CLEAN_A2_P2

Study group 4

21/9/2020

Loading packages

```
pacman::p_load(readr,dplyr,stringr,lmerTest,Metrics,caret, data.table, tidyr)
```

Exercise 1) Testing model performance

- recreate the models you chose last time (just write the model code again and apply it to your training data (from the first assignment))

```
# Load training Data
train_df <- read_csv("data_clean.csv")
```

```
## Parsed with column specification:
## cols(
##   .default = col_double(),
##   Ethnicity = col_character(),
##   Diagnosis = col_character(),
##   Gender = col_character()
## )
```

```
## See spec(...) for full column specifications.
```

```
train_df <- train_df[!is.na(train_df$Socialization), 1:22]
train_df <- train_df[!is.na(train_df$CHI_MLU), 1:22]

# Model from last time
max <-lmer(CHI_MLU ~
             I(Visit^2)+Visit*Diagnosis+ Socialization*Diagnosis + verbalIQ1+
             (1|Child.ID),
           train_df,
           REML = F)
```

- create the test dataset (apply the code from assignment 1 to clean up the 3 test datasets)

```
CleanUpData <- function(Demo,LU,Word){

  Speech <- merge(LU, Word) %>%
    rename(
```

```
      Child.ID = SUBJ,
      Visit=VISIT) %>%
    mutate(
      Visit = as.numeric(str_extract(Visit, "\\d")),
      Child.ID = gsub("\\.","", Child.ID)
      ) %>%
    dplyr::select(
      Child.ID, Visit, MOT_MLU, CHI_MLU, types_MOT, types_CHI, tokens_MOT, tokens_CHI
    )

  Demo <- Demo %>%
    dplyr::select(
      Child.ID, Visit, Ethnicity, Diagnosis, Gender, Age, ADOS, MullenRaw, ExpressiveLangRaw, Socializa
    ) %>%
    mutate(
      Child.ID = gsub("\\.","", Child.ID)
    )

  Data=merge(Demo,Speech,all=T)

  Data1= Data %>%
     subset(Visit=="1") %>%
     dplyr::select(Child.ID, ADOS, ExpressiveLangRaw, MullenRaw, Socialization) %>%
     rename(Ados1 = ADOS,
            verbalIQ1 = ExpressiveLangRaw,
            nonverbalIQ1 = MullenRaw,
            Socialization1 = Socialization)

  Data=merge(Data, Data1, all=T) %>%
    mutate(
      Child.ID = as.numeric(as.factor(as.character(Child.ID))),
      Visit = as.numeric(as.character(Visit)),
      Gender = recode(Gender,
         "1" = "M",
         "2" = "F"),
      Diagnosis = recode(Diagnosis,
         "A"  = "ASD",
         "B"  = "TD")
    )

  return(Data)
}


#- create the test dataset (apply the code from assignment 1 or my function to clean up the 3 test data
# Test data
Demo <- read_csv("demo_test.csv")
```

```
## Parsed with column specification:
## cols(
##   .default = col_double(),
##   Child.ID = col_character(),
##   Ethnicity = col_character(),
```

```
##   Diagnosis = col_character(),
##   Birthdate = col_character()
## )

## See spec(...) for full column specifications.
```

```
LU <- read_csv("LU_test.csv")
```

```
## Parsed with column specification:
## cols(
##   SUBJ = col_character(),
##   VISIT = col_character(),
##   MOT_MLU = col_double(),
##   MOT_LUstd = col_double(),
##   MOT_LU_q1 = col_double(),
##   MOT_LU_q2 = col_double(),
##   MOT_LU_q3 = col_double(),
##   CHI_MLU = col_double(),
##   CHI_LUstd = col_double(),
##   CHI_LU_q1 = col_double(),
##   CHI_LU_q2 = col_double(),
##   CHI_LU_q3 = col_double()
## )
```

```
token <- read_csv("token_test.csv")
```

```
## Parsed with column specification:
## cols(
##   SUBJ = col_character(),
##   VISIT = col_character(),
##   types_MOT = col_double(),
##   types_CHI = col_double(),
##   types_shared = col_double(),
##   tokens_MOT = col_double(),
##   tokens_CHI = col_double(),
##   X = col_logical()
## )
```

```
df_test <- CleanUpData(Demo, LU, token)
df_test <- df_test[!is.na(df_test$CHI_MLU), 1:20]
```

- calculate performance of the model on the training data: root mean square error is a good measure. (Tip: google the function rmse())

```
rmse(train_df$CHI_MLU, fitted(max))
```

```
## [1] 0.3952478
```

```
#0.45

mean(train_df$CHI_MLU)
```

```
## [1] 1.993246
```

```
#1.99
```

The rmse is quite big compared to the mean, which could indicate that our model is not good enough to predict the values.

- test the performance of the model on the test data

```
#This shows the predicted MLU
predict_chi <- predict(max, newdata = df_test, allow.new.levels = TRUE)
predict_chi
```

```
##        1        3        4        5        6        7        8        9
## 1.196107 1.644384 1.717515 1.698474 1.587259 2.528492 2.938933 2.938528
##       10       11       12       13       14       15       16       17
## 3.049900 3.056352 2.855909 1.558558 1.905265 2.083316 1.710306 1.678518
##       18       19       20       21       22       23       24       25
## 2.026191 1.678355 2.201443 2.579728 2.951364 3.211090 3.404957 1.340814
##       26       27       28       29       30       31       32       33
## 1.942847 2.393498 2.692767 2.926178 3.060837 1.132990 1.669236 2.133044
##       34       35       36
## 2.517837 2.790720 2.971430
```

```
#the fit of the predicted
rmse(df_test$CHI_MLU, predict_chi)
```

```
## [1] 0.5705706
```

```
mean(df_test$CHI_MLU)
```

```
## [1] 2.103934
```

```
#This doesn't seem like a good performance at all! :(
```

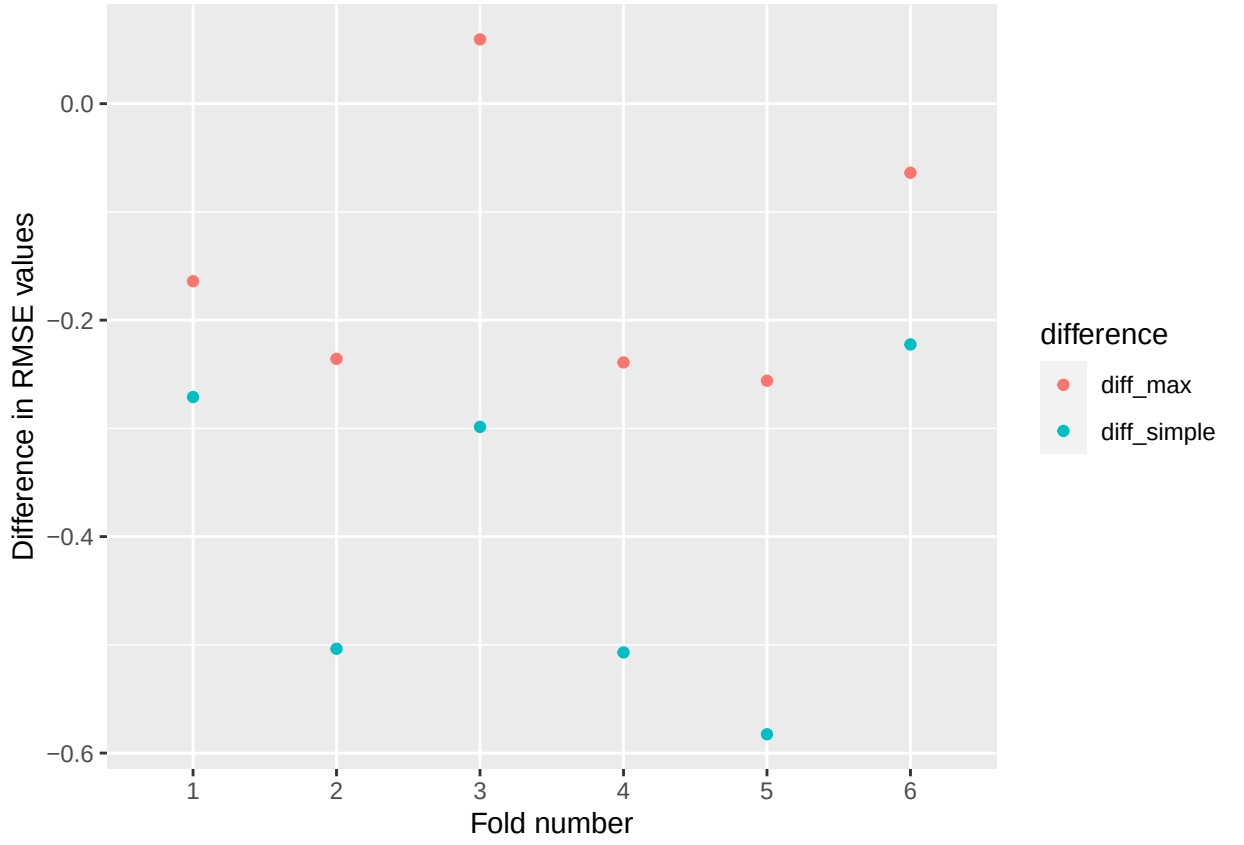
Exercise 2) Model Selection via Cross-validation (N.B: ChildMLU!)

- Use cross-validation to compare your model from last week with the basic model (Child MLU as a function of Time and Diagnosis, and don't forget the random effects!)

```
#Comparing basic model with our model in the same folds

#Creating folds
k = 6
folds = createFolds(unique(train_df$Child.ID), k = k, list = TRUE, returnTrain = FALSE)

#Creating empty lists for RMSE values
trainRMSE_our= rep(NA, k)
trainRMSE_simple= rep(NA,k)
testRMSE_our = rep(NA, k)
```

```
testRMSE_simple = rep(NA, k)

#Looping through each loop
i = 1
for (fold in folds){
  #Creating training data set and test data set
  train_os = subset(train_df, !(Child.ID %in% fold))
  test_os = subset(train_df, Child.ID %in% fold)
  #Making models
  our_model = lmer(CHI_MLU ~ I(Visit^2)+Visit*Diagnosis+ Socialization*Diagnosis + verbalIQ1 +
                     (1|Child.ID), train_os, REML = F)
  simple_model = lmer(CHI_MLU ~Visit*Diagnosis + (1+Visit|Child.ID), train_os, REML = FALSE)
  #Making prediction columns predicting the model
    #For our model
  test_os$prediction = predict(our_model, test_os, allow.new.levels = TRUE)
  train_os$prediction = fitted(our_model)
    #For simple model
  test_os$prediction2 = predict(simple_model, test_os, allow.new.levels = TRUE)
  train_os$prediction2 = fitted(simple_model)
  #Calculating RMSE values
    #For our model
  trainRMSE_our[i] = RMSE(train_os$CHI_MLU, fitted(our_model))
  testRMSE_our[i] = RMSE(test_os$CHI_MLU, test_os$prediction)
    #For simple model
  trainRMSE_simple[i] = RMSE(train_os$CHI_MLU, fitted(simple_model))
  testRMSE_simple[i] = RMSE(test_os$CHI_MLU, test_os$prediction2)
  i = i + 1
}

#Making dataframe with RMSE values
  #Making difference columns for our model
diff_max <- trainRMSE_our - testRMSE_our
  #Making difference columns for simple model
diff_simple <- trainRMSE_simple - testRMSE_simple

#Dataframe
our_vs_simple <- data.frame(trainRMSE_our, testRMSE_our, diff_max,
                            trainRMSE_simple, testRMSE_simple, diff_simple)
  #Fold column
our_vs_simple$folds <- c(1:6)

#Plotting our model against the simple model
our_vs_simple$folds <- as.factor(our_vs_simple$folds)
diff_df <- select(our_vs_simple, diff_simple, diff_max, folds)
#selecting relevant data
diff_df <- gather(our_vs_simple, difference, value, diff_max, diff_simple, factor_key = T)
#changing data format to long

#plotting
diff_df$difference <- as.factor(diff_df$difference)
ggplot(diff_df, aes(x = folds, y = value)) +
  geom_point(aes(color = difference)) +
  ylab('Difference in RMSE values') +
```

```
  xlab('Fold number')
```




```
#clearly 'our' model has the shortest distance to zero, and therefore it is better than the simple mode


#Mean (we want closest to 0)
  #our model on the training data is closest to the test data (report mean and SD)
mean(our_vs_simple$diff_our)
```

```
## Warning in mean.default(our_vs_simple$diff_our): argument is not numeric or
## logical: returning NA
```

```
## [1] NA
```

```
sd(our_vs_simple$diff_our)
```

```
## [1] NA
```

```
mean(our_vs_simple$diff_simple)
```

```
## [1] -0.3975565
```

```r
sd(our_vs_simple$diff_simple)
```

```
## [1] 0.1509526
```

- Now try to find the best possible predictive model of ChildMLU, that is, the one that produces the best cross-validated results.

```r
# Finding the best model in different folds for simplicity
  #Cross validation function
    #Out of the function we create some folds
k = 6
folds = createFolds(unique(train_df$Child.ID), k = k, list = TRUE, returnTrain = FALSE)

    #Making function that allows us to just put a test in
cross_valid <- function(data, m){

#Creating empty lists
trainRMSE = rep(NA, k)
testRMSE = rep(NA, k)

#Making loop
i = 1
for (fold in folds){
  #Datasets
  train = subset(train_df, !(Child.ID %in% fold))
  test = subset(train_df, Child.ID %in% fold)
  #Model
  m = lmer(m, train, REML = FALSE)
  #Prediction columns
  test$prediction = predict(m, test, allow.new.levels = TRUE)
  train$prediction = fitted(m)
  #Calculating RMSE
  trainRMSE[i] = RMSE(train$CHI_MLU, fitted(m))
  testRMSE[i] = RMSE(test$CHI_MLU, test$prediction)
  i = i + 1
}
#Putting results into dataframe
result = data.frame(trainRMSE, testRMSE)
  return(result)
}

#testing it
growth <-  CHI_MLU ~ I(Visit^2) + Visit*Diagnosis + (1+I(Visit^2)|Child.ID)
test_1 <- cross_valid(train_df, growth)
```

```
## Warning in checkConv(attr(opt, "derivs"), opt$par, ctrl = control$checkConv, :
## Model failed to converge with max|grad| = 0.00273114 (tol = 0.002, component 1)

## Warning in checkConv(attr(opt, "derivs"), opt$par, ctrl = control$checkConv, :
## Model failed to converge with max|grad| = 0.110543 (tol = 0.002, component 1)

## Warning in checkConv(attr(opt, "derivs"), opt$par, ctrl = control$checkConv, :
## Model failed to converge with max|grad| = 0.0401415 (tol = 0.002, component 1)
```

```
## Warning in checkConv(attr(opt, "derivs"), opt$par, ctrl = control$checkConv, :
## Model failed to converge with max|grad| = 0.00535001 (tol = 0.002, component 1)
```

```
test_1$diff1 <- test_1$trainRMSE - test_1$testRMSE

bmaxnv <-  CHI_MLU ~ I(Visit^2)+Visit*Diagnosis+ Socialization*Diagnosis + nonVerbalIQ1+(1|Child.ID)
test_2 <- cross_valid(train_df, bmaxnv)
test_2$diff2 <- test_2$trainRMSE - test_2$testRMSE

bmaxv <-  CHI_MLU ~ I(Visit^2)+Visit*Diagnosis+ Socialization*Diagnosis + verbalIQ1+(1|Child.ID)
test_3 <- cross_valid(train_df, bmaxv)
test_3$diff3 <- test_3$trainRMSE - test_3$testRMSE

bv <-  CHI_MLU ~ I(Visit^2)+Visit*Diagnosis + verbalIQ1+(1|Child.ID)
test_4 <- cross_valid(train_df, bv)
test_4$diff4 <- test_4$trainRMSE - test_4$testRMSE

bn <-  CHI_MLU ~ I(Visit^2)+Visit*Diagnosis + nonVerbalIQ1+(1|Child.ID)
test_5 <- cross_valid(train_df, bn)
test_5$diff5 <- test_5$trainRMSE - test_5$testRMSE

#Mean and sd on the test results
m1 <- mean(test_1$diff1)
sd1 <- sd(test_1$diff1)
m2 <- mean(test_2$diff2)
sd2 <- sd(test_2$diff2)
m3 <- mean(test_3$diff3)
sd3 <- sd(test_3$diff3)
m4 <- mean(test_4$diff4)
sd4 <- sd(test_4$diff4)
m5 <- mean(test_5$diff5)
sd5 <- sd(test_5$diff5)

#Making dataframe
models <- c(growth, bmaxnv, bmaxv, bv, bn)
models <- as.character(models)
meansRMSE <- c(m1,m2,m3,m4,m5)
sdRMSE <- c(sd1,sd2,sd3,sd4,sd5)

comparison_df <- data.frame(models, meansRMSE, sdRMSE)
```

Exercise 3) Assessing the single child

- how does the child fare in ChildMLU compared to the average TD child at each visit? Define the distance in terms of absolute difference between this Child and the average TD.

```
#Finding BERNIE
Bernie <- subset(df_test, Child.ID == 4)
mean(Bernie$CHI_MLU)
```

```
## [1] 2.510762
```

```
Bernie$CHI_MLU
```

```
## [1] 1.651685 2.035928 2.303167 2.531034 3.774336 2.768421
```

```
#Subsetting for TD
test_TD <- subset(df_test, Diagnosis == "TD")

#average for TD child at each visit

meanMLUTD <- aggregate(test_TD[, 12], list(test_TD$Visit), mean)

meanMLUTD <- rename(meanMLUTD, "Visit" = "Group.1")
meanMLUTD <- rename(meanMLUTD, "mean_MLU_TD" = "x")

#Inserting the mean in Bernies dataset
Bernie <- merge(Bernie, meanMLUTD, by = "Visit")

#Calculating the absolute difference
Bernie$AD_TD <- (Bernie$CHI_MLU - Bernie$mean_MLU_TD )

mean(Bernie$AD_TD)
```

```
## [1] 0.1844681
```

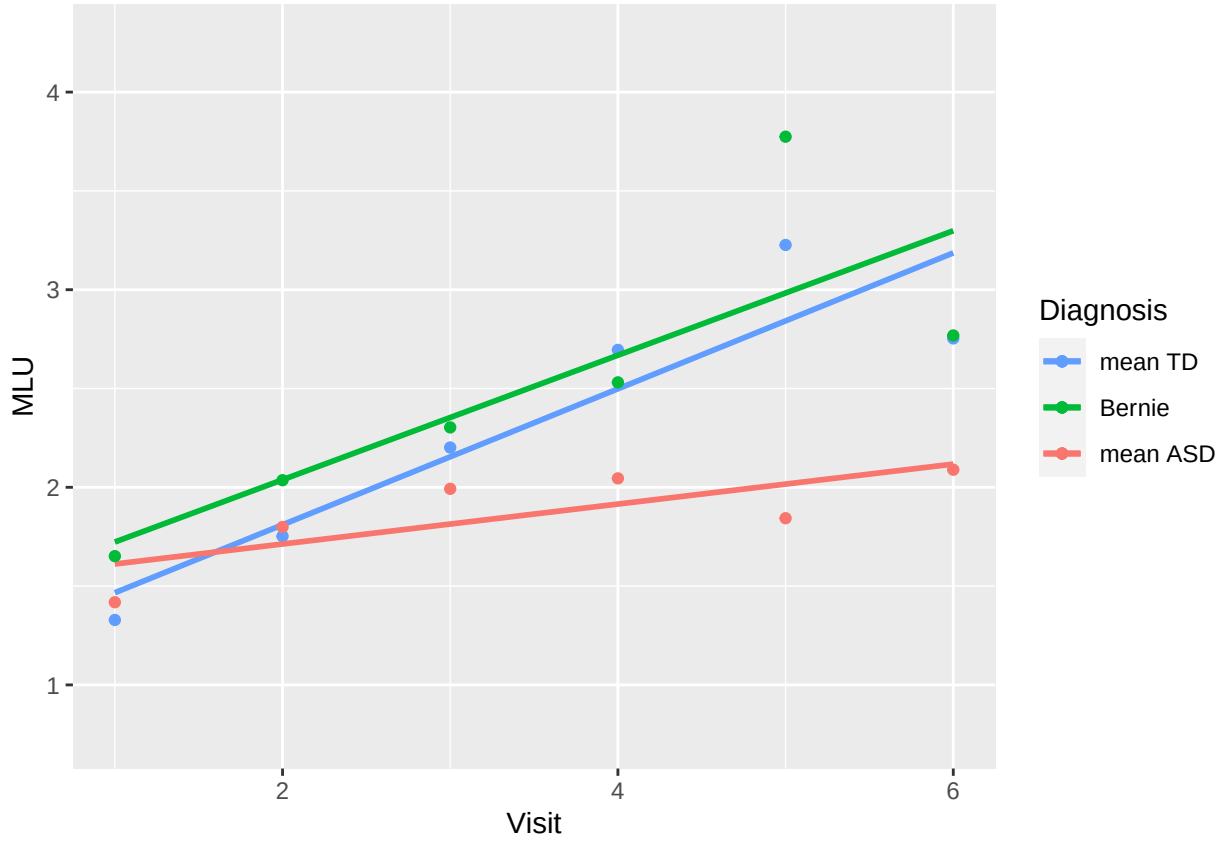
```
#Subsetting for ASD
test_ASD <- subset(df_test, Diagnosis == "ASD")

#average for TD child at each visit

meanMLUASD <- aggregate(test_ASD[, 12], list(test_ASD$Visit), mean)

meanMLUASD <- rename(meanMLUASD, c("Visit" = "Group.1", "mean_MLU_ASD" = "x"))

#Inserting the mean in Bernies dataset
Bernie <- merge(Bernie, meanMLUASD, by = "Visit")

#Calculating the absolute difference
Bernie$AD_ASD <- (Bernie$CHI_MLU - Bernie$mean_MLU_ASD )

mean(Bernie$AD_ASD)
```

```
## [1] 0.646119
```

```
#Plotting Bernie against means for TD and ASD (geom_smooth er ikke nødvendigvis det pæneste men synes d
ggplot(Bernie) +
  geom_point(aes(y=mean_MLU_TD , x=Visit, color = "red" , )) +
  geom_smooth(aes(y=mean_MLU_TD, x=Visit, color = "red"), method = lm , alpha = 0) +
  geom_point(aes(y=mean_MLU_ASD , x=Visit, color = "blue")) +
  geom_smooth(aes(y=mean_MLU_ASD, x=Visit, color = "blue"), method = lm, alpha = 0) +
  geom_point(aes(y=CHI_MLU, x=Visit, color = "green")) +
  geom_smooth(aes(y=CHI_MLU, x=Visit, color = "green"), method = lm, alpha = 0) +
```

```
  scale_color_discrete(name="Diagnosis",
                       breaks=c("red", "green", "blue"),
                       labels=c("mean TD", "Bernie", "mean ASD")) +
  ylab("MLU")
```

```
## `geom_smooth()` using formula 'y ~ x'
## `geom_smooth()` using formula 'y ~ x'
## `geom_smooth()` using formula 'y ~ x'
```



- how does the child fare compared to the model predictions at Visit 6? Is the child below or above expectations? (tip: use the predict() function on Bernie's data only and compare the prediction with the actual performance of the child) #See below

```
# predicted values for each visit
Bernie$predict_max <- predict(max, newdata = Bernie, allow.new.levels = TRUE)

# Calculating the difference between predicted and and actual values
Bernie$dif_max <- (Bernie$predict_max - Bernie$CHI_MLU)
Bernie$dif_max
```

```
## [1]  0.02666965  0.16551504  0.27656084  0.42032912 -0.56324674  0.63653612
```

```
# Plotting the values
ggplot(Bernie)+
  geom_point(aes(y = predict_max , x = Visit, color = "red")) +
  geom_point(aes(y = CHI_MLU, x = Visit, color = "blue")) +
  scale_color_discrete(name=" ",
                       breaks=c("red", "blue"),
                       labels=c("Predicted", "Bernie MLU")) + ylab("MLU")
```