Regresja Liniowa

Mikołaj Malec

April 20, 2020

Czym jest algorytm Regresji Liniowej?

Jest to algorytm, który zakłada liniową zależność między zmiennymi. Dzięki danym ze zbioru testowego uzyskamy liniowy model gotowy do przewidywania kolejnych próbek.

Wczytanie danych i bibliotek

```
x_test <- read.csv( "x_test.csv")[-1]
x_train <- read.csv( "x_train.csv")[-1]
y_test <- read.csv( "y_test.csv")[-1]
y_train <- read.csv( "y_train.csv")[-1]
```

O naszych danych

Dane te były wcześniej przez nasz zespół odpowiednio przygotowane, dzięki czemu możemy od razu przejść do trenowania naszego modelu.

Metryka dokładności

Do pomocy w określeniu jakości naszego modelu napiszemy funkcje, która wyliczy proc. poprawnych klasyfikacji.

```
accuracy <- function( table_in){
sum( diag( table_in)) / sum( table_in)
}
```

Tworzenie Modelu

Do stworzenia modelu użyjemy funkcji lm() z bazowego pakietu stats. Sama funkcja nie przyjmuje danych i celu treningowego tylko wszystkie dane w jednym pliku. To na wszystkich kolumnach jest tworzony nasz model.

```
train <- cbind( x_train, y_train) # Łączenie naszych ramek danych
train_lm <- lm( x~., train)
pred_train <- predict( train_lm, x_test)
```

Punkt odcięcia

Nasz model nie zwraca jednak wartości binarnych. Zamiast tego podaje on prawdopodobieństwa, jakie występują dla każdego rekordu. Wykorzystując naszą wcześniej zdefiniowaną metrykę, znajdziemy, jaki jest najlepszy punkt odcięcia tak, aby nasz model działał jak najlepiej. Naturalnym punktem wydaje się podział po 50%, jednakże z powodu danych wejściowych możliwa jest przychylność modelu w jedną ze stron. Sprawdźmy to teraz.

```
acc <- rep(0, 21)
ftest <- rep(0, 21)
rec <- rep(0, 21)
pre <- rep(0, 21)
maxi=0
for( i in 0:20){
predicted_lab <- ifelse( pred_train > i*0.05, 1, 0)
acc[i] <- accuracy( table( y_test[,1], predicted_lab))
if(maxi<accuracy( table( y_test[,1], predicted_lab))){
maxi = accuracy( table( y_test[,1], predicted_lab))
}
}
best_split <- which.max( acc) *0.05
```

Nasz model uzyskuje najlepszą celność przy podziale na ‘r toString((1-best_split)*100)’% dla 1 (Dobry klient) oraz ’r toString((1-best_split)*100)’% dla 0 (Zły klient)

Podsumowanie

Nasz model regresji liniowej osiąga dla podziału ‘r toString((1-best_split)*100)’%/’r toString((best_split)*100)’% osiągając przy tym ’r toString(maxi*100)’% celności.