the training and cross-validation framework. By systematically exploring a range of mtry values, the final tuned Random Forest model was selected based on the best cross-validation performance, indicating the importance of careful model calibration rather than relying on default settings.

To enhance the performance of the Random Forest model, hyperparameter tuning focused on optimizing the number of variables considered at each split (mtry). A grid search approach was employed, exploring values from 10 to 25, combined with five-fold cross-validation for robust evaluation. To address the potential class imbalance within the dataset, SMOTE (Synthetic Minority Over-sampling Technique) was incorporated into the cross-validation framework. The optimal mtry value was determined to be 15, which maximized the model's accuracy during cross-validation. This adjustment led to a final Random Forest model with an out-of-bag (OOB) error rate of 4.43%, confirming the importance of hyperparameter tuning in achieving competitive performance.

To provide a final external evaluation, predictions were made using the selected model on the provided test data. These predictions were submitted to a Kaggle competition, where the model achieved a ranking of 114 and a prediction score of 0.90705. This external benchmark is valuable because it evaluates model performance on a completely unseen dataset and under standardized scoring criteria. Achieving a score of 0.90705 is indicative that the chosen modeling pipeline, particularly the tuned Random Forest model, was robust and competitive in the broader context of that competition.

Stepwise logistic regression models based on AIC and BIC were also examined to investigate the benefits of simpler, more interpretable models. Although stepwise feature selection can identify parsimonious subsets of predictors, the final

performance on the validation set did not indicate substantial improvement over the tuned Random Forest. In a predictive setting where complex interactions and potential non-linear effects might be crucial, the Random Forest maintained its lead, especially after hyperparameter tuning.

In conclusion, the Random Forest model, especially after tuning and addressing class imbalance, demonstrated superior performance compared to logistic regression, LDA, and Naive Bayes. The final chosen model's predictions, as judged by the external Kaggle evaluation, confirmed that the modeling strategy was sound. Although simpler models guided by AIC or BIC were considered, these did not outperform the carefully tuned ensemble. Thus, the main finding is that a well-tuned Random Forest approach can provide reliable and accurate predictions for the given classification problem, as evidenced both by internal validation and external benchmarking.

4. Discussion and Future Work

Although the modeling pipeline was comprehensive, certain limitations deserve mention. The results and comparisons between models were mostly based on accuracy and confusion matrices, and while the code included the potential for ROC and AUC assessments, the final numeric results were not explicitly reported. Without a full set of evaluation metrics, it is challenging to characterize performance in terms of sensitivity, specificity, or calibration. More nuanced metrics might offer greater insights, especially in a clinical context where misclassification costs differ between false positives and false negatives.

Another limitation lies in the imputation and feature engineering strategies. The approach relied on simple median imputation for numeric variables, and the selection of categorical factors followed a straightforward approach without domain-specific transformations. Additional domain knowledge, such as known risk factors for the diagnosis or clinically relevant biomarkers, might have helped engineer more informative features or identify interaction terms that could enhance model performance.

Moreover, while the internal train/validation splitting procedure and cross-validation for hyperparameter tuning were methodologically sound, there was no explicit use of a fully independent external dataset beyond the Kaggle test. Although the Kaggle leaderboard evaluation provides a form of external validation, further independent validation, potentially from another clinical dataset, could establish greater confidence in the model's generalizability.

A related consideration is model interpretability. Although random forests offer excellent predictive performance, they are not as straightforward to interpret as logistic regression. While the variable importance plot gives some sense of the relative influence of predictors, additional explainability techniques such as SHAP values or partial dependence plots could be employed to clarify how specific features affect the model's predictions. This would be particularly valuable if the model is intended for clinical decision support, where transparency and trust are

paramount.

Finally, it is important to acknowledge that stepwise selection methods guided by AIC or BIC may not always yield models that generalize well. Although these criteria help reduce model complexity, they do not always lead to improved predictive performance. Combining domain knowledge, regularization methods, or advanced feature selection techniques might provide a more stable path to model simplicity without losing predictive accuracy.

5. References

5.1 Dataset

The dataset was sourced from the Kaggle competition "Classification of Alzheimer's Disease": https://www.kaggle.com/competitions/classification-of-the-alzheimers-disease

5.2 Tool

RStudio (via JupyterHub): https://posit.co/products/open-source/rstudio/

6. Appendix

Figure 1. EDA Plots (Univariate Analysis)

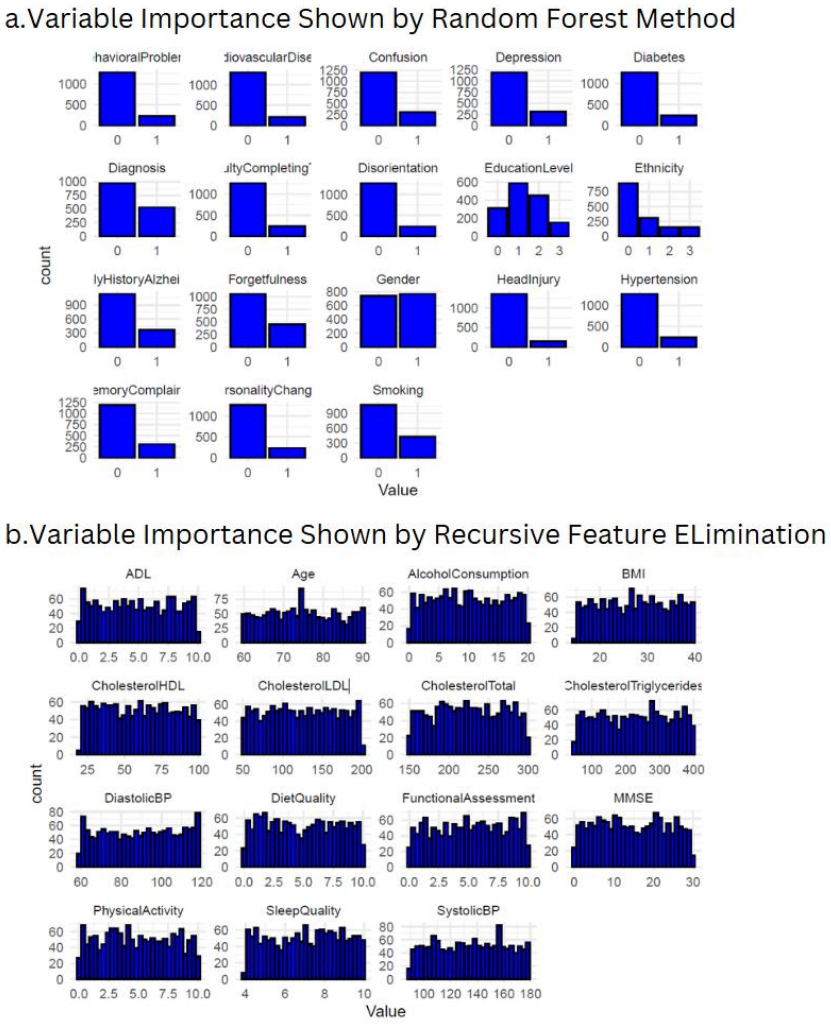


Figure 2. EDA Plots (Bivariate Analysis)

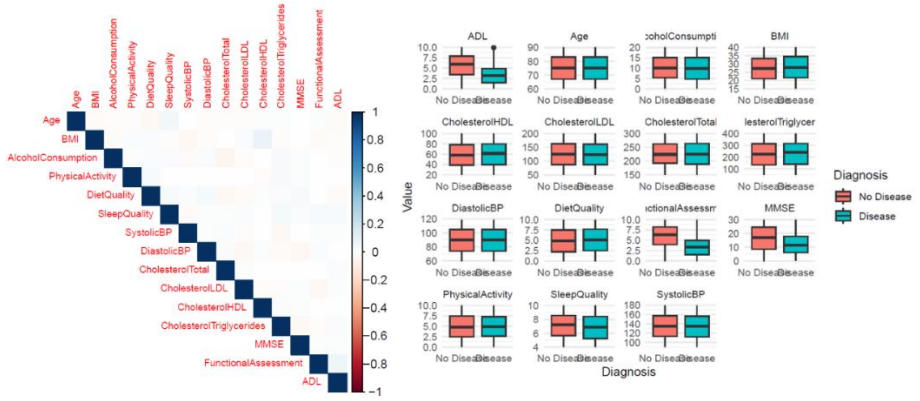


Figure 3. ROC Curves of the Models

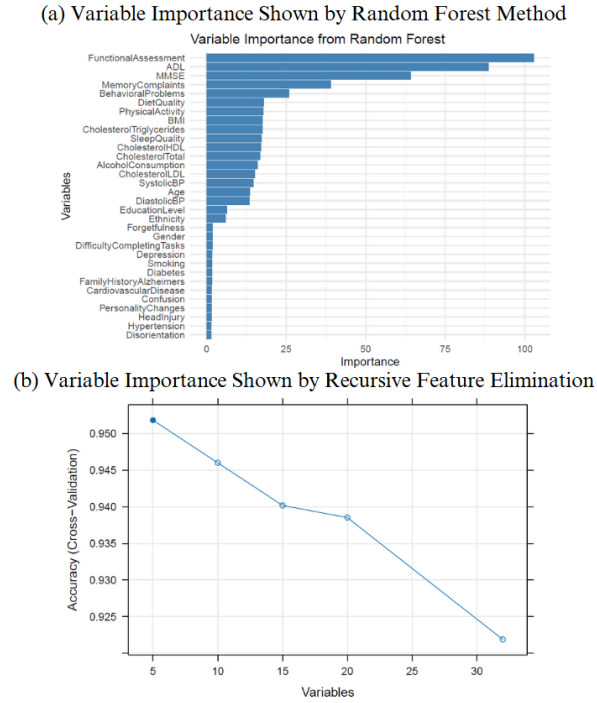
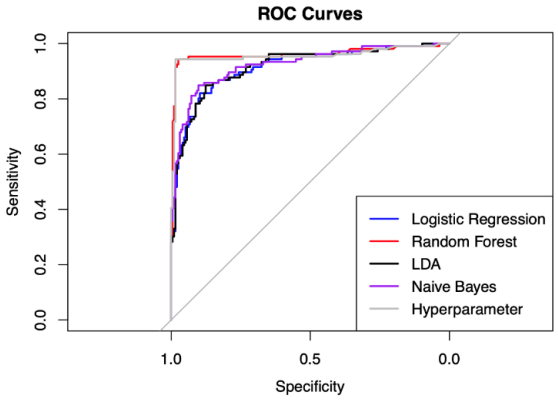


Figure 4. ROC Curves of the Models



7. Code to reproduce

The Classification of the Alzheimer's Disease

2024-12-08

1. Exploratory Data Analysis (EDA)

1.1. Load the Data

```
# Install required packages if not already installed
install.packages(c('tidyverse', 'ggplot2', 'corrplot', 'caret', 'randomForest', 'e1071', 'leaps', 'pROC
```

```
## Installing packages into '/usr/local/lib/R/site-library'
## (as 'lib' is unspecified)
```

```
## also installing the dependencies 'RANN', 'ROSE'
```

```
# Load libraries
library(tidyverse)
```

```
## -- Attaching core tidyverse packages ------------------------ tidyverse 2.0.0 --
## v dplyr     1.1.4     v readr     2.1.5
## v forcats   1.0.0     v stringr   1.5.1
## v ggplot2   3.5.0     v tibble    3.2.1
## v lubridate 1.9.3     v tidyr     1.3.1
## v purrr     1.0.2
```

```
## -- Conflicts ------------------------------------------ tidyverse_conflicts() --
## x dplyr::filter() masks stats::filter()
## x dplyr::lag()    masks stats::lag()
## i Use the conflicted package (<http://conflicted.r-lib.org/>) to force all conflicts to become error
```

```
library(ggplot2)
library(corrplot)
```

```
## corrplot 0.92 loaded
```

```
library(caret)
```

```
## Loading required package: lattice
##
## Attaching package: 'caret'
##
## The following object is masked from 'package:purrr':
##
##     lift
```

```
library(randomForest)
```

```
## randomForest 4.7-1.1
## Type rfNews() to see new features/changes/bug fixes.
##
## Attaching package: 'randomForest'
##
## The following object is masked from 'package:dplyr':
##
```

```
##     combine
##
## The following object is masked from 'package:ggplot2':
##
##     margin
```

```
library(e1071)
library(leaps)
library(pROC)
```

```
## Type 'citation("pROC")' for a citation.
##
## Attaching package: 'pROC'
##
## The following objects are masked from 'package:stats':
##
##     cov, smooth, var
```

```
library(Amelia)
```

```
## Loading required package: Rcpp
## ##
## ## Amelia II: Multiple Imputation
## ## (Version 1.8.1, built: 2022-11-18)
## ## Copyright (C) 2005-2024 James Honaker, Gary King and Matthew Blackwell
## ## Refer to http://gking.harvard.edu/amelia/ for more information
## ##
```

```
library(themis)
```

```
## Loading required package: recipes
##
## Attaching package: 'recipes'
##
## The following object is masked from 'package:stringr':
##
##     fixed
##
## The following object is masked from 'package:stats':
##
##     step
```

```
# Load the datasets
train <- read.csv('train.csv', stringsAsFactors = FALSE)
test <- read.csv('test.csv', stringsAsFactors = FALSE)
```

1.2. Data Overview

1.2.1. Inspect Data Structure

```
# View the first few rows
head(train)
```

```
##   PatientID Age Gender Ethnicity EducationLevel      BMI Smoking
## 1         1  67      0         3              0 37.20518       0
## 2         2  65      1         0              0 35.14184       1
## 3         3  62      0         1              1 17.87510       0
## 4         4  67      0         0              1 37.50344       1
```

```
## 5         5  65      1         0              2 29.18786       1
## 6         6  88      1         0              1 25.67389       0
##   AlcoholConsumption PhysicalActivity DietQuality SleepQuality
## 1         12.2156770         7.780544    6.433890     6.744820
## 2         17.1114042         6.645284    1.112379     7.568751
## 3         13.5255456         9.585769    4.266008     8.247084
## 4         19.9520140         1.953946    6.797333     7.666498
## 5          0.5332093         8.759570    6.364302     6.231143
## 6         10.3799869         8.043196    2.354424     4.764378
##   FamilyHistoryAlzheimers CardiovascularDisease Diabetes Depression HeadInjury
## 1                       0                     0        0          0          0
## 2                       0                     0        0          1          0
## 3                       0                     0        1          0          0
## 4                       0                     1        1          1          0
## 5                       0                     1        0          0          0
## 6                       0                     0        0          0          1
##   Hypertension SystolicBP DiastolicBP CholesterolTotal CholesterolLDL
## 1            0        137         114         270.1677      118.89108
## 2            0        111          82         227.2657      100.58877
## 3            0        131         108         202.4197      184.97482
## 4            0        121          76         235.9426      150.74410
## 5            0        158         117         292.3378      125.42963
## 6            0        126          83         270.9545       85.93615
##   CholesterolHDL CholesterolTriglycerides       MMSE FunctionalAssessment
## 1       78.04944                272.80402  0.6946002             9.986441
## 2       21.15240                156.87964 23.7899987             6.197277
## 3       36.97303                288.78818  6.5920715             9.572719
## 4       62.16979                195.95458 25.3426163             2.487042
## 5       82.86545                295.68221  6.6277415             7.521358
## 6       65.78187                 59.47537 22.1699224             6.592334
##   MemoryComplaints BehavioralProblems      ADL Confusion Disorientation
## 1                1                  0 6.009376         0              0
## 2                0                  0 7.519209         0              0
## 3                0                  0 8.573933         0              0
## 4                0                  0 6.217530         0              0
## 5                1                  0 5.193683         1              0
## 6                0                  0 9.420887         0              1
##   PersonalityChanges DifficultyCompletingTasks Forgetfulness Diagnosis
## 1                  0                         1             1         0
## 2                  0                         0             1         0
## 3                  0                         0             0         0
## 4                  0                         0             1         0
## 5                  0                         0             1         0
## 6                  1                         0             0         0
##   DoctorInCharge
## 1      XXXConfid
## 2      XXXConfid
## 3      XXXConfid
## 4      XXXConfid
## 5      XXXConfid
## 6      XXXConfid
```

```r
# Get the structure of the data
str(train)
```

```
## 'data.frame':    1504 obs. of  35 variables:
##  $ PatientID                : int  1 2 3 4 5 6 7 8 9 10 ...
##  $ Age                      : int  67 65 62 67 65 88 83 72 90 61 ...
##  $ Gender                   : int  0 1 0 0 1 1 1 1 1 1 ...
##  $ Ethnicity                : int  3 0 1 0 0 0 1 1 3 0 ...
##  $ EducationLevel           : int  0 0 1 1 2 1 3 1 2 0 ...
##  $ BMI                      : num  37.2 35.1 17.9 37.5 29.2 ...
##  $ Smoking                  : int  0 1 0 1 1 0 0 0 0 0 ...
##  $ AlcoholConsumption       : num  12.216 17.111 13.526 19.952 0.533 ...
##  $ PhysicalActivity         : num  7.78 6.65 9.59 1.95 8.76 ...
##  $ DietQuality              : num  6.43 1.11 4.27 6.8 6.36 ...
##  $ SleepQuality             : num  6.74 7.57 8.25 7.67 6.23 ...
##  $ FamilyHistoryAlzheimers  : int  0 0 0 0 0 0 0 1 1 0 ...
##  $ CardiovascularDisease    : int  0 0 0 1 1 0 0 0 0 0 ...
##  $ Diabetes                 : int  0 0 1 1 0 0 0 0 0 0 ...
##  $ Depression               : int  0 1 0 1 0 0 0 1 0 0 ...
##  $ HeadInjury               : int  0 0 0 0 0 1 0 0 0 0 ...
##  $ Hypertension             : int  0 0 0 0 0 0 0 0 0 0 ...
##  $ SystolicBP               : int  137 111 131 121 158 126 165 117 115 126 ...
##  $ DiastolicBP              : int  114 82 108 76 117 83 91 102 90 70 ...
##  $ CholesterolTotal         : num  270 227 202 236 292 ...
##  $ CholesterolLDL           : num  119 101 185 151 125 ...
##  $ CholesterolHDL           : num  78 21.2 37 62.2 82.9 ...
##  $ CholesterolTriglycerides : num  273 157 289 196 296 ...
##  $ MMSE                     : num  0.695 23.79 6.592 25.343 6.628 ...
##  $ FunctionalAssessment     : num  9.99 6.2 9.57 2.49 7.52 ...
##  $ MemoryComplaints         : int  1 0 0 0 1 0 0 0 0 0 ...
##  $ BehavioralProblems       : int  0 0 0 0 0 0 0 0 0 0 ...
##  $ ADL                      : num  6.01 7.52 8.57 6.22 5.19 ...
##  $ Confusion                : int  0 0 0 0 1 0 0 0 0 1 ...
##  $ Disorientation           : int  0 0 0 0 0 1 0 0 1 0 ...
##  $ PersonalityChanges       : int  0 0 0 0 0 1 0 0 0 0 ...
##  $ DifficultyCompletingTasks: int  1 0 0 0 0 0 0 0 0 1 ...
##  $ Forgetfulness            : int  1 1 0 1 1 0 0 0 0 1 ...
##  $ Diagnosis                : int  0 0 0 0 0 0 0 0 0 0 ...
##  $ DoctorInCharge           : chr  "XXXConfid" "XXXConfid" "XXXConfid" "XXXConfid" ...
```

```r
# Summary statistics
summary(train)
```

```
##    PatientID           Age            Gender         Ethnicity
##  Min.   :   1.0   Min.   :60.00   Min.   :0.0000   Min.   :0.0000
##  1st Qu.: 376.8   1st Qu.:67.00   1st Qu.:0.0000   1st Qu.:0.0000
##  Median : 752.5   Median :75.00   Median :1.0000   Median :0.0000
##  Mean   : 752.5   Mean   :74.91   Mean   :0.5086   Mean   :0.7114
##  3rd Qu.:1128.2   3rd Qu.:83.00   3rd Qu.:1.0000   3rd Qu.:1.0000
##  Max.   :1504.0   Max.   :90.00   Max.   :1.0000   Max.   :3.0000
##  EducationLevel       BMI           Smoking       AlcoholConsumption
##  Min.   :0.000   Min.   :15.01   Min.   :0.0000   Min.   : 0.002003
##  1st Qu.:1.000   1st Qu.:21.37   1st Qu.:0.0000   1st Qu.: 5.204286
##  Median :1.000   Median :27.76   Median :0.0000   Median : 9.924320
##  Mean   :1.296   Mean   :27.55   Mean   :0.2839   Mean   :10.030205
##  3rd Qu.:2.000   3rd Qu.:33.78   3rd Qu.:1.0000   3rd Qu.:15.140505
##  Max.   :3.000   Max.   :39.93   Max.   :1.0000   Max.   :19.988291
##  PhysicalActivity    DietQuality        SleepQuality    FamilyHistoryAlzheimers
```

```
##  Min.   :0.003616   Min.   :0.009385   Min.   : 4.003   Min.   :0.0000
##  1st Qu.:2.538671   1st Qu.:2.302514   1st Qu.: 5.480   1st Qu.:0.0000
##  Median :4.790574   Median :4.979274   Median : 7.100   Median :0.0000
##  Mean   :4.914426   Mean   :4.937305   Mean   : 7.042   Mean   :0.2447
##  3rd Qu.:7.452197   3rd Qu.:7.576618   3rd Qu.: 8.550   3rd Qu.:0.0000
##  Max.   :9.987429   Max.   :9.998346   Max.   :10.000   Max.   :1.0000
##  CardiovascularDisease    Diabetes        Depression       HeadInjury
##  Min.   :0.0000        Min.   :0.0000   Min.   :0.0000   Min.   :0.00000
##  1st Qu.:0.0000        1st Qu.:0.0000   1st Qu.:0.0000   1st Qu.:0.00000
##  Median :0.0000        Median :0.0000   Median :0.0000   Median :0.00000
##  Mean   :0.1343        Mean   :0.1596   Mean   :0.2081   Mean   :0.09508
##  3rd Qu.:0.0000        3rd Qu.:0.0000   3rd Qu.:0.0000   3rd Qu.:0.00000
##  Max.   :1.0000        Max.   :1.0000   Max.   :1.0000   Max.   :1.00000
##   Hypertension      SystolicBP     DiastolicBP     CholesterolTotal
##  Min.   :0.0000   Min.   : 90.0   Min.   : 60.00   Min.   :150.1
##  1st Qu.:0.0000   1st Qu.:112.0   1st Qu.: 74.00   1st Qu.:190.5
##  Median :0.0000   Median :135.0   Median : 90.00   Median :224.4
##  Mean   :0.1516   Mean   :134.7   Mean   : 89.71   Mean   :225.2
##  3rd Qu.:0.0000   3rd Qu.:156.0   3rd Qu.:105.00   3rd Qu.:262.5
##  Max.   :1.0000   Max.   :179.0   Max.   :119.00   Max.   :300.0
##  CholesterolLDL   CholesterolHDL  CholesterolTriglycerides      MMSE
##  Min.   : 50.40   Min.   :20.00   Min.   : 50.41           Min.   : 0.0353
##  1st Qu.: 87.52   1st Qu.:39.15   1st Qu.:136.31           1st Qu.: 7.1155
##  Median :124.52   Median :59.59   Median :229.55           Median :14.3225
##  Mean   :124.88   Mean   :59.51   Mean   :226.90           Mean   :14.6491
##  3rd Qu.:161.96   3rd Qu.:78.91   3rd Qu.:313.06           3rd Qu.:21.8386
##  Max.   :199.97   Max.   :99.98   Max.   :399.94           Max.   :29.9914
##  FunctionalAssessment MemoryComplaints BehavioralProblems      ADL
##  Min.   :0.00046      Min.   :0.0000   Min.   :0.0000     Min.   :0.004354
##  1st Qu.:2.65883      1st Qu.:0.0000   1st Qu.:0.0000     1st Qu.:2.358590
##  Median :5.19113      Median :0.0000   Median :0.0000     Median :4.877862
##  Mean   :5.13989      Mean   :0.2055   Mean   :0.1516     Mean   :4.903536
##  3rd Qu.:7.61636      3rd Qu.:0.0000   3rd Qu.:0.0000     3rd Qu.:7.517219
##  Max.   :9.99647      Max.   :1.0000   Max.   :1.0000     Max.   :9.972663
##    Confusion      Disorientation   PersonalityChanges DifficultyCompletingTasks
##  Min.   :0.0000   Min.   :0.0000   Min.   :0.0000     Min.   :0.0000
##  1st Qu.:0.0000   1st Qu.:0.0000   1st Qu.:0.0000     1st Qu.:0.0000
##  Median :0.0000   Median :0.0000   Median :0.0000     Median :0.0000
##  Mean   :0.2028   Mean   :0.1562   Mean   :0.1569     Mean   :0.1622
##  3rd Qu.:0.0000   3rd Qu.:0.0000   3rd Qu.:0.0000     3rd Qu.:0.0000
##  Max.   :1.0000   Max.   :1.0000   Max.   :1.0000     Max.   :1.0000
##  Forgetfulness      Diagnosis      DoctorInCharge
##  Min.   :0.0000   Min.   :0.0000   Length:1504
##  1st Qu.:0.0000   1st Qu.:0.0000   Class :character
##  Median :0.0000   Median :0.0000   Mode  :character
##  Mean   :0.2999   Mean   :0.3537
##  3rd Qu.:1.0000   3rd Qu.:1.0000
##  Max.   :1.0000   Max.   :1.0000
```

1.2.2. Check for Missing Values

```
# Check for missing values in training data
sapply(train, function(x) sum(is.na(x)))
```

```
##                PatientID                      Age                   Gender
```

```
##                      0                      0                      0
##              Ethnicity        EducationLevel                    BMI
##                      0                      0                      0
##                Smoking     AlcoholConsumption      PhysicalActivity
##                      0                      0                      0
##            DietQuality          SleepQuality FamilyHistoryAlzheimers
##                      0                      0                      0
##  CardiovascularDisease               Diabetes             Depression
##                      0                      0                      0
##            HeadInjury           Hypertension             SystolicBP
##                      0                      0                      0
##           DiastolicBP      CholesterolTotal        CholesterolLDL
##                      0                      0                      0
##        CholesterolHDL CholesterolTriglycerides                   MMSE
##                      0                      0                      0
##   FunctionalAssessment       MemoryComplaints    BehavioralProblems
##                      0                      0                      0
##                   ADL              Confusion        Disorientation
##                      0                      0                      0
##     PersonalityChanges DifficultyCompletingTasks        Forgetfulness
##                      0                      0                      0
##              Diagnosis        DoctorInCharge
##                      0                      0
```

```r
# Load the Amelia library
library(Amelia)

# Visualize missing data
missmap(train, main = "Missing values in Training Data", col = c("yellow", "black"), legend = FALSE)
```

1.3. Data Cleaning

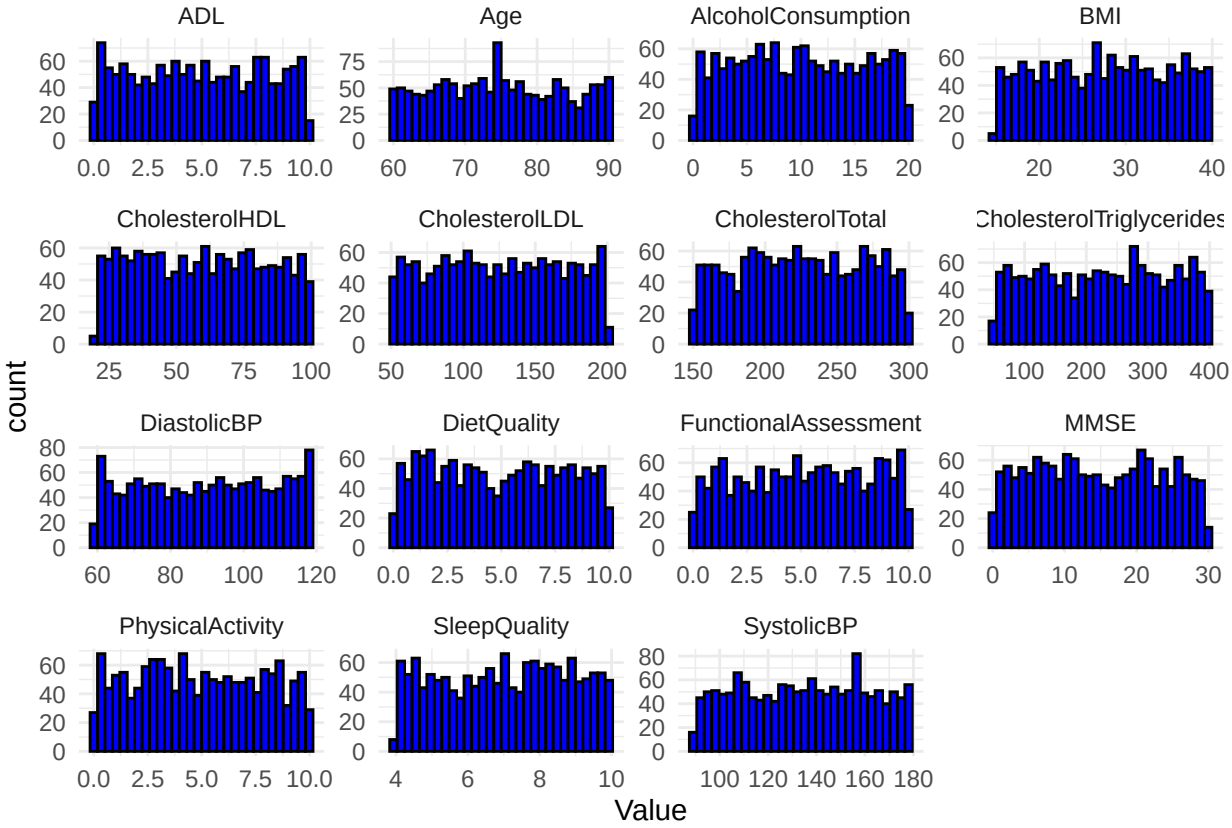
```
library(dplyr)
# Remove 'PatientID' and 'DoctorInCharge' from train
train <- train %>% select(-PatientID, -DoctorInCharge)
# Identify categorical variables
categorical_vars <- c('Gender', 'Ethnicity', 'EducationLevel', 'Smoking', 'MemoryComplaints',
                      'BehavioralProblems', 'FamilyHistoryAlzheimers', 'CardiovascularDisease',
                      'Diabetes', 'Depression', 'HeadInjury', 'Hypertension', 'Confusion',
                      'Disorientation', 'PersonalityChanges', 'DifficultyCompletingTasks',
                      'Forgetfulness', 'Diagnosis')

# Convert to factors
train[categorical_vars] <- lapply(train[categorical_vars], as.factor)
# For numerical variables, use median imputation
numerical_vars <- setdiff(names(train), categorical_vars)
train[numerical_vars] <- lapply(train[numerical_vars], function(x) ifelse(is.na(x), median(x, na.rm = T
```

1.4. Univariate Analysis

```
# Plot histograms for numerical variables
library(ggplot2)
library(tidyverse)

train %>%
  select(numerical_vars) %>%
  gather(key = "Variable", value = "Value") %>%
  ggplot(aes(x = Value)) +
```

```
  geom_histogram(bins = 30, fill = 'blue', color = 'black') +
  facet_wrap(~ Variable, scales = 'free') +
  theme_minimal()
```

```
## Warning: Using an external vector in selections was deprecated in tidyselect 1.1.0.
## i Please use `all_of()` or `any_of()` instead.
##   # Was:
##   data %>% select(numerical_vars)
##
##   # Now:
##   data %>% select(all_of(numerical_vars))
##
## See <https://tidyselect.r-lib.org/reference/faq-external-vector.html>.
## This warning is displayed once every 8 hours.
## Call `lifecycle::last_lifecycle_warnings()` to see where this warning was
## generated.
```

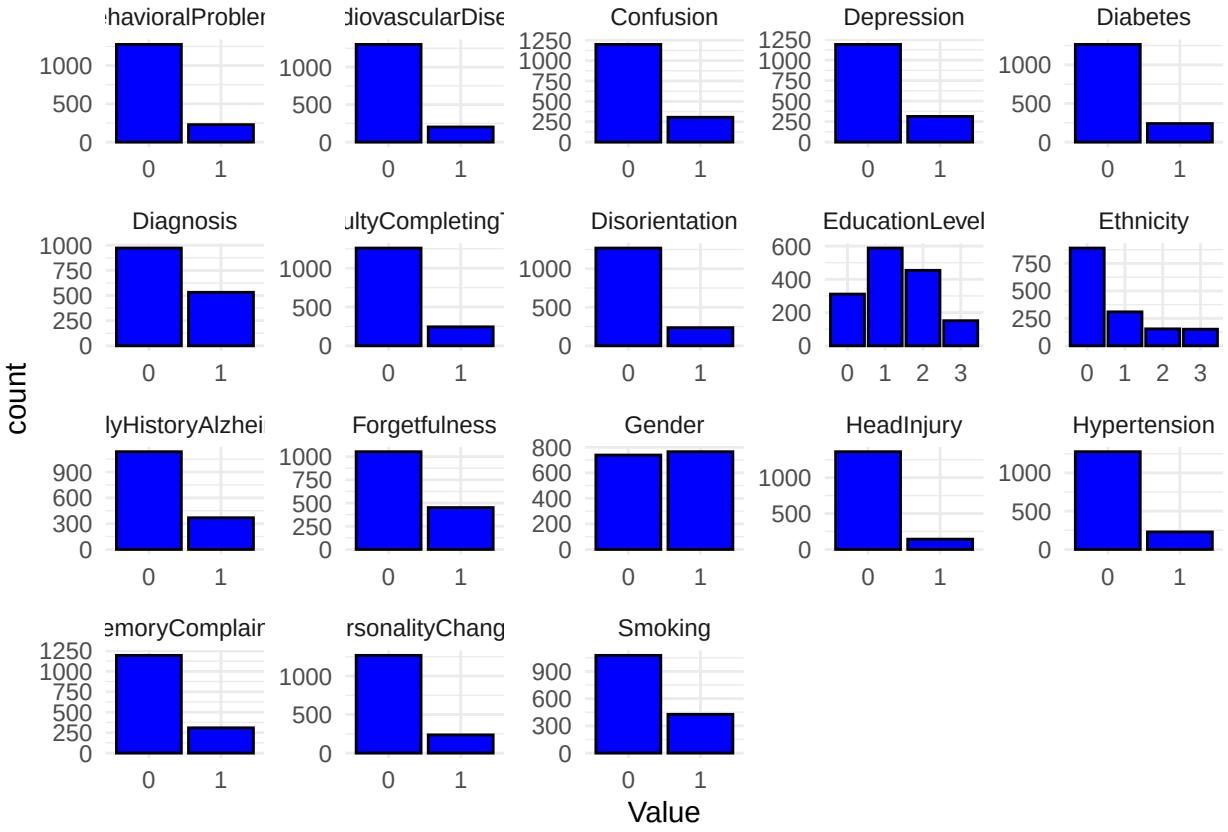



```
# Plot bar charts for categorical variables
train %>%
  select(categorical_vars) %>%
  gather(key = "Variable", value = "Value") %>%
  ggplot(aes(x = Value)) +
  geom_bar(fill = 'blue', color = 'black') +
  facet_wrap(~ Variable, scales = 'free') +
  theme_minimal()
```

```
## Warning: Using an external vector in selections was deprecated in tidyselect 1.1.0.
## i Please use `all_of()` or `any_of()` instead.
```

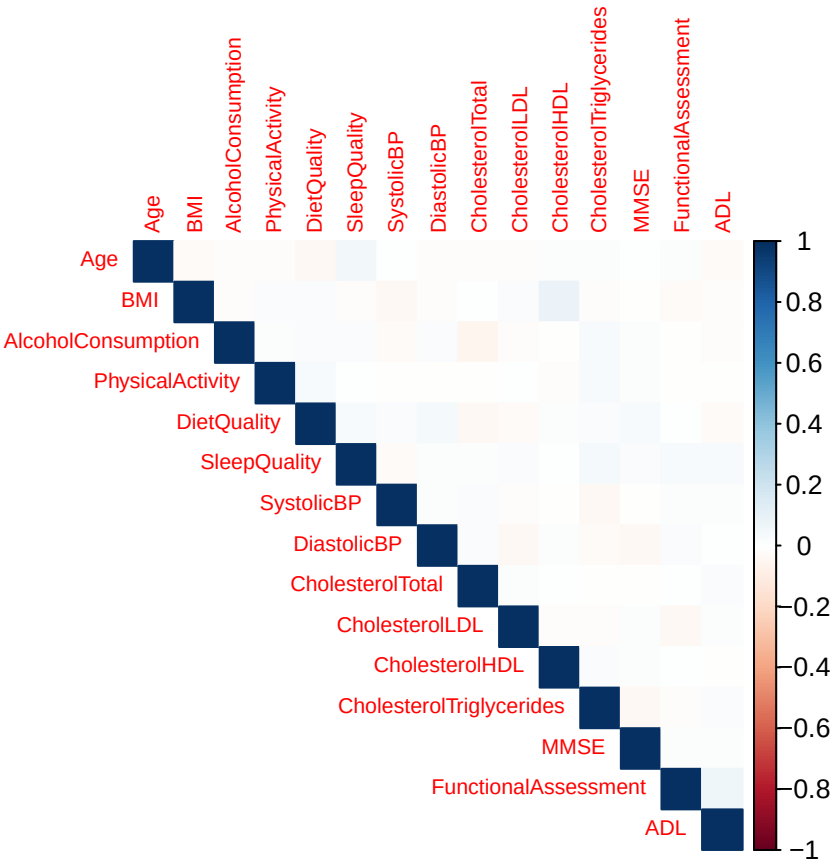
```
##   # Was:
##   data %>% select(categorical_vars)
##
##   # Now:
##   data %>% select(all_of(categorical_vars))
##
## See <https://tidyselect.r-lib.org/reference/faq-external-vector.html>.
## This warning is displayed once every 8 hours.
## Call `lifecycle::last_lifecycle_warnings()` to see where this warning was
## generated.
```

```
## Warning: attributes are not identical across measure variables; they will be
## dropped
```

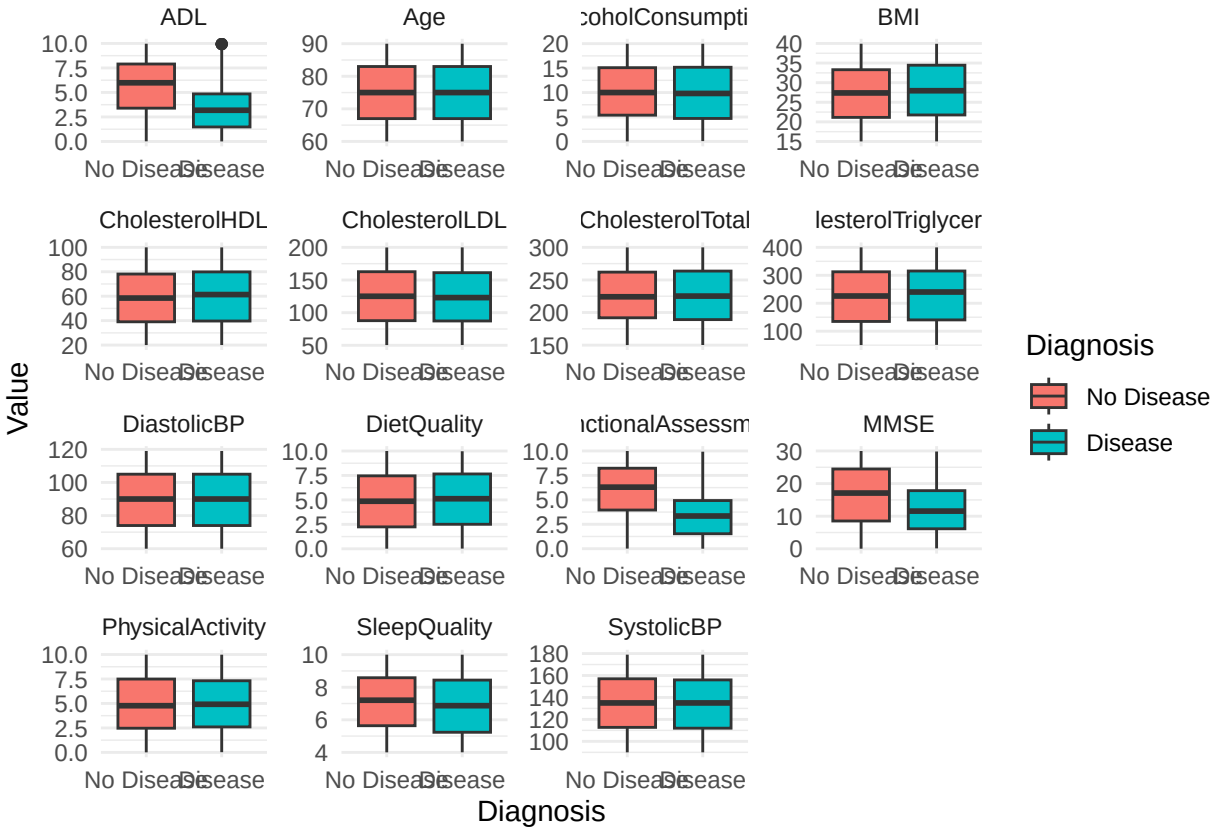


1.5. Bivariate Analysis

```r
# Calculate correlation matrix
corr_matrix <- cor(train %>% select(numerical_vars) %>% mutate_all(as.numeric))

# Plot correlation matrix
corrplot(corr_matrix, method = 'color', type = 'upper', tl.cex = 0.7)
```

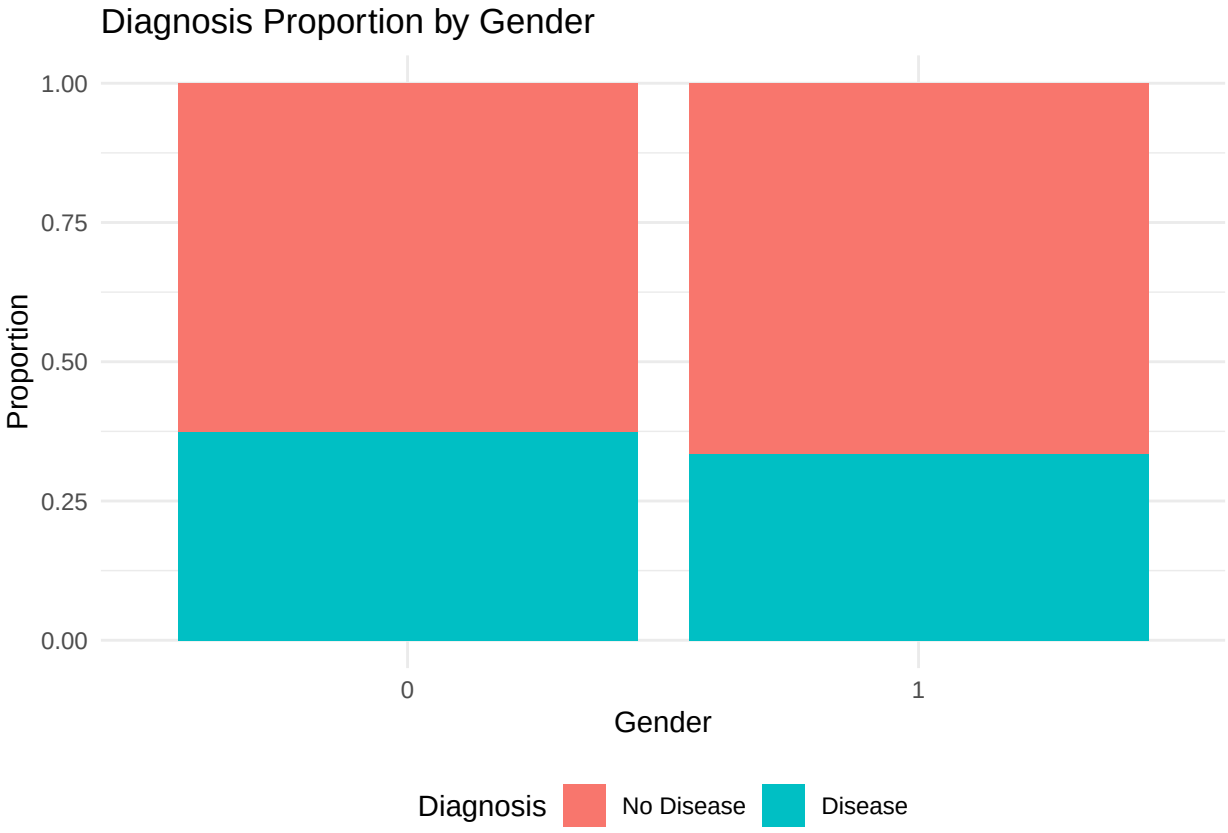



```r
# Boxplots for numerical variables vs Diagnosis
library(tidyr)
library(dplyr)
library(ggplot2)
train %>%
  select(numerical_vars, Diagnosis) %>%
  gather(key = "Variable", value = "Value", -Diagnosis) %>%
  ggplot(aes(x = Diagnosis, y = Value, fill = Diagnosis)) +
  geom_boxplot() +
  facet_wrap(~ Variable, scales = 'free') +
  theme_minimal()+
  scale_x_discrete(labels = c('0' = 'No Disease', '1' = 'Disease'))+
  scale_fill_discrete(labels = c('0' = 'No Disease', '1' = 'Disease'))
```

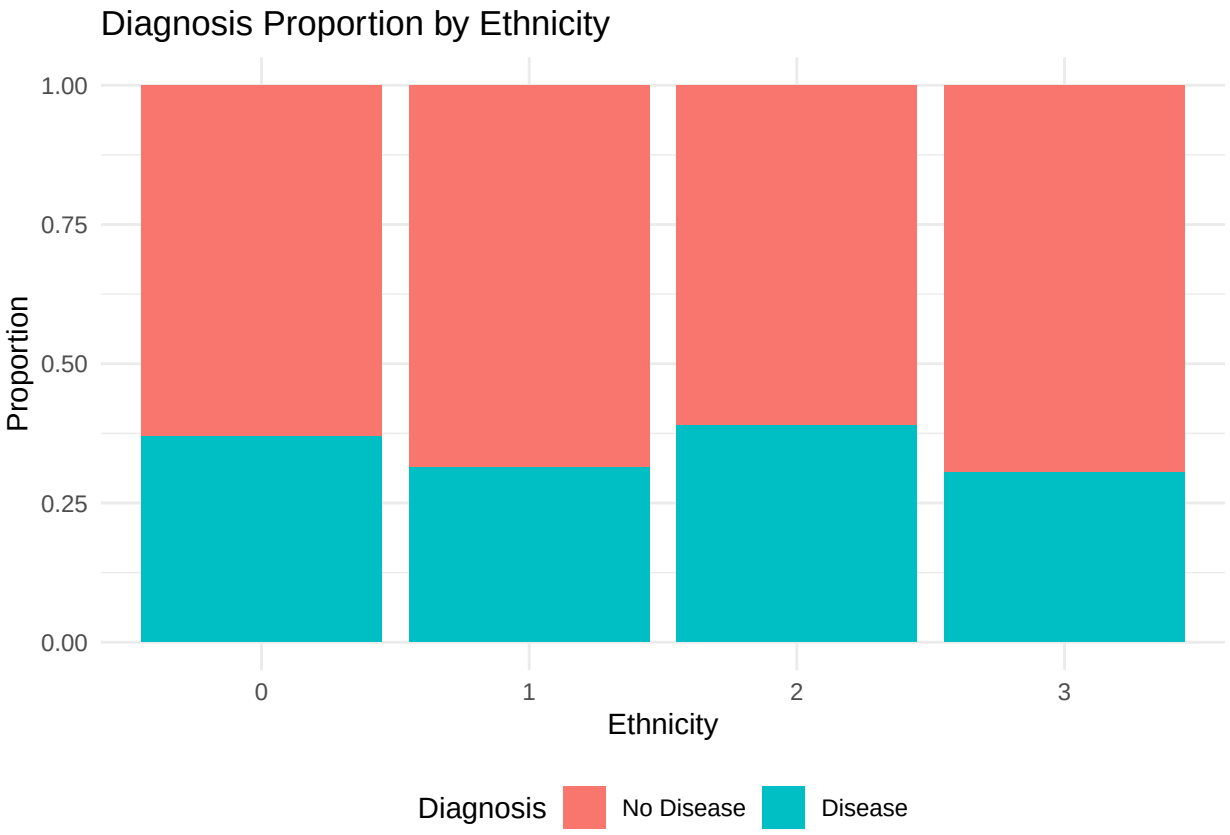



```r
# Load necessary libraries
library(ggplot2)

# Bar charts for categorical variables vs Diagnosis
for (var in setdiff(categorical_vars, 'Diagnosis')) {
  # Create ggplot object
  p <- ggplot(train, aes(x = .data[[var]], fill = Diagnosis)) +
    geom_bar(position = 'fill') +
    labs(y = 'Proportion') +
    theme_minimal() +
    ggtitle(paste('Diagnosis Proportion by', var)) +
    theme(legend.position = 'bottom') +
    scale_fill_brewer(palette = 'Set1')+
    scale_fill_discrete(labels = c('0' = 'No Disease', '1' = 'Disease'))

  # Print the plot
  print(p)
}
```

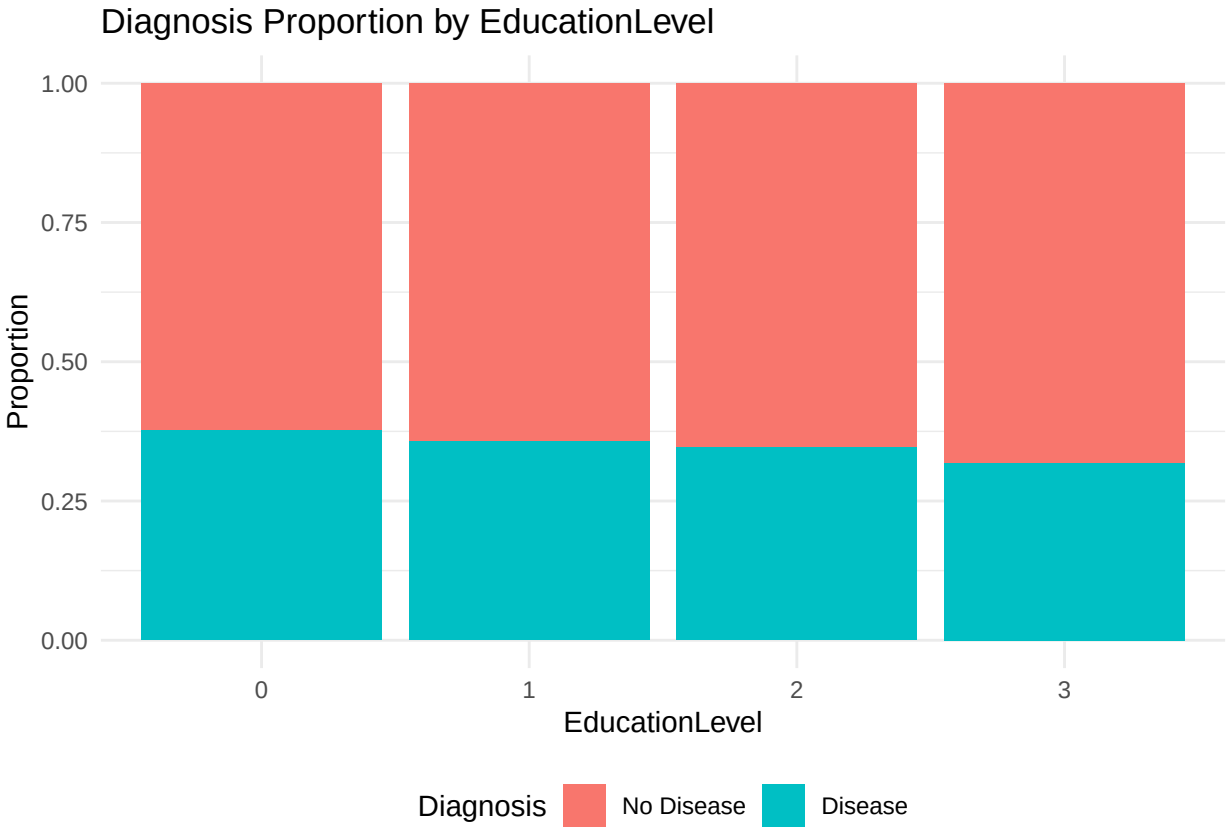
```
## Scale for fill is already present.
## Adding another scale for fill, which will replace the existing scale.
## Scale for fill is already present.
## Adding another scale for fill, which will replace the existing scale.
```



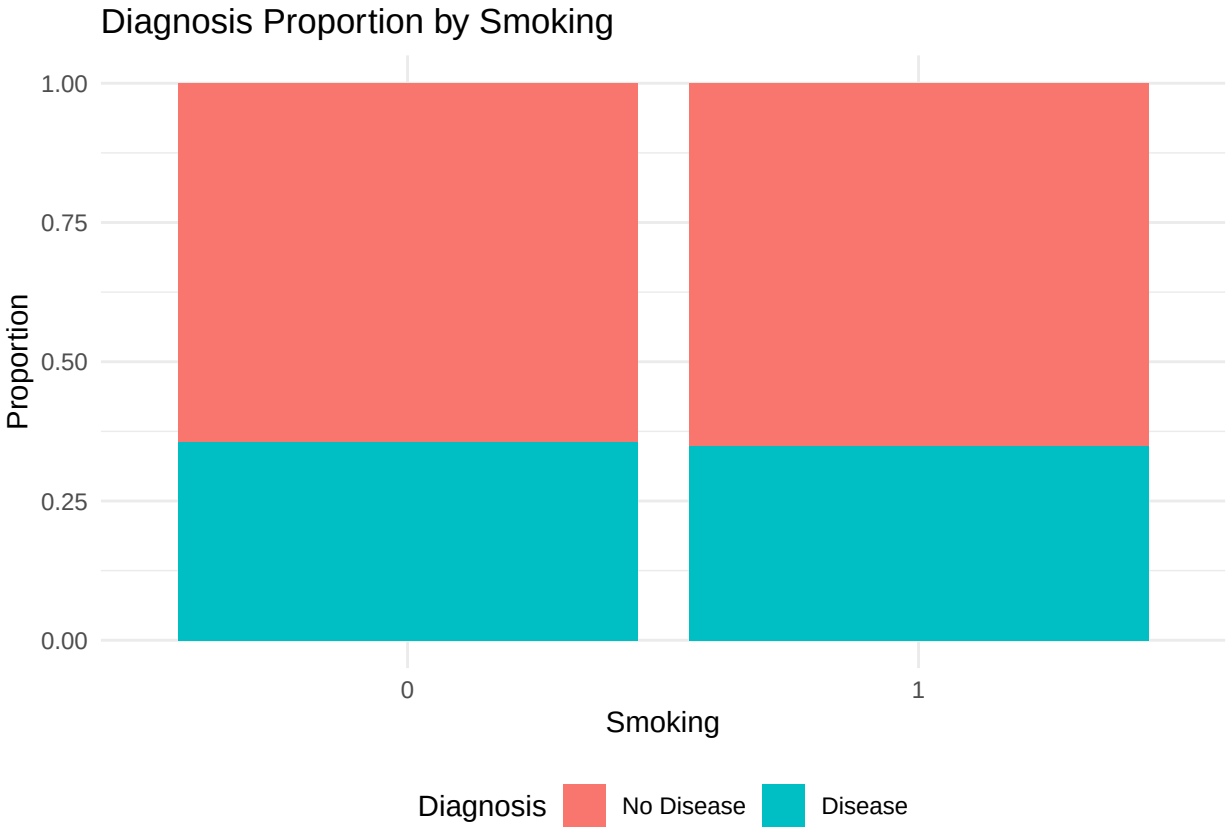

```
## Scale for fill is already present.
## Adding another scale for fill, which will replace the existing scale.
```



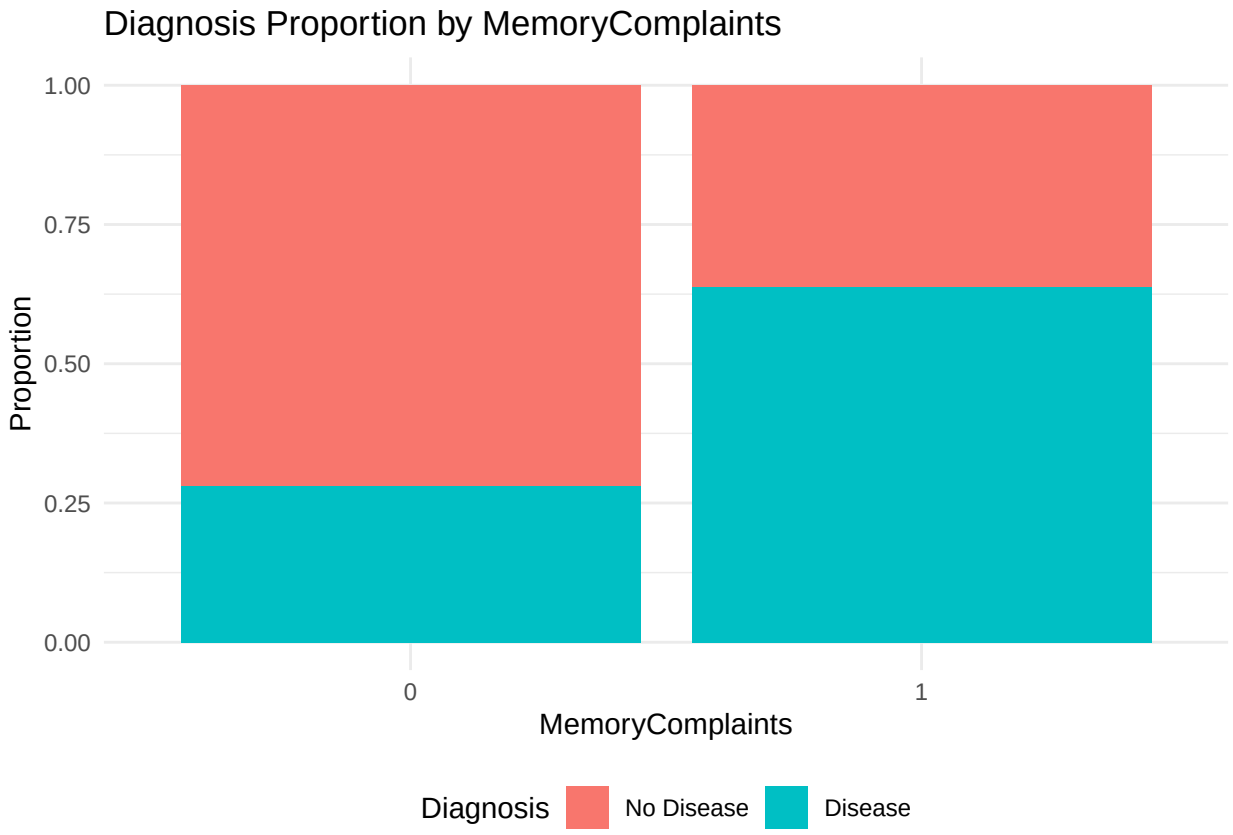

```
## Scale for fill is already present.
## Adding another scale for fill, which will replace the existing scale.
```



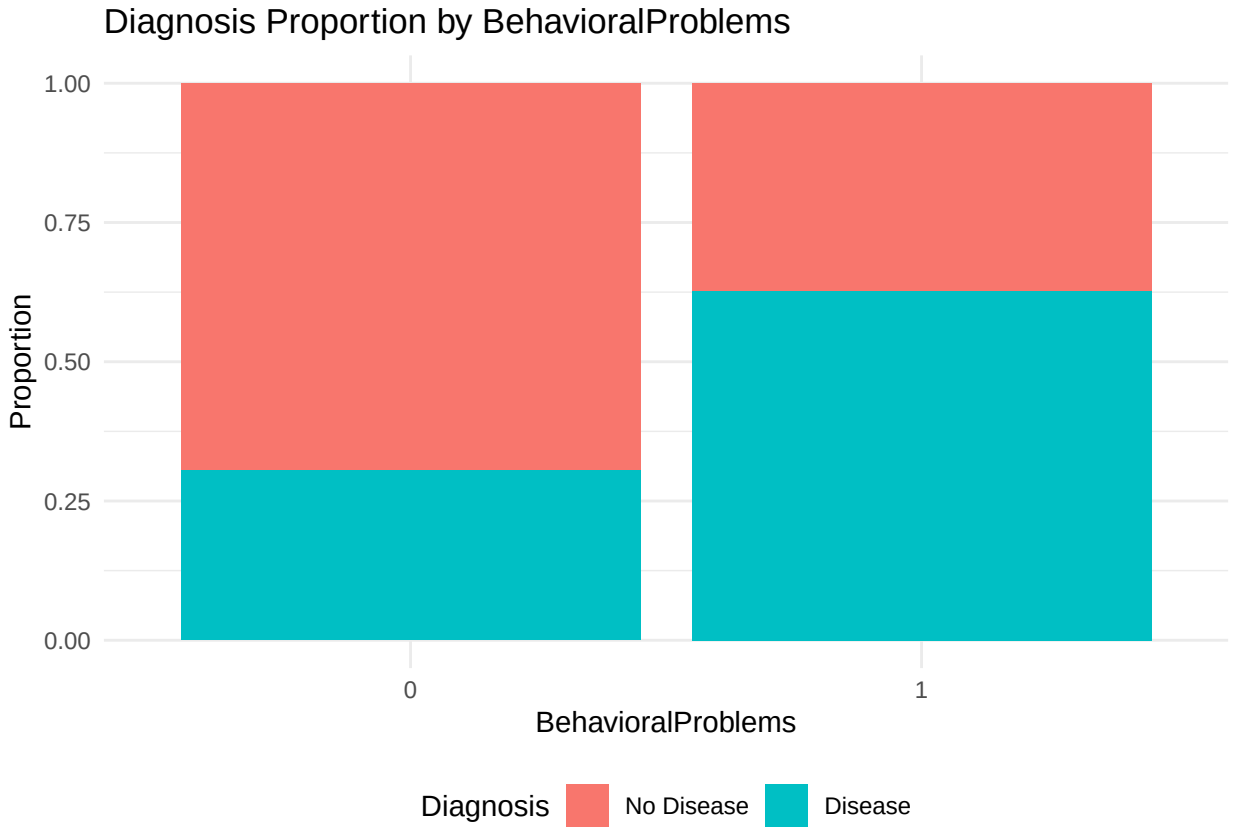

```
## Scale for fill is already present.
## Adding another scale for fill, which will replace the existing scale.
```



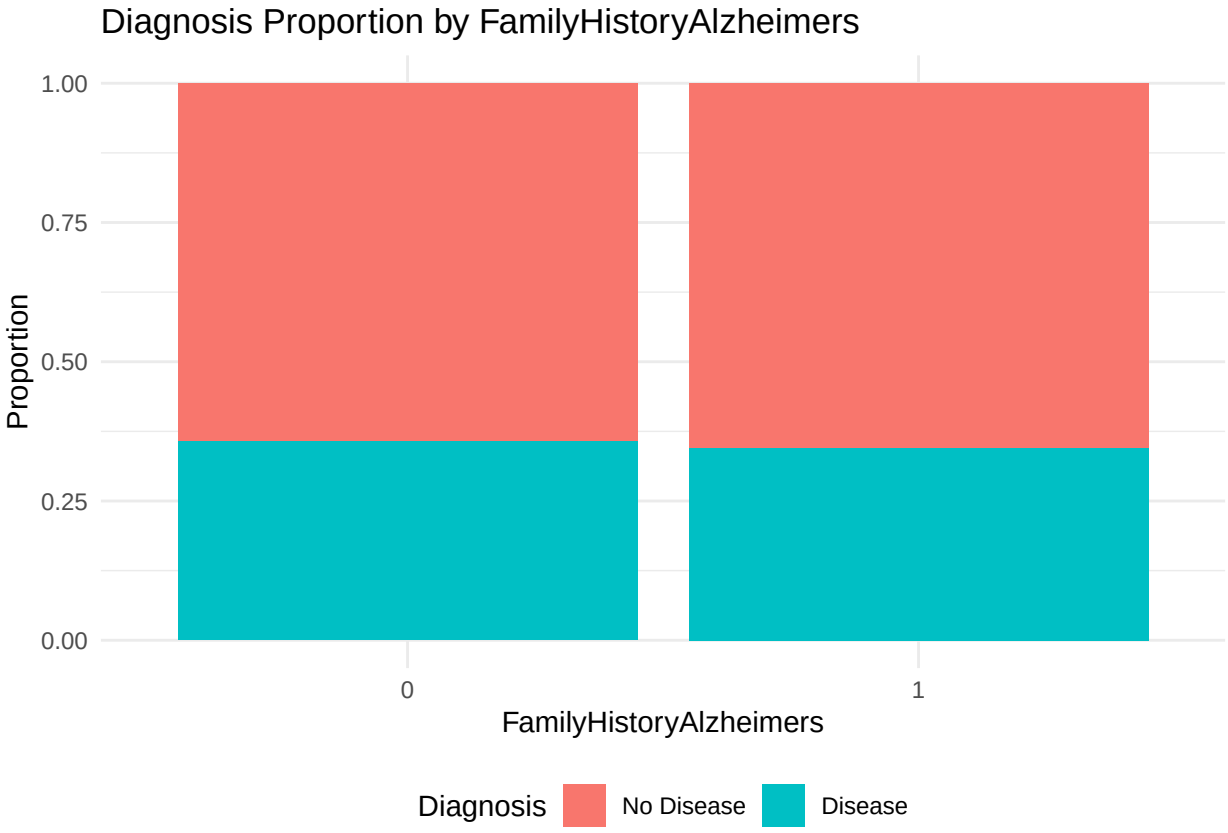

```
## Scale for fill is already present.
## Adding another scale for fill, which will replace the existing scale.
```



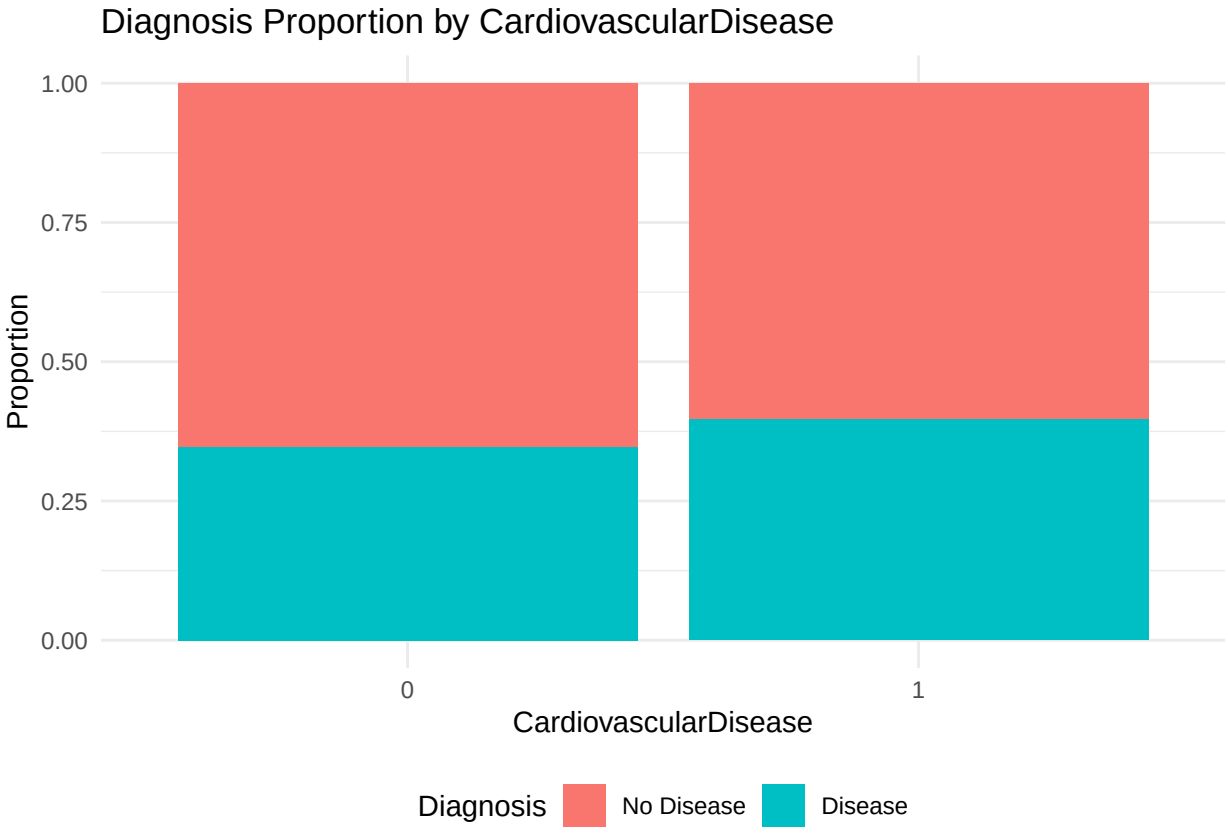

```
## Scale for fill is already present.
## Adding another scale for fill, which will replace the existing scale.
```



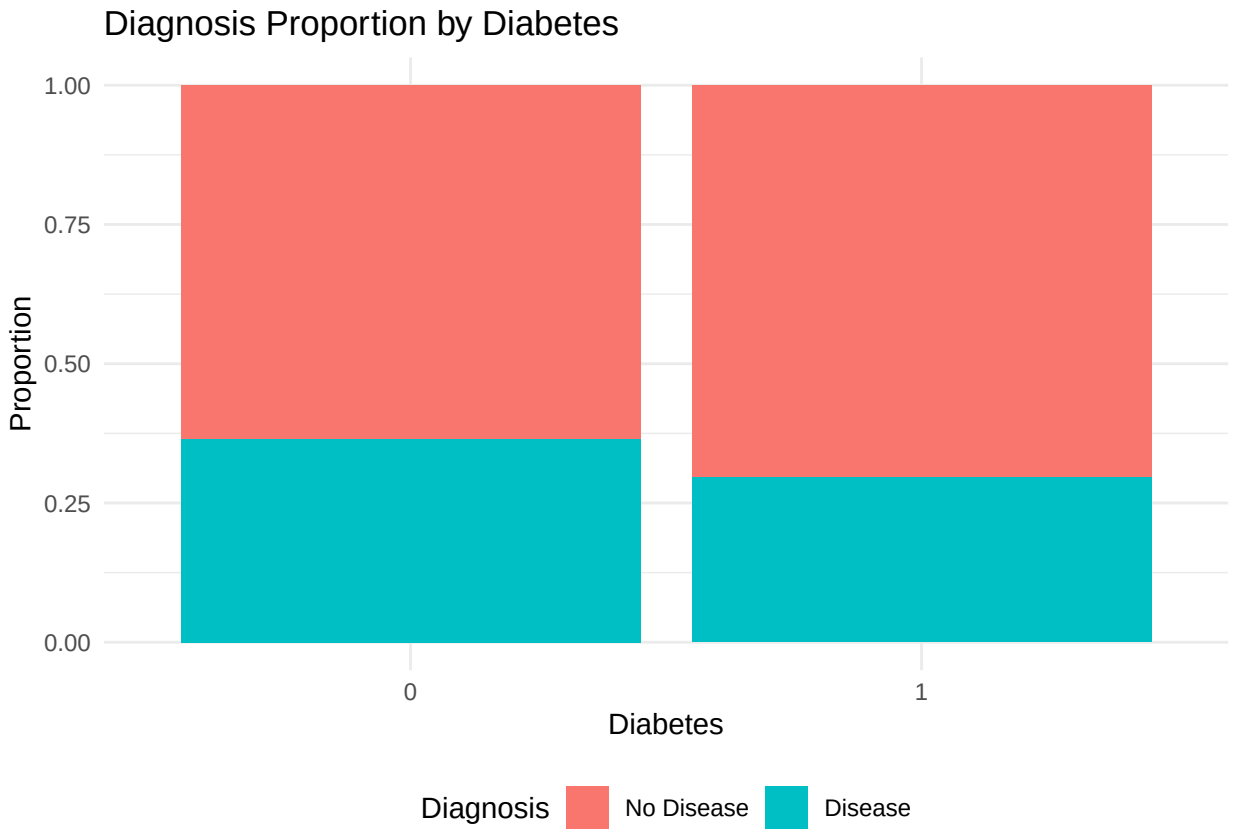

```
## Scale for fill is already present.
## Adding another scale for fill, which will replace the existing scale.
```



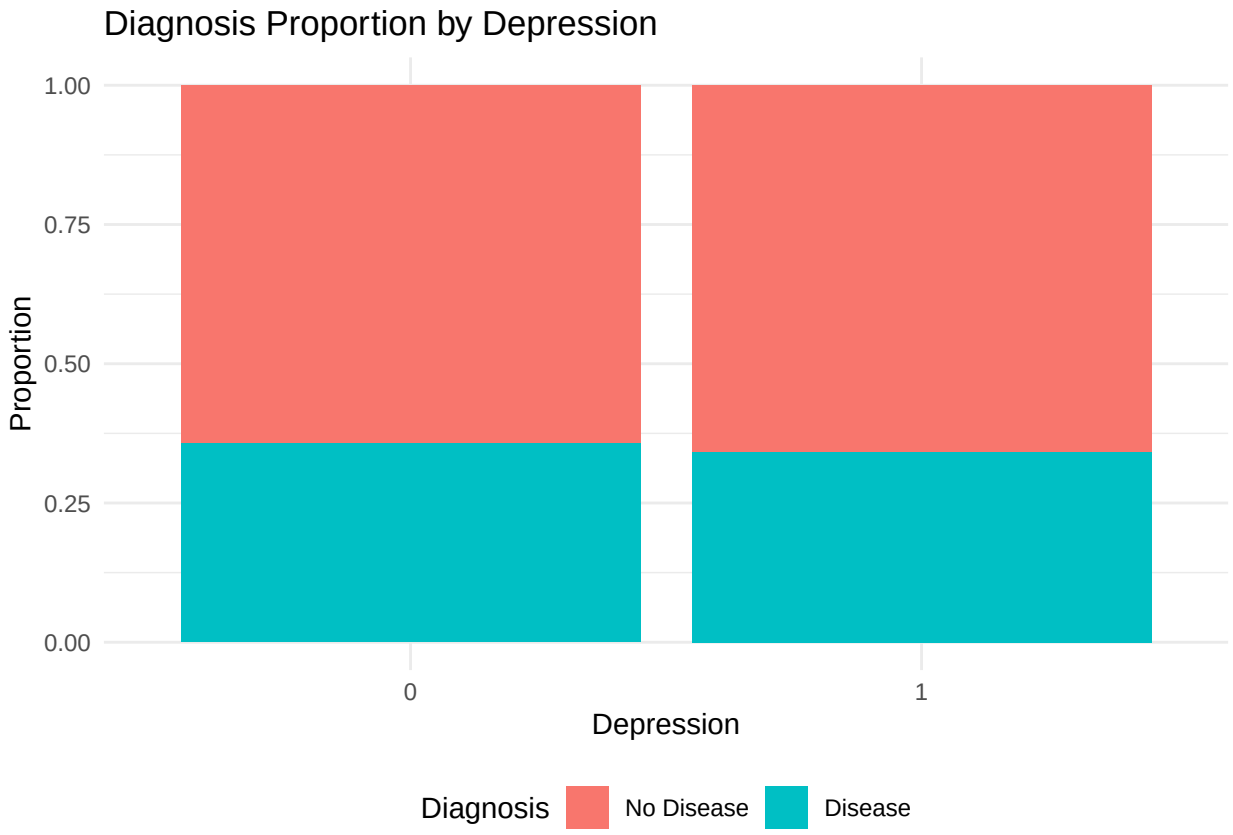

```
## Scale for fill is already present.
## Adding another scale for fill, which will replace the existing scale.
```



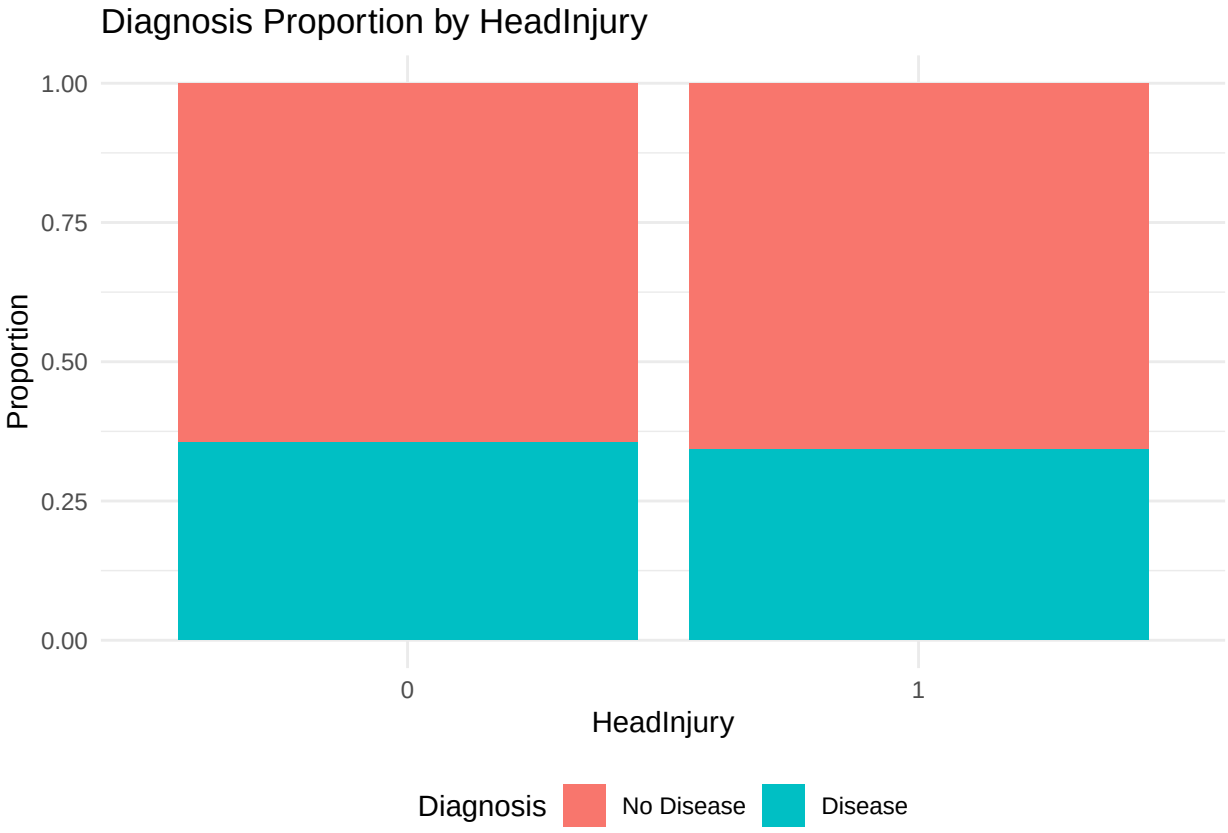

```
## Scale for fill is already present.
## Adding another scale for fill, which will replace the existing scale.
```



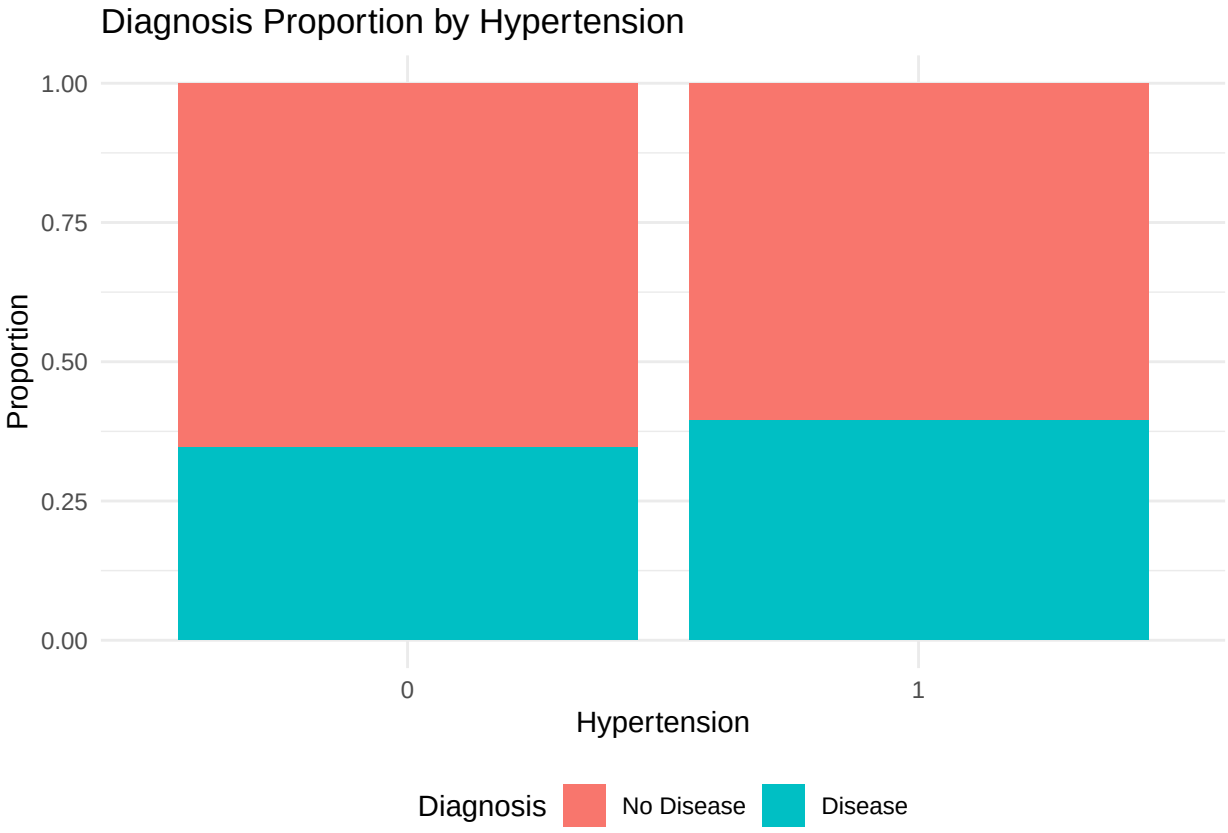

```
## Scale for fill is already present.
## Adding another scale for fill, which will replace the existing scale.
```



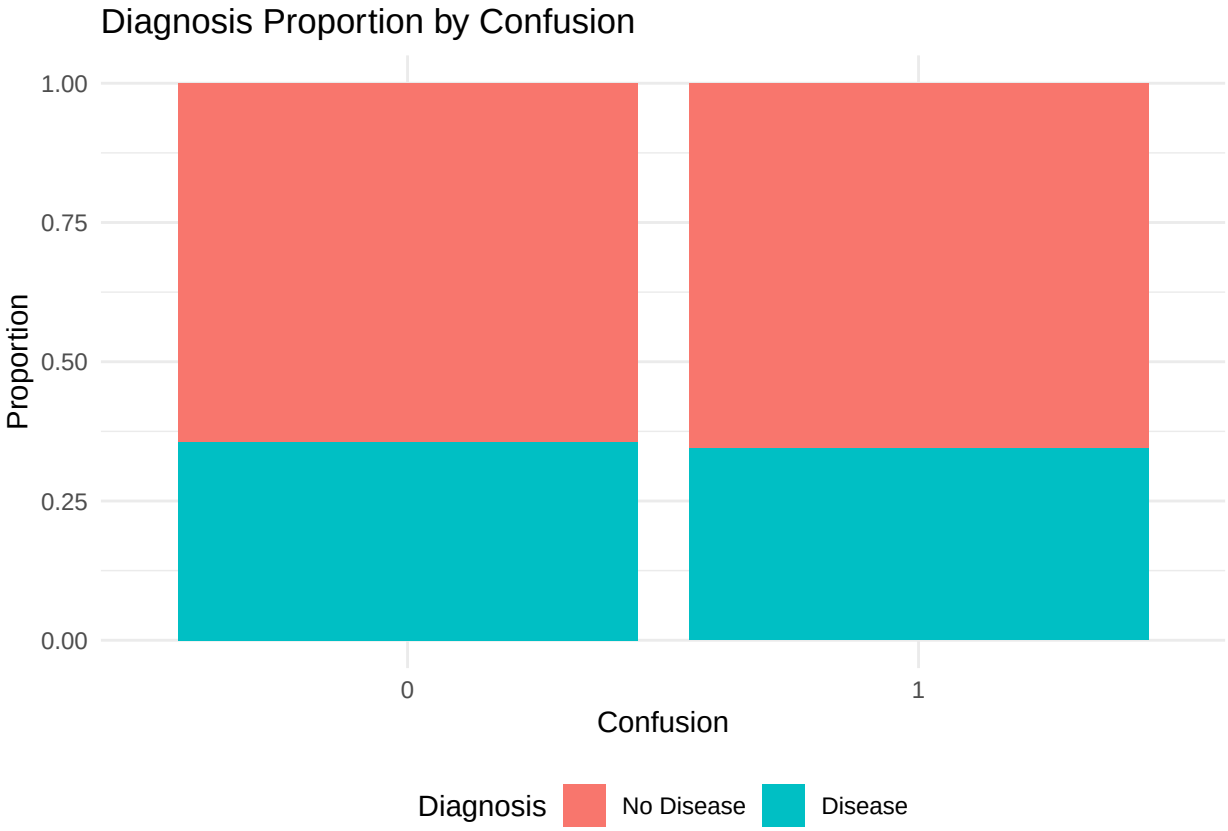

```
## Scale for fill is already present.
## Adding another scale for fill, which will replace the existing scale.
```



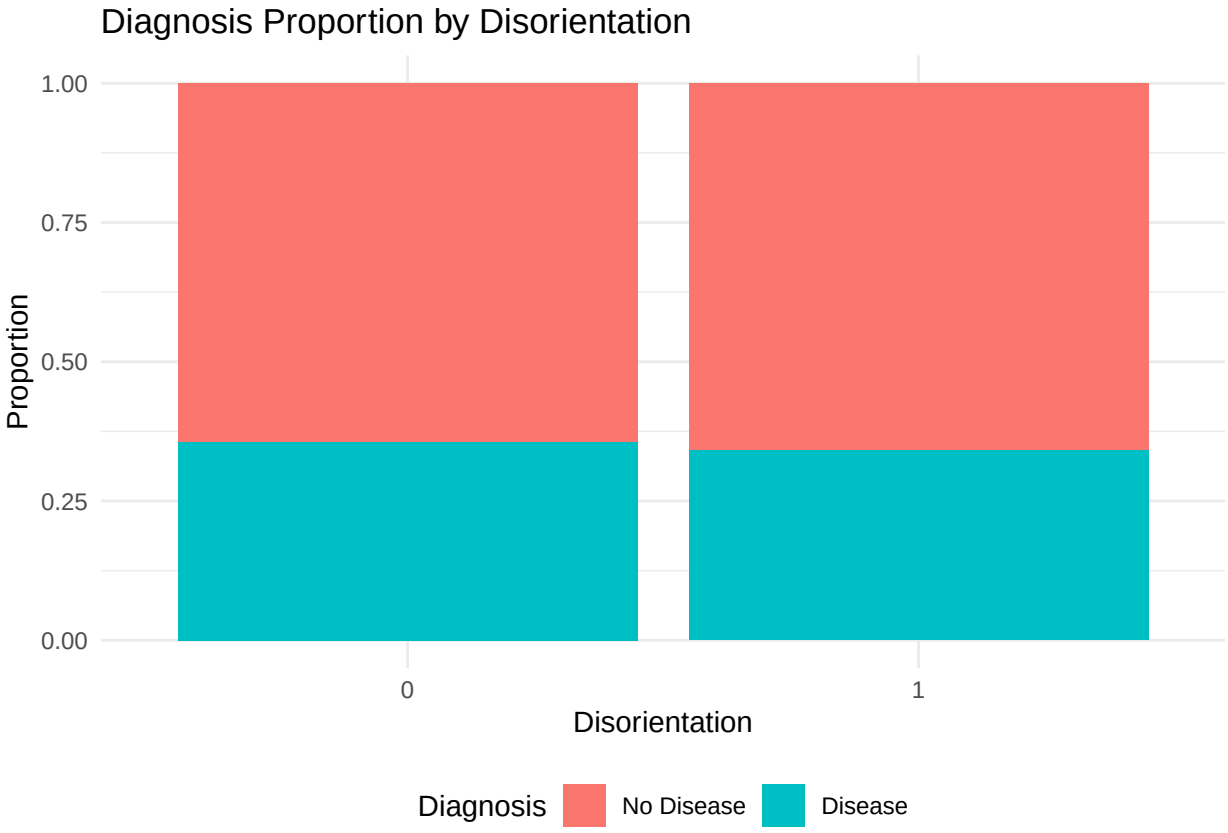

```
## Scale for fill is already present.
## Adding another scale for fill, which will replace the existing scale.
```



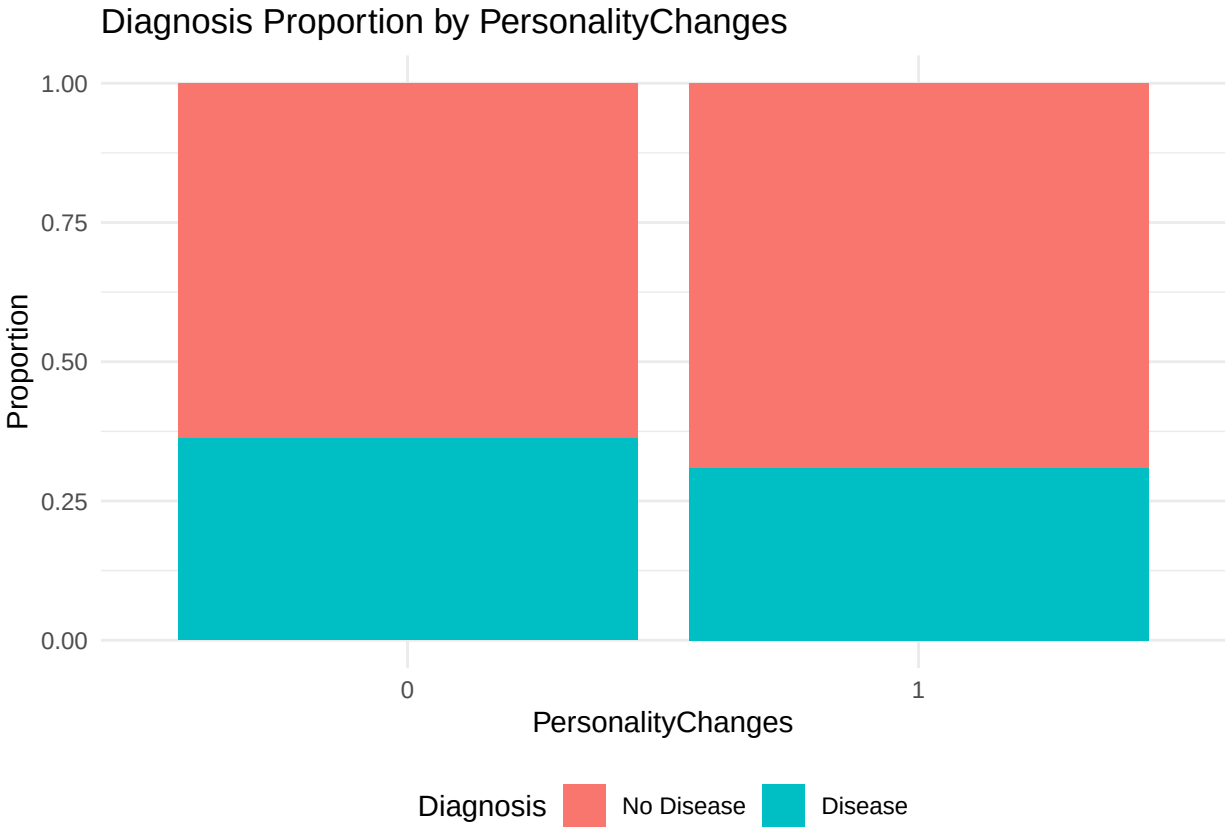

```
## Scale for fill is already present.
## Adding another scale for fill, which will replace the existing scale.
```



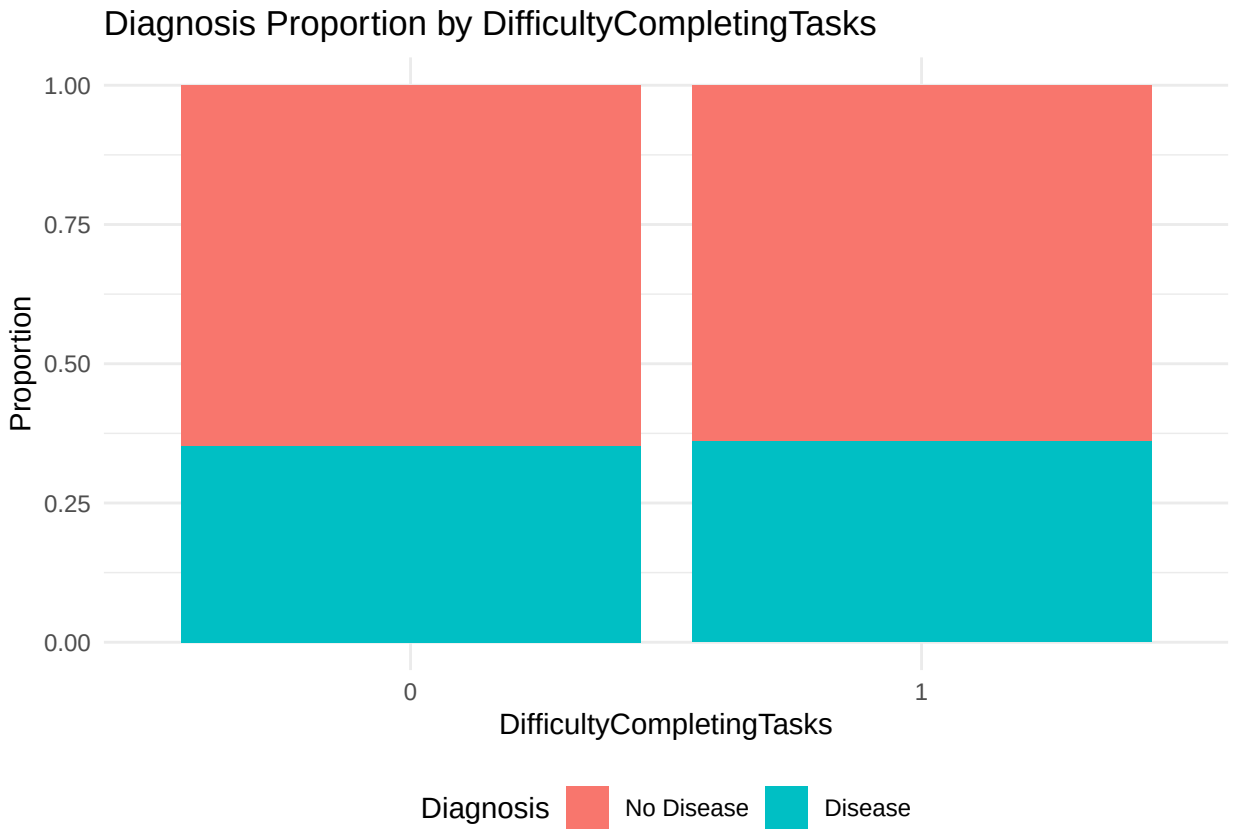

```
## Scale for fill is already present.
## Adding another scale for fill, which will replace the existing scale.
```

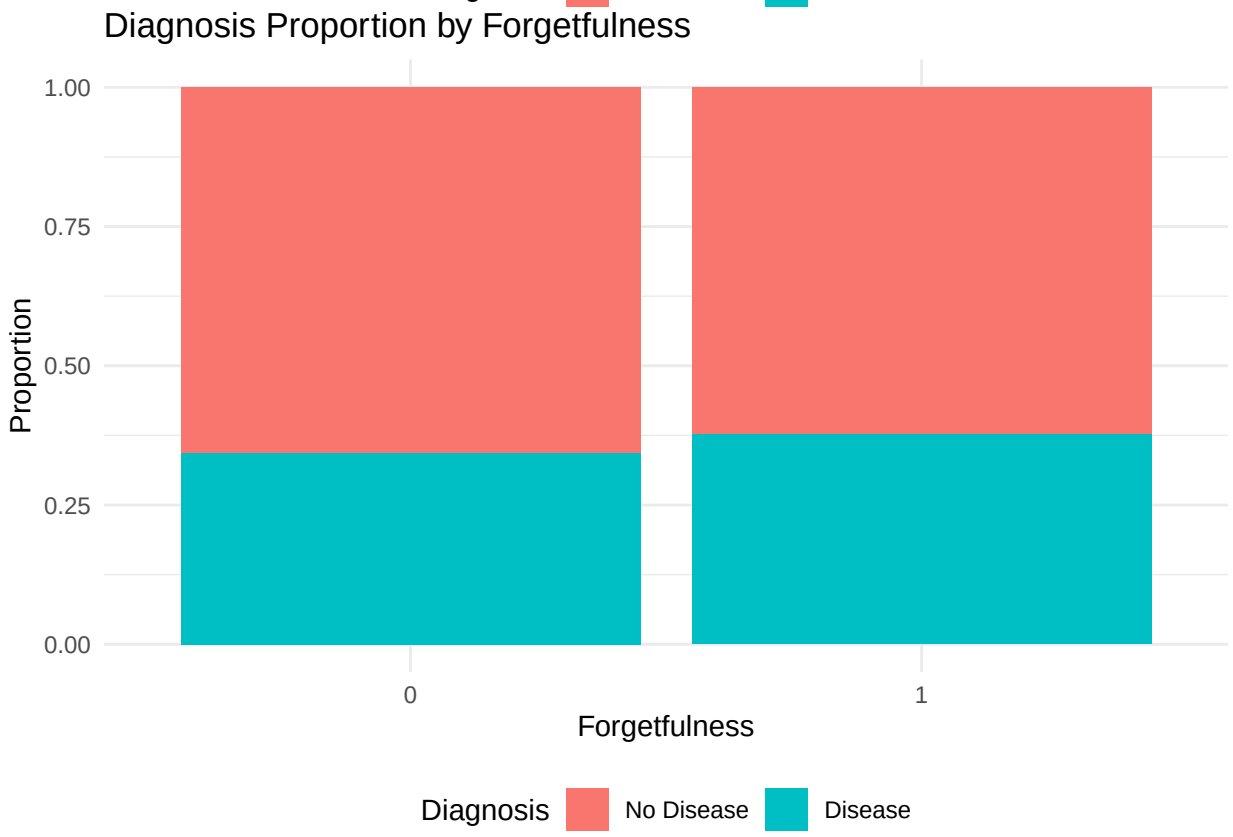



```
## Scale for fill is already present.
## Adding another scale for fill, which will replace the existing scale.
```




```
## Scale for fill is already present.
## Adding another scale for fill, which will replace the existing scale.
```





2. Prediction of the Target Variable (Diagnosis)

2.1. Data Preparation

```r
library(caret)
set.seed(123)
train_index <- createDataPartition(train$Diagnosis, p = 0.8, list = FALSE)
train_data <- train[train_index, ]
valid_data <- train[-train_index, ]
```

2.2. Logistic Regression

```r
# Build logistic regression model
glm_model <- glm(Diagnosis ~ ., data = train_data, family = binomial)

# Summary of the model
summary(glm_model)
```

```
##
## Call:
## glm(formula = Diagnosis ~ ., family = binomial, data = train_data)
##
## Coefficients:
##                              Estimate Std. Error z value Pr(>|z|)
## (Intercept)                 3.8921148  1.2854894   3.028  0.00246 **
## Age                        -0.0063797  0.0095330  -0.669  0.50335
## Gender1                    -0.2703788  0.1712112  -1.579  0.11429
## Ethnicity1                 -0.3707940  0.2246670  -1.650  0.09886 .
## Ethnicity2                  0.0396992  0.3019202   0.131  0.89539
## Ethnicity3                 -0.2845124  0.3094524  -0.919  0.35788
## EducationLevel1            -0.1315280  0.2313651  -0.568  0.56970
## EducationLevel2             0.0732989  0.2426823   0.302  0.76262
## EducationLevel3            -0.2520521  0.3291101  -0.766  0.44376
## BMI                         0.0057538  0.0118589   0.485  0.62754
## Smoking1                   -0.1789221  0.1943481  -0.921  0.35725
## AlcoholConsumption         -0.0064046  0.0148680  -0.431  0.66664
## PhysicalActivity           -0.0137382  0.0292531  -0.470  0.63862
## DietQuality                 0.0332433  0.0298383   1.114  0.26523
## SleepQuality               -0.0619417  0.0491758  -1.260  0.20781
## FamilyHistoryAlzheimers1    0.1512638  0.2002817   0.755  0.45010
## CardiovascularDisease1      0.2548065  0.2339869   1.089  0.27616
## Diabetes1                  -0.0271540  0.2364758  -0.115  0.90858
## Depression1                 0.2251362  0.2064860   1.090  0.27557
## HeadInjury1                -0.1827456  0.2878455  -0.635  0.52551
## Hypertension1               0.1965996  0.2373693   0.828  0.40753
## SystolicBP                  0.0021187  0.0033137   0.639  0.52258
## DiastolicBP                 0.0038221  0.0047613   0.803  0.42212
## CholesterolTotal           -0.0001103  0.0019985  -0.055  0.95598
## CholesterolLDL             -0.0002450  0.0020092  -0.122  0.90294
## CholesterolHDL              0.0039594  0.0037458   1.057  0.29050
## CholesterolTriglycerides    0.0006160  0.0008376   0.735  0.46208
## MMSE                       -0.1085405  0.0110849  -9.792  < 2e-16 ***
## FunctionalAssessment       -0.4329772  0.0351008 -12.335  < 2e-16 ***
## MemoryComplaints1           2.4297325  0.2228794  10.902  < 2e-16 ***
## BehavioralProblems1         2.5422929  0.2553676   9.955  < 2e-16 ***
## ADL                        -0.4089033  0.0349830 -11.689  < 2e-16 ***
## Confusion1                 -0.2629195  0.2107773  -1.247  0.21226
```

```
## Disorientation1            -0.0173571  0.2367329  -0.073  0.94155
## PersonalityChanges1        -0.0547501  0.2423071  -0.226  0.82124
## DifficultyCompletingTasks1  0.1356912  0.2346503   0.578  0.56308
## Forgetfulness1             -0.0455041  0.1890322  -0.241  0.80977
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 1564.67  on 1203  degrees of freedom
## Residual deviance:  895.94  on 1167  degrees of freedom
## AIC: 969.94
##
## Number of Fisher Scoring iterations: 6
```

```r
# Predictions on validation set
glm_probs <- predict(glm_model, valid_data, type = 'response')
glm_preds <- ifelse(glm_probs > 0.5, 1, 0)

# Confusion matrix
glm_cm <- confusionMatrix(as.factor(glm_preds), valid_data$Diagnosis)
print(glm_cm)
```

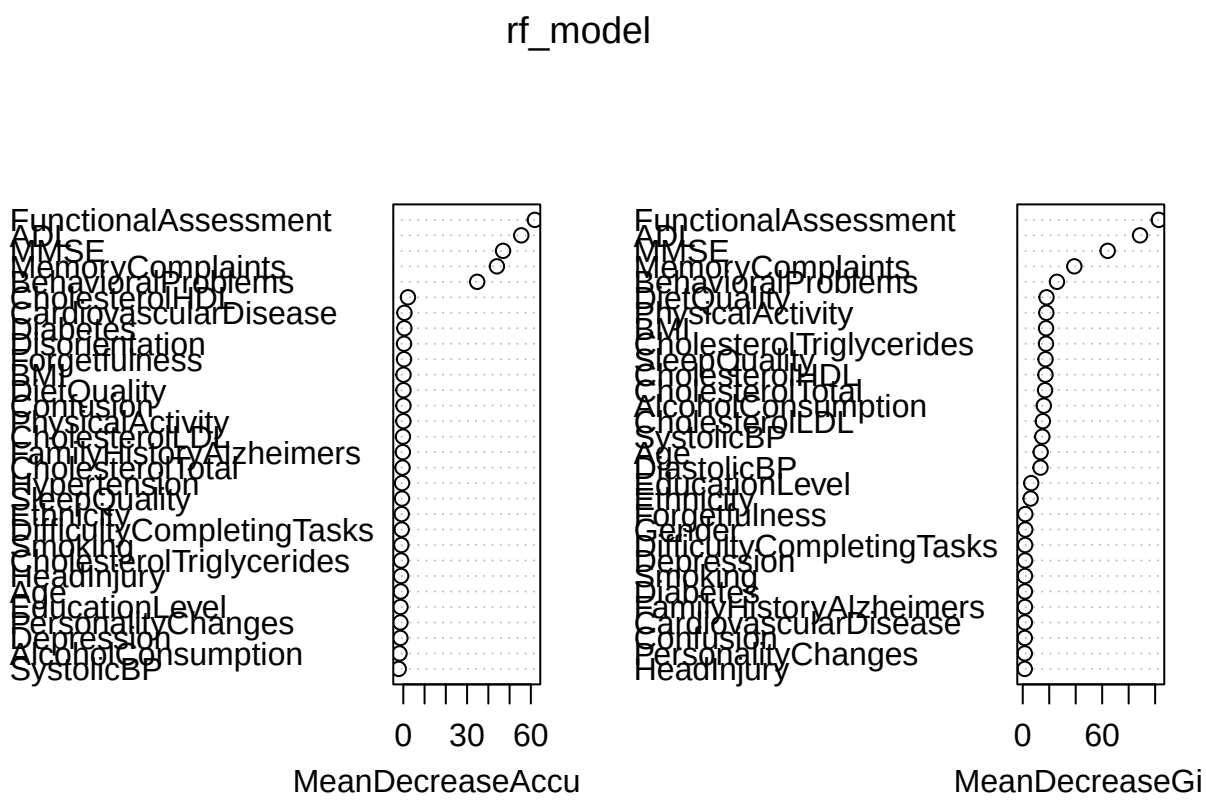
```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction   0   1
##          0 175  22
##          1  19  84
##
##                Accuracy : 0.8633
##                  95% CI : (0.8192, 0.9001)
##     No Information Rate : 0.6467
##     P-Value [Acc > NIR] : <2e-16
##
##                   Kappa : 0.699
##
##  Mcnemar's Test P-Value : 0.7548
##
##             Sensitivity : 0.9021
##             Specificity : 0.7925
##          Pos Pred Value : 0.8883
##          Neg Pred Value : 0.8155
##              Prevalence : 0.6467
##          Detection Rate : 0.5833
##    Detection Prevalence : 0.6567
##       Balanced Accuracy : 0.8473
##
##        'Positive' Class : 0
##
```

```r
# Calculate accuracy
glm_accuracy <- mean(glm_preds == valid_data$Diagnosis)
print(paste("Logistic Regression Accuracy:", round(glm_accuracy * 100, 2), "%"))
```

```
## [1] "Logistic Regression Accuracy: 86.33 %"
```

2.3. Random Forest

```
# Build random forest model
library(randomForest)
set.seed(123)
rf_model <- randomForest(Diagnosis ~ ., data = train_data, ntree = 500, importance = TRUE)

# Variable importance plot
varImpPlot(rf_model)
```




```
# Predictions on validation set
rf_preds <- predict(rf_model, valid_data)

# Confusion matrix
rf_cm <- confusionMatrix(rf_preds, valid_data$Diagnosis)
print(rf_cm)
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction   0   1
##          0 191  14
##          1   3  92
##
##                Accuracy : 0.9433
##                  95% CI : (0.9108, 0.9666)
##     No Information Rate : 0.6467
##     P-Value [Acc > NIR] : < 2e-16
##
##                   Kappa : 0.873
```

```
##
##  Mcnemar's Test P-Value : 0.01529
##
##             Sensitivity : 0.9845
##             Specificity : 0.8679
##          Pos Pred Value : 0.9317
##          Neg Pred Value : 0.9684
##              Prevalence : 0.6467
##          Detection Rate : 0.6367
##    Detection Prevalence : 0.6833
##       Balanced Accuracy : 0.9262
##
##        'Positive' Class : 0
##
```

```
# Calculate accuracy
rf_accuracy <- mean(rf_preds == valid_data$Diagnosis)
print(paste("Random Forest Accuracy:", round(rf_accuracy * 100, 2), "%"))
```

```
## [1] "Random Forest Accuracy: 94.33 %"
```

2.4. Linear Discriminant Analysis (LDA)

```
# Load dplyr and MASS

library(dplyr)
library(MASS)
```

```
##
## Attaching package: 'MASS'
```

```
## The following object is masked from 'package:dplyr':
##
##     select
```

```
library(caret)

# Fit LDA model
lda_model <- lda(Diagnosis ~ ., data = train_data)

# Print model summary
print(lda_model)
```

```
## Call:
## lda(Diagnosis ~ ., data = train_data)
##
## Prior probabilities of groups:
##         0         1
## 0.6461794 0.3538206
##
## Group means:
##        Age   Gender1 Ethnicity1 Ethnicity2 Ethnicity3 EducationLevel1
## 0 74.88046 0.5334190  0.2133676 0.09511568 0.10539846       0.3920308
## 1 74.80986 0.4741784  0.1830986 0.10798122 0.08685446       0.3873239
##   EducationLevel2 EducationLevel3      BMI  Smoking1 AlcoholConsumption
## 0       0.3071979      0.09897172 27.32396 0.2802057          10.130905
## 1       0.3075117      0.08920188 27.96298 0.2746479           9.948297
```

```
##   PhysicalActivity DietQuality SleepQuality FamilyHistoryAlzheimers1
## 0         4.944288    4.835363     7.160377                0.2455013
## 1         4.917248    4.999600     6.958219                0.2535211
##   CardiovascularDisease1 Diabetes1 Depression1 HeadInjury1 Hypertension1
## 0              0.1311054 0.1645244   0.2082262  0.09640103     0.1478149
## 1              0.1619718 0.1455399   0.1948357  0.09154930     0.1760563
##   SystolicBP DiastolicBP CholesterolTotal CholesterolLDL CholesterolHDL
## 0   134.6311    89.98972         225.2968       125.2710       58.86803
## 1   134.8192    90.08685         225.0832       124.7023       61.23251
##   CholesterolTriglycerides     MMSE FunctionalAssessment MemoryComplaints1
## 0                 227.1754 16.04460             5.977068         0.1092545
## 1                 232.6116 11.80636             3.762495         0.3544601
##   BehavioralProblems1      ADL Confusion1 Disorientation1 PersonalityChanges1
## 0          0.07969152 5.632153  0.2082262       0.1619537           0.1696658
## 1          0.25821596 3.540788  0.1995305       0.1455399           0.1361502
##   DifficultyCompletingTasks1 Forgetfulness1
## 0                  0.1516710      0.2827763
## 1                  0.1643192      0.2957746
##
## Coefficients of linear discriminants:
##                                       LD1
## Age                         -0.0031298602
## Gender1                     -0.1048469013
## Ethnicity1                  -0.1851671870
## Ethnicity2                   0.0369142134
## Ethnicity3                  -0.1241550146
## EducationLevel1             -0.0863701595
## EducationLevel2              0.0302193105
## EducationLevel3             -0.0965470095
## BMI                          0.0044450949
## Smoking1                    -0.0871835147
## AlcoholConsumption          -0.0014390817
## PhysicalActivity            -0.0082078076
## DietQuality                  0.0123216784
## SleepQuality                -0.0431747893
## FamilyHistoryAlzheimers1     0.0791679580
## CardiovascularDisease1       0.1613434490
## Diabetes1                    0.0020964082
## Depression1                  0.1241455056
## HeadInjury1                 -0.0949781102
## Hypertension1                0.1298414929
## SystolicBP                   0.0013070684
## DiastolicBP                  0.0012543748
## CholesterolTotal             0.0001494060
## CholesterolLDL              -0.0002221857
## CholesterolHDL               0.0019657492
## CholesterolTriglycerides     0.0003558030
## MMSE                        -0.0548221303
## FunctionalAssessment        -0.2287531941
## MemoryComplaints1            1.4209330852
## BehavioralProblems1          1.3786160168
## ADL                         -0.2102776442
## Confusion1                  -0.1331946883
## Disorientation1             -0.0315523419
```

```
## PersonalityChanges1          0.0002448730
## DifficultyCompletingTasks1   0.0407550635
## Forgetfulness1              -0.0090783603
```

```
# Predict on validation set
lda_preds <- predict(lda_model, valid_data)$class

# Confusion matrix
lda_cm <- confusionMatrix(lda_preds, valid_data$Diagnosis)
print(lda_cm)
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction   0   1
##          0 173  22
##          1  21  84
##
##                Accuracy : 0.8567
##                  95% CI : (0.8118, 0.8943)
##     No Information Rate : 0.6467
##     P-Value [Acc > NIR] : 2.716e-16
##
##                   Kappa : 0.6857
##
##  Mcnemar's Test P-Value : 1
##
##             Sensitivity : 0.8918
##             Specificity : 0.7925
##          Pos Pred Value : 0.8872
##          Neg Pred Value : 0.8000
##              Prevalence : 0.6467
##          Detection Rate : 0.5767
##    Detection Prevalence : 0.6500
##       Balanced Accuracy : 0.8421
##
##        'Positive' Class : 0
##
```

```
# Calculate accuracy
lda_accuracy <- mean(lda_preds == valid_data$Diagnosis)
print(paste("LDA Accuracy:", round(lda_accuracy * 100, 2), "%"))
```

```
## [1] "LDA Accuracy: 85.67 %"
```

2.5. Naive Bayes

```
# Fit Naive Bayes model
library(e1071)
nb_model <- naiveBayes(Diagnosis ~ ., data = train_data)

# Print model summary
print(nb_model)
```

```
##
## Naive Bayes Classifier for Discrete Predictors
##
```

```
## Call:
## naiveBayes.default(x = X, y = Y, laplace = laplace)
##
## A-priori probabilities:
## Y
##         0         1
## 0.6461794 0.3538206
##
## Conditional probabilities:
##    Age
## Y       [,1]     [,2]
##   0 74.88046 8.866077
##   1 74.80986 9.180491
##
##    Gender
## Y           0         1
##   0 0.4665810 0.5334190
##   1 0.5258216 0.4741784
##
##    Ethnicity
## Y            0          1          2          3
##   0 0.58611825 0.21336761 0.09511568 0.10539846
##   1 0.62206573 0.18309859 0.10798122 0.08685446
##
##    EducationLevel
## Y            0          1          2          3
##   0 0.20179949 0.39203085 0.30719794 0.09897172
##   1 0.21596244 0.38732394 0.30751174 0.08920188
##
##    BMI
## Y       [,1]     [,2]
##   0 27.32396 7.185095
##   1 27.96298 7.231224
##
##    Smoking
## Y           0         1
##   0 0.7197943 0.2802057
##   1 0.7253521 0.2746479
##
##    AlcoholConsumption
## Y        [,1]     [,2]
##   0 10.130905 5.687622
##   1  9.948297 5.716819
##
##    PhysicalActivity
## Y       [,1]     [,2]
##   0 4.944288 2.884589
##   1 4.917248 2.879578
##
##    DietQuality
## Y       [,1]     [,2]
##   0 4.835363 2.918154
##   1 4.999600 2.971120
##
```

```
##    SleepQuality
## Y       [,1]     [,2]
##   0 7.160377 1.739710
##   1 6.958219 1.744981
##
##    FamilyHistoryAlzheimers
## Y           0         1
##   0 0.7544987 0.2455013
##   1 0.7464789 0.2535211
##
##    CardiovascularDisease
## Y           0         1
##   0 0.8688946 0.1311054
##   1 0.8380282 0.1619718
##
##    Diabetes
## Y           0         1
##   0 0.8354756 0.1645244
##   1 0.8544601 0.1455399
##
##    Depression
## Y           0         1
##   0 0.7917738 0.2082262
##   1 0.8051643 0.1948357
##
##    HeadInjury
## Y            0          1
##   0 0.90359897 0.09640103
##   1 0.90845070 0.09154930
##
##    Hypertension
## Y           0         1
##   0 0.8521851 0.1478149
##   1 0.8239437 0.1760563
##
##    SystolicBP
## Y       [,1]     [,2]
##   0 134.6311 25.41813
##   1 134.8192 25.98502
##
##    DiastolicBP
## Y       [,1]     [,2]
##   0 89.98972 17.75378
##   1 90.08685 17.49513
##
##    CholesterolTotal
## Y       [,1]     [,2]
##   0 225.2968 41.82247
##   1 225.0832 43.65304
##
##    CholesterolLDL
## Y       [,1]     [,2]
##   0 125.2710 43.07952
##   1 124.7023 43.25795
```

```
##
##    CholesterolHDL
## Y       [,1]     [,2]
##   0 58.86803 22.90476
##   1 61.23251 23.02539
##
##    CholesterolTriglycerides
## Y       [,1]     [,2]
##   0 227.1754 102.4079
##   1 232.6116 101.9677
##
##    MMSE
## Y       [,1]     [,2]
##   0 16.04460 8.966840
##   1 11.80636 7.119611
##
##    FunctionalAssessment
## Y       [,1]     [,2]
##   0 5.977068 2.759017
##   1 3.762495 2.600731
##
##    MemoryComplaints
## Y           0         1
##   0 0.8907455 0.1092545
##   1 0.6455399 0.3544601
##
##    BehavioralProblems
## Y           0          1
##   0 0.92030848 0.07969152
##   1 0.74178404 0.25821596
##
##    ADL
## Y       [,1]     [,2]
##   0 5.632153 2.830142
##   1 3.540788 2.577678
##
##    Confusion
## Y           0         1
##   0 0.7917738 0.2082262
##   1 0.8004695 0.1995305
##
##    Disorientation
## Y           0         1
##   0 0.8380463 0.1619537
##   1 0.8544601 0.1455399
##
##    PersonalityChanges
## Y           0         1
##   0 0.8303342 0.1696658
##   1 0.8638498 0.1361502
##
##    DifficultyCompletingTasks
## Y           0         1
##   0 0.8483290 0.1516710
```

```
##   1 0.8356808 0.1643192
##
##    Forgetfulness
## Y           0         1
##   0 0.7172237 0.2827763
##   1 0.7042254 0.2957746
```

```r
# Predict on validation set
nb_preds <- predict(nb_model, valid_data)

# Confusion matrix
nb_cm <- confusionMatrix(nb_preds, valid_data$Diagnosis)
print(nb_cm)
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction   0   1
##          0 182  28
##          1  12  78
##
##                Accuracy : 0.8667
##                  95% CI : (0.8229, 0.903)
##     No Information Rate : 0.6467
##     P-Value [Acc > NIR] : < 2e-16
##
##                   Kappa : 0.6979
##
##  Mcnemar's Test P-Value : 0.01771
##
##             Sensitivity : 0.9381
##             Specificity : 0.7358
##          Pos Pred Value : 0.8667
##          Neg Pred Value : 0.8667
##              Prevalence : 0.6467
##          Detection Rate : 0.6067
##    Detection Prevalence : 0.7000
##       Balanced Accuracy : 0.8370
##
##        'Positive' Class : 0
##
```

```r
# Calculate accuracy
nb_accuracy <- mean(nb_preds == valid_data$Diagnosis)
print(paste("Naive Bayes Accuracy:", round(nb_accuracy * 100, 2), "%"))
```

```
## [1] "Naive Bayes Accuracy: 86.67 %"
```

2.6. Hyperparameter

```r
#Define Hyperparameter Grid
# Define a grid for the 'mtry' parameter in Random Forest
tuneGrid <- expand.grid(mtry = c(10:25))
#Cross-Validation Setup (using stratified sampling)
library(caret)
control <- trainControl(method = "cv",
                        number = 5,
```

```
                        summaryFunction = defaultSummary,
                        savePredictions = TRUE,
                        classProbs = FALSE,
                        sampling = "smote")  # Handle class imbalance in small datasets
# Model Training with Hyperparameter Tuning
# Train the Random Forest model using the 'caret' package and grid search
library(themis)
model <- train(Diagnosis ~ .,
               data = train_data,
               method = "rf",
               metric = "Accuracy",   # Set the metric explicitly for classification
               trControl = control,
               tuneGrid = tuneGrid,
               allowParallel = TRUE)
```

```
# Print the Best Tuned Model
print(model$bestTune)   # Output the best 'mtry' value
```

```
##   mtry
## 6   15
```

```
print(model)
```

```
## Random Forest
##
## 1204 samples
##   32 predictor
##    2 classes: '0', '1'
##
## No pre-processing
## Resampling: Cross-Validated (5 fold)
## Summary of sample sizes: 963, 964, 963, 963, 963
## Addtional sampling using SMOTE
##
## Resampling results across tuning parameters:
##
##   mtry  Accuracy   Kappa
##   10    0.9385477  0.8653593
##   11    0.9443534  0.8780547
##   12    0.9443534  0.8782919
##   13    0.9443499  0.8782092
##   14    0.9443465  0.8783467
##   15    0.9493326  0.8889993
##   16    0.9451833  0.8800479
##   17    0.9451833  0.8802500
##   18    0.9443465  0.8784436
##   19    0.9435166  0.8770358
##   20    0.9443430  0.8785835
##   21    0.9476694  0.8853683
##   22    0.9460097  0.8819662
##   23    0.9476763  0.8854381
##   24    0.9435097  0.8766191
##   25    0.9443465  0.8784236
##
## Accuracy was used to select the optimal model using the largest value.
```

```
## The final value used for the model was mtry = 15.
```

```
library(randomForest)
# choose best mtry
best_mtry <- model$bestTune$mtry

# use best mtry fit final model
final_model <- randomForest(
  Diagnosis ~ .,
  data = train_data,
  mtry = best_mtry
)

print(model$results)
```

```
##    mtry  Accuracy     Kappa  AccuracySD    KappaSD
## 1    10 0.9385477 0.8653593 0.012876783 0.02785057
## 2    11 0.9443534 0.8780547 0.009543346 0.02117667
## 3    12 0.9443534 0.8782919 0.013629776 0.02935481
## 4    13 0.9443499 0.8782092 0.013959545 0.02982843
## 5    14 0.9443465 0.8783467 0.013349639 0.02901719
## 6    15 0.9493326 0.8889993 0.012937979 0.02825621
## 7    16 0.9451833 0.8800479 0.015072679 0.03253412
## 8    17 0.9451833 0.8802500 0.011510444 0.02495441
## 9    18 0.9443465 0.8784436 0.018252914 0.03926280
## 10   19 0.9435166 0.8770358 0.019393416 0.04137937
## 11   20 0.9443430 0.8785835 0.014306363 0.03062035
## 12   21 0.9476694 0.8853683 0.012338043 0.02687689
## 13   22 0.9460097 0.8819662 0.015542090 0.03334288
## 14   23 0.9476763 0.8854381 0.014836854 0.03252920
## 15   24 0.9435097 0.8766191 0.013712685 0.02956718
## 16   25 0.9443465 0.8784236 0.015718849 0.03382019
```

```
print(model$bestTune)
```

```
##   mtry
## 6   15
```

```
# final parameter
model$finalModel
```

```
##
## Call:
##  randomForest(x = x, y = y, mtry = param$mtry, allowParallel = TRUE)
##                Type of random forest: classification
##                      Number of trees: 500
## No. of variables tried at each split: 15
##
##         OOB estimate of  error rate: 4.43%
## Confusion matrix:
##     0   1 class.error
## 0 752  26  0.03341902
## 1  43 735  0.05526992
```

```
print(model$finalModel)
```

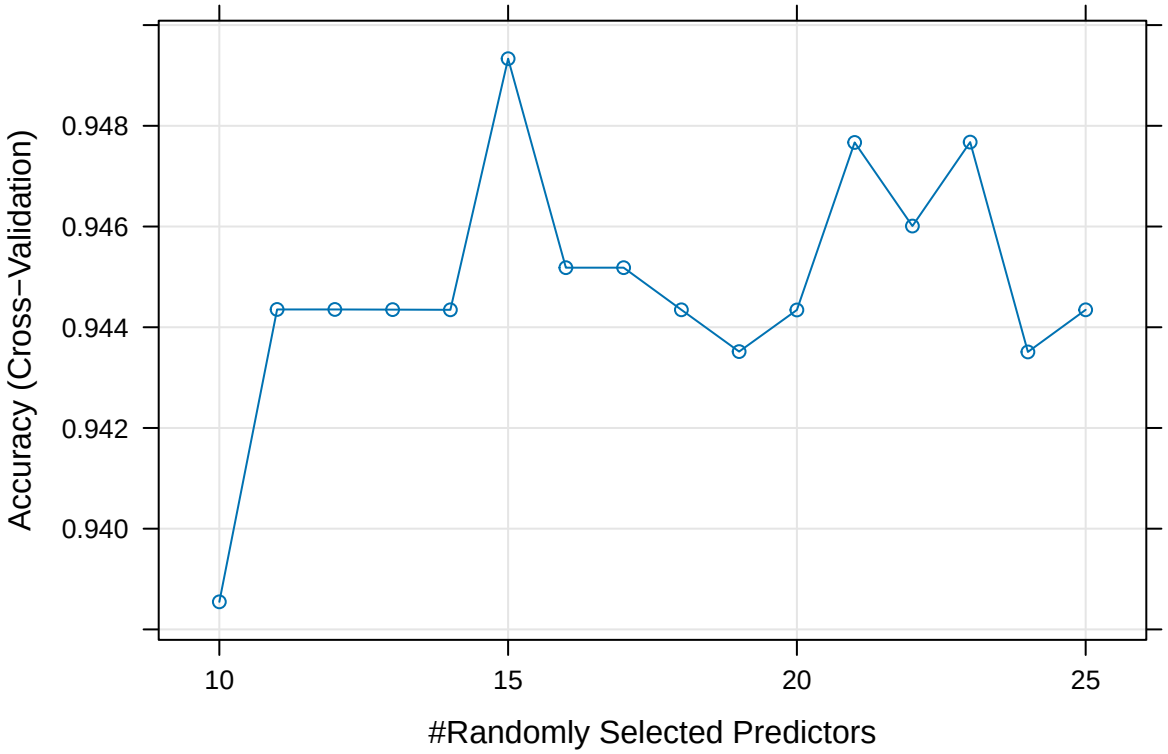
```
##
```

```
## Call:
##  randomForest(x = x, y = y, mtry = param$mtry, allowParallel = TRUE)
##                Type of random forest: classification
##                      Number of trees: 500
## No. of variables tried at each split: 15
##
##         OOB estimate of  error rate: 4.43%
## Confusion matrix:
##     0   1 class.error
## 0 752  26  0.03341902
## 1  43 735  0.05526992
```

```
summary(model$finalModel)
```

```
##                 Length Class      Mode
## call               5   -none-     call
## type               1   -none-     character
## predicted       1556   factor     numeric
## err.rate        1500   -none-     numeric
## confusion          6   -none-     numeric
## votes           3112   matrix     numeric
## oob.times       1556   -none-     numeric
## classes            2   -none-     character
## importance        36   -none-     numeric
## importanceSD       0   -none-     NULL
## localImportance    0   -none-     NULL
## proximity          0   -none-     NULL
## ntree              1   -none-     numeric
## mtry               1   -none-     numeric
## forest            14   -none-     list
## y               1556   factor     numeric
## test               0   -none-     NULL
## inbag              0   -none-     NULL
## xNames            36   -none-     character
## problemType        1   -none-     character
## tuneValue          1   data.frame list
## obsLevels          2   -none-     character
## param              1   -none-     list
```

```
# Visualize the Tuning Results
# Plot the performance of different hyperparameter values
plot(model)
```



2.7. ROC Curves

```
# ROC for logistic regression
library(pROC)
glm_roc <- roc(valid_data$Diagnosis, glm_probs)
```

```
## Setting levels: control = 0, case = 1
```

```
## Setting direction: controls < cases
```

```
plot(glm_roc, col = 'blue', main = 'ROC Curves')
auc(glm_roc)
```

```
## Area under the curve: 0.9149
```

```
# ROC for random forest
rf_probs <- predict(rf_model, valid_data, type = 'prob')[,2]
rf_roc <- roc(valid_data$Diagnosis, rf_probs)
```

```
## Setting levels: control = 0, case = 1
## Setting direction: controls < cases
```

```
plot(rf_roc, col = 'red', add = TRUE)
auc(rf_roc)
```

```
## Area under the curve: 0.9616
```

```
# ROC for lda
lda_probs <- predict(lda_model, valid_data)$posterior[, 2]
lda_roc <- roc(valid_data$Diagnosis, lda_probs)
```

```
## Setting levels: control = 0, case = 1
## Setting direction: controls < cases
```

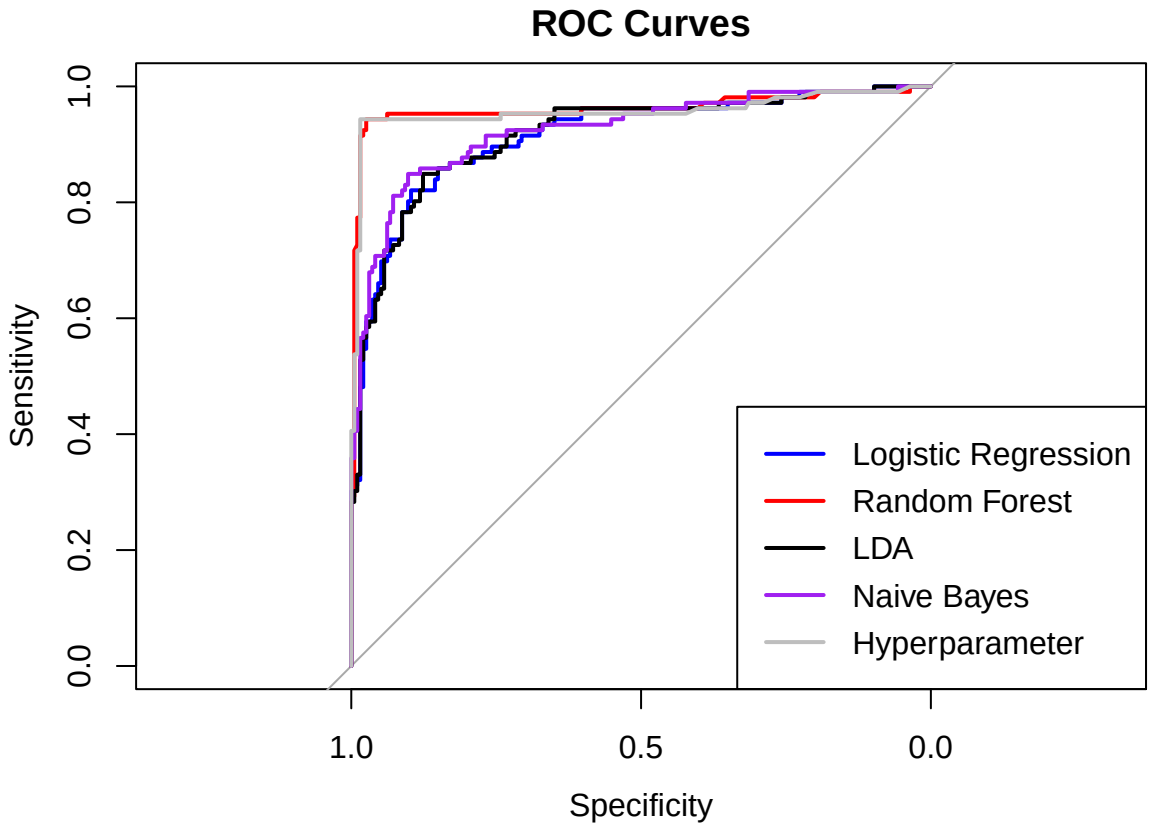
```r
plot(lda_roc, col = 'black', add = TRUE)
auc_value <- auc(lda_roc)

#ROC for Naive Bayes
nb_probs <- predict(nb_model, valid_data, type = 'raw')[, 2]
nb_roc <- roc(valid_data$Diagnosis, nb_probs)
```

```
## Setting levels: control = 0, case = 1
## Setting direction: controls < cases
```

```r
plot(nb_roc, col = 'purple', add = TRUE)
auc_value <- auc(nb_roc)

# ROC of Hyperparameter
rf_probs <- predict(final_model, valid_data, type = "prob")[, 2]
rf_roc <- roc(valid_data$Diagnosis, rf_probs)
```

```
## Setting levels: control = 0, case = 1
## Setting direction: controls < cases
```

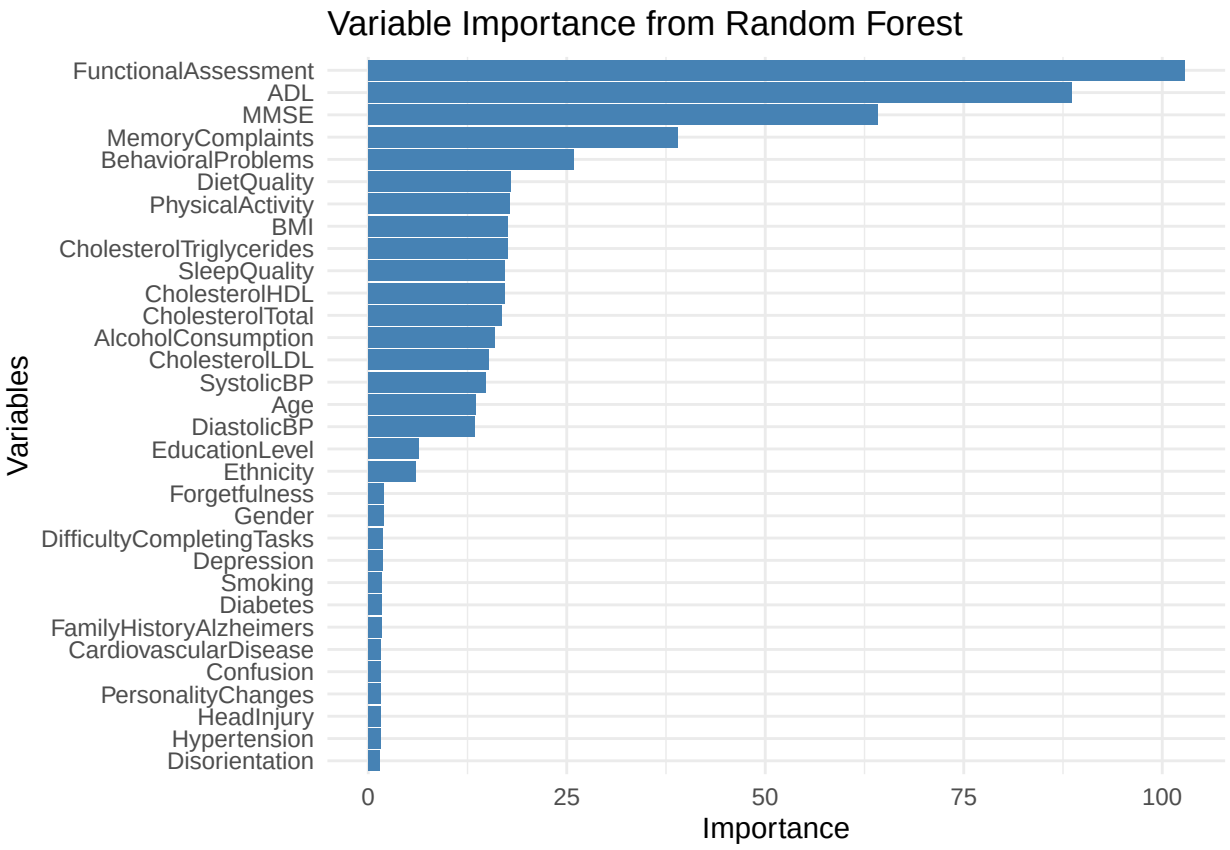
```r
plot(rf_roc, col = 'grey', add = TRUE)
auc_value <- auc(rf_roc)

legend('bottomright',
       legend = c('Logistic Regression', 'Random Forest', 'LDA', 'Naive Bayes','Hyperparameter'),
       col = c('blue', 'red','black','purple','grey'), lwd = 2)
```



3. Feature Selection

3.1. Variable Importance from Random Forest

```
# Get variable importance from random forest
library(randomForest)
importance <- importance(rf_model)
var_importance <- data.frame(Variables = row.names(importance), Importance = importance[, 'MeanDecrease

library(ggplot2)
# Plot variable importance
ggplot(var_importance, aes(x = reorder(Variables, Importance), y = Importance)) +
  geom_bar(stat = 'identity', fill = 'steelblue') +
  coord_flip() +
  theme_minimal() +
  labs(title = 'Variable Importance from Random Forest', x = 'Variables', y = 'Importance')
```



3.2. Recursive Feature Elimination (RFE)

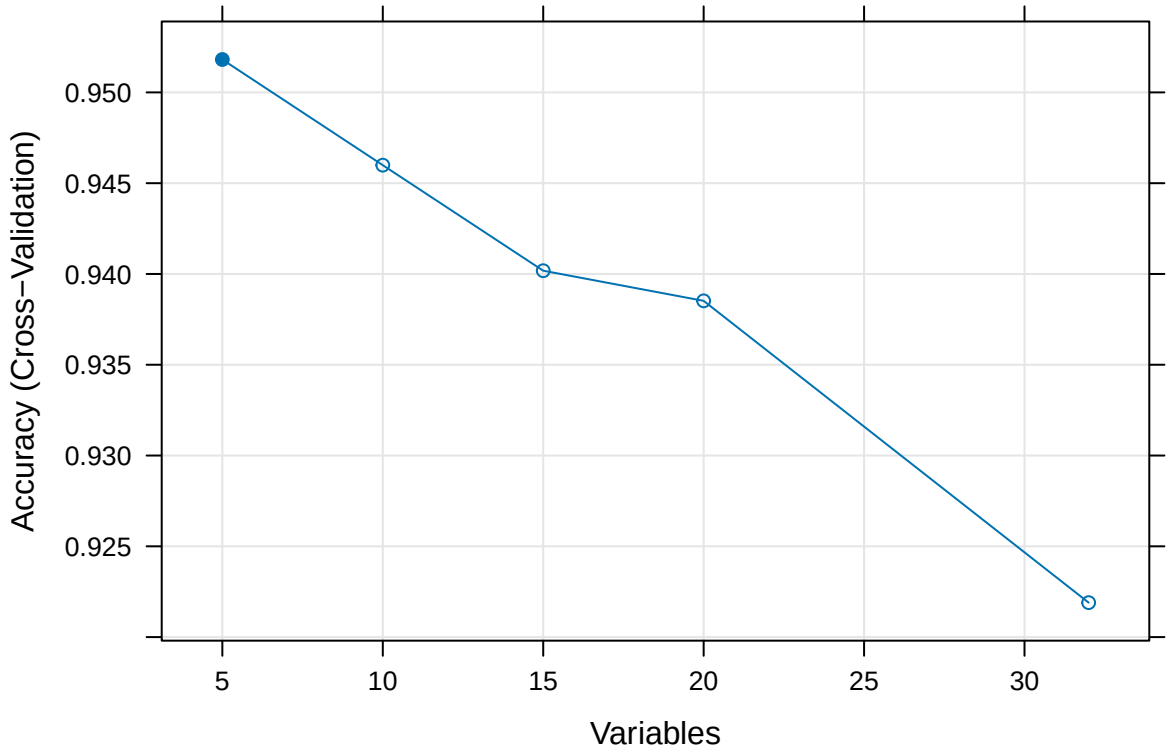
```
# Use caret's RFE with random forest
library(caret)
control <- rfeControl(functions = rfFuncs, method = 'cv', number = 5)
set.seed(123)
rfe_model <- rfe(train_data[, -which(names(train_data) == 'Diagnosis')], train_data$Diagnosis, sizes = 

# Optimal variables
optimal_vars <- rfe_model$optVariables
print("Optimal Variables Selected:")
```

```
## [1] "Optimal Variables Selected:"
```

```r
print(optimal_vars)
```

```
## [1] "FunctionalAssessment" "ADL"                  "MMSE"
## [4] "MemoryComplaints"     "BehavioralProblems"
```

```r
# Plot RFE results
plot(rfe_model, type = c('g', 'o'))
```



4. Statistical Inference

4.1. Hypothesis Testing

```r
#4.1.1. Comparing Means
# T-test for 'Age' variable
t_test_age <- t.test(Age ~ Diagnosis, data = train)
print(t_test_age)
```

```
##
##  Welch Two Sample t-test
##
## data:  Age by Diagnosis
## t = 0.38758, df = 1071.3, p-value = 0.6984
## alternative hypothesis: true difference in means between group 0 and group 1 is not equal to 0
## 95 percent confidence interval:
##  -0.7653557  1.1421310
## sample estimates:
## mean in group 0 mean in group 1
##        74.97222        74.78383
```

```r
# Wilcoxon test for 'MMSE' if not normally distributed
wilcox_test_mmse <- wilcox.test(MMSE ~ Diagnosis, data = train)
print(wilcox_test_mmse)
```

```
##
##  Wilcoxon rank sum test with continuity correction
##
## data:  MMSE by Diagnosis
## W = 333042, p-value < 2.2e-16
## alternative hypothesis: true location shift is not equal to 0
```

```
#4.1.2. Chi-Squared Test for Categorical Variables
# Chi-squared test for 'Gender'
table_gender <- table(train$Gender, train$Diagnosis)
chi_test_gender <- chisq.test(table_gender)
print(chi_test_gender)
```

```
##
##  Pearson's Chi-squared test with Yates' continuity correction
##
## data:  table_gender
## X-squared = 2.3131, df = 1, p-value = 0.1283
```

4.2. Confidence Intervals and Odds Ratios

```
# Confidence intervals for logistic regression coefficients
confint_glm <- confint(glm_model)
```

```
## Waiting for profiling to be done...
```

```
print(confint_glm)
```

```
##                                 2.5 %       97.5 %
## (Intercept)                1.388765824  6.433636386
## Age                       -0.025119736  0.012290173
## Gender1                   -0.607313884  0.064517849
## Ethnicity1                -0.815834214  0.065970342
## Ethnicity2                -0.556288291  0.628940577
## Ethnicity3                -0.898958503  0.315977015
## EducationLevel1           -0.585156091  0.322851072
## EducationLevel2           -0.401787218  0.550630522
## EducationLevel3           -0.901892433  0.390177949
## BMI                       -0.017504461  0.029032371
## Smoking1                  -0.562244867  0.200455498
## AlcoholConsumption        -0.035602421  0.022742193
## PhysicalActivity          -0.071210824  0.043583663
## DietQuality               -0.025126150  0.091977048
## SleepQuality              -0.158515658  0.034463882
## FamilyHistoryAlzheimers1  -0.242475479  0.543547233
## CardiovascularDisease1    -0.205347481  0.713224950
## Diabetes1                 -0.494457851  0.433816004
## Depression1               -0.180840132  0.629589491
## HeadInjury1               -0.754217243  0.376207163
## Hypertension1             -0.270579756  0.661179069
## SystolicBP                -0.004373535  0.008630021
## DiastolicBP               -0.005499528  0.013184058
## CholesterolTotal          -0.004031215  0.003811052
## CholesterolLDL            -0.004187059  0.003697898
## CholesterolHDL            -0.003378804  0.011320514
## CholesterolTriglycerides  -0.001024752  0.002262310
## MMSE                      -0.130757272 -0.087258212
```

```
## FunctionalAssessment       -0.503698523 -0.365949004
## MemoryComplaints1           2.002028391  2.876764308
## BehavioralProblems1         2.051924648  3.054235053
## ADL                        -0.479378649 -0.342087607
## Confusion1                 -0.679496213  0.147712813
## Disorientation1            -0.485358449  0.443968762
## PersonalityChanges1        -0.534040036  0.417311094
## DifficultyCompletingTasks1 -0.327100101  0.593997380
## Forgetfulness1             -0.417800109  0.324032250
```

```
# Odds ratios and confidence intervals
odds_ratios <- exp(cbind(OR = coef(glm_model), confint(glm_model)))
```

```
## Waiting for profiling to be done...
```

```
print(odds_ratios)
```

```
##                                   OR     2.5 %      97.5 %
## (Intercept)                49.0144311 4.0098981 622.4332452
## Age                         0.9936406 0.9751931   1.0123660
## Gender1                     0.7630903 0.5448123   1.0666446
## Ethnicity1                  0.6901861 0.4422702   1.0681950
## Ethnicity2                  1.0404978 0.5733332   1.8756224
## Ethnicity3                  0.7523810 0.4069933   1.3715987
## EducationLevel1             0.8767548 0.5570189   1.3810597
## EducationLevel2             1.0760522 0.6691231   1.7343462
## EducationLevel3             0.7772042 0.4058010   1.4772436
## BMI                         1.0057704 0.9826479   1.0294579
## Smoking1                    0.8361710 0.5699282   1.2219592
## AlcoholConsumption          0.9936159 0.9650239   1.0230028
## PhysicalActivity            0.9863557 0.9312655   1.0445474
## DietQuality                 1.0338020 0.9751869   1.0963397
## SleepQuality                0.9399377 0.8534096   1.0350646
## FamilyHistoryAlzheimers1    1.1633035 0.7846830   1.7221047
## CardiovascularDisease1      1.2902119 0.8143643   2.0405614
## Diabetes1                   0.9732114 0.6099015   1.5431349
## Depression1                 1.2524933 0.8345688   1.8768400
## HeadInjury1                 0.8329800 0.4703787   1.4567489
## Hypertension1               1.2172566 0.7629370   1.9370749
## SystolicBP                  1.0021209 0.9956360   1.0086674
## DiastolicBP                 1.0038294 0.9945156   1.0132714
## CholesterolTotal            0.9998897 0.9959769   1.0038183
## CholesterolLDL              0.9997550 0.9958217   1.0037047
## CholesterolHDL              1.0039672 0.9966269   1.0113848
## CholesterolTriglycerides    1.0006162 0.9989758   1.0022649
## MMSE                        0.8971425 0.8774307   0.9164404
## FunctionalAssessment        0.6485752 0.6042915   0.6935382
## MemoryComplaints1          11.3558443 7.4040592  17.7567248
## BehavioralProblems1        12.7087770 7.7828660  21.2049586
## ADL                         0.6643785 0.6191680   0.7102860
## Confusion1                  0.7688038 0.5068723   1.1591799
## Disorientation1             0.9827926 0.6154765   1.5588818
## PersonalityChanges1         0.9467217 0.5862318   1.5178746
## DifficultyCompletingTasks1  1.1453282 0.7210116   1.8112141
## Forgetfulness1              0.9555157 0.6584938   1.3826919
```

5. Model Selection Techniques

5.1. Model Selection with AIC and BIC

5.1.1. AIC-based Selection

```
# Full model
full_model <- glm(Diagnosis ~ ., data = train_data, family = binomial)

# Null model
null_model <- glm(Diagnosis ~ 1, data = train_data, family = binomial)

# reduction of the model
reduced_formula <- as.formula(paste("Diagnosis ~", paste(optimal_vars, collapse = "+")))
reduced_model <- glm(reduced_formula, data = train_data, family = binomial)

# Stepwise selection based on AIC
step_model_aic <- stepAIC(reduced_model, direction = "both", trace = FALSE)
summary(step_model_aic)
```

```
##
## Call:
## glm(formula = Diagnosis ~ FunctionalAssessment + ADL + MMSE +
##     MemoryComplaints + BehavioralProblems, family = binomial,
##     data = train_data)
##
## Coefficients:
##                       Estimate Std. Error z value Pr(>|z|)
## (Intercept)            3.74991    0.30057  12.476   <2e-16 ***
## FunctionalAssessment -0.42257    0.03395 -12.447   <2e-16 ***
## ADL                  -0.40222    0.03393 -11.854   <2e-16 ***
## MMSE                 -0.10584    0.01072  -9.872   <2e-16 ***
## MemoryComplaints1     2.45117    0.21848  11.219   <2e-16 ***
## BehavioralProblems1   2.53497    0.24783  10.229   <2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 1564.67  on 1203  degrees of freedom
## Residual deviance:  915.26  on 1198  degrees of freedom
## AIC: 927.26
##
## Number of Fisher Scoring iterations: 6
```

```
# AIC value
aic_value <- AIC(step_model_aic)
print(paste("AIC of selected model:", aic_value))
```

```
## [1] "AIC of selected model: 927.257577904444"
```

```
# Stepwise selection based on BIC
n <- nrow(train_data)
step_model_bic <- stepAIC(reduced_model, direction = "both", k = log(n), trace = FALSE)

# Summary of the model
summary(step_model_bic)
```

```
##
## Call:
## glm(formula = Diagnosis ~ FunctionalAssessment + ADL + MMSE +
##     MemoryComplaints + BehavioralProblems, family = binomial,
##     data = train_data)
##
## Coefficients:
##                      Estimate Std. Error z value Pr(>|z|)
## (Intercept)           3.74991    0.30057  12.476   <2e-16 ***
## FunctionalAssessment -0.42257    0.03395 -12.447   <2e-16 ***
## ADL                  -0.40222    0.03393 -11.854   <2e-16 ***
## MMSE                 -0.10584    0.01072  -9.872   <2e-16 ***
## MemoryComplaints1     2.45117    0.21848  11.219   <2e-16 ***
## BehavioralProblems1   2.53497    0.24783  10.229   <2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 1564.67  on 1203  degrees of freedom
## Residual deviance:  915.26  on 1198  degrees of freedom
## AIC: 927.26
##
## Number of Fisher Scoring iterations: 6
```

```
# BIC value
bic_value <- BIC(step_model_bic)
print(paste("BIC of selected model:", bic_value))
```

```
## [1] "BIC of selected model: 957.818005659656"
```

```
# AIC model predictions
aic_preds_prob <- predict(step_model_aic, valid_data, type = 'response')
aic_preds <- ifelse(aic_preds_prob > 0.5, 1, 0)

# Confusion matrix
aic_cm <- confusionMatrix(as.factor(aic_preds), valid_data$Diagnosis)
print(aic_cm)
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction   0   1
##          0 173  20
##          1  21  86
##
##                Accuracy : 0.8633
##                  95% CI : (0.8192, 0.9001)
##     No Information Rate : 0.6467
##     P-Value [Acc > NIR] : <2e-16
##
##                   Kappa : 0.7016
##
##  Mcnemar's Test P-Value : 1
##
##             Sensitivity : 0.8918
```

```
##             Specificity : 0.8113
##          Pos Pred Value : 0.8964
##          Neg Pred Value : 0.8037
##              Prevalence : 0.6467
##          Detection Rate : 0.5767
##    Detection Prevalence : 0.6433
##       Balanced Accuracy : 0.8515
##
##        'Positive' Class : 0
##
```

```
# BIC model predictions
bic_preds_prob <- predict(step_model_bic, valid_data, type = 'response')
bic_preds <- ifelse(bic_preds_prob > 0.5, 1, 0)

# Confusion matrix
bic_cm <- confusionMatrix(as.factor(bic_preds), valid_data$Diagnosis)
print(bic_cm)
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction   0   1
##          0 173  20
##          1  21  86
##
##                Accuracy : 0.8633
##                  95% CI : (0.8192, 0.9001)
##     No Information Rate : 0.6467
##     P-Value [Acc > NIR] : <2e-16
##
##                   Kappa : 0.7016
##
##  Mcnemar's Test P-Value : 1
##
##             Sensitivity : 0.8918
##             Specificity : 0.8113
##          Pos Pred Value : 0.8964
##          Neg Pred Value : 0.8037
##              Prevalence : 0.6467
##          Detection Rate : 0.5767
##    Detection Prevalence : 0.6433
##       Balanced Accuracy : 0.8515
##
##        'Positive' Class : 0
##
```

6. Comparing All Models

6.1. Summary of Model Performances

```
model_performance <- data.frame(
  Model = c('Logistic Regression', 'Random Forest', 'LDA', 'Naive Bayes', 'AIC Logistic', 'BIC Logistic
  Accuracy = c(
    glm_accuracy * 100,
    rf_accuracy * 100,
```

```
    lda_accuracy * 100,
    nb_accuracy * 100,
    mean(aic_preds == valid_data$Diagnosis) * 100,
    mean(bic_preds == valid_data$Diagnosis) * 100
  )
)
```

7. Final Prediction on Test Data

7.1. Data Preparation

```
# Load necessary library
library(dplyr)

# Remove 'PatientID' and 'DoctorInCharge' from test data
test_ids <- test$PatientID
test_data <- test %>% dplyr::select(-PatientID, -DoctorInCharge)

# Convert categorical variables to factors
test_data[categorical_vars[-length(categorical_vars)]] <- lapply(test_data[categorical_vars[-length(cat

# Ensure levels match between train and test
for (var in categorical_vars[-length(categorical_vars)]) {
  levels(test_data[[var]]) <- levels(train_data[[var]])
}

# Handle missing values if any
test_data[numerical_vars] <- lapply(test_data[numerical_vars], function(x) ifelse(is.na(x), median(x, na
# Predict on test data
test_preds <- predict(model, test_data)

# Prepare submission file
submission <- data.frame(PatientID = test_ids, Diagnosis = test_preds)

# Convert 'Diagnosis' to integer
submission$Diagnosis <- as.integer(as.character(submission$Diagnosis))

# Save submission file
write.csv(submission, 'submission.csv', row.names = FALSE)
```