Alzheimer's Analysis

Sneha Jacob, Alina Bangash, Jeremy Udo, Sana Akhtar, Jake Schwartz

Introduction

Our team decided to analyze the Alzheimer's Disease dataset created by Rabei El Kharoua on Kaggle. We decided to proceed with this topic since Alzheimer's is a very common illness without a cure that affects millions of people both directly and indirectly. This data set includes data from 2,149 patients and has 34 variables.

Which factors are strongly associated with a diagnosis of Alzheimer's Disease?

E.g. "1. Neural Networks (Jake Schwartz, Alina Akhtar)"

```
alzheimers_data = alzheimers_data <- read.csv("alzheimers_disease_data.csv")
colnames(alzheimers_data)
```

```
##  [1] "PatientID"                 "Age"
##  [3] "Gender"                    "Ethnicity"
##  [5] "EducationLevel"            "BMI"
##  [7] "Smoking"                   "AlcoholConsumption"
##  [9] "PhysicalActivity"          "DietQuality"
## [11] "SleepQuality"              "FamilyHistoryAlzheimers"
## [13] "CardiovascularDisease"     "Diabetes"
## [15] "Depression"                "HeadInjury"
## [17] "Hypertension"              "SystolicBP"
## [19] "DiastolicBP"               "CholesterolTotal"
## [21] "CholesterolLDL"            "CholesterolHDL"
## [23] "CholesterolTriglycerides"  "MMSE"
## [25] "FunctionalAssessment"      "MemoryComplaints"
## [27] "BehavioralProblems"        "ADL"
## [29] "Confusion"                 "Disorientation"
## [31] "PersonalityChanges"        "DifficultyCompletingTasks"
## [33] "Forgetfulness"             "Diagnosis"
## [35] "DoctorInCharge"
```

```
table(alzheimers_data$Diagnosis)
```

```
##
##    0    1
## 1389  760
```

Initialize Model, 2 hidden layers, uses ReLU activation, and binary cross entropy with logits loss because it says in torch modules it is more stable than sigmoid

```
library(torch)
```

```
## Warning: package 'torch' was built under R version 4.3.3
```

```
alzheimers_net = nn_module(
  "class_net",

  initialize = function(){
    self$layer1 = nn_linear(in_features = ncol(x), out_features = 64)
    self$layer2 = nn_linear(in_features = 64, out_features = 32)
    self$output = nn_linear(in_features = 32, out_features = 1)
  },
  forward = function(x){
    x %>%
      self$layer1() %>%
      nnf_relu() %>%
      self$layer2() %>%
```

```
      nnf_relu() %>%
      self$output()
  }
)
```

Convert columns to numeric

```
alzheimers_data = alzheimers_data[, -which(names(alzheimers_data) == "DoctorInCharge")]
alzheimers_data = alzheimers_data[, -which(names(alzheimers_data) == "PatientID")]

alzheimers_data$Gender <- as.numeric(factor(alzheimers_data$Gender))
alzheimers_data$Ethnicity <- as.numeric(factor(alzheimers_data$Ethnicity))
alzheimers_data$EducationLevel <- as.numeric(factor(alzheimers_data$EducationLevel))
alzheimers_data$Smoking <- as.numeric(factor(alzheimers_data$Smoking))
alzheimers_data$AlcoholConsumption <- as.numeric(factor(alzheimers_data$AlcoholConsumption))
alzheimers_data$PhysicalActivity <- as.numeric(factor(alzheimers_data$PhysicalActivity))
alzheimers_data$DietQuality <- as.numeric(factor(alzheimers_data$DietQuality))
alzheimers_data$SleepQuality <- as.numeric(factor(alzheimers_data$SleepQuality))
alzheimers_data$FamilyHistoryAlzheimers <- as.numeric(factor(alzheimers_data$FamilyHistoryAlzheimers))
alzheimers_data$CardiovascularDisease <- as.numeric(factor(alzheimers_data$CardiovascularDisease))
alzheimers_data$Diabetes <- as.numeric(factor(alzheimers_data$Diabetes))
alzheimers_data$Depression <- as.numeric(factor(alzheimers_data$Depression))
alzheimers_data$HeadInjury <- as.numeric(factor(alzheimers_data$HeadInjury))
alzheimers_data$Hypertension <- as.numeric(factor(alzheimers_data$Hypertension))

# Normalize the numeric columns, but the -ncol removes the last column which is diagnosis because we do

alzheimers_data[, -ncol(alzheimers_data)] <- scale(alzheimers_data[, -ncol(alzheimers_data)])

#Applies as.numeric to all data because it wasnt working with above earlier
alzheimers_data = data.frame(lapply(alzheimers_data, as.numeric))

#See classifications in table
table(alzheimers_data$Diagnosis)
```

```
##
##    0    1
## 1389  760
```

Split Data and Convert Into Tensors

```
x = as.matrix(alzheimers_data[, -32]) #All but diagnose column
y = alzheimers_data$Diagnosis

x_tensor = torch_tensor(as.matrix(x), dtype = torch_float())
y_tensor = torch_tensor(as.numeric(y), dtype = torch_float())
y_tensor
```

```
## torch_tensor
##  0
##  0
##  0
```

```
##  0
##  0
##  0
##  0
##  1
##  0
##  0
##  0
##  0
##  0
##  1
##  0
##  1
##  1
##  1
##  0
##  1
##  1
##  0
##  0
##  1
##  1
##  0
##  0
##  0
##  0
##  0
## ... [the output was truncated (use n=-1 to disable)]
## [ CPUFloatType{2149} ]
```

Neural Net Steps with K-Folds cross validation

```r
library(caret)
```

```
## Loading required package: ggplot2
```

```
## Loading required package: lattice
```

```r
# Define K-fold cross-validation parameters
k <- 10
folds <- createFolds(y, k = k, list = TRUE, returnTrain = TRUE)

result <- numeric(k)

for (i in 1:k) {
  cat("Fold:", i, "\n")

  # Split the data
  train_indices <- folds[[i]]
  test_indices <- setdiff(seq_len(nrow(alzheimers_data)), train_indices)

  x_train <- x_tensor[train_indices, ]
  y_train <- y_tensor[train_indices]$unsqueeze(2)
  x_test <- x_tensor[test_indices, ]
```

```
  y_test <- y_tensor[test_indices]$unsqueeze(2)
  # Initialize the model
  neural_model <- alzheimers_net()
  optimizer <- optim_adam(neural_model$parameters, lr = 0.001)
  criterion <- nn_bce_with_logits_loss()

  # Training loop
  num_epochs <- 100
  for (epoch in 1:num_epochs) {
    optimizer$zero_grad()
    outputs <- neural_model(x_train)
    loss <- criterion(outputs, y_train)
    loss$backward()
    optimizer$step()


    if (epoch %% 10 == 0) {
    cat("Epoch:", epoch, "Loss:", loss$item(), "\n")
    }
  }

  # Evaluate the model
  neural_model$eval()
  with_no_grad({
    outputs <- neural_model(x_test)
    predictions <- torch_sigmoid(outputs) > 0.5
    accuracy <- mean(as_array(predictions) == as_array(y_test))
    result[i] <- accuracy
  })
}
```

```
## Fold: 1
## Epoch: 10 Loss: 0.6548127
## Epoch: 20 Loss: 0.6171016
## Epoch: 30 Loss: 0.5649083
## Epoch: 40 Loss: 0.4927339
## Epoch: 50 Loss: 0.4077709
## Epoch: 60 Loss: 0.3250087
## Epoch: 70 Loss: 0.2560219
## Epoch: 80 Loss: 0.2022445
## Epoch: 90 Loss: 0.1597737
## Epoch: 100 Loss: 0.1244614
## Fold: 2
## Epoch: 10 Loss: 0.6562297
## Epoch: 20 Loss: 0.6070468
## Epoch: 30 Loss: 0.5470157
## Epoch: 40 Loss: 0.4722339
## Epoch: 50 Loss: 0.3912062
## Epoch: 60 Loss: 0.3151656
## Epoch: 70 Loss: 0.253592
## Epoch: 80 Loss: 0.206868
## Epoch: 90 Loss: 0.1699495
## Epoch: 100 Loss: 0.1379056
## Fold: 3
```

```
## Epoch: 10 Loss: 0.6644116
## Epoch: 20 Loss: 0.6303245
## Epoch: 30 Loss: 0.5837034
## Epoch: 40 Loss: 0.5214015
## Epoch: 50 Loss: 0.4472862
## Epoch: 60 Loss: 0.3695334
## Epoch: 70 Loss: 0.2972417
## Epoch: 80 Loss: 0.2365288
## Epoch: 90 Loss: 0.1880703
## Epoch: 100 Loss: 0.1489289
## Fold: 4
## Epoch: 10 Loss: 0.671301
## Epoch: 20 Loss: 0.63283
## Epoch: 30 Loss: 0.5869412
## Epoch: 40 Loss: 0.5256845
## Epoch: 50 Loss: 0.4495069
## Epoch: 60 Loss: 0.3660335
## Epoch: 70 Loss: 0.2893146
## Epoch: 80 Loss: 0.230259
## Epoch: 90 Loss: 0.1868162
## Epoch: 100 Loss: 0.1524627
## Fold: 5
## Epoch: 10 Loss: 0.6763661
## Epoch: 20 Loss: 0.6351676
## Epoch: 30 Loss: 0.5947539
## Epoch: 40 Loss: 0.5492331
## Epoch: 50 Loss: 0.4930807
## Epoch: 60 Loss: 0.428286
## Epoch: 70 Loss: 0.3595619
## Epoch: 80 Loss: 0.2942329
## Epoch: 90 Loss: 0.2380775
## Epoch: 100 Loss: 0.1914133
## Fold: 6
## Epoch: 10 Loss: 0.6847839
## Epoch: 20 Loss: 0.6369051
## Epoch: 30 Loss: 0.5914015
## Epoch: 40 Loss: 0.5373437
## Epoch: 50 Loss: 0.4724266
## Epoch: 60 Loss: 0.3992122
## Epoch: 70 Loss: 0.3256584
## Epoch: 80 Loss: 0.2599944
## Epoch: 90 Loss: 0.2076306
## Epoch: 100 Loss: 0.1667324
## Fold: 7
## Epoch: 10 Loss: 0.6640218
## Epoch: 20 Loss: 0.6219987
## Epoch: 30 Loss: 0.5739584
## Epoch: 40 Loss: 0.5108392
## Epoch: 50 Loss: 0.4336388
## Epoch: 60 Loss: 0.3501763
## Epoch: 70 Loss: 0.2754579
## Epoch: 80 Loss: 0.2174327
## Epoch: 90 Loss: 0.1730146
## Epoch: 100 Loss: 0.1367406
```

```
## Fold: 8
## Epoch: 10 Loss: 0.6500049
## Epoch: 20 Loss: 0.6161366
## Epoch: 30 Loss: 0.5710318
## Epoch: 40 Loss: 0.5081824
## Epoch: 50 Loss: 0.4279924
## Epoch: 60 Loss: 0.3402181
## Epoch: 70 Loss: 0.2642025
## Epoch: 80 Loss: 0.2079965
## Epoch: 90 Loss: 0.1658514
## Epoch: 100 Loss: 0.1307195
## Fold: 9
## Epoch: 10 Loss: 0.6708925
## Epoch: 20 Loss: 0.6287234
## Epoch: 30 Loss: 0.5769619
## Epoch: 40 Loss: 0.510579
## Epoch: 50 Loss: 0.4320328
## Epoch: 60 Loss: 0.3494756
## Epoch: 70 Loss: 0.2738889
## Epoch: 80 Loss: 0.2124735
## Epoch: 90 Loss: 0.1658315
## Epoch: 100 Loss: 0.129988
## Fold: 10
## Epoch: 10 Loss: 0.6495581
## Epoch: 20 Loss: 0.6188761
## Epoch: 30 Loss: 0.576013
## Epoch: 40 Loss: 0.5133113
## Epoch: 50 Loss: 0.432047
## Epoch: 60 Loss: 0.3440225
## Epoch: 70 Loss: 0.2674099
## Epoch: 80 Loss: 0.2103252
## Epoch: 90 Loss: 0.1677094
## Epoch: 100 Loss: 0.1327588
```

```r
# Print cross-validation results
cat("Cross-Validation Accuracy:", mean(result), "\n")
```

```
## Cross-Validation Accuracy: 0.9306629
```