hm8

Robert Benke

23 maja 2019

Read data - quake.csv

https://www.openml.org/d/209

```
quake_dfr <- fread("quake.csv")

test_quake_dfr <- quake_dfr[ind <<- sample(1:nrow(quake_dfr), 400, replace = FALSE),]
train_quake_dfr <- quake_dfr[-ind,]
```

model 1 - neural network 3-8-12-1

```
model <- keras_model_sequential()
model %>%
  layer_dense(units = 6, activation = "relu", input_shape = c(3)) %>%
  layer_dense(units = 16, activation = "relu") %>%
  layer_dense(units = 1, activation = "relu")

model %>% compile(
  loss = "mean_absolute_percentage_error",
  optimizer = optimizer_rmsprop(),
  metrics = "mae"
)

history <- model %>% fit(
  as.matrix(train_quake_dfr[,-4]), train_quake_dfr$col_4,
  epochs = 60, batch_size = 200,
  validation_split = 0.2
)

prediction_NN <- predict(model,as.matrix(test_quake_dfr[,-4]))
result_NN <- sum((prediction_NN - test_quake_dfr$col_4)^2)/nrow(test_quake_dfr)
result_NN
```

```
## [1] 2.583671
```

model 2 - XGBoost

```
model_xgb <- xgboost(as.matrix(train_quake_dfr[,-4]), label = train_quake_dfr$col_4, nrounds = 200, verb
prediction_xgb <- predict(model_xgb, as.matrix(test_quake_dfr[,-4]))
result_xgb <- sum((prediction_xgb - test_quake_dfr$col_4)^2)/nrow(test_quake_dfr)
result_xgb
```

```
## [1] 0.04921828
```

model 3 - linear model

```
model_lm <- lm(col_4~. , data = train_quake_dfr)
prediction_lm <- predict(model_lm, test_quake_dfr)
result_lm <- sum((prediction_lm - test_quake_dfr$col_4)^2)/nrow(test_quake_dfr)
result_lm
```

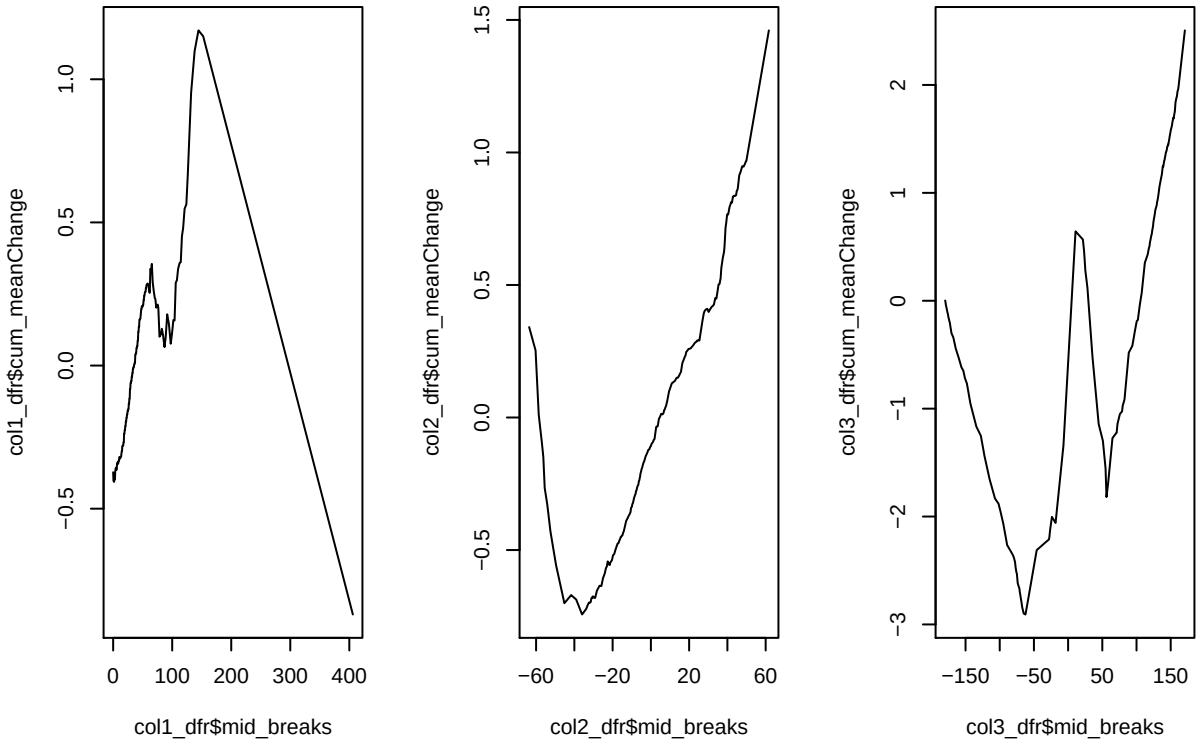
```
## [1] 0.03847336
```

Już tutaj widoczna jest przewaga prostego modelu liniowego nad modelami bardziej złożonymi. Modele liniowe są odporniejsze na przetrenowanie niż sieci neuronowe i xgboost. Nie mają tak dużych możliwości dopasowania się do skomplikowanych zależności i nie modelują wpływu wspólnego kilku zmiennych, za to znacznie mniej różnią się pomiędzy próbami z populacji (mają małą wariancję, są stabilne).

New features

```
col1_dfr <- ALE(model, train_quake_dfr, "col_1", 200)
col2_dfr <- ALE(model, train_quake_dfr, "col_2", 200)
col3_dfr <- ALE(model, train_quake_dfr, "col_3", 200)

par(mfrow=c(1,3))
plot(col1_dfr$mid_breaks, col1_dfr$cum_meanChange, type = 'l')
plot(col2_dfr$mid_breaks, col2_dfr$cum_meanChange, type = 'l')
plot(col3_dfr$mid_breaks, col3_dfr$cum_meanChange, type = 'l')
```




```
par(mfrow=c(1,1))


library(changepoint)
```

```
## Loading required package: zoo
```

```
##
## Attaching package: 'zoo'
```

```
## The following objects are masked from 'package:base':
##
##     as.Date, as.Date.numeric
```

```
## Successfully loaded changepoint package version 2.2.2
##  NOTE: Predefined penalty values changed in version 2.2.  Previous penalty values with a postfix 1 i
```

```
col1_cpt <- cpt.meanvar(col1_dfr$cum_meanChange, method = "PELT", pen.value = 0.05)
col2_cpt <- cpt.meanvar(col2_dfr$cum_meanChange, method = "PELT", pen.value = 0.05)
col3_cpt <- cpt.meanvar(col3_dfr$cum_meanChange, method = "PELT", pen.value = 0.05)

breaks1 <- c(-Inf, col1_dfr$mid_breaks[col1_cpt@cpts], Inf)
breaks2 <- c(-Inf, col2_dfr$mid_breaks[col2_cpt@cpts], Inf)
breaks3 <- c(-Inf, col3_dfr$mid_breaks[col3_cpt@cpts], Inf)

categorical_train_quake_dfr <- train_quake_dfr %>% mutate(col1 = cut(col_1, breaks1),
                                                          col2 = cut(col_2, breaks2),
                                                          col3 = cut(col_3, breaks3)) %>%
  select(col1,col2,col3,col_4)
```

Model 4 - linear model with categorized variables

```
model_lm_cat <- lm(col_4~., data = categorical_train_quake_dfr)


categorical_test_quake_dfr <-test_quake_dfr %>% mutate(col1 = cut(col_1, breaks1),
                                                       col2 = cut(col_2, breaks2),
                                                       col3 = cut(col_3, breaks3)) %>%
  select(col1,col2,col3,col_4)
prediction_lm_cat <- predict(model_lm_cat, categorical_test_quake_dfr)

result_cat_lm <- sum((prediction_lm_cat - test_quake_dfr$col_4)^2)/nrow(test_quake_dfr)
result_cat_lm
```

```
## [1] 0.04022873
```

Średni błąd kwadratowy dla ostatnego modelu jest nieco wyższy niż dla zwykłej regresji liniowych co może być pewnym zaskoczeniem ponieważ wcześniej zauważyliśmy że błąd ten da się skutecznie ograniczać zmniejszając złożoność modelu (zmniejszając wariancję jednocześnie godząc się na możliwy nieco większy bias). Jednak gdy spojrzymy na liczbę stopnii swobody ostatniego modelu to jest ona wyższa niż zwykłego modelu liniowego. Zatem rzeczywiście, istnieje tutaj liniowa relacja pomiędzy złożonością modelu a estymacją błędu generalizacji. Ta relacja nie jest oczywista, szczególnie gdy patrzymy na ALE ploty zmiennych objaśniających. Idealny ALE plot dla regresji liniowej powinien przyjmować wartość stałą równą zero (nie jest to jednak warunek wystarczający, ponieważ nie bierze pod uwagę interakcji).