# MA335 - Final Project

Word count: 1230

2212098

20-06-2023

# Contents

# Introduction

Alzheimer's is a neurological condition that affects a person's cognitive functionality. Millions of individuals across the globe have encountered this disorder due to various reasons. In this report, I have analysed a comprehensive dataset containing few of the many characteristics associated with Alzheimer's. The primary objective was to identify the relationship between these characteristics and the diagnosis of Alzheimer's as "Demented" or "Non-Demented" and classify individuals into these two classes/clusters.

The tasks involved in the project were **preliminary data analysis** which included descriptive and statistical analysis to understand the data, **data pre-processing** to handle missing values, standardize numeric values and format variable types, **clustering** to form and visualize the diagnosis of individuals based on the characteristics as "Demented"/"Non-Demented", **feature selection** to identify the most significant characteristics out of the list that helps in classifying the patients and lastly, **classification** to categorize the individuals into two groups of diagnosis.

Overall, this study contributes to better understand Alzheimer's disease and it's potential characteristics.

# Analysis

The dataset contains 10 columns with one being the diagnosis (*Group*) and rest are the corresponding factors influencing the diagnosis to name a few, *Gender*, *Socioeconomic status (SES)*, *Estimated total intracranial volume(eTIV)*, *Clinical dementia rating (CDR)* etc for about 350 individuals.

## Data Wrangling

This section aims to clean and pre-process the data by performing the following tasks:

#### Remove `Converted` class from diagnosis variable *Group*

The rows with diagnosis recorded as 'converted' are removed from the data ensuring that only the relevant classes in Alzheimer's diagnosis are included for further analysis.

#### Encoding text data to numerical values *Group* & *Gender*

To handle text data and convert them to corresponding numeric categorical values, *Label Encoder* object and base R function *factor* have been used.

#### Handling missing values (*SES* & *MMSE*)

*SES* is a categorical variable, therefore the **NA** values are replaced with the mode. Additionally, *MMSE* is a continuous variable therefore, the **NA** values are replaced with the median of the column.

#### Standardizing volume variables (*eTIV*, *nWBV*, *ASF*)

Numeric continuous variables with values of different scale can cause inconsistencies during analysis and modelling. Therefore, the numerical columns are standardized by subtracting each data point with the mean and dividing the result by the standard deviation.

## Descriptive Analysis

In the given dataset, it can be observed that `Non-demented` diagnosis has a higher count of **190** and only **3** patients having severe conditions of dementia with *CDR* of 2. Out of the 3 patients with severe conditions, 2 are **Female**. On the other hand, majority of the female gender have no or very mild dementia whereas the male patients have an even distribution between mild and considerable dementia.

In terms of **Estimated total intracranial volume**, female patients on average had `eTIV` distributed evenly with 2 outlier points indicating the same results with CDR with severe conditions. On the other hand, the eTIV for male was distributed on the higher end. Additionally, on average female patients had higher intracranial volume as compared to male patients.

The **Whole brain volume (nWBV)** for age group 60-70 was higher for non-demented patients than the demented. Similarly, for older age groups $> 80$, the volume can be seen decreasing more for the demented patients in comparison to non-demented who stay on the neutral end.

## Cluster Analysis

### Fuzzy k-means clustering algorithm

In this section, the diagnosis Group has been clustered using **Fuzzy k-means clustering** algorithm which is a variation of the traditional k-means algorithm. FKM is more flexible and has a probabilistic approach that allows data points to belong to more than 1 cluster by assigning membership degree values. The membership values represent the probability of a data point belonging to a particular cluster. This process happens in iteration until convergence.

### Implementation

For the dataset, firstly the **`fclust`** method is run iteratively for every value of $K$ from 1 - 4 and the optimal value of clusters is chosen from the results. Then, the data is fed to the clustering algorithm **`fanny`** with the optimal number of chosen clusters to separate the data points into k number of Groups.

### Results

From the cluster output, it can be seen that some of the data points are common between both clusters as the algorithm has given a membership degree almost equal for both cluster groups. Also, some of the points have exceptional characteristics therefore they don't belong to either of the clusters (outliers). Overall, the fuzzy k-means algorithm clustered the data points correctly into the respective diagnosis `Group`.

## Logistic Regression

### Implementation

The aim of this section was to classify the data points into the 2 diagnosis groups by training a binary classification model based on the given characteristics. *Logistic Regression* was chosen because the data that we were classifying has a binary class (Demented/Non-demented) as the response variable. All the variables were taken as predictors and 80% of the data was taken as training and rest 20% for testing the model.

The values of parameters `mixture` and `penalty` which corresponds, to the amount of regularization applied to the model, were decided by *hyper-parameter tuning* using **grid**

**search cross validation**. The optimal values returned by the cross-validation were chosen to train the logistic regression model.

### Results

From the grid search, the optimal values of `mixture` was **0** indicating a *ridge regularization model* and `penalty` was **1e-9**. Taking all the variables as predictors, the trained logistic model was able to predict the classes of the test data with 100% accuracy.

On one side, this may seem good but when observed closely variable **CDR** has a very correlation of almost 90% with Group thereby causing the model to predict accurately. This leads to the implementation of feature selection method to identify significant variables.

## Feature Selection Using Variable Importance

### Implementation

Using `Cor()` we have got the correlation coefficients of all predictors and find that *eTIV* and *ASF* have a high correlation. Then, the function `findcorrelation()` was used to identify the variables to remove that had a coefficient value > **0.75**. Following this observation, we have removed the `eTIV` and `CDR` variables from the data. The remaining variables are then taken as predictors are fed to the feature selection method **`filterVarImp`** which assigns a score to each predictor corresponding to their significance in predicting the Group class.

### Results

From the scores, it can be observed that `MMSE`, `Age`, `Gender` and `EDUC` are the top 4 significant variables with a score above 50%.

Training the logistic regression model again with only the selected features gives an accuracy of approximately 80% in predicting the class of diagnosis Group.

## Model Comparison using Anova

This section aims to compare the 2 classification models using `anova()`.

**H0:** *Model 1: Group ~ Gender + Age + EDUC + SES + MMSE + CDR + eTIV + nWBV + ASF*

**H1:** *Model 2: Group ~ MMSE + Age + Gender + EDUC*

Supporting the above analysis and observation, the results of anova indicate that the model with all variables as predictors does a better job at predicting the classes of group than the model with the selected features.

# Conclusion

In conclusion, a comprehensive analysis of the data was performed to investigate the relationship between various characteristics contributing to Alzheimer's disease and the diagnosis. The implementation of preliminary descriptive and statistical analysis, clustering and classification modelling provided a integrated approach is recognizing the patterns in the factors and identifying the significant factors associated with Alzheimer's diagnosis.

# Appendix - R Code

```r
# set the working directory
setwd("S:/MA335_Modelling and observational data in R/Final
↪  project/Alzheimer-classification")

# import the libraries
library(dplyr)
library(ggplot2)
library(superml)
library(tidyr)
library(plotly)
library(corrplot)
library(RColorBrewer)
library(fclust)
library(tidymodels)
library(glmnet)
library(caret)
library(factoextra)
library(cluster)
library(knitr)
library(formatR)
library(modelsummary)
```

Table 1: Project data

| Group | M.F | Age | EDUC | SES | MMSE | CDR | eTIV | nWBV | ASF |
|---|---|---|---|---|---|---|---|---|---|
| Nondemented | M | 87 | 14 | 2 | 27 | 0.0 | 1987 | 0.696 | 0.883 |
| Nondemented | M | 88 | 14 | 2 | 30 | 0.0 | 2004 | 0.681 | 0.876 |
| Demented | M | 75 | 12 | NA | 23 | 0.5 | 1678 | 0.736 | 1.046 |

```r
# read the csv file as data frame
raw_data <- read.csv("project data.csv", header = TRUE)

kable(raw_data[1:3, ], caption = "Project data") %>%
  kableExtra::kable_styling(bootstrap_options = "bordered")
```

```r
# DATA CLEANING

# find the row numbers with Group = "Converted"
converted_group_rows <- which(raw_data$Group=="Converted")

# remove the rows with Group value "Converted"
raw_data <- raw_data[-converted_group_rows, ]

# Convert text data to numerical values

# 1. Group (Demented - 1, Non-Demented - 0)

# create encoder object
group_encoder = LabelEncoder$new()

# fit and transform the Group column using the created object
```

```r
raw_data$Group <- group_encoder$fit_transform(raw_data$Group)

# 2. M.F (M - 0, F - 1)

raw_data$M.F <- factor(raw_data$M.F,
                        levels = c("M", "F"),
                        labels = c(0, 1))

# rename column name from "M.F" to "Gender"
raw_data <- rename(raw_data,
                   Gender = M.F
                   )

# replace missing values

# 1. Replace NA values with MODE of column - SES

# function to calculate the mode of a column
calculate_mode <- function(col_to_calculate) {
  unique_values <- unique(col_to_calculate)
  mode_of_col <-
↪   unique_values[which.max(tabulate(match(col_to_calculate,
↪   unique_values)))]
  return(mode_of_col)
}

# get the mode of SES
mode_SES <- calculate_mode(raw_data$SES)
```

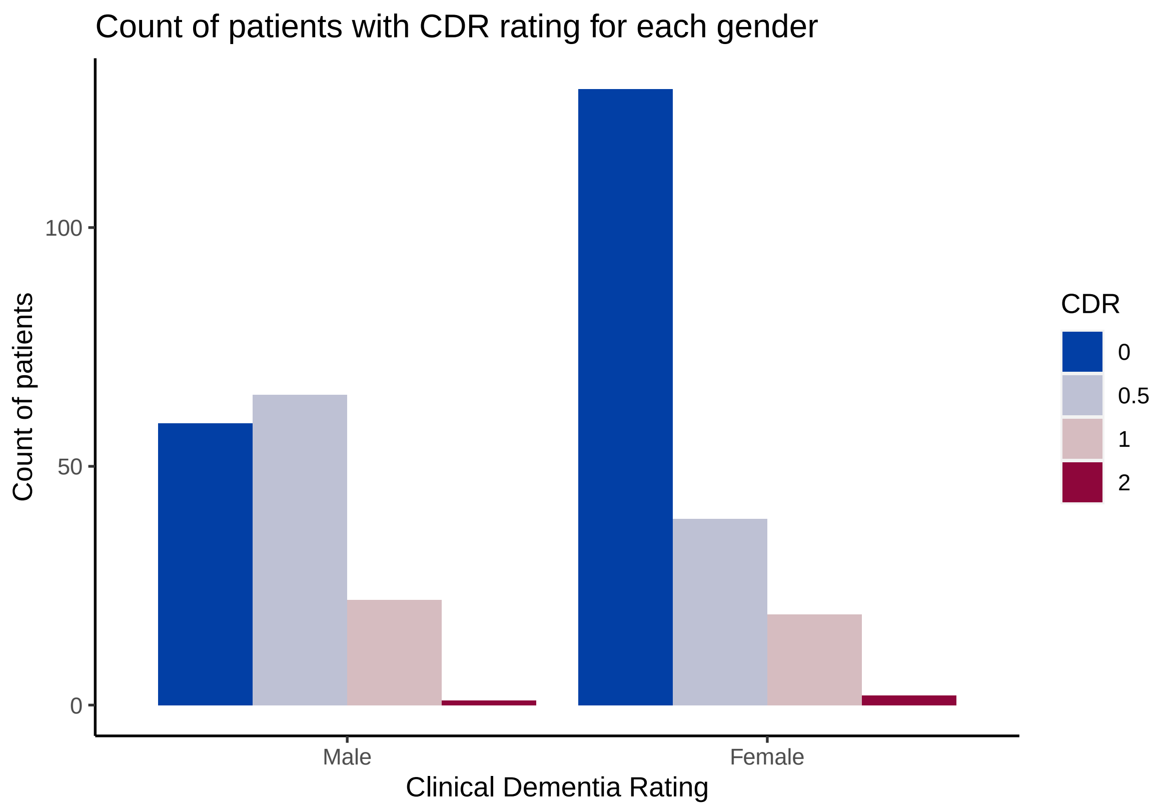
```
# get the median of MMSE
median_MMSE <- median(raw_data$MMSE, na.rm = TRUE)

# replace the NA values with mode of SES (categorical column) and
↪  median of MMSE (continuous column)
raw_data <- raw_data %>% replace_na(list(SES = mode_SES, MMSE =
↪  median_MMSE))

# Standardize the volume columns to have same scale

# function to calculate the mean and standard deviation of column
calculate_mean_sd <-  function(column_name) {
  column_name <- as.numeric(column_name)
  mean_sd_list <- list("mean_col" = mean(column_name), "sd_col" =
↪  sd(column_name))
  return(mean_sd_list)
}

# get the mean and sd
eTIV_mean_sd <- calculate_mean_sd(raw_data$eTIV)
nWBV_mean_sd <- calculate_mean_sd(raw_data$nWBV)
asf_mean_sd <- calculate_mean_sd(raw_data$ASF)

# calculate the standardized value of every data point of the column
raw_data$eTIV <- (raw_data$eTIV - eTIV_mean_sd$mean_col)/
↪  eTIV_mean_sd$sd_col
raw_data$nWBV <- (raw_data$nWBV - nWBV_mean_sd$mean_col)/
↪  nWBV_mean_sd$sd_col
raw_data$ASF <- (raw_data$ASF - asf_mean_sd$mean_col)/
↪  asf_mean_sd$sd_col
```

Table 2: Cleaned data

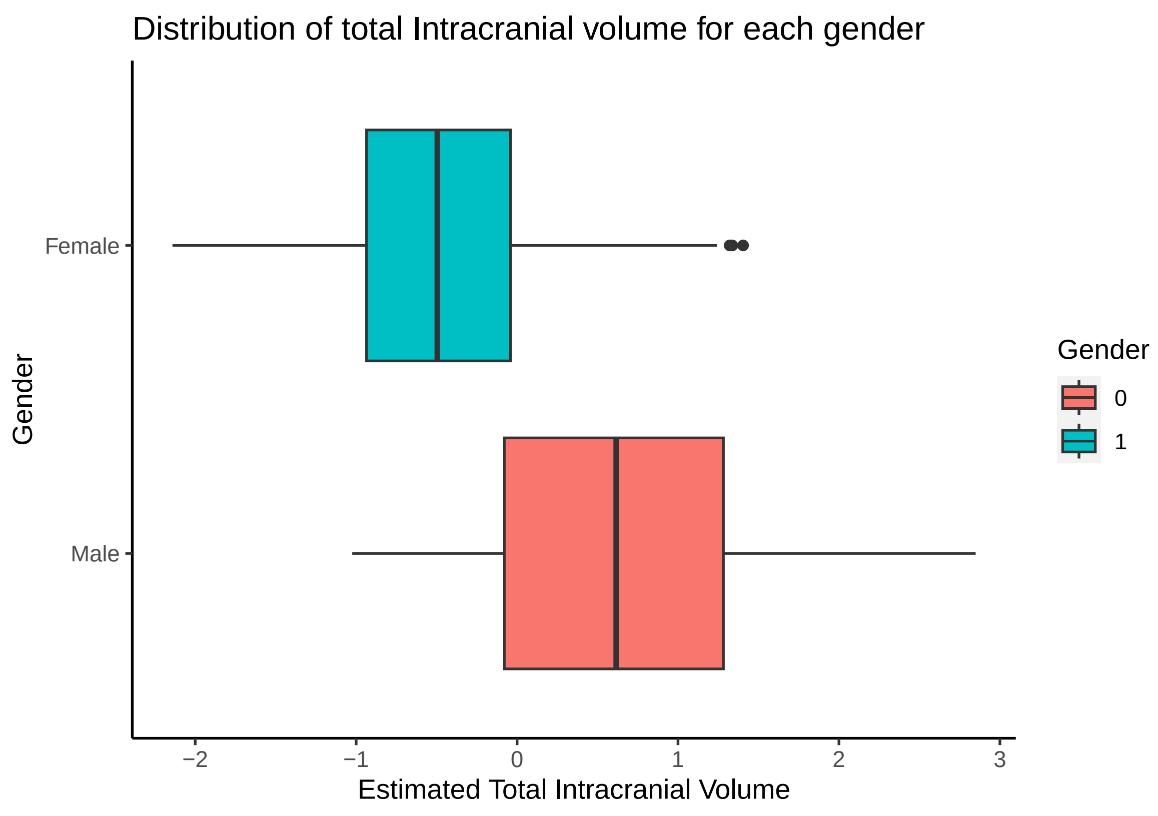
| Group | Gender | Age | EDUC | SES | MMSE | CDR | eTIV | nWBV | ASF |
|---|---|---|---|---|---|---|---|---|---|
| 0 | 0 | 87 | 14 | 2 | 27 | 0 | 2.754767 | -0.9168807 | -2.204152 |
| 0 | 0 | 88 | 14 | 2 | 30 | 0 | 2.849242 | -1.3188884 | -2.253827 |
| 1 | 0 | 75 | 12 | 2 | 23 | 0.5 | 1.037530 | 0.1551399 | -1.047416 |

```r
# convert the CDR variable to factor type
raw_data$CDR <- as.factor(raw_data$CDR)

# make a copy of the cleaned data
clean_data <- raw_data

kable(clean_data[1:3, ], caption = "Cleaned data") %>%
  kableExtra::kable_styling(bootstrap_options = "bordered")
```

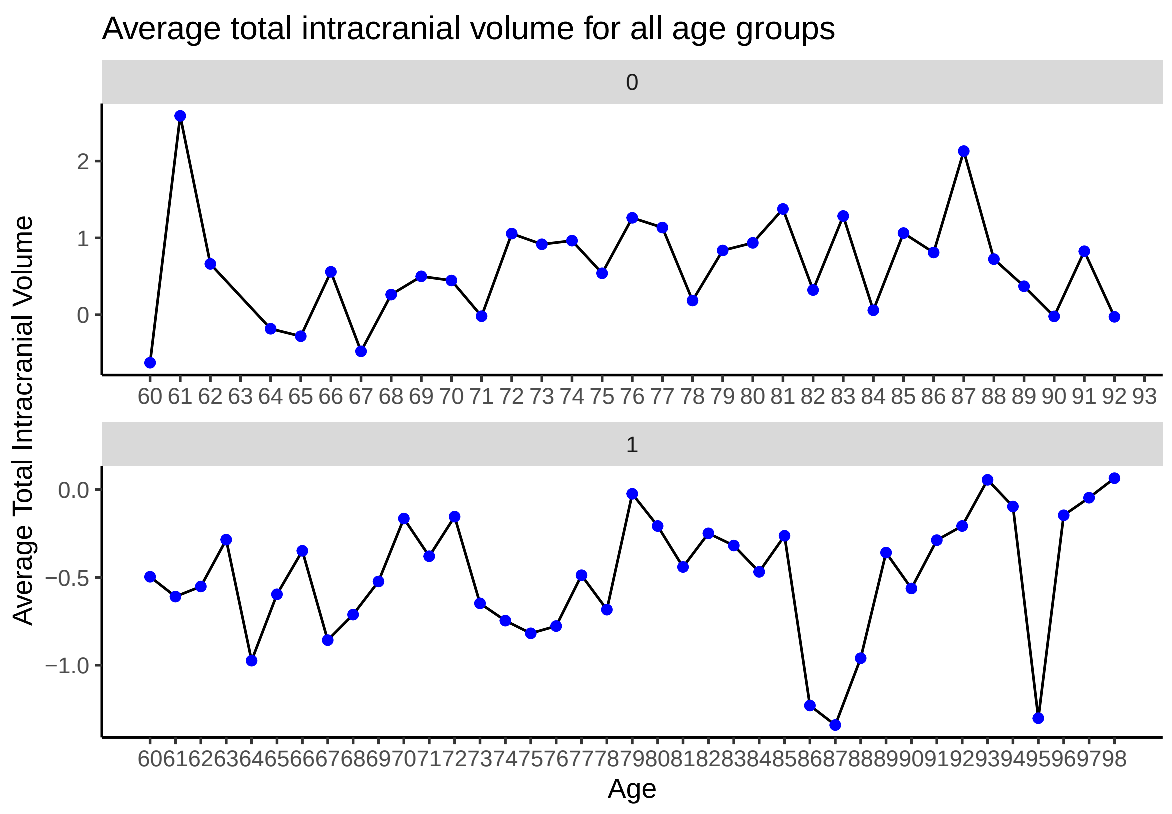
```r
# DESCRIPTIVE ANALYSIS

# Number of patients with different CDR for each gender
clean_data %>%
  select(c("Gender", "CDR")) %>%
  group_by(Gender, CDR) %>%
  summarise(count_n = n()) %>%
  ggplot(aes(x = Gender, y = count_n, fill = CDR)) +
  geom_bar(stat = "identity", position = position_dodge()) +
  scale_fill_manual(values = colorspace::diverge_hcl(4)) +
  scale_x_discrete(labels = c("Male", "Female")) +
  theme(
    panel.background = element_blank(),
    panel.grid.major = element_blank(),
    panel.grid.minor = element_blank(),
    axis.line = element_line(colour = "black")
  ) +
  labs(title = "Count of patients with CDR rating for each gender",
       x = "Clinical Dementia Rating", y = "Count of patients")
```



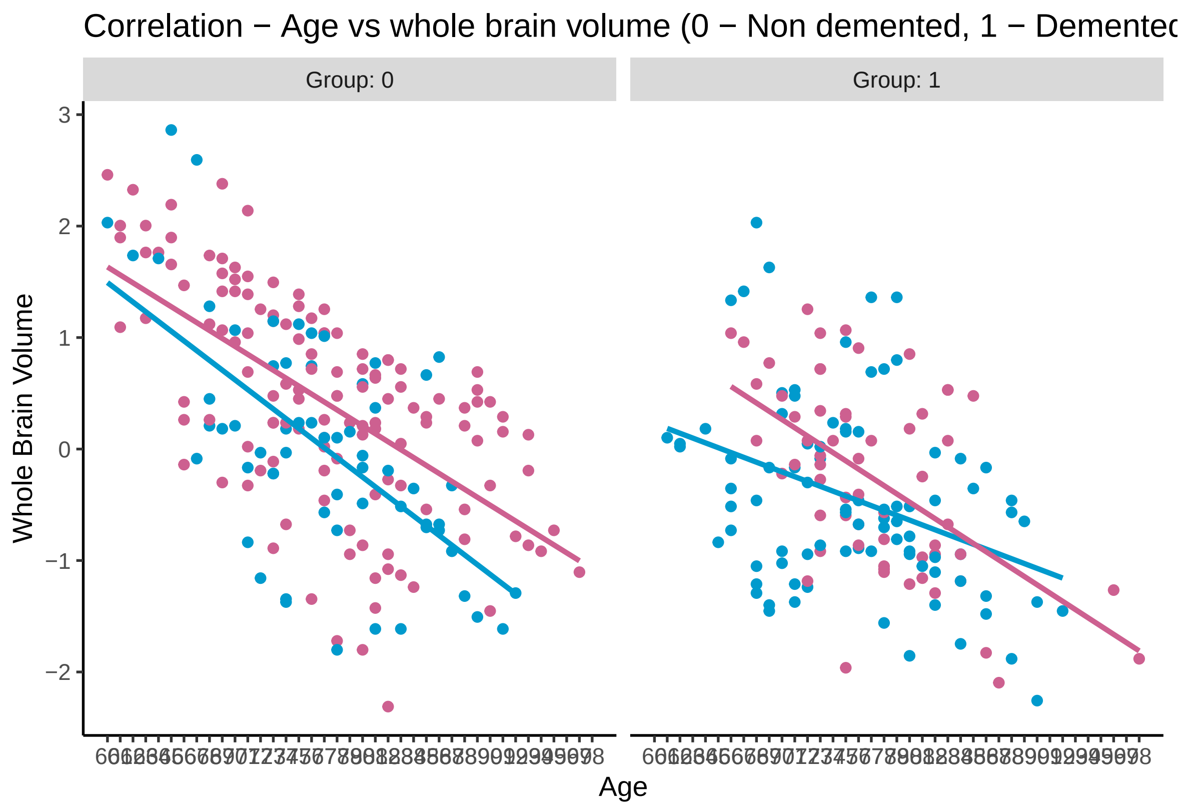
```
# distribution of total intracranial volume for each gender
clean_data %>%
  group_by(Gender) %>%
  ggplot(aes(x=Gender, y=eTIV, fill=Gender)) +
  geom_boxplot() +
  scale_x_discrete(labels = c("Male", "Female")) +
  coord_flip() +
  theme(
    panel.background = element_blank(),
    panel.grid.major = element_blank(),
    panel.grid.minor = element_blank(),
    axis.line = element_line(colour = "black")
  ) +
  labs(title = "Distribution of total Intracranial volume for each
  ↪  gender",
       x = "Gender", y = "Estimated Total Intracranial Volume")
```



```
# distribution of eTIV for all age groups
clean_data %>%
  group_by(Age, Gender) %>%
  summarise(avg_eTIV = mean(eTIV)) %>%
  ggplot(aes(x=Age, y=avg_eTIV, group=1)) +
  geom_line() + geom_point(color="blue") +
  scale_x_continuous(breaks = unique(clean_data$Age)) +
  theme(
    panel.background = element_blank(),
    panel.grid.major = element_blank(),
    panel.grid.minor = element_blank(),
    axis.line = element_line(colour = "black")
  ) +
  labs(title = "Average total intracranial volume for all age groups",
       x = "Age", y = "Average Total Intracranial Volume") +
  facet_wrap(vars(Gender), scales = "free", nrow = 2)
```



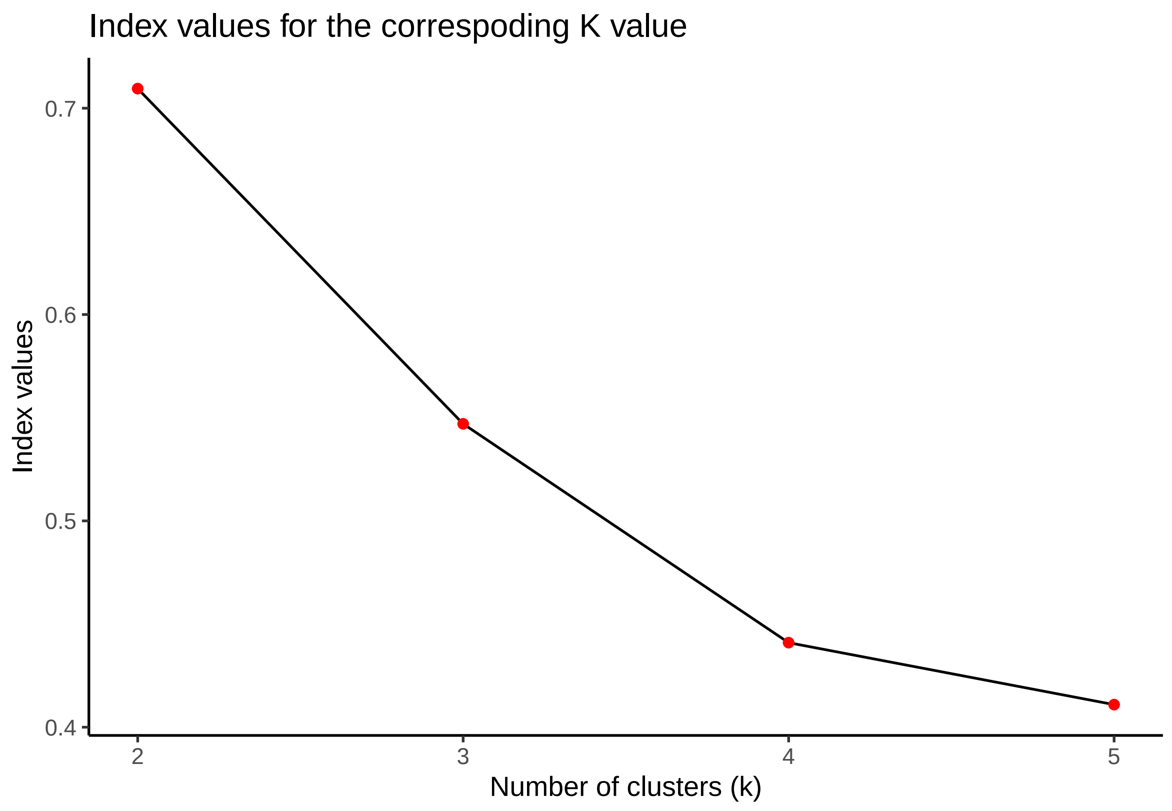
```
# scatter plot between Age and nWBV for each gender group
clean_data %>%
   ggplot(aes(x=Age, y=nWBV, color=Gender)) +
   geom_point() +
   geom_smooth(method = lm, se = FALSE) +
   scale_color_manual(values = c("deepskyblue3", "hotpink3"),
                      labels = c("Male", "Female")) +
   scale_x_continuous(breaks = unique(clean_data$Age)) +
   theme(
     panel.background = element_blank(),
     axis.line = element_line(colour = "black"),
     legend.position = "none"
   ) +
   labs(title = "Correlation - Age vs whole brain volume (0 - Non
    ↪  demented, 1 - Demented)",
        x = "Age", y = "Whole Brain Volume") +
   facet_wrap(vars(Group), labeller = "label_both")
```



```
# CLUSTERING ALGORITHM

# maximum clusters we want to form
max_number_of_clusters <- 5

# vector to store the values of index returned by the method for each
↪  k value
index_values <- numeric(max_number_of_clusters - 1)

# function to calculate the index for each k value and choose the
↪  optimal k
get_optimal_cluster <- function(data, max_clust) {

  # iterate over k values from 2 to max value
  for (k in 2:max_clust) {
    # perform fuzzy clustering
    cluster_result <- Fclust(data.matrix(data), k)
    # get the index value for the used k value
    index_values[k-1] <- Fclust.index(cluster_result)
  }

  # create a data frame containing clusters and index values
  index_data <- data.frame(clusters = 2:max_clust, indices =
↪   index_values)

  # select the k value with maximum index
  optimal_index <- which.max(index_values) + 1

  # return the data frame and chosen optimal k
  return(list(optimal_value = optimal_index, index_df = index_data))

}

# get the output returned by function
result <- get_optimal_cluster(clean_data, max_number_of_clusters)
```

The standard FkM algorithm has been chosen The default options have been set, to specify different options, use FKM The default value alpha=1 has been set for computing SIL.F The standard FkM algorithm has been chosen The default options have been set, to specify different options, use FKM The default value alpha=1 has been set for computing SIL.F The standard FkM algorithm has been chosen The default options have been set, to specify different options, use FKM The default value alpha=1 has been set for computing SIL.F The standard FkM algorithm has been chosen The default options have been set, to specify different options, use FKM The default value alpha=1 has been set for computing SIL.F

```r
# optimal k value returned by function
optimal_k <- result$optimal_value

# data frame returned by function with k and index values
cluster_index_df <- result$index_df

# plot the clusters and corresponding indices
ggplot(cluster_index_df, aes(x = clusters, y = indices)) +
  geom_line(linewidth = 0.5) +
  geom_point(color = "red") +
  labs(title = "Index values for the correspoding K value",
       x = "Number of clusters (k)",
       y = "Index values") +
  theme_classic()
```

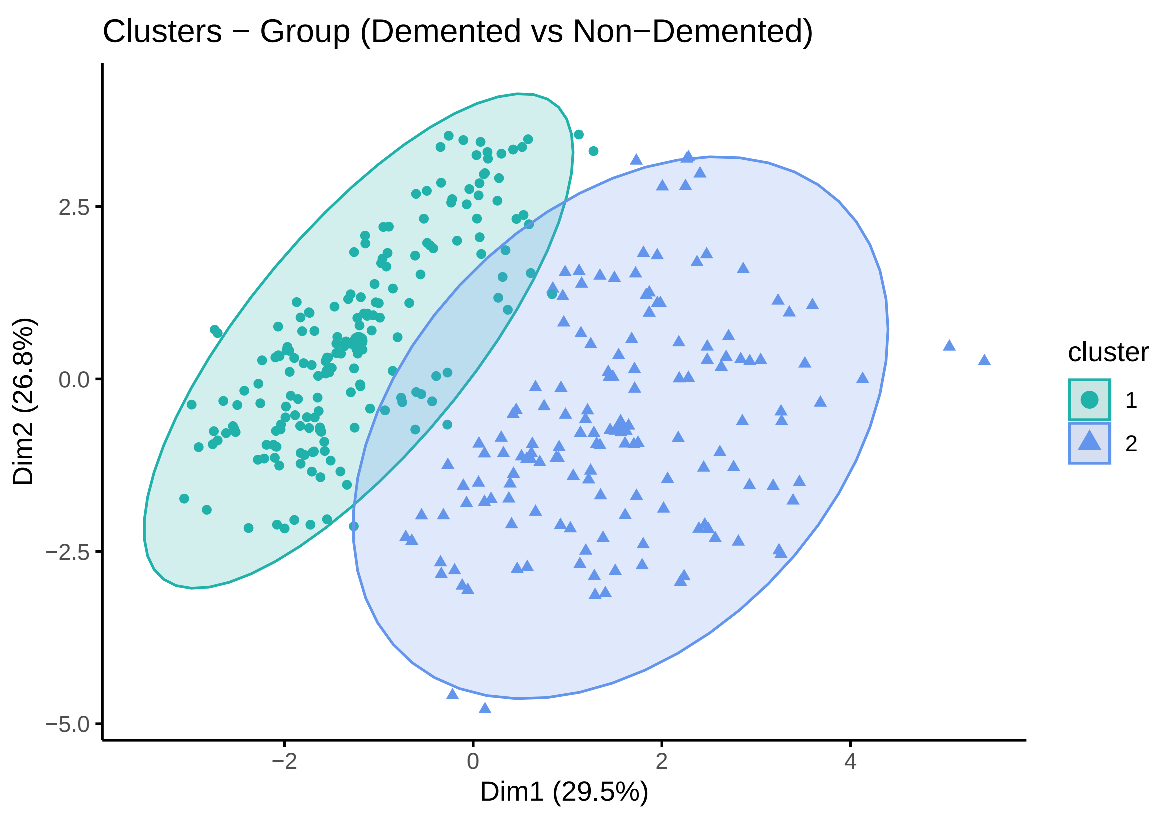


```
cat("Optimal value of k: ", optimal_k)
```

Optimal value of k: 2

```
# Perform fuzzy k-means clustering
clustering_FKM <-  function(data, k_clusters){
  fkm_output <- fanny(data.matrix(data),
                      k =k_clusters,
                      stand = T,
                      memb.exp = 1.15
                    )


  return(fkm_output)
}
```

```r
# call the function for clustering algorithm
fuzzy_k_means_result <- clustering_FKM(clean_data, optimal_k)



# plot the clusters
fviz_cluster(
    object = fuzzy_k_means_result,
    geom = "point",
    ellipse.type = "norm",
    show.clust.cent = T,
    repel = T,
    palette = c("lightseagreen", "cornflowerblue"),
    main = "Clusters - Group (Demented vs Non-Demented)",
    ggtheme = theme(
      panel.background = element_blank(),
      panel.grid.major = element_blank(),
      panel.grid.minor = element_blank(),
      axis.line = element_line(colour = "black")
    )
  )
```



```r
# LOGISTIC REGRESSION

# convert Group variable to factor type
clean_data$Group <- as.factor(clean_data$Group)

# SPLIT INTO TRAIN/TEST DATA

# set random state for splitting data
set.seed(345)

# split data into training and test set
data_split <- initial_split(data = clean_data,
                            prop = 0.8,
                            strata = Group)
```

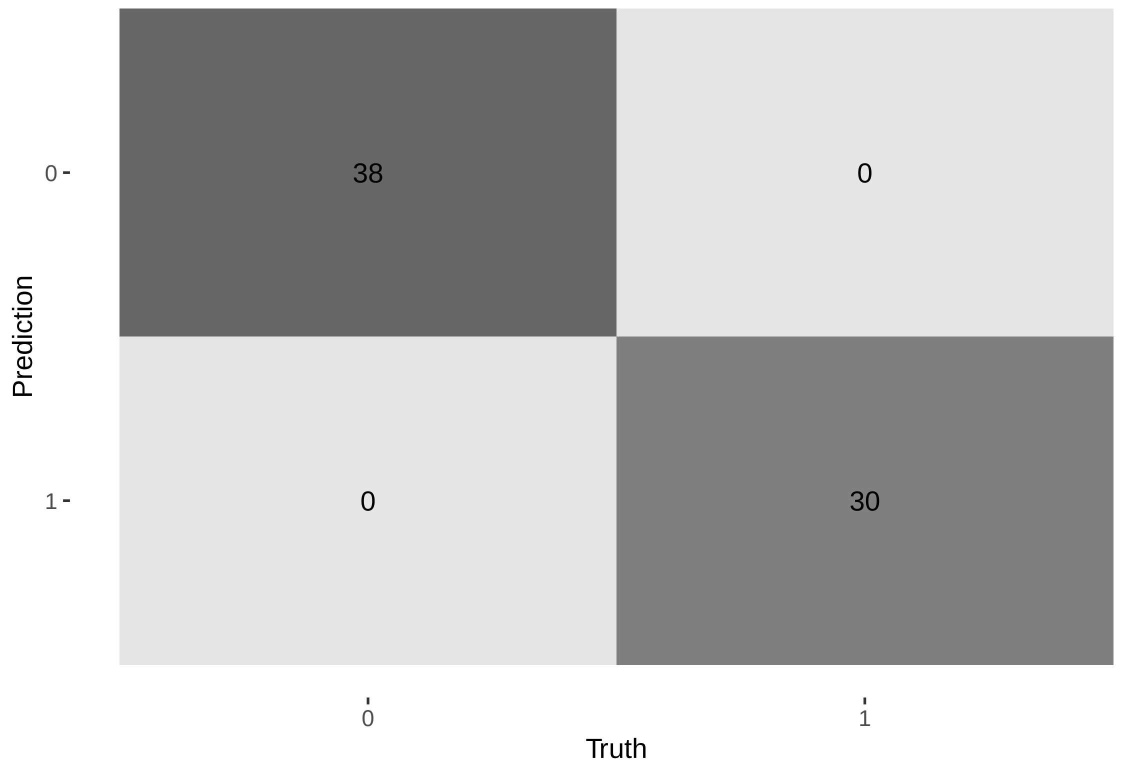
```r
# create the train and tests data set
train_data <- data_split %>% training()

test_data <- data_split %>% testing()

# HYPER PARAMTER TUNING

# hyper-parameter tuning for `mixture` and `penalty`

# define the logistic regression model
logis_model <- logistic_reg(mixture = tune(),
                            penalty = tune(),
                            engine = "glmnet")

# define a grid search for hyper-parameters
grid_Search <- grid_regular(mixture(),
                            penalty(),
                            levels = c(mixture = 4, penalty = 4))

# define the workflow for parameter tuning and building model
log_reg_workflow <- workflow() %>%
  add_model(logis_model) %>% # add the above created model
  add_formula(Group ~ .) # set the formula: response variable ~
   ↪  predictors

# define cross validation method for grid search
cv_folds <- vfold_cv(train_data,
                     v = 7)
```

```
# tune the parameters using the above defined methods
tune_logis_reg <- tune_grid(
  object = log_reg_workflow,
  resamples = cv_folds,
  grid = grid_Search,
  control = control_grid(save_pred = TRUE)
)

# choose the best values for the logistic regression model parameters
chosen_params <- select_best(tune_logis_reg, metric = "accuracy")

# print the optimal values of mixture and penalty
cat("Mixture: ", chosen_params$mixture, " ", "Penalty: ",
↪  chosen_params$penalty)
```

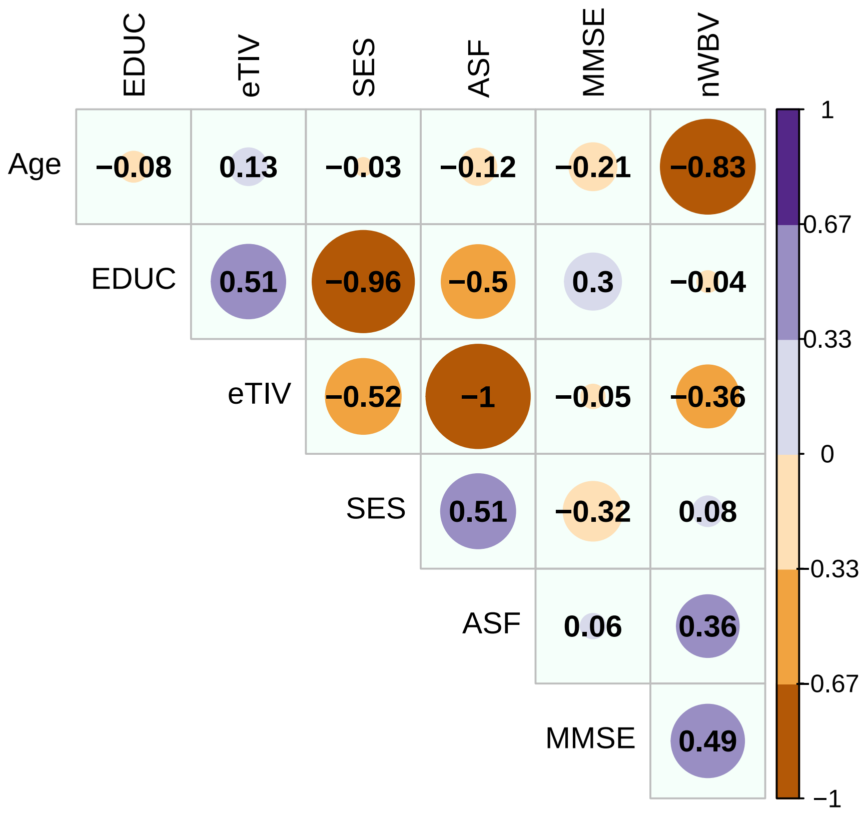
Mixture: 0 Penalty: 1e-10

```
# LOGISTIC REGRESSION MODEL

# fit the model with the chosen parameter values
final_model <- logistic_reg(
  mixture = chosen_params$mixture,
  penalty = chosen_params$penalty) %>%
  set_engine("glmnet") %>%
  set_mode("classification") %>%
  fit(Group ~ ., data = train_data)



# Predict the results on test data
```

```
pred_results <- predict(final_model,
                        new_data = test_data,
                        type = "class")



# combine the actual and predicted class variables in one data frame
predicted_class <- test_data %>%
  select("Group") %>%
  bind_cols(pred_results)

# print the confusion matrix
autoplot(conf_mat(predicted_class,
         truth = Group,
         estimate = .pred_class), type = "heatmap")
```



```
# FEATURE SELECTION

# correlation matrix
corrplot(
  cor(cor(data.matrix(clean_data[, -c(1, 2, 7)]))),
  type="upper",
  order = "hclust",
  col = brewer.pal(n=6, "PuOr"),
  bg = "mintcream",
  tl.col = "black",
  diag = FALSE,
  addCoef.col = "black"
)
```



```r
# find the redundant variables and remove them from the data
findCorrelation(cor(data.matrix(clean_data[, -c(1, 2, 7)])), cutoff =
↪  0.75)
```

[1] 5

```r
# Remove `eTIV` and `CDR` from the data
temp_data <- clean_data[,-c(7,8)]

temp_data$Group <- as.numeric(temp_data$Group)

# calculate the importance value for each feature
fs <- filterVarImp(x = temp_data[, 2:8],
                   y = temp_data[, 1])

# bind the features and corresponding score in a data frame
fs <-  data.frame(cbind(feature = rownames(fs), score =
↪  as.double(fs[,1])))

# print the significance of features and their corresponding scores
kable(fs[order(fs$score, decreasing = TRUE),], caption = "Feature
↪  scores")
```

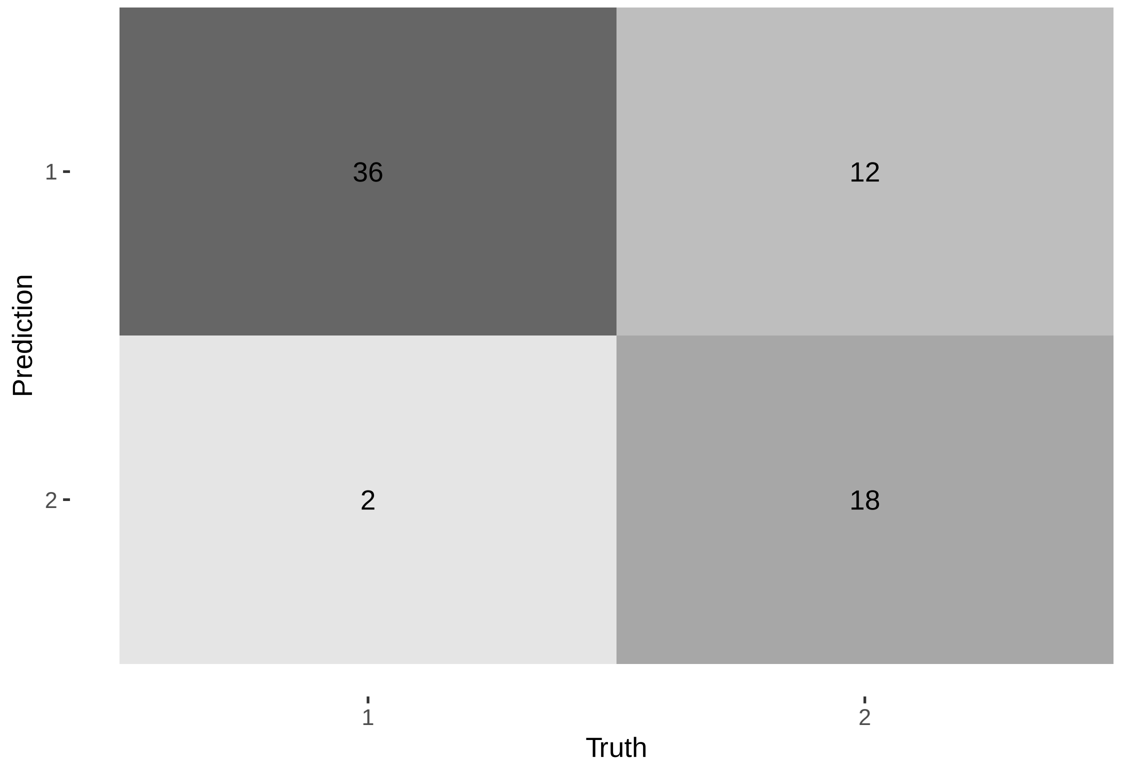
```r
# selecting the top 4 features as new predictors
new_data <- temp_data[, c("Group", "MMSE", "Age", "Gender", "EDUC")]

new_data$Group <- as.factor(new_data$Group)

# LOGISTIC REGRESSION WITH SELECTED FEATURES
```

Table 3: Feature scores

| | feature | score |
|---|---|---|
| 5 | MMSE | 7.90805966306536e-14 |
| 1 | Gender | 5.07937615597167 |
| 3 | EDUC | 0.712008028438323 |
| 2 | Age | 0.578213328044115 |
| 7 | ASF | 0.459674134140075 |
| 6 | nWBV | 0.38834676318609 |
| 4 | SES | 0.0717282178596737 |

```r
# set random state for splitting data
set.seed(45)

# split data into training and test set
new_split <- initial_split(data = new_data,
                            prop = 0.8,
                            strata = Group)

# create the train and tests data set
new_train_data <- new_split %>% training()

new_test_data <- new_split %>% testing()

# train the logistic regression model with the selected features
log_model2 <- logistic_reg(
  mixture = chosen_params$mixture,
  penalty = chosen_params$penalty) %>%
  set_engine("glmnet") %>%
  set_mode("classification") %>%
  fit(Group ~ ., data = new_train_data)
```

```r
# Predict the results on test data
pred_results2 <- predict(log_model2,
                         new_data = new_test_data,
                         type = "class")

# combine the actual and predicted class variables in one data frame
predicted_class2 <- new_test_data %>%
  select(Group) %>%
  bind_cols(pred_results2)

# print the confusion matrix
autoplot(conf_mat(predicted_class2,
         truth = Group,
         estimate = .pred_class), type = "heatmap")
```



```
# MODEL COMPARISON BEFORE_AFTER FEATURE SELECTION

# Logistic regression model with all variables as predictors
model1 <- glm(Group~., family = "binomial", data = train_data)

modelsummary(model1,
             output = "kableExtra",
             stars = T,
             title = "Summary of model with all variables as
              ↪  predictors")
```

```
# Logistic regression model with only selected features
model2 <- glm(Group~., family = "binomial", data = new_train_data)
```

Table 4: Summary of model with all variables as predictors

| | (1) |
|---|---|
| (Intercept) | 560.800 |
| | (451 792.019) |
| Gender1 | −29.741 |
| | (42 440.148) |
| Age | −4.978 |
| | (4341.988) |
| EDUC | 4.199 |
| | (7226.179) |
| SES | −19.210 |
| | (19 228.826) |
| MMSE | −9.334 |
| | (12 368.150) |
| CDR0.5 | 162.938 |
| | (69 732.595) |
| CDR1 | 119.415 |
| | (86 617.978) |
| CDR2 | 92.232 |
| | (278 983.324) |
| eTIV | −97.069 |
| | (140 070.423) |
| nWBV | −23.093 |
| | (35 946.980) |
| ASF | −59.665 |
| | (157 479.931) |
| Num.Obs. | 268 |
| AIC | 24.0 |
| BIC | 67.1 |
| Log.Lik. | 0.000 |
| F | 0.000 |
| RMSE | 0.00 |

+ $p < 0.1$, * $p < 0.05$, ** $p < 0.01$, *** $p < 0.001$

Table 5: Summary of model with only selected variables as predictors

| | (1) |
|---|---|
| (Intercept) | 37.309*** |
| | (5.522) |
| MMSE | −1.082*** |
| | (0.153) |
| Age | −0.061* |
| | (0.027) |
| Gender1 | −1.010** |
| | (0.379) |
| EDUC | −0.144* |
| | (0.073) |
| Num.Obs. | 268 |
| AIC | 191.8 |
| BIC | 209.7 |
| Log.Lik. | −90.882 |
| F | 14.360 |
| RMSE | 0.33 |

+ $p < 0.1$, * $p < 0.05$, ** $p < 0.01$, *** $p < 0.001$

```
modelsummary(model2,
             output = "kableExtra",
             stars = T,
             title = "Summary of model with only selected variables as
              ↪  predictors")
```

```
# comparison of models using anova testing
anova(model1, model2, test = "LR") %>%
  tidy() %>%
  mutate(
    p.value = scales::pvalue(p.value),
    term = c("Model 1", "Model 2")
  ) %>%
```

Table 6: ANOVA summary

| term | df.residual | residual.deviance | df | deviance | p.value |
|---|---|---|---|---|---|
| Model 1 | 256 | 0.0000 | NA | NA | NA |
| Model 2 | 263 | 181.7642 | -7 | -181.7642 | <0.001 |

```
kable(
  caption = "ANOVA summary"
) %>%
kableExtra::kable_styling(bootstrap_options = "bordered")
```