Pseudo Code

function boostedrirt(data):

checking\_inputs(data)

rmse\_results # Initialisiere Ergebnisobjekt

param\_grid = expand.grid(subsample, colsample) # Tuningparametergrid

for (param in paramgrid){ # Iterieren über Tuningparameter

cv\_data <- cv\_ziehen(datensatz) #Crossvalidation Folds ziehen

for ( i in 1:folds){ # Iterien über Crossvalidation Folds

test\_data <- cv\_data[i]

train\_data <-cv\_data[-i]

f\_train # Schritt 1: Initalisieren von f\_0

f\_test

improv\_count = 0

iter = 1

while iter <= itermax and improv\_count < 5{ # Schritt 2: adaptierter EM-Algorithmus

train\_data$neg\_grad<- berechne\_neg\_grad(train\_data) #b)

para\_data <- ziehe\_stichprobe(train\_data,param) #a)

cart\_modell <- schaetze\_cart(para\_data) #c)

para\_data$temp\_res <- berechne\_temp\_res(para\_data, cart\_modell, f\_train) # d)

lmm\_modell <- schaetze\_lmm(para\_data) #e)

train\_data$f\_iter\_act <- berechne\_aktuelle\_vorhersage(train\_data, lmm\_modell, f\_train, cart\_modell) # f)

test\_data$f\_iter\_act<- berechne\_aktuelle\_vorhersage(test\_data, lmm\_modell, f\_test, cart\_modell)

f\_train <- train\_data$f\_iter\_act # Aktualisieren von f\_train

rmse\_train <- berechne\_rmse(train\_data)

rmse\_test <- berechne\_rmse(test\_data) #g)

if (iter > 1 && rmse\_test >= min(rmse\_results[rmse\_results$fold == i,]$rmse\_test)) {

improv\_count <- improv\_count + 1}

if(improv\_count == 5){iter\_final <- iter -5}

if(iter == itermax){iter\_final <- iter} #h)

rmse\_results # speichern in Ergebnisobjekt

iter #hochsetzen

} # Ende der while-Schleife

} # Ende der CV- Schleife

} # Ende der Parameterschleife

final\_model <- berechne\_finales\_model(data, rmse\_results) # Finales Model berechnen

}