Assignment 1 - Language Development in ASD - part 3

Maria Mareckova, Katarina Michelova, Natalia Makhova, Alexandra Pasternakova

October 3th, 2018

## Welcome to the third exciting part of the Language Development in ASD exercise

In this exercise we will delve more in depth with different practices of model comparison and model selection, by first evaluating your models from last time, then learning how to cross-validate models and finally how to systematically compare models.

N.B. There are several datasets for this exercise, so pay attention to which one you are using!

1. The (training) dataset from last time (the awesome one you produced :-) ).
2. The (test) datasets on which you can test the models from last time:

* Demographic and clinical data: <https://www.dropbox.com/s/ra99bdvm6fzay3g/demo_test.csv?dl=1>
* Utterance Length data: <https://www.dropbox.com/s/uxtqqzl18nwxowq/LU_test.csv?dl=1>
* Word data: <https://www.dropbox.com/s/1ces4hv8kh0stov/token_test.csv?dl=1>

### Exercise 1) Testing model performance

How did your models from last time perform? In this exercise you have to compare the results on the training data () and on the test data. Report both of them. Compare them. Discuss why they are different.

-Arecreate the models you chose last time (just write the model code again and apply it to your training data (from the first assignment)) B calculate performance of the model on the training data: root mean square error is a good measure. (Tip: google the function rmse()) C create the test dataset (apply the code from assignment 1 part 1 to clean up the 3 test datasets) D test the performance of the models on the test data (Tips: google the functions “predict()”) F optional: predictions are never certain, can you identify the uncertainty of the predictions? (e.g. google predictinterval())

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

merged\_data <- read.csv("merged\_data.csv")  
model <- lmer(CHI\_MLU~VISIT\* Diagnosis+I(VISIT^2)\*Diagnosis +I(VISIT^3)\* Diagnosis +(1+VISIT+I(VISIT^2) +I(VISIT^3)|SUBJ),merged\_data, REML = F)

## Warning in optwrap(optimizer, devfun, getStart(start, rho$lower, rho$pp), :  
## convergence code 1 from bobyqa: bobyqa -- maximum number of function  
## evaluations exceeded

## singular fit

summary(model)

## Linear mixed model fit by maximum likelihood . t-tests use  
## Satterthwaite's method [lmerModLmerTest]  
## Formula:   
## CHI\_MLU ~ VISIT \* Diagnosis + I(VISIT^2) \* Diagnosis + I(VISIT^3) \*   
## Diagnosis + (1 + VISIT + I(VISIT^2) + I(VISIT^3) | SUBJ)  
## Data: merged\_data  
##   
## AIC BIC logLik deviance df.resid   
## 505.3 578.7 -233.7 467.3 333   
##   
## Scaled residuals:   
## Min 1Q Median 3Q Max   
## -2.3525 -0.5183 -0.0914 0.4893 2.9537   
##   
## Random effects:  
## Groups Name Variance Std.Dev. Corr   
## SUBJ (Intercept) 0.0666874 0.25824   
## VISIT 0.2800597 0.52921 -0.22   
## I(VISIT^2) 0.0335104 0.18306 0.46 -0.91   
## I(VISIT^3) 0.0002949 0.01717 -0.53 0.76 -0.96  
## Residual 0.0994937 0.31543   
## Number of obs: 352, groups: SUBJ, 61  
##   
## Fixed effects:  
## Estimate Std. Error df t value Pr(>|t|)   
## (Intercept) 0.721491 0.221041 142.425839 3.264 0.00138 \*\*  
## VISIT 0.714296 0.267478 111.601454 2.670 0.00871 \*\*  
## DiagnosisB 0.383758 0.303682 141.950990 1.264 0.20841   
## I(VISIT^2) -0.168734 0.087160 101.060716 -1.936 0.05567 .   
## I(VISIT^3) 0.013563 0.008258 109.766523 1.642 0.10335   
## VISIT:DiagnosisB -0.660489 0.366629 110.715868 -1.802 0.07434 .   
## DiagnosisB:I(VISIT^2) 0.347278 0.119316 100.361713 2.911 0.00444 \*\*  
## DiagnosisB:I(VISIT^3) -0.036428 0.011309 109.905621 -3.221 0.00168 \*\*  
## ---  
## Signif. codes: 0 '\*\*\*' 0.001 '\*\*' 0.01 '\*' 0.05 '.' 0.1 ' ' 1  
##   
## Correlation of Fixed Effects:  
## (Intr) VISIT DgnssB I(VISIT^2 I(VISIT^3 VISIT: DB:I(VISIT^2  
## VISIT -0.893   
## DiagnosisB -0.728 0.650   
## I(VISIT^2) 0.862 -0.974 -0.627   
## I(VISIT^3) -0.828 0.931 0.603 -0.988   
## VISIT:DgnsB 0.652 -0.730 -0.892 0.711 -0.679   
## DB:I(VISIT^2 -0.630 0.712 0.861 -0.730 0.722 -0.974   
## DB:I(VISIT^3 0.605 -0.680 -0.827 0.721 -0.730 0.930 -0.988   
## convergence code: 1  
## singular fit

#doing B  
##root mean square error = rmse  
  
preds <- predict(model, type= "response")  
  
rmse(merged\_data$CHI\_MLU, preds)#0.2570438

## [1] 0.2593025

#doiong C  
demo\_test<- read.csv("demo\_test.csv")  
#we see dots in the end of the names, extra variables, rename collumn, mullenrae into nonverbalIQ, expressivelangraw into verbalIQ  
demo\_test <- dplyr::rename(demo\_test, SUBJ = Child.ID)  
demo\_test$SUBJ <- str\_replace\_all(demo\_test$SUBJ, "\\.", "")  
demo\_test <- dplyr::rename(demo\_test, VISIT = Visit)  
demo\_test <- dplyr::select(demo\_test, SUBJ, VISIT, Ethnicity, Diagnosis, Gender, Age, ADOS, MullenRaw, ExpressiveLangRaw )  
  
demo\_test <- dplyr::rename(demo\_test, nonverbalIQ = MullenRaw)  
demo\_test <- dplyr::rename(demo\_test, verbalIQ = ExpressiveLangRaw)  
  
LU\_test <- read.csv("LU\_test.csv")  
LU\_test$SUBJ <- str\_replace\_all(LU\_test$SUBJ, "\\.", "")  
LU\_test$VISIT <- str\_extract(LU\_test$VISIT,"[^vVisit.]")  
LU\_test <- dplyr::select(LU\_test, SUBJ, VISIT, MOT\_MLU, MOT\_LUstd, CHI\_MLU, CHI\_LUstd)  
  
token\_test<- read.csv("token\_test.csv")  
token\_test$VISIT <- str\_extract(token\_test$VISIT,"[^vVisit.]")  
token\_test$SUBJ <- str\_replace\_all(token\_test$SUBJ, "\\.", "")  
LU\_test$VISIT <- as.numeric(LU\_test$VISIT)  
token\_test$VISIT <- as.numeric(token\_test$VISIT)  
  
m1 <- full\_join(LU\_test, token\_test, by = c("SUBJ", "VISIT"))  
  
demo\_test\_real <- left\_join(m1, demo\_test, by = c("SUBJ", "VISIT"))  
demo\_test\_real$X <- NULL

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

##root mean square error = rmse  
preds <- predict(model, type= "response")  
  
rmse(merged\_data$CHI\_MLU, preds)#0.2593025

## [1] 0.2593025

#this is a raw measure of the size of the error of your prediction - everytime you make a prediction you expect it to be off by 0.2593025 (the smaller the better)  
#test data we have random effect for kids individually and for train we don't  
#why is it bigger number for train data? we look for specifity for kids  
#simple model was worse it has higher error

test the performance of the models on the test data (Tips: google the functions “predict()”) - optional: predictions are never certain, can you identify the uncertainty of the predictions? (e.g. google predictinterval())

##root mean square error = rmse  
preds1 <- predict(model, newdata=demo\_test\_real,allow.new.levels=T)  
  
rmse(demo\_test\_real$CHI\_MLU, preds1) #0.7222786

## [1] 0.7222786

link to gitlab: <https://gitlab.com/majka1308/assignment1/blob/master/A1_P3_LangASD_prediction_katka.Rmd>

We tested our model on training data and testing data using rmse function. Root mean square error is a measure of differences between our model predictions and actual data we observed. When tested on training data our model performed well with value of rmse being 0.2570438 on the other side, when tested on new testing data our model performed worse with value of error being 0.7222786. This difference would mean that our model is much better at predicting the data it was trained on than other sample of data and therefore shows that we over fitted our model.

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

In this exercise you have to use cross-validation to calculate the predictive error of your models and use this predictive error to select the best possible model. - 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!) - (Tips): google the function “createFolds”; loop through each fold, train both models on the other folds and test them on the fold)

#merging data   
demo\_test\_real$X <- NULL  
demo\_test\_real$types\_shared <- NULL  
merged\_data$X <- NULL  
  
full\_data = rbind(demo\_test\_real, merged\_data)

## Warning in `[<-.factor`(`\*tmp\*`, ri, value = c(19.8, 23.93, 27.7, 32.9, :  
## invalid factor level, NA generated

write.csv(full\_data, file = "full\_data.csv")  
  
full\_data$SUBJ = as.factor(full\_data$SUBJ)  
full\_data$SUBJ = as.integer(full\_data$SUBJ)  
  
  
#Create fold with unique SUBJ  
library(caret)

## Loading required package: lattice

##   
## Attaching package: 'caret'

## The following objects are masked from 'package:ModelMetrics':  
##   
## confusionMatrix, precision, recall, sensitivity, specificity

## The following object is masked from 'package:purrr':  
##   
## lift

## The following objects are masked from 'package:Metrics':  
##   
## precision, recall

folds=createFolds(full\_data$SUBJ, list=TRUE, returnTrain = TRUE, k = 5)  
  
#Create matarix to store results  
cv\_empty=matrix(ncol=3, nrow=5)  
cv\_empty1= matrix(ncol=3,nrow = 5)  
  
  
#Loop  
#Indexing, reset when using the loop again  
n=1  
  
set.seed(101)  
  
for (k in folds){  
#------ Split into training and test data ------   
#Create training dataset, data not in fold k  
data\_train=subset(full\_data, !(SUBJ %in% k))  
#Create test dataset, data in fold k  
data\_test=subset(full\_data, SUBJ %in% k)  
   
#------ train model - apply model to data\_train ------  
model2 = lmer(CHI\_MLU ~ VISIT \* Diagnosis + (1+ VISIT|SUBJ), data\_train)  
#Make predictions based on model  
Prediction=predict(model2, type='response')  
#Prints the prediction against the test data  
results\_train=ModelMetrics::rmse(data\_train$CHI\_MLU, Prediction)   
  
#------ test the model - test model on data\_test (last quarter) ------  
#Make predictions based on modeVIS  
Prediction\_test=predict(model2, data\_test, type='response', allow.new.levels = TRUE)  
#  
results\_test=ModelMetrics::rmse(data\_test$CHI\_MLU, Prediction\_test)  
   
#------ save the performance ------   
#Fills out the kth row in cv\_empty with iteration and rmse's  
cv\_empty[n,]=cbind(n,results\_train,results\_test)  
  
######################################################  
   
#------ train second model - apply model to data\_train ------  
model3 = lmer(CHI\_MLU~VISIT\* Diagnosis+I(VISIT^2)\*Diagnosis +I(VISIT^3)\* Diagnosis +(1+VISIT+I(VISIT^2) +I(VISIT^3)|SUBJ),data\_train)  
#Make predictions based on model  
Prediction1=predict(model3, type='response')  
#Prints the prediction against the test data  
results\_train1=ModelMetrics::rmse(data\_train$CHI\_MLU, Prediction1)   
  
#------ test the model  
Prediction\_test1=predict(model3, data\_test, type='response', allow.new.levels = TRUE)  
#  
results\_test1=ModelMetrics::rmse(data\_test$CHI\_MLU, Prediction\_test1)  
   
#------ save the performance ------   
cv\_empty1[n,]=cbind(n,results\_train1,results\_test1)  
n=n+1  
}

## Warning in optwrap(optimizer, devfun, getStart(start, rho$lower, rho$pp), :  
## convergence code 1 from bobyqa: bobyqa -- maximum number of function  
## evaluations exceeded

## singular fit

## Warning in optwrap(optimizer, devfun, getStart(start, rho$lower, rho$pp), :  
## convergence code 1 from bobyqa: bobyqa -- maximum number of function  
## evaluations exceeded

## singular fit

## Warning: Model failed to converge with 1 negative eigenvalue: -4.4e-01

## Warning in optwrap(optimizer, devfun, getStart(start, rho$lower, rho$pp), :  
## convergence code 1 from bobyqa: bobyqa -- maximum number of function  
## evaluations exceeded

## singular fit

## Warning: Model failed to converge with 2 negative eigenvalues: -2.2e-01  
## -8.4e-01  
  
## Warning: convergence code 1 from bobyqa: bobyqa -- maximum number of  
## function evaluations exceeded

## singular fit

## Warning: Model failed to converge with 1 negative eigenvalue: -8.7e-03

## singular fit

## Warning in optwrap(optimizer, devfun, getStart(start, rho$lower, rho$pp), :  
## convergence code 1 from bobyqa: bobyqa -- maximum number of function  
## evaluations exceeded

## singular fit

#------ Make the matrix into a data frame ------   
cv\_results=data.frame(cv\_empty)  
cv\_results1=data.frame(cv\_empty1)  
#Rename variables   
cv\_results=setnames(cv\_results,"X1", "iteration")  
cv\_results=setnames(cv\_results,"X2", "train")  
cv\_results=setnames(cv\_results,"X3", "test")  
cv\_results1=setnames(cv\_results1,"X1", "iteration1")  
cv\_results1=setnames(cv\_results1,"X2", "train1")  
cv\_results1=setnames(cv\_results1,"X3", "test1")  
  
results\_models <- data.frame(mean(cv\_results$train), mean(cv\_results$test), mean(cv\_results1$train), mean(cv\_results1$test))  
  
results\_models=setnames(results\_models,"mean.cv\_results.train.", "train\_1st")  
results\_models=setnames(results\_models,"mean.cv\_results.test.", "test\_1st")  
results\_models=setnames(results\_models,"mean.cv\_results1.test.", "test\_2nd")  
results\_models=setnames(results\_models,"mean.cv\_results1.train.", "train\_2nd")  
  
#after calculating the mean rmse for cross validation, we can see that the second model performed a little bit better, even though the difference is minimal

Which model is better at predicting new data: the one you selected last week or the one chosen via cross-validation this week?

* Test both of them on the test data.
* Report the results and comment on them.
* Now try to find the best possible predictive model of ChildMLU, that is, the one that produces the best cross-validated results.
* Bonus Question 1: What is the effect of changing the number of folds? Can you plot RMSE as a function of number of folds?
* Bonus Question 2: compare the cross-validated predictive error against the actual predictive error on the test data

Through cross-validation we tested simpler model: model2= lmer(CHI\_MLU ~ VISIT \* Diagnosis + (1+ VISIT|SUBJ), train\_data). This model, however, performed about the same as our previous model with results on average being 0,3443552 for training data and 0,7649187 for testing data. This means that the model is just like previous one much worse in explaining other data samples. We chose to continue with this model because the first one seemed to have bigger issues with over fitting.

### Exercise 3) Assessing the single child

Let’s get to business. This new kiddo - Bernie - has entered your clinic. This child has to be assessed according to his group’s average and his expected development.

Bernie is one of the six kids in the test dataset, so make sure to extract that child alone for the following analysis.

You want to evaluate:

* 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. (Tip: recreate the equation of the model: Y=Intercept+BetaX1+BetaX2, etc; input the average of the TD group for each parameter in the model as X1, X2, etc.).
* 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)

bernie <- dplyr::filter(full\_data, CHI\_MLU == 1.9844560 & MOT\_MLU == 3.414397)  
#bernie is a subject n 14  
  
#Creating a dataset without bernie (Bernie is number 63)  
no\_bernie\_data = subset(full\_data, (SUBJ != 14))  
#ONLY bernie  
only\_bernie\_data = subset(full\_data, (SUBJ == 14))  
  
#----- Create model and make average TD child at each visit ------  
# creating final model  
model\_full = lmer(CHI\_MLU~VISIT\* Diagnosis+I(VISIT^2)\*Diagnosis +I(VISIT^3)\* Diagnosis +(1+VISIT+I(VISIT^2) +I(VISIT^3)|SUBJ),no\_bernie\_data)

## Warning in optwrap(optimizer, devfun, getStart(start, rho$lower, rho$pp), :  
## convergence code 1 from bobyqa: bobyqa -- maximum number of function  
## evaluations exceeded

## singular fit

#Get betas  
summary(model\_full)

## Linear mixed model fit by REML. t-tests use Satterthwaite's method [  
## lmerModLmerTest]  
## Formula:   
## CHI\_MLU ~ VISIT \* Diagnosis + I(VISIT^2) \* Diagnosis + I(VISIT^3) \*   
## Diagnosis + (1 + VISIT + I(VISIT^2) + I(VISIT^3) | SUBJ)  
## Data: no\_bernie\_data  
##   
## REML criterion at convergence: 544.6  
##   
## Scaled residuals:   
## Min 1Q Median 3Q Max   
## -2.40805 -0.51765 -0.08487 0.48294 2.98304   
##   
## Random effects:  
## Groups Name Variance Std.Dev. Corr   
## SUBJ (Intercept) 0.0422267 0.20549   
## VISIT 0.2138997 0.46249 0.11   
## I(VISIT^2) 0.0250513 0.15828 0.22 -0.88   
## I(VISIT^3) 0.0002195 0.01482 -0.34 0.71 -0.95  
## Residual 0.1003524 0.31678   
## Number of obs: 381, groups: SUBJ, 66  
##   
## Fixed effects:  
## Estimate Std. Error df t value Pr(>|t|)   
## (Intercept) 0.719226 0.213194 39.261068 3.374 0.00168 \*\*  
## VISIT 0.705025 0.256400 37.620431 2.750 0.00912 \*\*  
## DiagnosisB 0.433837 0.290956 38.925838 1.491 0.14400   
## I(VISIT^2) -0.167896 0.083250 18.270035 -2.017 0.05866 .   
## I(VISIT^3) 0.013542 0.007885 21.192486 1.717 0.10047   
## VISIT:DiagnosisB -0.715139 0.348755 37.096031 -2.051 0.04743 \*   
## DiagnosisB:I(VISIT^2) 0.369063 0.113077 17.887281 3.264 0.00434 \*\*  
## DiagnosisB:I(VISIT^3) -0.038680 0.010715 20.978235 -3.610 0.00165 \*\*  
## ---  
## Signif. codes: 0 '\*\*\*' 0.001 '\*\*' 0.01 '\*' 0.05 '.' 0.1 ' ' 1  
##   
## Correlation of Fixed Effects:  
## (Intr) VISIT DgnssB I(VISIT^2 I(VISIT^3 VISIT: DB:I(VISIT^2  
## VISIT -0.893   
## DiagnosisB -0.733 0.654   
## I(VISIT^2) 0.860 -0.975 -0.630   
## I(VISIT^3) -0.827 0.932 0.606 -0.988   
## VISIT:DgnsB 0.656 -0.735 -0.891 0.717 -0.685   
## DB:I(VISIT^2 -0.633 0.718 0.859 -0.736 0.728 -0.974   
## DB:I(VISIT^3 0.609 -0.686 -0.825 0.727 -0.736 0.931 -0.988   
## convergence code: 1  
## singular fit

# getting means of variables for the TD kids, used for input as "x" in model  
visit\_mean <- full\_data %>% filter(Diagnosis == "B") %>% dplyr::group\_by(VISIT) %>% dplyr::summarise(mean(CHI\_MLU))  
  
MLU\_visit1 = 0.72 + 0.71\*1.31 - 0.17\*1.31 + 0.01\*1.31 -0.72\*1.31 +0.37\*1.31 -0.04\*1.31  
  
MLU\_visit2 = 0.72 + 0.71\*1.76 - 0.17\*1.76 + 0.01\*1.76- 0.72\*1.76 +0.37\*1.76 -0.04 \*1.76  
  
MLU\_visit3 = 0.72 + 0.71\*2.23 - 0.17\*2.23 + 0.01\*2.23 -0.72\*2.23 +0.37\*2.23 -0.04\*2.23  
  
MLU\_visit4 = 0.72 + 0.71\*2.73 - 0.17\*2.73 + 0.01\*2.73 -0.72\*2.73 +0.37\*2.73 -0.04\*2.73   
  
MLU\_visit5 = 0.72 + 0.71\*3 - 0.17\*3 + 0.01\*3 -0.72\*3 +0.37\*3 - 0.04\*3   
  
MLU\_visit6 = 0.72 + 0.71\*2.91 - 0.17\*2.91 + 0.01\*2.91 - 0.72\*2.91 +0.37\*2.91 -0.04\*2.91  
  
# making a list of MLU of TD kids at the different visits as calculated above.  
MLU\_list = c(MLU\_visit1, MLU\_visit2, MLU\_visit3, MLU\_visit4, MLU\_visit5, MLU\_visit6)  
  
#getting the difference in MLU between Bernie and the TD per visit  
only\_bernie\_data[,5] - MLU\_list #Bernie has higher MLU than average TD child

## [1] 1.054856 1.542844 2.276391 2.026299 1.973252 2.262813

#getting the difference between Bernie and his predicted score  
Prediction3=predict(model\_full, only\_bernie\_data[6,], allow.new.levels = TRUE)  
  
only\_bernie\_data[6,5]-Prediction3 #he performed better than expected

## 11   
## 1.618309

* Bernie is very skilled therefore when compared to average child without autism he performs better with all values being positive meaning his score is bigger than the average we subtracted. Good work Bernie! (values after substraction: 1.054856 1.542844 2.276391 2.026299 1.973252 2.262813)
* At Visit 6 Bernie performs better than is the prediction of our model. The difference between Bernie’s actual MLU and predicted value was 1.62.

### OPTIONAL: Exercise 4) Model Selection via Information Criteria

Another way to reduce the bad surprises when testing a model on new data is to pay close attention to the relative information criteria between the models you are comparing. Let’s learn how to do that!

Re-create a selection of possible models explaining ChildMLU (the ones you tested for exercise 2, but now trained on the full dataset and not cross-validated).

Then try to find the best possible predictive model of ChildMLU, that is, the one that produces the lowest information criterion.

* Bonus question for the optional exercise: are information criteria correlated with cross-validated RMSE? That is, if you take AIC for Model 1, Model 2 and Model 3, do they co-vary with their cross-validated RMSE?

### OPTIONAL: Exercise 5): Using Lasso for model selection

Welcome to the last secret exercise. If you have already solved the previous exercises, and still there’s not enough for you, you can expand your expertise by learning about penalizations. Check out this tutorial: <http://machinelearningmastery.com/penalized-regression-in-r/> and make sure to google what penalization is, with a focus on L1 and L2-norms. Then try them on your data!