Raport do zadania 6. Karol Księżopolski

2025-06-15

Wybór modelu

Rozważałem trzy modele trenowania danych: ElasticNet, Random Forest i Tree Booster z pakietu caret. Ostatecznie wybrałem xgbTree, bo osiągał zdecydowanie lepsze wyniki od pozostałych modeli.

Transformacje zmiennych i redukcja wymiaru

Najpierw usuwam z danych cechy o wariancji zbliżonej do zera, ponieważ nie wnoszą prawie żadnej informacji, a potencjalnie zmniejsza to czas trenowania.

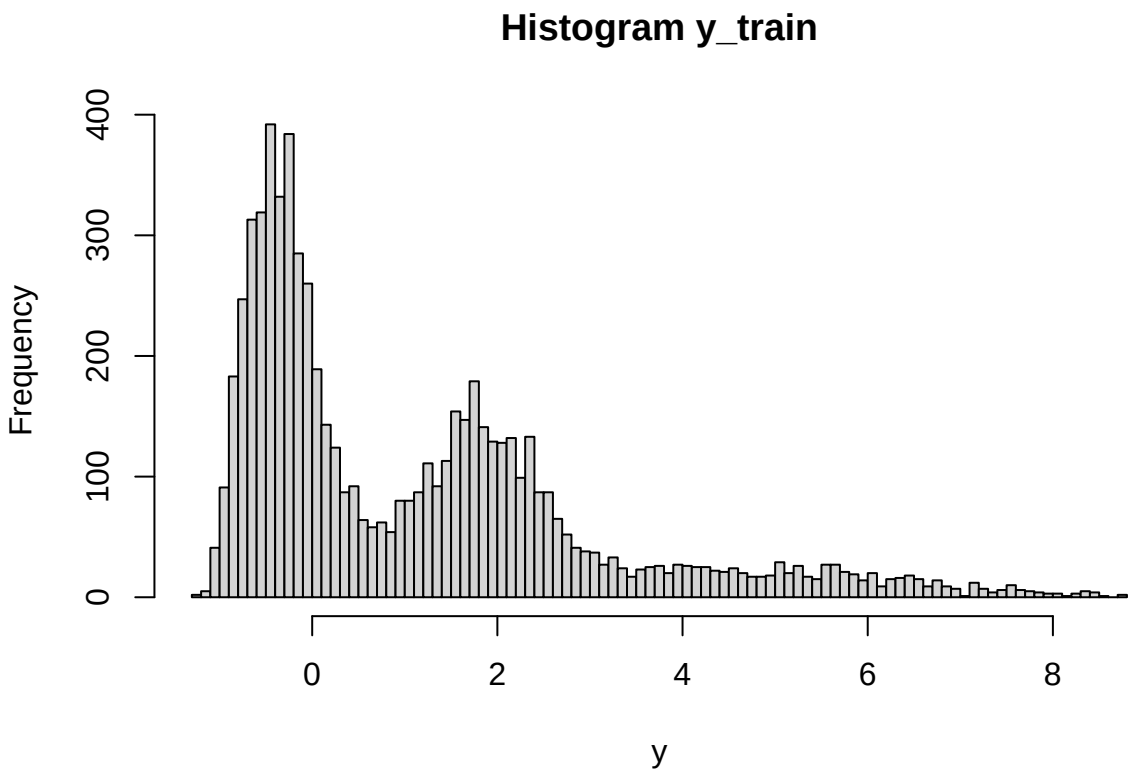
```
nzv = nearZeroVar(X_train)

if(length(nzv) > 0) {
  X_train_filt = X_train[, -nzv]
  X_test_filt = X_test[, -nzv]
} else {
  X_train_filt = X_train
  X_test_filt = X_test
}
```

Następnie usuwam cechy, które mają w zbiorze danych mocno z nimi skorelowane cechy, ponownie, żeby zmniejszyć czas pracy algorytmu kosztem zbędnych informacji.

```
cor_mat = cor(X_train_filt)
high_cor = findCorrelation(cor_mat, cutoff = 0.95)
if(length(high_cor) > 0) {
  X_train_filt = X_train_filt[, -high_cor]
  X_test_filt = X_test_filt[, -high_cor]
}
```

Patrząc na histogram zmiennej y_train widać, że jej rozkład rozciąga się w kierunku większych wartości, więc żeby dane miały bardziej symetryczny rozkład i lepiej spełniały założenia modeli regresyjnych, stosuję transformację logarytmiczną. Przed użyciem logarytmu przesuwam dane, żeby zapewnić brak problemów z logarytmem przez ujemne wartości. Wyniki po zastosowaniu transformacji logarytmicznej znacząco się poprawiły.

```
y_train = read.csv("y_train.csv", header = TRUE)
y_train_vec = as.numeric(y_train[, 1])
hist(y_train_vec, breaks = 100, main = "Histogram y_train", xlab = "y")
```



Liczba foldów walidacji krzyżowej i hiperparametry

Używam 5 foldów w walidacji krzyżowej. W tak dużym zbiorze danych 5 foldów wydaje mi się lepszym pogodzeniem dokładności z czasem niż 10.

Model Tree Booster trenowałem na początku na domyślnych parametrach ze strony https://xgboost.readthedocs.io/en/stable/parameter.html#parameters-for-tree-booster i do niego zbliżonych z wyjątkiem max_depth, w którym zacząłem od mniejszych wartości dla krótszego czasu. Starałem się żeby przy każdej próbie siatka hiperparametrów miała maksymalnie 2 * 2 * 3 = 12 kombinacji parametrów, żeby czas trenowania był znośny. W następnych próbach zazwyczaj siatka różniła się tym, że nie używałem hiperparametrów, które nie zostały użyte w najoptymalniejszych modelach z poprzednich prób.

Wewnątrz funkcji train dodałem jeszcze centrowanie i skalowanie danych oraz analizę składowych głównych, żeby jeszcze bardziej poprawić czas działania algorytmu i zmniejszyć ryzyko przeuczenia.