Classification of the Alzheimer's Disease Using a SVM Model*

Kaggle Competition Report

Rahma Binth Mohammad - 1006327226
Kaggle Username - rahmabm Final Private Score - 0.91346
Final Kaggle Ranking - 84th

This report presents a supervised machine learning approach to classify individuals diagnosed with Alzheimer's disease. After data preprocessing and feature selection, we evaluated several algorithms and selected the best-performing model, a Support Vector Machine (SVM) based on predictive accuracy. Our final model demonstrates strong classification performance and provides insights into the most informative features associated with the disease.

Table of contents

*Code and data are available at: https://github.com/rahmabinth/alz_classification.

1 Introduction

There were over 55 million people worldwide living with dementia in 2020 and by 2050, it is projected to affect 139 million people (International, n.d.). Given the scale of this health issue, it is important to identify the characteristics and lifestyle choices of the people it affects in order to identify patterns. These patterns can assist with early detection, which would allow for better prevention. Thus, this paper focuses on predicting Alzheimer's Disease in patients using a dataset from Kaggle that contains data collected directly from patients. We treated the diagnosis as a binary outcome (1 = Yes, 0 = No) and used machined learning algorithms to find the best model for performance. Our final and best performing model achieved a private score of 0.91346 and ranked 84th on the leaderboard. It allows for early detection of Alzheimers and demonstrates the potential of machine learning in highlighting key predictive factors in patient data.

The remainder of this paper is structured as follows. Section 2 describes the dataset, including how the variables were measured, the data preprocessing and feature engineering steps, as well as the final selection of features. Section 3 presents the modeling approach and details the model that achieved the best predictive performance. Section 4 provides our findings and the evaluation metrics for the final model used such as accuracy, precision, and recall. Section 5 outlines the limitations of our work and improvements for future work.

2 Data

2.1 Overview

We conducted our complete analysis using the statistical programming language `R` (R Core Team 2023) and packages `patchwork` (Pedersen 2024), `dplyr` (Wickham et al. 2023), `caret` (Kuhn and Max 2008), `ggplot2` (Wickham 2016), `e1071` (Meyer et al. 2024), `pROC` (Robin et al. 2011), and `tidyverse` (Wickham et al. 2019). Our data was pulled from the Kaggle Competition called Classification of the Alzheimer's Disease dataset (Bing 2024). The dataset provides patient data for 2149 patients. Each row contains a different individuals patient data with 33 different variables (Bing 2024). It contains demographic details, lifestyle factors, medical history, cognitive assessments, and symptoms for each patient.

2.2 Measurement

While the dataset is based on real-world patient assessments and health records, certain variables are expressed as measurable features with the use of standardized clinic tests. This includes several of the cognitive and function assessments in the reported data such as Mini-Mental State Examination (MMSE) Score, Functional Assessment Score, and Behavioural Problems. Others have been self-reported as binary variables, such as the symptoms. Variables such as `Confusion`, measured the presence of confusion, along with variables like `Disorientation` and `Forgetfulness`.

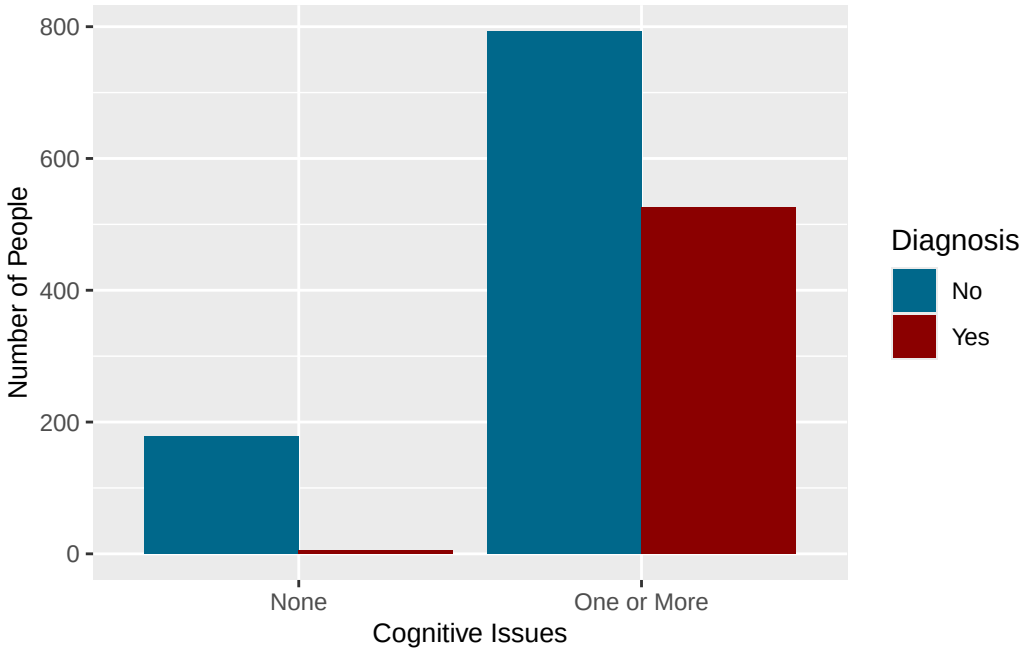

Figure 1: Diagnosis vs. Cognitive Issues

2.3 Data Preprocessing and Feature Engineering

We started by inspecting the data for missing values and irrelevant variables. We found variables such as the `DoctorInCharge`, and `PatientID` to be irrelevant to our problem and removed them. This also allowed for randomness in our data. We looked for patterns within the data by segregating the data into separate categories. We identified those with any one type of cognitive issue and looked for a pattern within those diagnosed and undiagnosed with Alzeheimers. As mentioned earlier, these issues were found through cognitive and functional assessments. They include `MMSE`, `FunctionalAssessment`, `MemoryComplaints`, `BehaviourProblems`, `ADL`. We found that 88% of the patients had one or more cognitive issue and approximately 40% of

them were diagnosed with the diseases. The most significant find was that only 1% of those undiagnosed with the disease had one or more cognitive issues. See Figure 1. This indicated a very high chance that if a patient had a cognitive issue, they were more likely to have the disease.

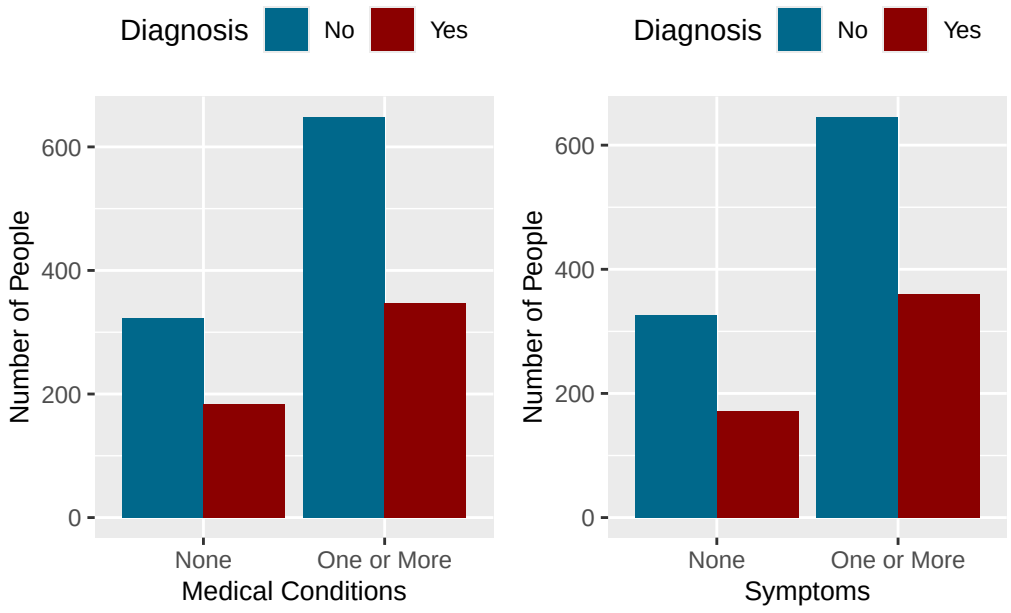

Figure 2: Diagnosis vs. Medical History and Diagnosis vs. Symptoms

We did the same with medical history. If the patient had one or more of the the following, we grouped them, a family history of the disease (`FamilyHistoryAlzheimers`), presence of cardiovascular disease (`CardiovascularDisease`), presence of diabetes (`Diabetes`), presence of depression (`Depression`), history of head injury (`HeadInjury`), presence of hypertension (`Hypertension`). We found approximately 66% of the patients had one or more medical condition and 35% of them were diagnosed with Alzheimer's. It was also shown that 35% of those diagnosed didn't have one or more of these medical condition. It was less conclusive then those with cognitive issues but we found it relevant to test out models with this feature later on, since a significant number of those with the condition had the disease. We saw a similar pattern with patient symptoms, where we grouped together those with one or more of the symptoms, `Confusion`, `Disorientation`, `PersonalityChanges`, `DifficultCompletingTasks`, and `Forgetful`. See Figure 2.

2.4 Feature Selection

In addition, we looked at how each patient feature played into the presence of the diseases (`Diagnosis`). Section A provides the code to replicate this by changing the name for the feature variable. Our analysis revealed that certain features had more of an impact than others. For example, by examining all the variables, we saw 61% of those diagnosed with the diseases were Caucasians, and 61% of those diagnosed either didn't attend high school or only attended high school and didn't hold a Bachelor's Degree or Higher Education. However, gender didn't seem to play a role as the number of men and women diagnosed were approximately the same.

3 Model

Our initial goal for our modelling strategy was twofold, balancing interpretability and predictive accuracy. With this in mind, we used the three main factors mentioned in Section 2.4, cognitive issues, medical conditions, and symptoms, and created a logistic regression model. We used cross validation method to train and test data but failed to get higher than 62% predictability rate. Hence, we prioritized predictive performance by choosing to do a Support Vector Machine (SVM), which offered limited interpretability. The SVM performed significantly better than our previous tested models. However, interpretability decreased. Section 5 identifies this tradeoff.

3.1 Model Set-up

We define the outcome variable $y_i \in \{0, 1\}$ as the diagnosis of Alzheimer's Disease, where $y_i = 1$ indicates a positive diagnosis and $y_i = 0$ indicates a negative diagnosis. To predict this outcome, we trained a Support Vector Machine (SVM) classifier using the following selected predictor variables where the feature for individual i is denoted by $x_i = (x_1, x_2, ..., x_p)$.

x_1: Ethnicity, x_2: Gender, x_3: MMSE score (Mini-Mental State Examination score), x_4: MemoryComplaints, x_5: ADL (Activities of Daily Living score), x_6: FunctionalAssessment, x_7: BehavioralProblems, x_8: HeadInjury, x_9: Forgetfulness

The model estimates a function $f(\mathbf{x}_i)$ such that:

$$\hat{y}_i = \begin{cases} 1 & \text{if } f(\mathbf{x}_i) \geq 0 \\ 0 & \text{if } f(\mathbf{x}_i) < 0 \end{cases}$$

We run the model in R (R Core Team 2023) using the `e1071` package (Meyer et al. 2024). In our SVM model, $f(\mathbf{x})$ is the decision function, it tells us how far a point is from the decision boundary, the separating surface between classes.

If $f(\mathbf{x}_i) \geq 0$, the model predicts class 1 (positive diagnosis).
If $f(\mathbf{x}_i) < 0$, it predicts class 0 (negative diagnosis).

3.2 Model Justification

We chose the SVM model for its strong performance in binary classification, particularly in high-dimensional settings. Given our dataset and the various factors provided in the dataset that affected the diagnosis, the high-dimensional settings proved useful. However, the relationship between the patient features and the Alzheimer's diagnosis was not linear, as some had more of an impact. Features like MMSE assessment score had a much stronger impact on the diagnosis than the ethnicity of the patient. Thus, we chose to do a Radial Basis Function (RBF) kernel rather than a linear one as it provided about 9% more accuracy by capturing the complex interactions of the features. While SVMs lack the interpretability of simpler models, their predictive accuracy made them a suitable choice for this task.

4 Results

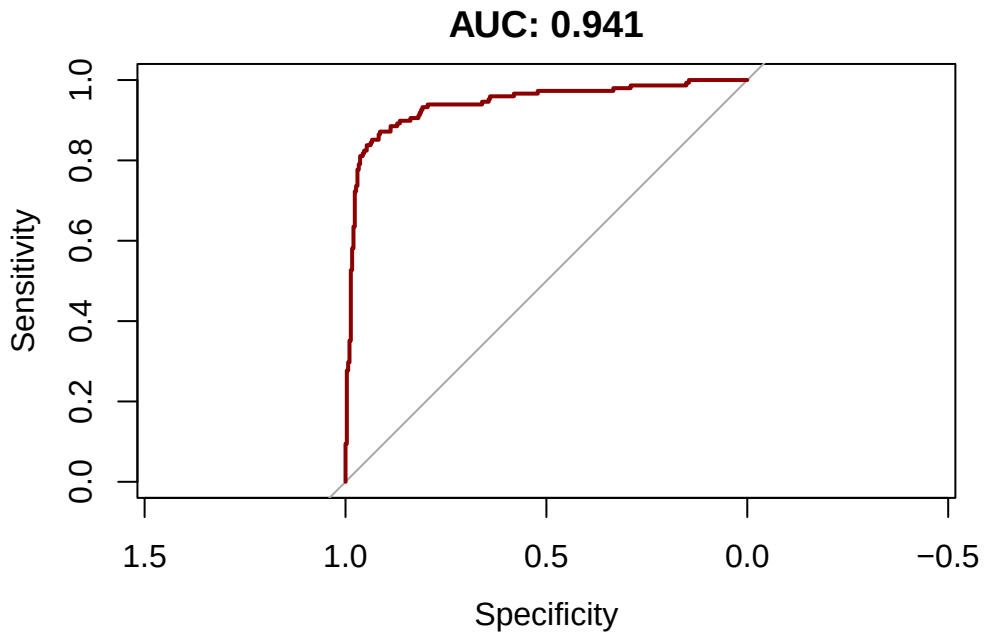

Figure 3: ROC curve

4.1 Evaluation Metrics and Model Performance

Before submitting on Kaggle, we worked with the "train_csv" file and split it into two, 70% of the data was used for training and 30% used for validation. Our validation set proved useful when testing the models. Hyperparameter tuning via 10-fold cross validation helped identify the optimal parameters, cost = 10 and gamma = 0.1. We found these numbers by tuning the SVM with different ranges and finding the best one. Section A provides the code for this. These parameters provided a cross-validated error rate of 0.0997, which shows that the model misclassifies about 10.26% of observations and will do so on new and similar data.

Our SVM classifier with these parameters achieved an accuracy of 89% and sensitivity (recall) of 90.1% (accurately identifying the undiagnosed patients) and specificity (precision) of 87% (accurately identifying the diagnosed patients) on the validation set. Figure 3 is the ROC curve that signifies the relationship and plots how well our model separates the two classes. The area under the curve (AUC) of 0.941 shows that our model has excellent ability to distinguish between Alzheimer's-positive and Alzheimer's-negative patients.

4.2 Feature Contribution to Diseases

As mentioned in Section 2.4, all the variables were examined to see a how they interacted with the diagnosis variable. However, when we simply inputted those features into the SVM, it only gave us an accuracy of 86%. To improve our model and better understand each feature's importance, we conducted a leave-one-feature-out sensitivity check, essentially removing one feature at a time, retraining the model, and checking how the accuracy changes. Using this, we found certain features to be extremely important to the model's predictive ability. For example, MMSE and FunctionalAssessment were very important. This check also allowed us identify which features to retain and exclude. The final set of features used is listen in Section 3.1 and reflects the most significant contributors to the Alzheimer's disease. Section 5 will examine the relevance of these features.

5 Discussion

As mentioned, using feature elimination, we were able to identify the most predictive features of the diseases. Patients with low MMSE scores, greater difficulty in daily living tasks (`ADL`) and symptoms such as `Forgetfulness` were significantly more likely to be diagnosed with Alzheimer's. These results suggest that cognitive assessments and behavioral symptoms are strong indicators of diagnosis.

While our SVM model demonstrated strong predictive performance, there are several limitations to consider. The model is not easily interpretable ue to the use of a non-linear SVM, making it difficult to understand why a specific prediction was made. We accepted this trade-off for higher accuracy. Also, the dataset contains other patient features that most doctors

and scientists consider to be vital information in predicting alzheimers or affecting diagnosis, but was ignored by our model, as to improve predictive performance. These include features like `FamilyMedicalHistory`. Additionally, we attempted to balance the classes using "class.weights" as best as we can. See Section A. However, class imbalance may still influence model bias. To improve our work, we could focus on gathering and mining data for patients over time. This could potentially improve predictive power and provide insights into disease progression.

A Appendix

The following is the code used for everything in the analysis. Please refer to the GitHub repository for clarity regarding the different R scripts. Each R script is divided up with a preamble below. This report was created using a Quarto script which can also be found in the GitHub repository.

https://github.com/rahmabinth/alz_classification

```
#### Preamble ####
# Purpose: Installs the necessary packages.
# Author: Rahma Binth Mohammad
# Contact: rahma.binthmohammad@mail.utoronto.ca
# License: MIT

#### Download packages ####
install.packages("e1071")
install.packages("ggplot2")
install.packages("dplyr")
install.packages("caret")
install.packages("patchwork")
install.packages("pROC")
```

```
#### Preamble ####
# Purpose: Cleans Data, Finds Trends, and Tests Different Models
# Author: Rahma Binth Mohammad
# Contact: rahma.binthmohammad@mail.utoronto.ca
# License: MIT
# Pre-requisites: Run the "00-install.packages.R "file

# Load packages, install beforehand if needed
library(dplyr)
library(ggplot2)
library(caret)
library(e1071)

# Read in the data
train_data <- read.csv("data/01-raw_data/train.csv", stringsAsFactors = FALSE)

test_data <- read.csv("data/01-raw_data/test.csv", stringsAsFactors = FALSE)

#Check for NA values, none found
colSums(is.na(train_data))
```

```r
colSums(is.na(test_data))

# Remove the 'DoctorInCharge' column, irrelevant to our problem
train_data <- select(train_data, -DoctorInCharge)
test_data <- select(test_data, -DoctorInCharge)

#Check data and trends

# Create a subset of data where Diagnosis indicates Yes (1)
alzheimers_data <- subset(train_data, Diagnosis == 1)

# Summary of all columns when Diagnosis == 1
summary(alzheimers_data)

#Check if gender makes a difference
d_women <- subset(alzheimers_data, Gender == 1)
nrow(d_women)

d_men <- subset(alzheimers_data, Gender == 0)
nrow(d_men)

#Check if ethnicity makes a difference
d_caucasian <- subset(alzheimers_data, Ethnicity == 0)
nrow(d_caucasian)

d_african_am <- subset(alzheimers_data, Ethnicity == 1)
nrow(d_african_am)

d_asian <- subset(alzheimers_data, Ethnicity == 2)
nrow(d_asian)

d_other_eth <- subset(alzheimers_data, Ethnicity == 3)
nrow(d_other_eth)

#Check if education makes a difference
d_none_ed <- subset(alzheimers_data, EducationLevel == 0)
nrow(d_none_ed)

d_highschool <- subset(alzheimers_data, EducationLevel == 1)
nrow(d_highschool)

d_bachelors <- subset(alzheimers_data, EducationLevel == 2)
```

```r
nrow(d_bachelors)

d_higher_ed <- subset(alzheimers_data, EducationLevel == 3)
nrow(d_higher_ed)

#Check how Lifestyle Factors differ
d_smoking <- subset(alzheimers_data, Smoking == 1)
nrow(d_smoking)

d_non_smoking <- subset(alzheimers_data, Smoking == 0)
nrow(d_non_smoking)

d_alcohol <- subset(alzheimers_data, AlcoholConsumption > 2)
nrow(d_alcohol)

d_sleep <- subset(alzheimers_data, SleepQuality >9)
nrow(d_sleep)

#Check Family History
d_family <- subset(alzheimers_data, FamilyHistoryAlzheimers == 1)
nrow(d_family)

#Check Systolic BP
d_systolicBP <- subset(alzheimers_data, SystolicBP >95)
nrow(d_systolicBP)

set.seed(123)
# Bar plots to check certain Cognitive Assessments against the Diagnosis
ggplot(train_data, aes(x = as.factor(BehavioralProblems),
                       fill = as.factor(Diagnosis))) +
  geom_bar(position = "dodge") +
  labs(title = "Alzheimer's Diagnosis vs. BehavioralProblems",
       x = "BehavioralProblems (0 = No, 1 = Yes)",
       y = "Count",
       fill = "Diagnosis (0 = No, 1 = Yes)") +
  theme_minimal()

ggplot(train_data, aes(x = ADL, fill = as.factor(Diagnosis))) +
  geom_histogram(binwidth = 5, position = "dodge", alpha = 0.7) +
  labs(title = "Alzheimer's Diagnosis vs. ADL",
       x = "ADL",
       y = "Count",
```

```
       fill = "Diagnosis (0 = No, 1 = Yes)") +
  theme_minimal()

# Create a table with the number of people who smoke, diagnosed and undiagnosed
train_data %>%
  filter(Smoking == 1) %>%
  group_by(Diagnosis) %>%
  summarise(Count = n())

# Create a table with the number of people who drink, diagnosed and undiagnosed
train_data %>%
  filter(AlcoholConsumption > 15) %>%
  group_by(Diagnosis) %>%
  summarise(Count = n())

# Create a table with the number of people who have a functional impairement
# Finding is that Lower Scores = greater impairement
train_data %>%
  filter(FunctionalAssessment < 5) %>%
  group_by(Diagnosis) %>%
  summarise(Count = n())

# Create a table with the number of people who have a functional impairement
# Lower Scores = greater impairement
train_data %>%
  filter(BehavioralProblems == 1) %>%
  group_by(Diagnosis) %>%
  summarise(Count = n())

train_data %>%
  filter(BehavioralProblems == 0) %>%
  group_by(Diagnosis) %>%
  summarise(Count = n())

train_data %>%
  filter(ADL < 5) %>%
  group_by(Diagnosis) %>%
  summarise(Count = n())

train_data %>%
  filter(Confusion == 1) %>%
  group_by(Diagnosis) %>%
```

```r
  summarise(Count = n())

train_data %>%
  filter(Confusion == 0) %>%
  group_by(Diagnosis) %>%
  summarise(Count = n())

# Add a column that indicates if the person has at least one medical condition
train_data <- train_data %>%
  mutate(HasMedicalCondition = ifelse(FamilyHistoryAlzheimers == 1 |
                                        CardiovascularDisease == 1 |
                                        Diabetes == 1 |
                                        Depression == 1 |
                                        HeadInjury == 1 |
                                        Hypertension == 1, 1, 0))

# Add a column that indicates if the person has at least one medical condition
test_data <- test_data %>%
  mutate(HasMedicalCondition = ifelse(FamilyHistoryAlzheimers == 1 |
                                        CardiovascularDisease == 1 |
                                        Diabetes == 1 |
                                        Depression == 1 |
                                        HeadInjury == 1 |
                                        Hypertension == 1, 1, 0))

# Count the number of people in each group (HasMedicalCondition vs Diagnosis)
summary_data <- train_data %>%
  group_by(HasMedicalCondition, Diagnosis) %>%
  summarise(Count = n(), .groups = "drop")

# Plot HasMedicalCondition vs Diagnosis
ggplot(summary_data, aes(x = as.factor(HasMedicalCondition),
                         y = Count, fill = as.factor(Diagnosis))) +
  geom_bar(stat = "identity", position = "dodge") +
  labs(title = "Alzheimer's Diagnosis vs. Medical History",
       x = "Has One or More Medical Conditions (0 = No, 1 = Yes)",
       y = "Number of People",
       fill = "Diagnosis (0 = No, 1 = Yes)") +
  theme_minimal()

# Add a column that indicates if the person has at least one cognitive issue
train_data <- train_data %>%
```

```r
  mutate(HasCognitiveIssues = ifelse(MMSE < 10 |
                                     FunctionalAssessment < 5 |
                                     MemoryComplaints == 1 |
                                     BehavioralProblems == 1 |
                                     ADL < 5, 1, 0))

# Add a column that indicates if the person has at least one cognitive issue
test_data <- test_data %>%
  mutate(HasCognitiveIssues = ifelse(MMSE < 10 |
                                     FunctionalAssessment < 5 |
                                     MemoryComplaints == 1 |
                                     BehavioralProblems == 1 |
                                     ADL < 5, 1, 0))

# Count the number of people in each group (HasCognitiveIssues vs Diagnosis)
summary_cognitive <- train_data %>%
  group_by(HasCognitiveIssues, Diagnosis) %>%
  summarise(Count = n(), .groups = "drop")

# Plot HasCognitiveIssues vs Diagnosis
ggplot(summary_cognitive, aes(x = as.factor(HasCognitiveIssues),
                              y = Count, fill = as.factor(Diagnosis))) +
  geom_bar(stat = "identity", position = "dodge") +
  labs(title = "Alzheimer's Diagnosis vs. Cognitive Issues",
       x = "Has One or More Congnitive Conditions (0 = No, 1 = Yes)",
       y = "Number of People",
       fill = "Diagnosis (0 = No, 1 = Yes)") +
  theme_minimal()

# Add a column that indicates if the person has at least one symptom
train_data <- train_data %>%
  mutate(HasSymptoms = ifelse(Confusion == 1 |
                                Disorientation == 1 |
                                PersonalityChanges == 1 |
                                DifficultyCompletingTasks == 1 |
                                Forgetfulness == 1, 1, 0))

# Add a column that indicates if the person has at least one symptom
test_data <- test_data %>%
  mutate(HasSymptoms = ifelse(Confusion == 1 |
                                Disorientation == 1 |
                                PersonalityChanges == 1 |
```

```
                                DifficultyCompletingTasks == 1 |
                                Forgetfulness == 1, 1, 0))

# Count the number of people in each group (HasSymptoms vs Diagnosis)
summary_symptoms <- train_data %>%
  group_by(HasSymptoms, Diagnosis) %>%
  summarise(Count = n(), .groups = "drop")

# Plot HasSymptoms vs Diagnosis
ggplot(summary_symptoms, aes(x = as.factor(HasSymptoms),
                             y = Count, fill = as.factor(Diagnosis))) +
  geom_bar(stat = "identity", position = "dodge") +
  labs(title = "Alzheimer's Diagnosis vs. Symptoms",
       x = "Has One or More Symptoms (0 = No, 1 = Yes)",
       y = "Number of People",
       fill = "Diagnosis (0 = No, 1 = Yes)") +
  theme_minimal()

# Split the data with similar proportions of diagnosed and undiagnoses
index_70 <- createDataPartition(train_data$Diagnosis, p = 0.7, list = FALSE)
train_70 <- train_data[index_70,]
validation_set <- train_data[-index_70,]

# Check the dimension of the data to ensure proportion & variables are present
dim(train_70)
dim(validation_set)

library(e1071)

train_70$Diagnosis <- as.factor(train_70$Diagnosis)
# Convert 0/1 columns to factors
binary_to_factor <- function(df) {
  binary_cols <- sapply(df, function(col) all(col %in% c(0, 1)))
  df[binary_cols] <- lapply(df[binary_cols], factor)
  return(df)
}

train_70 <- binary_to_factor(train_70)
validation_set <- binary_to_factor(validation_set)


# Remove the 'PAtientID' column, irrelevant to our problem
```

```r
train_70 <- select(train_70, -PatientID)
validation_set <- select(validation_set, -PatientID)


# Select features to be used in SVM based on the trends found previously
selected_features <- c("Ethnicity",
                       "Gender",
                       "MMSE",
                       "MemoryComplaints",
                       "ADL",
                       "FunctionalAssessment",
                       "BehavioralProblems",
                       "HeadInjury",
                       "Forgetfulness")


# Subset to selected features + target
train_subset <- train_70[, c(selected_features, "Diagnosis")]
test_subset <- validation_set[, c(selected_features, "Diagnosis")]


for (feature in selected_features) {
  subset <- setdiff(selected_features, feature)
  train_temp <- train_subset[, c(subset, "Diagnosis")]
  svm_model_temp <- svm(Diagnosis ~ ., data = train_temp,
                        kernel = "radial", cost = 10, gamma = 0.1)
  preds <- predict(svm_model_temp, newdata = train_temp[, subset])
  acc <- mean(preds == train_temp$Diagnosis)
  cat("Without", feature, "- Accuracy:", round(acc, 4), "\n")
}

set.seed(123)
svm_model_test <- svm(Diagnosis ~ .,
                  data = train_subset,
                  kernel = "radial",
                  cost = 10,
                  gamma = 0.1,
                  scale = TRUE,
                  class.weights = c("0" = 1, "1" = 1.3))
predictions <- predict(svm_model_test,
                       newdata = test_subset[, selected_features])
```

```r
confusionMatrix(predictions, as.factor(test_subset$Diagnosis))

# Save model in case to be used later if needed
saveRDS(
  svm_model_test,
  file = "models/svm_model_test.rds"
)

#Use this code to load the model later if needed
# loaded_model <- readRDS("models/svm_model.rds")

tune_result <- tune(svm,
                    Diagnosis ~ .,
                    data = train_subset,
                    kernel = "linear",
                    ranges = list(cost = c(1, 10, 100),
                                  gamma = c(0.001, 0.01, 0.1)),
                    class.weights = c("0" = 1, "1" = 1.3))

summary(tune_result)

best_model <- tune_result$best.model

predictions <- predict(best_model, newdata = test_subset)

confusionMatrix(predictions, as.factor(test_subset$Diagnosis))


library(pROC)

# Get decision values
svm_model_prob <- predict(svm_model_test,
                          newdata = test_subset[, selected_features],
                          decision.values = TRUE)

# Extract decision values for ROC
decision_values <- attributes(svm_model_prob)$decision.values

# Create ROC curve
roc_obj <- roc(test_subset$Diagnosis, as.numeric(decision_values))

#Plot ROC curve with value
```

```r
plot(roc_obj, col = "darkred", main = paste("ROC Curve - AUC:",
                                            round(auc(roc_obj), 3)))


#Previously tried models
set.seed(123)

model_1 <- glm(Diagnosis ~ HasMedicalCondition + HasSymptoms,
               data = train_70,
               family = binomial())

# View the summary
summary(model_1)

model_2 <- glm(Diagnosis ~ HasCognitiveIssues + HasSymptoms,
                 data = train_70,
                 family = binomial())

# View the summary
summary(model_2)

model_3 <- glm(Diagnosis ~ HasMedicalCondition + HasCognitiveIssues +
                   HasSymptoms,
                 data = train_70,
                 family = binomial())

# View the summary
summary(model_3)


model_4 <- glm(Diagnosis ~ HasCognitiveIssues,
                 data = train_70,
                 family = binomial())

# View the summary
summary(model_4)

# Make predictions on the test set
predictions <- predict(model_2, validation_set, type = "response")

# Convert probabilities to binary outcomes (0 or 1)
predicted_class <- ifelse(predictions > 0.35, 1, 0)
```

```r
# Check the performance with confusion matrix
confusionMatrix(as.factor(predicted_class), as.factor(validation_set$Diagnosis))

#### Preamble ####
# Purpose: Final Model using SVM
# Author: Rahma Binth Mohammad
# Contact: rahma.binthmohammad@mail.utoronto.ca
# License: MIT

# Workspace setup
library(tidyverse)
library(e1071)
library(dplyr)

#Set working directory
setwd("/Users/Rahma/alzheimers")

# Read in the data
train_data <- read.csv("data/01-raw_data/train.csv", stringsAsFactors = FALSE)
test_data <- read.csv("data/01-raw_data/test.csv", stringsAsFactors = FALSE)

#Check for NA values, none found
colSums(is.na(train_data))
colSums(is.na(test_data))

# Remove the 'DoctorInCharge' column, irrelevant to our problem
train_data <- select(train_data, -DoctorInCharge)
test_data <- select(test_data, -DoctorInCharge)

# Remove the 'PatientID' column from training data irrelevant to our problem
train_data <- select(train_data, -PatientID)

# Convert all 0/1 columns to factors, just in case
binary_to_factor <- function(df) {
  binary_cols <- sapply(df, function(col) all(col %in% c(0, 1)))
  df[binary_cols] <- lapply(df[binary_cols], factor)
  return(df)
}

train_data <- binary_to_factor(train_data)
test_data <- binary_to_factor(test_data)
```

```r
# Select features to be used in SVM based on the trends found previously
selected_features <- c("Ethnicity",
                       "Gender",
                       "MMSE",
                       "MemoryComplaints",
                       "ADL",
                       "FunctionalAssessment",
                       "BehavioralProblems",
                       "HeadInjury",
                       "Forgetfulness")

# Subset to contain selected features & target variable for the training data
train_selected <- train_data[, c(selected_features, "Diagnosis")]
# Subset to contain  selected features & the Patient ID for the test data
test_selected <- test_data[, c(selected_features, "PatientID")]

# Set seed for SVM model
set.seed(123)

#SVM - fine tuned
svm_model <- svm(Diagnosis ~ .,
                 data = train_selected,
                 kernel = "radial",
                 cost = 20,
                 gamma = 0.1,
                 scale = TRUE,
                 class.weights = c("0" = 1, "1" = 1.3))

predictions_for_sub <- predict(svm_model,
                               newdata = test_selected[, c(selected_features,
                                                           "PatientID")])

# Create data table with PatientID and predictions
prediction_results <- data.frame(
  PatientID = test_selected$PatientID,
  Diagnosis = as.numeric(as.character(predictions_for_sub))
)

# Writing data to a CSV file
write.csv(prediction_results, "data/02-prediction_data/submission.csv",
          row.names = FALSE)
```

```r
# Save model in case to be used later if needed
saveRDS(
  svm_model,
  file = "models/svm_model.rds"
)
```

References

Bing, Xin. 2024. *Classification of the Alzheimer's Disease*. https://www.kaggle.com/competitions/classification-of-the-alzheimers-disease/data.

International, Alzheimer's Disease. n.d. *Dementia Statistics*. https://www.alzint.org/about/dementia-facts-figures/dementia-statistics/.

Kuhn, and Max. 2008. "Building Predictive Models in r Using the Caret Package." *Journal of Statistical Software* 28 (5): 1–26. https://doi.org/10.18637/jss.v028.i05.

Meyer, David, Evgenia Dimitriadou, Kurt Hornik, Andreas Weingessel, and Friedrich Leisch. 2024. *E1071: Misc Functions of the Department of Statistics, Probability Theory Group (Formerly: E1071), TU Wien*. https://CRAN.R-project.org/package=e1071.

Pedersen, Thomas Lin. 2024. *Patchwork: The Composer of Plots*. https://CRAN.R-project.org/package=patchwork.

R Core Team. 2023. *R: A Language and Environment for Statistical Computing*. Vienna, Austria: R Foundation for Statistical Computing. https://www.R-project.org/.

Robin, Xavier, Natacha Turck, Alexandre Hainard, Natalia Tiberti, Frédérique Lisacek, Jean-Charles Sanchez, and Markus Müller. 2011. "pROC: An Open-Source Package for r and s+ to Analyze and Compare ROC Curves." *BMC Bioinformatics* 12: 77.

Wickham, Hadley. 2016. *Ggplot2: Elegant Graphics for Data Analysis*. Springer-Verlag New York. https://ggplot2.tidyverse.org.

Wickham, Hadley, Mara Averick, Jennifer Bryan, Winston Chang, Lucy D'Agostino McGowan, Romain François, Garrett Grolemund, et al. 2019. "Welcome to the tidyverse." *Journal of Open Source Software* 4 (43): 1686. https://doi.org/10.21105/joss.01686.

Wickham, Hadley, Romain François, Lionel Henry, Kirill Müller, and Davis Vaughan. 2023. *Dplyr: A Grammar of Data Manipulation*. https://CRAN.R-project.org/package=dplyr.