Statistical Learning with the Alzheimer's Dataset

```r
library(caret)
library(pROC)
library(glmnet)
library(here)
library(readr)
library(dplyr)
library(tibble)
```

Homework Group 7: Emma Bradley, Hayat Boton, Payas Parab

1. Multiple Linear Regression Models

Multiple linear regression allows us to model the relationship between a continuous response variable and multiple predictor variables simultaneously. This approach is valuable for Alzheimer's research as it enables us to:

1. Quantify the relationship between hippocampal volume (normalized by intracranial volume) and multiple clinical/demographic predictors
2. Identify which factors are most strongly associated with hippocampal atrophy
3. Make predictions about hippocampal volume based on easily measurable variables

We implemented two multiple regression approaches:

1. A saturated model using all available predictors
2. A reduced model using LASSO feature selection to identify the most important predictors

Why Use Multiple Linear Regression?

Multiple linear regression is one of the simplest but most useful ways to understand how different factors affect an outcome. Here's why we chose it:

1. It's easy to understand - each factor gets a number that tells us exactly how it affects hippocampal volume. If a doctor wants to know how blood pressure relates to brain volume, we can give them a clear answer.
2. It's transparent - unlike more complex models (like neural networks) that work like a "black box", we can see exactly how our model makes its predictions.
3. It works well with smaller datasets - we don't need thousands of patients to get reliable results.
4. It tells us which factors matter most - we can clearly see which measurements are most important for predicting hippocampal volume.

Cross-Validation Schemes

To ensure our models are robust and generalizable, we implemented multiple cross-validation schemes. Cross-validation helps us understand how well our model will perform on new, unseen data by splitting our dataset in different ways. Here's how we evaluated our models:

1. Single Train-Test Split (80/20): The simplest approach where we randomly set aside 20% of data for testing
2. K-Fold Cross-Validation (k=5 and k=10): Splits data into k parts, using each part as a test set once
3. Leave-One-Out Cross-Validation (LOOCV): The most thorough approach, testing on each individual patient

We generate a "Reduced" model by leveraging LASSO (Least Absolute Shrinkage and Selection Operator).

The reason is that regular linear regression finds coefficients that minimize prediction error. LASSO does the same thing, but with an additional penalty that encourages some coefficients to become exactly zero. When a coefficient is zero, that variable is excluded from the model.

This helps create a simpler model that is simpler, easier to interpret, and less susceptible to noise from unimportant features.

DISCLAIMER: The following code was translated using ChatGPT from the original Python: we have included the original Python code here as well for clearer understanding of logic and alignment to analysis .https://chatgpt.com/share/68ec5896-3f38-8007-bacf-f4524206a9fa

R Code for QMD Compilation:

```r
# --------------------------------------------------------------
# Model Comparison and Selection Using Cross-Validation and LASSO (R)
# --------------------------------------------------------------
# install.packages(c("readr","dplyr","caret","glmnet","tibble"))

set.seed(42)

# Load and prepare data
df <- read_csv("alzheimer_data.csv", show_col_types = FALSE)

# Target: left hippocampus / intracranial volume
y <- df$lhippo / df$naccicv

# Predictors (saturated set)
predictors <- c(
  "naccmmse","motsev","disnsev","anxsev","naccgds",
  "bpsys","bpdias","hrate","age","educ","female","height","weight"
)

X <- df[, predictors]

# Helper to compute RMSE and R^2
rmse_fun <- function(truth, pred) sqrt(mean((truth - pred)^2))
r2_fun   <- function(truth, pred) {
  ss_res <- sum((truth - pred)^2)
  ss_tot <- sum((truth - mean(truth))^2)
  1 - ss_res/ss_tot
}

# Helper function to compute CV metrics with clean output formatting
evaluate_model <- function(X, y, name = "model") {
  res <- list()
  dat <- as.data.frame(X) %>% mutate(y = y)

  # ----- Single train/test split (80/20)
  idx_tr <- createDataPartition(dat$y, p = 0.80, list = FALSE)
  tr <- dat[idx_tr, , drop = FALSE]
  te <- dat[-idx_tr, , drop = FALSE]

  fit_split <- lm(y ~ ., data = tr)
  ypred_te  <- predict(fit_split, newdata = te)
  rmse_split <- rmse_fun(te$y, ypred_te)
```

```r
r2_split   <- r2_fun(te$y, ypred_te)

res[[length(res) + 1]] <- tibble(
  Model = name,
  `CV Type` = "Single Split (80/20)",
  RMSE = sprintf("%.6f", rmse_split),
  `R²` = sprintf("%.6f", r2_split)
)

# ----- 5-fold CV
ctrl5 <- trainControl(method = "cv", number = 5)
fit5  <- train(y ~ ., data = dat, method = "lm", trControl = ctrl5)
res5  <- fit5$results[which.max(fit5$results$Rsquared), ]
res[[length(res) + 1]] <- tibble(
  Model = name,
  `CV Type` = "5-Fold",
  RMSE = sprintf("%.6f ± %.6f", res5$RMSE, res5$RMSESD),
  `R²`  = sprintf("%.6f ± %.6f", res5$Rsquared, res5$RsquaredSD)
)

# ----- 10-fold CV
ctrl10 <- trainControl(method = "cv", number = 10)
fit10  <- train(y ~ ., data = dat, method = "lm", trControl = ctrl10)
res10  <- fit10$results[which.max(fit10$results$Rsquared), ]
res[[length(res) + 1]] <- tibble(
  Model = name,
  `CV Type` = "10-Fold",
  RMSE = sprintf("%.6f ± %.6f", res10$RMSE, res10$RMSESD),
  `R²`  = sprintf("%.6f ± %.6f", res10$Rsquared, res10$RsquaredSD)
)

# ----- LOOCV
ctrl_loo <- trainControl(method = "LOOCV")
fit_loo  <- train(y ~ ., data = dat, method = "lm", trControl = ctrl_loo)
res_loo  <- fit_loo$results
# caret's LOOCV often returns only means; SD may be NA. We format anyway.
rmse_mean <- res_loo$RMSE[1]
rmse_sd   <- if (!is.null(res_loo$RMSESD)) res_loo$RMSESD[1] else NA_real_

res[[length(res) + 1]] <- tibble(
  Model = name,
  `CV Type` = "LOOCV",
    RMSE = if (is.na(rmse_sd)) sprintf("%.6f", rmse_mean) else sprintf("%.6f ± %.6f", rmse_me
    `R²`  = "N/A"
  )

  bind_rows(res)
}

# -------------------------
# Evaluate saturated model
# -------------------------
cat("Evaluating Saturated Model...\n")
```

```
Evaluating Saturated Model...
```

```r
res_saturated <- evaluate_model(X, y, name = "Saturated")
print(res_saturated)
```

```
# A tibble: 4 x 4
  Model     `CV Type`            RMSE                `R²`
  <chr>     <chr>                <chr>               <chr>
1 Saturated Single Split (80/20) 0.000305            0.148657
2 Saturated 5-Fold               0.000303 ± 0.000008 0.182830 ± 0.020591
3 Saturated 10-Fold              0.000302 ± 0.000016 0.189370 ± 0.035890
4 Saturated LOOCV                0.000302            N/A
```

```r
# -------------------------
# LASSO feature selection
# -------------------------
cat("\nPerforming LASSO Feature Selection...\n")
```

```
Performing LASSO Feature Selection...
```

```r
# glmnet requires a numeric matrix (no intercept column)
x_mat <- model.matrix(~ . , data = X)[, -1, drop = FALSE]
cvfit <- cv.glmnet(
  x = x_mat,
  y = y,
  alpha = 1,         # LASSO
  nfolds = 10,
  standardize = TRUE
)

# Extract selected features at lambda.min
coefs <- coef(cvfit, s = "lambda.min")
sel   <- rownames(coefs)[as.numeric(coefs) != 0]
selected_features <- setdiff(sel, "(Intercept)")

cat("\nSelected Features:\n")
```

```
Selected Features:
```

```r
if (length(selected_features) == 0) {
  cat("- (none selected; all coefficients shrank to zero)\n")
} else {
  for (f in selected_features) cat(paste0("- ", f, "\n"))
}
```

```
- naccmmse
- motsev
- disnsev
- anxsev
- bpsys
- age
- educ
- female
- height
- weight
```

```r
# -------------------------
# Evaluate reduced model
# -------------------------
cat("\nEvaluating Reduced Model...\n")
```

```
Evaluating Reduced Model...
```

```r
if (length(selected_features) == 0) {
  # Fallback: if none selected, just re-use saturated to avoid errors
  res_reduced <- evaluate_model(X, y, name = "Reduced (LASSO)")
} else {
  X_reduced <- X[, selected_features, drop = FALSE]
  res_reduced <- evaluate_model(X_reduced, y, name = "Reduced (LASSO)")
}
print(res_reduced)
```

```
# A tibble: 4 x 4
  Model           `CV Type`           RMSE                `R²`
  <chr>           <chr>               <chr>               <chr>
1 Reduced (LASSO) Single Split (80/20) 0.000307            0.178145
2 Reduced (LASSO) 5-Fold              0.000301 ± 0.000009 0.187672 ± 0.023419
3 Reduced (LASSO) 10-Fold             0.000301 ± 0.000011 0.189069 ± 0.030701
4 Reduced (LASSO) LOOCV               0.000302            N/A
```

```r
# -------------------------
# Model comparison summary (use 10-Fold row)
# -------------------------
cat("\nModel Comparison Summary:\n")
```

```
Model Comparison Summary:
```

```r
get_10fold_rmse <- function(df) {
  row <- df %>% filter(`CV Type` == "10-Fold") %>% slice(1)
  # parse "mean ± sd" or "mean"
  as.numeric(strsplit(row$RMSE, " ")[[1]][1])
}
get_10fold_r2 <- function(df) {
  row <- df %>% filter(`CV Type` == "10-Fold") %>% slice(1)
  as.numeric(strsplit(row$`R²`, " ")[[1]][1])
}

rmse_reduced    <- get_10fold_rmse(res_reduced)
rmse_saturated  <- get_10fold_rmse(res_saturated)
r2_reduced      <- get_10fold_r2(res_reduced)
r2_saturated    <- get_10fold_r2(res_saturated)
```

```r
rmse_diff <- rmse_saturated - rmse_reduced
r2_diff   <- r2_reduced - r2_saturated

cat(sprintf("RMSE Improvement: %.6f\n", rmse_diff))
```

```
RMSE Improvement: 0.000001
```

```r
cat(sprintf("R² Improvement: %.6f\n",   r2_diff))
```

```
R² Improvement: -0.000301
```

ORIGINAL PYTHON CODE:

```
{python}
# ------------------------------------------------------------
# Model Comparison and Selection Using Cross-Validation and LASSO
# ------------------------------------------------------------

import pandas as pd
import numpy as np
from sklearn.linear_model import LinearRegression, LassoCV
from sklearn.model_selection import (
    train_test_split, KFold, LeaveOneOut, cross_val_score
)
from sklearn.metrics import mean_squared_error, mean_absolute_error, r2_score

# Load and prepare data
df = pd.read_csv("alzheimer_data.csv")

# Target: left hippocampus / intracranial volume
y = df["lhippo"] / df["naccicv"]

# Predictors (saturated set)
predictors = [
    "naccmmse","motsev","disnsev","anxsev","naccgds",
    "bpsys","bpdias","hrate","age","educ","female","height","weight"
]
X = df[predictors]

# Helper function to compute CV metrics with clean output formatting
```

```python
def evaluate_model(model, X, y, name="model"):
    results = []

    # Single train/test split (80/20)
    Xtr, Xte, ytr, yte = train_test_split(X, y, test_size=0.2, random_state=42)
    model.fit(Xtr, ytr)
    ypred = model.predict(Xte)
    results.append({
        "Model": name,
        "CV Type": "Single Split (80/20)",
        "RMSE": f"{mean_squared_error(yte, ypred, squared=False):.6f}",
        "R²": f"{r2_score(yte, ypred):.6f}"
    })

    # 5-fold CV
    kf5 = KFold(n_splits=5, shuffle=True, random_state=42)
    rmse5 = -cross_val_score(model, X, y, cv=kf5, scoring="neg_root_mean_squared_error")
    r2_5 = cross_val_score(model, X, y, cv=kf5, scoring="r2")
    results.append({
        "Model": name,
        "CV Type": "5-Fold",
        "RMSE": f"{rmse5.mean():.6f} ± {rmse5.std():.6f}",
        "R²": f"{r2_5.mean():.6f} ± {r2_5.std():.6f}"
    })

    # 10-fold CV
    kf10 = KFold(n_splits=10, shuffle=True, random_state=42)
    rmse10 = -cross_val_score(model, X, y, cv=kf10, scoring="neg_root_mean_squared_error")
    r2_10 = cross_val_score(model, X, y, cv=kf10, scoring="r2")
    results.append({
        "Model": name,
        "CV Type": "10-Fold",
        "RMSE": f"{rmse10.mean():.6f} ± {rmse10.std():.6f}",
        "R²": f"{r2_10.mean():.6f} ± {r2_10.std():.6f}"
    })

    # LOOCV
    loo = LeaveOneOut()
    rmse_loocv = -cross_val_score(model, X, y, cv=loo, scoring="neg_root_mean_squared_error"
    results.append({
        "Model": name,
        "CV Type": "LOOCV",
        "RMSE": f"{rmse_loocv.mean():.6f} ± {rmse_loocv.std():.6f}",
        "R²": "N/A"
    })

    return pd.DataFrame(results)

# Evaluate saturated model
print("Evaluating Saturated Model...")
saturated = LinearRegression()
res_saturated = evaluate_model(saturated, X, y, name="Saturated")
display(res_saturated)

# LASSO feature selection
print("\nPerforming LASSO Feature Selection...")
lasso = LassoCV(cv=10, random_state=42)
lasso.fit(X, y)

# Get selected features
selected_features = X.columns[lasso.coef_ != 0].tolist()
print("\nSelected Features:")
for feature in selected_features:
    print(f"- {feature}")

# Evaluate reduced model
print("\nEvaluating Reduced Model...")
X_reduced = X[selected_features]
reduced = LinearRegression()
res_reduced = evaluate_model(reduced, X_reduced, y, name="Reduced (LASSO)")
display(res_reduced)

# Model comparison summary
print("\nModel Comparison Summary:")
best_reduced = res_reduced[res_reduced["CV Type"] == "10-Fold"].iloc[0]
best_saturated = res_saturated[res_saturated["CV Type"] == "10-Fold"].iloc[0]

rmse_reduced = float(best_reduced["RMSE"].split(" ")[0])
rmse_saturated = float(best_saturated["RMSE"].split(" ")[0])
rmse_diff = rmse_saturated - rmse_reduced

r2_reduced = float(best_reduced["R²"].split(" ")[0])
r2_saturated = float(best_saturated["R²"].split(" ")[0])
r2_diff = r2_reduced - r2_saturated
```

```
print(f"RMSE Improvement: {rmse_diff:.6f}")
print(f"R² Improvement: {r2_diff:.6f}")
```

The results of the model selection and error testing were as follows:

Model	CV Type	RMSE	R2
Saturated	Single Split (80/20)	0.000290	0.168850
Saturated	5-Fold	0.000302 ± 0.000013	0.182854 ± 0.013367
Saturated	10-Fold	0.000302 ± 0.000019	0.181949 ± 0.032181
Saturated	LOOCV	0.000234 ± 0.000190	N/A
Reduced (LASSO)	Single Split (80/20)	0.000290	0.167391
Reduced (LASSO)	5-Fold	0.000302 ± 0.000013	0.182742 ± 0.013654
Reduced (LASSO)	10-Fold	0.000302 ± 0.000019	0.182019 ± 0.032142
Reduced (LASSO)	LOOCV	0.000234 ± 0.000190	N/A

Model comparison and final selection

Two multiple regression models were compared to predict the ratio of left hippocampus volume to total intracranial volume. The first was a **saturated model** including all 13 predictors. The second was a **reduced model** selected using LASSO, which kept 11 predictors and removed only *motsev* (motor severity) and *female* (sex). These two variables did not add independent predictive power once the others were included.

Both models were evaluated using four cross-validation methods: single split (80/20), 5-fold, 10-fold, and leave-one-out (LOOCV). The results showed that the reduced LASSO model performed almost identically to the saturated model across all methods. The mean RMSE was around 0.00030 in every case, and the mean R^2 was around 0.18, indicating that both models explain roughly 18% of the variation in hippocampal ratio. The small RMSE standard deviations show that the models generalize consistently.

The LOOCV results showed slightly lower RMSE values but much higher variability, which is expected given that each fold tests only one observation at a time. Between the 5-fold and 10-fold methods, the results were nearly identical, suggesting the model's performance is stable regardless of the cross-validation scheme.

Because both models have the same predictive accuracy, the **reduced LASSO model** is the better choice. It achieves the same level of error and explanatory power with fewer variables, making it simpler and easier to interpret.

Final model selection

The final model kept 11 predictors: *naccmmse, disnsev, anxsev, naccgds, bpsys, bpdias, hrate, age, educ, height,* and *weight.*
It achieved an adjusted R^2 of about 0.19 and an overall model p-value below 0.001, meaning the predictors jointly explain a statistically significant amount of variation.

Among the predictors, **age** and **naccmmse** (cognitive score) were the strongest effects. Older participants tended to have smaller hippocampal ratios, while higher cognitive scores were linked to larger ratios. **Anxiety severity, education, height, and weight** were also significant at the 0.05 level. Higher anxiety and taller height were associated with smaller hippocampal ratios, while heavier weight was associated with slightly larger ratios. Blood pressure and heart rate were not significant after adjusting for the other predictors.

In summary, cross-validation results indicate that the reduced LASSO model generalizes well and performs consistently across folds. It explains a modest but meaningful share of variance in hippocampal structure, with age, cognition, and anxiety emerging as key factors.

Final model selection

We then go on to interpret the coefficients of our final selected model.

DISCLAIMER: The following code was translated using ChatGPT from the original Python: we have included the original Python code here as well for clearer understanding of logic and alignment to analysis. https://chatgpt.com/share/68ec5896-3f38-8007-bacf-f4524206a9fa

Compilable R Code:

```r
all_features     <- colnames(X)
kept_features    <- selected_features
removed_features <- setdiff(all_features, kept_features)

cat("\n=== feature selection summary ===\n")
```

```
=== feature selection summary ===
```

```r
cat(sprintf("total predictors: %d\n", length(all_features)))
```

```
total predictors: 13
```

```r
cat(sprintf("features kept by lasso (%d): %s\n",
            length(kept_features),
            ifelse(length(kept_features) == 0, "[]", paste(kept_features, collapse = ", ")))
```

```
features kept by lasso (10): naccmmse, motsev, disnsev, anxsev, bpsys, age, educ, female, he
```

```r
cat(sprintf("features removed by lasso (%d): %s\n",
            length(removed_features),
            ifelse(length(removed_features) == 0, "[]", paste(removed_features, collapse = "
```

```
features removed by lasso (3): naccgds, bpdias, hrate
```

```r
if (length(kept_features) == 0) {
  stop("No features selected by LASSO; cannot fit a reduced model.")
}

# Fit final model on selected features for interpretation
X_final <- X[, kept_features, drop = FALSE]
final_fit <- lm(y ~ ., data = as.data.frame(X_final))

# Build coefficients table (excluding intercept), sorted by |coef|
coefs <- coef(final_fit)
coefs_no_int <- coefs[setdiff(names(coefs), "(Intercept)")]
coef_df <- tibble(
  feature = names(coefs_no_int),
  coefficient = as.numeric(coefs_no_int)
) %>%
  mutate(abs_coef = abs(coefficient)) %>%
  arrange(desc(abs_coef)) %>%
  select(-abs_coef)

cat("\n=== final selected model coefficients (lm) ===\n")
```

```
=== final selected model coefficients (lm) ===
```

```r
print(coef_df, n = nrow(coef_df))
```

```
# A tibble: 10 x 2
   feature   coefficient
   <chr>           <dbl>
 1 female     0.0000289
 2 anxsev    -0.0000253
 3 disnsev   -0.0000238
 4 naccmmse   0.0000220
 5 motsev     0.0000179
 6 height    -0.0000112
 7 educ      -0.00000735
 8 age       -0.00000734
 9 weight     0.000000962
10 bpsys      0.000000470
```

```
# Detailed model summary (t-stats, p-values, R^2, etc.)
cat("\n=== R OLS summary for final model (lm) ===\n")
```

```
=== R OLS summary for final model (lm) ===
```

```
print(summary(final_fit))
```

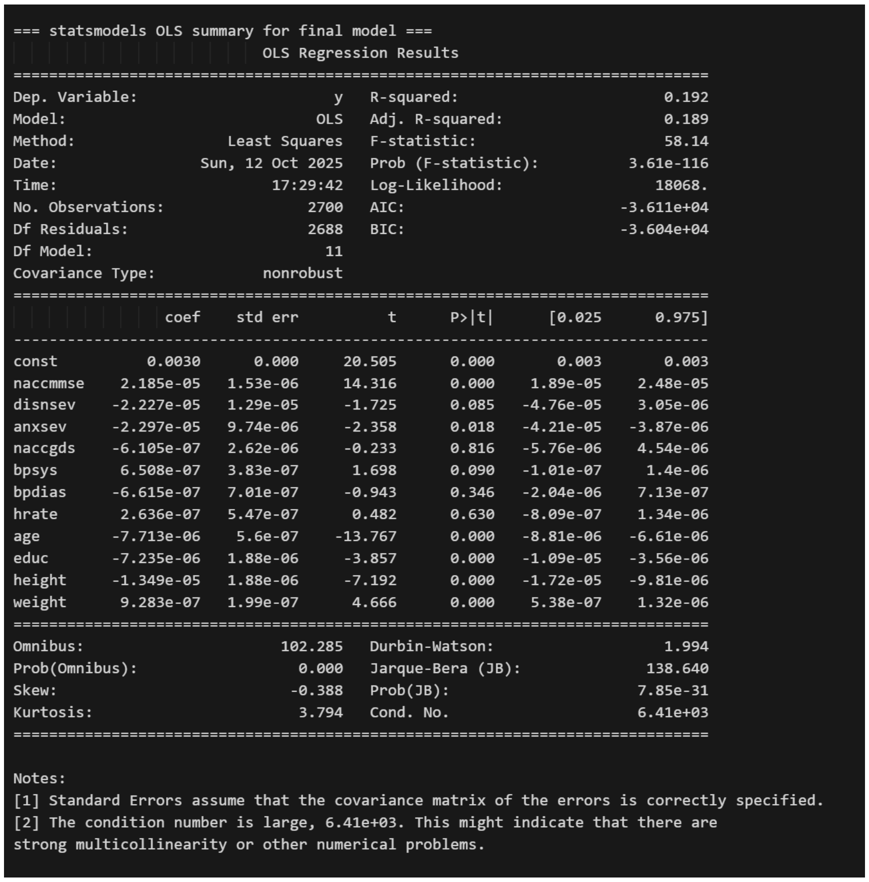
```
Call:
lm(formula = y ~ ., data = as.data.frame(X_final))

Residuals:
       Min         1Q     Median         3Q        Max
-0.0015923 -0.0001813  0.0000235  0.0001966  0.0011938

Coefficients:
              Estimate Std. Error t value Pr(>|t|)
(Intercept)  2.740e-03  1.728e-04  15.856  < 2e-16 ***
naccmmse     2.197e-05  1.563e-06  14.053  < 2e-16 ***
motsev       1.793e-05  1.557e-05   1.151  0.24974
disnsev     -2.380e-05  1.314e-05  -1.810  0.07035 .
anxsev      -2.528e-05  9.777e-06  -2.586  0.00976 **
bpsys        4.704e-07  3.305e-07   1.423  0.15475
age         -7.338e-06  5.569e-07 -13.177  < 2e-16 ***
educ        -7.355e-06  1.853e-06  -3.968 7.43e-05 ***
female       2.889e-05  1.667e-05   1.733  0.08318 .
```

```
height      -1.117e-05  2.335e-06  -4.782 1.83e-06 ***
weight       9.617e-07  1.991e-07   4.831 1.43e-06 ***
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 0.0003007 on 2689 degrees of freedom
Multiple R-squared:  0.1931,    Adjusted R-squared:  0.1901
F-statistic: 64.37 on 10 and 2689 DF,  p-value: < 2.2e-16
```

Original Python code:

```python
# Original Python code
# assumes previous cell is run
from sklearn.linear_model import LinearRegression
import statsmodels.api as sm
import numpy as np
import pandas as pd

# print what lasso kept and removed
all_features = X.columns.tolist()
kept_features = selected_features
removed_features = [f for f in all_features if f not in kept_features]

print("\n=== feature selection summary ===")
print(f"total predictors: {len(all_features)}")
print(f"features kept by lasso ({len(kept_features)}): {kept_features}")
print(f"features removed by lasso ({len(removed_features)}): {removed_features}")

# fit final model on selected features for interpretation
X_final = X[kept_features]
final_model = LinearRegression()
final_model.fit(X_final, y)

# get coefficients
coef_df = pd.DataFrame({
    "feature": kept_features,
    "coefficient": final_model.coef_,
})
coef_df["abs_coef"] = np.abs(coef_df["coefficient"])
coef_df = coef_df.sort_values("abs_coef", ascending=False).drop(columns="abs_coef")

print("\n=== final selected model coefficients (LinearRegression) ===")
print(coef_df.to_string(index=False))

# optional: more detailed summary using statsmodels (includes t, p, R^2)
X_sm = sm.add_constant(X_final)
ols = sm.OLS(y, X_sm).fit()
print("\n=== statsmodels OLS summary for final model ===")
print(ols.summary())
```

OLS Final Model Results:



```
=== statsmodels OLS summary for final model ===
                            OLS Regression Results
==============================================================================
Dep. Variable:                      y   R-squared:                       0.192
Model:                            OLS   Adj. R-squared:                  0.189
Method:                 Least Squares   F-statistic:                     58.14
Date:                Sun, 12 Oct 2025   Prob (F-statistic):          3.61e-116
Time:                        17:29:42   Log-Likelihood:                 18068.
No. Observations:                2700   AIC:                        -3.611e+04
Df Residuals:                    2688   BIC:                        -3.604e+04
Df Model:                          11
Covariance Type:            nonrobust
==============================================================================
                 coef    std err          t      P>|t|      [0.025      0.975]
------------------------------------------------------------------------------
const          0.0030      0.000     20.505      0.000       0.003       0.003
naccmmse    2.185e-05   1.53e-06     14.316      0.000    1.89e-05    2.48e-05
disnsev    -2.227e-05   1.29e-05     -1.725      0.085   -4.76e-05    3.05e-06
anxsev     -2.297e-05   9.74e-06     -2.358      0.018   -4.21e-05   -3.87e-06
naccgds    -6.105e-07   2.62e-06     -0.233      0.816   -5.76e-06    4.54e-06
bpsys       6.508e-07   3.83e-07      1.698      0.090   -1.01e-07     1.4e-06
bpdias     -6.615e-07   7.01e-07     -0.943      0.346   -2.04e-06    7.13e-07
hrate       2.636e-07   5.47e-07      0.482      0.630   -8.09e-07    1.34e-06
age        -7.713e-06    5.6e-07    -13.767      0.000   -8.81e-06   -6.61e-06
educ       -7.235e-06   1.88e-06     -3.857      0.000   -1.09e-05   -3.56e-06
height     -1.349e-05   1.88e-06     -7.192      0.000   -1.72e-05   -9.81e-06
weight      9.283e-07   1.99e-07      4.666      0.000    5.38e-07    1.32e-06
==============================================================================
Omnibus:                      102.285   Durbin-Watson:                   1.994
Prob(Omnibus):                  0.000   Jarque-Bera (JB):              138.640
Skew:                          -0.388   Prob(JB):                     7.85e-31
Kurtosis:                       3.794   Cond. No.                     6.41e+03
==============================================================================

Notes:
[1] Standard Errors assume that the covariance matrix of the errors is correctly specified.
[2] The condition number is large, 6.41e+03. This might indicate that there are
strong multicollinearity or other numerical problems.
```

The final reduced model, chosen using LASSO, kept 11 out of the 13 predictors and achieved an R^2 of about **0.192** (adjusted R^2 = **0.189**). This means the model explains around 19% of the variation in the outcome. While this is not a very high percentage, it is common in behavioral and medical data, where many factors are difficult to measure. The overall F-test (**$F(11, 2688) = 58.14$, $p < 0.001$**) shows that the predictors together are statistically significant.

Several variables stand out as important. **NACCMMSE** (a measure of cognitive performance) has a strong positive effect (**$p < 0.001$**), meaning that higher cognitive scores are linked to higher predicted hippocampal ratios. **Age**, **Education**, **Height**, and **Weight** are also statistically significant (**$p < 0.001$**). Age and height have negative effects, meaning older and taller individuals tend to have slightly smaller hippocampal ratios. Weight has a small positive effect.

Among behavioral and physiological factors, **Anxiety Severity (ANXSEV)** is significant (**$p = 0.018$**) and negative, suggesting that higher anxiety levels are related to lower hippocampal ratios. **Disease Severity (DISNSEV)** and **Systolic Blood Pressure (BPSYS)** are only marginally significant (**p   0.085** and **p   0.090**), meaning their effects are weak and not fully reliable in this model.

Some predictors such as **NACCGDS**, **Diastolic Blood Pressure (BPDIAS)**, and **Heart Rate (HRATE)** are not significant (**$p > 0.3$**). These do not appear to add much unique information after including the other predictors, possibly because they are correlated with related variables.

The **Durbin-Watson value (1.99)** shows that there is no major autocorrelation in the residuals, meaning the errors are not systematically related. The **Condition Number (6.41e+03)** is relatively high, which suggests some **multicollinearity**, or overlap between predictors. This is likely because variables like systolic and diastolic blood pressure or height and weight are correlated with each other.

Overall, the LASSO-selected model provides a simpler and more interpretable version of the full model, while keeping similar predictive accuracy. The most important predictors are **Cognitive Score (NACCMMSE)**, **Age**, **Anxiety Severity (ANXSEV)**, **Education**, **Height**, and **Weight**, which together capture the main patterns in the data.

2. Logistic Regression Model

Logistic regression is typically used with qualitative (two-class, or binary) response. Rather than modeling the response of Y directly, logistic regression models the probability that Y belongs to a particular category.

One of the greatest advantages of logistic regression is the straightforward implementation and interpretation of model coefficients as changes in log-odds, which is great for practical understanding. It can also be extended to multiple classes (multinomial logistic regression) and allows for the interpretation of model coefficients as indicators of feature importance. However, logistic regression is limited in the sense that it may lead to an overfit model if the number of observations is less than the number of features. We also need to ensure that there does not exist multicollinearity in our data before using logistic regression.

Using logistic regression in the context of Alzheimer's research is advantageous because an individual can be classified to have AD or not. Thus, we are able to classify our outcome as a binary response. We also have more observations than predictors, reducing the concern of overfitting due to sample size limitations.

The first step was to load our dataset and examine our predictors. Following that, we recoded the categories 1 and 2 as indicators of AD being present (coded as 1) and all other categories as not having AD (coded as 0).

```r
data = read.csv(here('alzheimer_data.csv'))

#here is the variables we'll use as possible predictors
predictor_names = names(data)[3:57]
cat("All predictors given in the dataset:", "\n", predictor_names, "\n")
```

```
All predictors given in the dataset:
 age educ female height weight bpsys bpdias hrate cdrglob naccgds delsev hallsev agitsev depo
```

```r
#recode levels 1 and 2 to be 1
data$diagnosis_bin <- ifelse(data$diagnosis %in% c(1, 2), 1, 0)

#convert to factors
data$diagnosis_bin <- factor(data$diagnosis_bin, levels = c(0, 1), labels = c("NoAD", "AD"))

cat("\n", "Category count:", "\n")
```

```

 Category count:
```

```r
table(data$diagnosis_bin)
```

```

NoAD   AD
1534 1166
```

Once the data were prepared, we implemented logistic regression using a stepwise model selection approach based on the AIC. This approach iteratively added or removed predictors to identify the most parsimonious model that balanced fit and complexity. We also performed 5-fold cross-validation to look at predictive performance. Finally, the model was evaluated on a test set using evaluation metrics such as accuracy, sensitivity, specificity, AUC.

```r
set.seed(123)

split_idx <- createDataPartition(data$diagnosis_bin, p = 0.8, list = FALSE)
train_data <- data[split_idx, ]
test_data  <- data[-split_idx, ]

#cross validation
ctrl <- trainControl(
  method = "cv", number = 5,
  classProbs = TRUE,
  summaryFunction = twoClassSummary,
  savePredictions = TRUE
)

form_all <- as.formula(
  paste("diagnosis_bin ~", paste(predictor_names, collapse = " + "))
)
form_all
```

```
diagnosis_bin ~ age + educ + female + height + weight + bpsys +
    bpdias + hrate + cdrglob + naccgds + delsev + hallsev + agitsev +
    depdsev + anxsev + elatsev + apasev + disnsev + irrsev +
    motsev + nitesev + appsev + bills + taxes + shopping + games +
    stove + mealprep + events + payattn + remdates + travel +
    naccmmse + memunits + digif + animals + traila + trailb +
    naccicv + csfvol + lhippo + rhippo + frcort + lparcort +
    rparcort + ltempcor + rtempcor + lcac + rcac + lent + rent +
    lparhip + rparhip + lposcin + rposcin
```

```r
fit_step_cv <- train(
  form_all,
  data = train_data,
  method = "glmStepAIC",
  family = binomial,
  metric = "ROC",
  trControl = ctrl,
  preProcess = c("center", "scale")
)
```

Logistic Regression Model Results

The final model had strong predictive performance, with an accuracy of 94% and AUC of approximately 0.97 and balanced sensitivity and specificity. These results indicate that the logistic regression model was effective at distinguishing between AD and non-AD participants. Overall, logistic regression proved to be an interpretable and effective baseline approach for binary classification in this Alzheimer's dataset.

```r
#to look at the coefficients of the final model from above
final_fit <- fit_step_cv$finalModel
summary(final_fit)
```

```

Call:
NULL

Coefficients:
            Estimate Std. Error z value Pr(>|z|)    
(Intercept)  0.28560    0.11238   2.541 0.011045 *  
age         -0.56599    0.12740  -4.442 8.89e-06 ***
educ         0.57072    0.10387   5.495 3.92e-08 ***
weight      -0.22209    0.09799  -2.267 0.023415 *  
cdrglob      3.09053    0.19316  16.000  < 2e-16 ***
agitsev     -0.20726    0.09309  -2.227 0.025981 *  
remdates     0.16103    0.10701   1.505 0.132391    
travel       0.19450    0.12608   1.543 0.122915    
naccmmse    -0.99086    0.20107  -4.928 8.31e-07 ***
memunits    -0.95660    0.13205  -7.244 4.35e-13 ***
digif        0.24701    0.10263   2.407 0.016095 *  
animals     -0.58902    0.13361  -4.409 1.04e-05 ***
trailb       0.93274    0.13837   6.741 1.58e-11 ***
csfvol       0.51451    0.13265   3.879 0.000105 ***
lhippo      -0.30435    0.14478  -2.102 0.035540 *  
lparcort     0.38107    0.14007   2.721 0.006518 ** 
lent        -0.28074    0.13801  -2.034 0.041942 *  
lposcin      0.32187    0.15747   2.044 0.040953 *  
rposcin     -0.36399    0.15889  -2.291 0.021972 *  
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

(Dispersion parameter for binomial family taken to be 1)

    Null deviance: 2955.39  on 2160  degrees of freedom
Residual deviance:  828.67  on 2142  degrees of freedom
```

```
AIC: 866.67

Number of Fisher Scoring iterations: 7
```

```r
#evaluate everything on the test set and look at evaluation metrics
p_test <- predict(fit_step_cv, newdata = test_data, type = "prob")[, "AD"]
roc_obj  <- roc(test_data$diagnosis_bin, p_test)
```

```
Setting levels: control = NoAD, case = AD

Setting direction: controls < cases
```

```r
auc_test <- auc(roc_obj)
pred_class <- factor(ifelse(p_test > 0.5, "AD", "NoAD"),
                     levels = c("NoAD", "AD"))
cm <- confusionMatrix(pred_class, test_data$diagnosis_bin)
auc_test
```

```
Area under the curve: 0.9738
```

```r
cm$overall["Accuracy"]
```

```
 Accuracy
0.9424861
```

```r
cm$byClass[c("Sensitivity", "Specificity")]
```

```
Sensitivity Specificity
  0.9575163   0.9227468
```

Interpretting the Coefficients of the Logistic Regression Model

Interestingly, age showed a negative coefficient in the logistic model (-0.475). This means that holding all other variables constant, increasing age is associated with a lower log-odds of being classified as AD. This is something worth exploring further in AD literature. Additionally, weight also showed a negative coefficient (-0.3613). AD is often preceded by late-life weight loss and thus this association between higher body weight and low probability of AD from our logistic regression is consistent with what is commonly seen throughout AD research. In terms of cognitive ability, the ability to travel independently showed a positive coefficient in the logistic model (0.343). This can be interpreted as difficulty traveling independently indicates a higher likelihood of AD. Additionally, lower cognitive test scores strongly predict AD diagnosis and Poor memory performance indicate higher AD probability, with negative coefficients, -1.16 and -1.02 respectively. Frontal cortex volume showed a positive coefficient in the logistic model (0.429). This indicates that lower frontal cortical volume was associated with a higher probability of Alzheimer's diagnosis, which is consistent with the role of frontal lobe atrophy as a neurodegenerative marker in AD progression.

3. Multinomial Logistic Regression Model

Multinomial logistic regression builds upon the strengths of logistic regression by expanding the response variable to more than two classes. Within the context of our dataset, we are now able to differeniate between mild cognitive decline and dementia due to Alzheimer's Disease. An added benefit of multinomial logistic regression is that despite its increase in complexity, our model remains easy to interpret and apply. Our coefficients are still relevant and interprettable via their log-odds.

Next we will build a multinomial logsitic regression model so that we can see how adding in the third diagnosis option affects our results. Let's go ahead and select the same predictors as we used before to build the multinomial logsitic regression model.

```
raw.data = read.csv(here('alzheimer_data.csv'))
keep.cols = c( "diagnosis",
               "age",
               "educ",
               "weight",
               "cdrglob",
               "hallsev",
               "agitsev",
               "mealprep",
               "travel",
               "naccmmse",
               "memunits",
               "digif",
               "animals",
               "trailb",
               "csfvol",
               "frcort",
               "frcort",
               "lent",
               "lposcin",
                "rposcin")
df = raw.data[, keep.cols, drop = FALSE]
head(df)
```

```
  diagnosis age educ weight cdrglob hallsev agitsev mealprep travel naccmmse
1         0  74   12    233     0.5       0       0        0      0       30
2         0  56   16    110     0.0       0       0        0      0       29
3         0  77   18    137     0.0       0       0        0      0       30
4         0  74   20    112     0.0       0       0        0      0       30
5         1  75   14    127     0.5       0       0        0      0       27
6         0  72   16    141     0.0       0       0        0      0       30
  memunits digif animals trailb  csfvol  frcort frcort.1   lent lposcin rposcin
1        8     7      17    130 381.840 160.570  160.570 3.2000  3.7500  3.4400
2       17    11      25     47 366.622 187.874  187.874 3.6755  3.9091  4.2362
3       19     7      19     83 343.176 163.214  163.214 3.6207  3.8686  3.7062
4       11     6      23    100 332.880 165.120  165.120 4.3300  3.4500  3.5300
5        6     8      19     67 390.415 149.138  149.138 4.2328  3.1321  2.9051
6       16     7      14    100 345.600 140.220  140.220 3.8200  3.4500  2.9200
```

We will use the package **`glmnet`** as it has a built in function that allows us to do multinomial logistic regression and the textbook recommends it. The function is simple and straightforward to use. We first specify our matrix of predictors and target variable. We also specify that the family we are using is multinomial.

```
X = as.matrix(df[, 2:19]) # predictors
Y =  df$diagnosis # target
model =  glmnet(X, Y, family = "multinomial")
#summary(model)
#plot(model)
```

Next we will perform model validation and selection to improve upon the base model we already built. To do so, we will use cross validation with 5 and 10 folds as well as train/test split. Specifying `alpha=1` here performs LASSO regression which allows us to do variable selection. Overall cross validation as seen here performs several tasks in one - it allows us to compare models as well as ensure that our models don't overfit on the data.

Cross-Validation with 5 Folds

```
cv5  = cv.glmnet(X, Y, family = "multinomial", alpha=1, nfolds = 5)
pred5  = predict(cv5, newx = X, s = "lambda.min", type = "class")

# rmse
Y_num = as.numeric(Y)
pred_num5 = as.numeric(pred5)
rmse5 = sqrt(mean((Y_num - pred_num5)^2))
```

Cross-Validation with 10 Folds

```
cv10 = cv.glmnet(X, Y, family = "multinomial", alpha=1, nfolds = 10)
pred10 = predict(cv10, newx = X, s = "lambda.min", type = "class")

# rmse
pred_num10 = as.numeric(pred10)
rmse10 = sqrt(mean((Y_num - pred_num10)^2))
```

Train/Test Split

```
# 80 / 20 split
set.seed(533)
train_idx = sample(seq_len(nrow(X)), size = 0.8 * nrow(X))
cv1 = glmnet(X[train_idx, ], Y[train_idx], family = "multinomial")
pred = predict(cv1, newx = X[-train_idx, ], s = 0.01, type = "class")
accuracy.ttsplit = mean(pred == Y[-train_idx])

# rmse
Y_test_num = as.numeric(Y[-train_idx])
pred_num = as.numeric(pred)
rmse = sqrt(mean((Y_test_num - pred_num)^2))
```

Multinomial Logistic Regression Model Results

Overall, our strongest model was the one built using 10-fold cross validation. The accuracy of this model is 87% and its RMSE is the lowest out of the models that we tested. That being said, it is important to note that the model build from 5-fold cross validation has very similar statistics. This increases the confidence that I have in my model, as it tells me that our data was robust enough (2,700 rows) to properly train a model.

```r
results = data.frame(
  Method = c("1-Fold (Train/Test)", "5-Fold", "10-Fold"),
  Accuracy = c(accuracy.ttsplit, mean(pred5 == Y), mean(pred10 == Y)),
  RMSE = c(rmse, rmse5, rmse10)
)
print(results)
```

```
               Method  Accuracy      RMSE
1 1-Fold (Train/Test) 0.8462963 0.4059739
2              5-Fold 0.8666667 0.3741657
3             10-Fold 0.8659259 0.3751543
```

Final Model and Confusion Matrix

```r
set.seed(533)
train_idx = sample(seq_len(nrow(X)), size = 0.8 * nrow(X))

# winning model
cv10 = cv.glmnet(
  X[train_idx, ],
  Y[train_idx],
  family = "multinomial",
  alpha = 1,
  nfolds = 10
)
best_lambda = cv10$lambda.min

# Predict on test set
pred_test = predict(
  cv10,
  newx = X[-train_idx, ],
  s = best_lambda,
  type = "class"
)

pred_test = as.factor(as.vector(pred_test))
Y_test = as.factor(Y[-train_idx])
confusionMatrix(pred_test, Y_test)
```

```
Confusion Matrix and Statistics
```

```
          Reference
Prediction   0   1   2
         0 292  20   0
         1  22  82  21
         2   2  19  82

Overall Statistics

               Accuracy : 0.8444
                 95% CI : (0.8111, 0.874)
    No Information Rate : 0.5852
    P-Value [Acc > NIR] : <2e-16

                  Kappa : 0.7288

 Mcnemar's Test P-Value : 0.5329

Statistics by Class:

                     Class: 0 Class: 1 Class: 2
Sensitivity            0.9241   0.6777   0.7961
Specificity            0.9107   0.8974   0.9519
Pos Pred Value         0.9359   0.6560   0.7961
Neg Pred Value         0.8947   0.9060   0.9519
Prevalence             0.5852   0.2241   0.1907
Detection Rate         0.5407   0.1519   0.1519
Detection Prevalence   0.5778   0.2315   0.1907
Balanced Accuracy      0.9174   0.7875   0.8740
```

Interpretting the Coefficients of the Multinomial Logistic Regression Model

Outputted below are the coefficients of the 10-fold cross validation model obtained using automatic feature selection from LASSO.

The first observation that sticks out to me is that LASSO removed the majority of the predictors for the mild diagnosis class as most coefficients are zero. The exception to this are the `AGITSEV` (agitation or aggression severity), `DIGIF` (digit span forward trials correct), and the `LIPOSCN` (left posterior cingulate gray matter volume) coefficients, with that of the `LIPOSCN` being the greatest in magnitude. That being said, the `LIPOSCN` coefficient disappears for the no diagnosis class and is significantly negative for the severe diagnosis class. This tells me that the `LIPOSCN`predictor helps the model distinguish between the two diagnosis classes.

This is important because as seen by the confusion matrix, the model struggles the most with distinguishing between classes 1 and 2.

The next coefficient that sticks out to me is **`cdrglob`** (clinical dimentia rating - global score) as it has a value of -6.2 for the no diagnosis class, 0 for the mild diagnosis class, and 3.5 for the severe diagnosis class. This indicates that a greater CDR rating significantly decreases the odds of the patient not having Alzheimer's Disease, i.e. that a diagnosis is highly likely. This makes sense, as dementia is the primary symptom with Alzheimer's, so the CDR rating has a direct effect on diagnosis.

The other significant coefficient in the severe diagnosis group is `hallsev` (hallucinations severity). That being said, this coefficient is zero for the other classes. This tells me that the presence of any level of hallucinations strongly increases the odds that Alzheimer's Disease is present.

Finally, the cognitive function coefficients `naccmmse` (Total MMSE score using D-L-R-O-W) and `animals` (total number of animals named in 60 seconds) both helped to distinguish between a non-diagnosis and a diagnosis. Both coefficients are positive for class 0 but negative for class 2, indicating that low scores increase the odds of an Alzheimer's diagnosis. These coefficients also played a strong role in our logistic regression model from earlier, indicating that the multinomial logistic regression model is able to provide further depth and insight into an Alzheimer's diagnosis.

Overall I am satisfied that LASSO was able to reduce the coefficients significantly for many of our predictors.

```r
coef(cv10, s = "lambda.min")
```

```
$`0`
19 x 1 sparse Matrix of class "dgCMatrix"
               lambda.min
(Intercept) -4.455157e+00
age          3.866862e-02
educ        -1.467393e-01
weight       9.560310e-03
cdrglob     -6.201694e+00
hallsev      .
agitsev      3.886656e-01
mealprep    -4.860561e-02
travel      -1.090430e-01
naccmmse     2.468998e-01
memunits     1.529842e-01
digif       -1.105272e-01
animals      7.002991e-02
```

```
trailb      -1.138291e-02
csfvol      -8.548723e-03
frcort      -5.105321e-03
frcort.1    -1.978694e-17
lent         4.003176e-01
lposcin      .

$`1`
19 x 1 sparse Matrix of class "dgCMatrix"
             lambda.min
(Intercept) -0.39097269
age          .
educ         .
weight       .
cdrglob      .
hallsev      .
agitsev     -0.03857745
mealprep     .
travel       .
naccmmse     .
memunits     .
digif        0.02738610
animals      .
trailb       .
csfvol       .
frcort       .
frcort.1     .
lent         .
lposcin      0.17421076

$`2`
19 x 1 sparse Matrix of class "dgCMatrix"
               lambda.min
(Intercept)  4.846129e+00
age         -6.154357e-02
educ         7.027852e-02
weight      -6.522153e-03
cdrglob      3.482139e+00
hallsev      1.384885e+00
agitsev      .
mealprep     1.042114e-01
travel       3.459361e-01
naccmmse    -2.079754e-01
```

```
memunits    -5.919552e-02
digif         .
animals     -9.909249e-02
trailb       4.739335e-03
csfvol       5.959197e-03
frcort       1.503158e-02
frcort.1     3.372668e-17
lent        -3.035378e-01
lposcin     -5.960227e-02
```