

Podstawy maszynowego uczenia na przykładzie klasyfikacji tekstu

Sztuczna Inteligencja i Inżyniera Wiedzy. Ćwiczenie 4

Michał Hetmańczyk

27 05 2020

Sprawozdanie powstało jak dokument typu R Markdown.

Etap 1

Data processing

W pierwszej kolejności rozpakowano piliki zip.

Następnie umieszczono zawartość plików w strukturze danych typu Data Frame. Z nazwy pliku wyekstrahowano kategorię dokumentu. Referencja na tak utworzoną strukturę. została nazwana 'texts_df'.

Opis struktury:

```
str(texts_df, vec.len = 0)
```

```
## 'data.frame': 9837 obs. of 2 variables:
## $ category: Factor w/ 34 levels "Albania","Amerykanscy-prozaicy",...: NULL ...
## $ text : chr ...
```

```
levels(texts_df$category)
```

```
## [1] "Albania"
## [2] "Amerykanscy-prozaicy"
## [3] "Arabowie"
## [4] "Astronautyka"
## [5] "Choroby"
## [6] "Egipt"
## [7] "Ekologia-roslin"
## [8] "Filmy-animowane"
## [9] "Galezie-prawa"
## [10] "Gry-komputerowe"
## [11] "Karkonosze"
## [12] "Katolicyzm"
## [13] "Komiksy"
## [14] "Komputery"
## [15] "Kotowate"
## [16] "Kultura-Chin"
## [17] "Monety"
## [18] "Muzyka-powazna"
## [19] "Narcciarstwo"
```

```
## [20] "Narkomania"
## [21] "Niemieccy-wojskowi"
## [22] "Optyka"
## [23] "Pierwiastki-chemiczne"
## [24] "Pilka-nozna"
## [25] "Propaganda-polityczna"
## [26] "Rachunkowosc"
## [27] "Samochody"
## [28] "Samoloty"
## [29] "Sporty-silowe"
## [30] "System-opieki-zdrowotnej-w-Polsce"
## [31] "Szachy"
## [32] "Wojska-pancerne"
## [33] "Zegluga"
## [34] "Zydzi"
```

Aby ujednolicić tekst oraz wstępnie usunąć potencjalnie zbędne cechy, dane zostały wyczyszczone poprzez usunięcie: wielkich liter, liczb, znaków interpunkcyjnych, zbędnych białych znaków oraz wyrazów ze stop-listy (plik stopwords.txt z polską stop-listą został zaczerpnięty z <https://github.com/bieli/stopwords/blob/master/polish.stopwords.txt>). Aby wykorzystać w tym celu funkcję `tm_map`, zawartość dokumentów umieszczono w przeznaczony do tego strukturze - `VCorpus` (taki typ danych jest obsługiwany przez `tm_map`).

```
text_labels <- texts_df$category
text_corpus <- VCorpus(VectorSource(texts_df$text))
text_corpus_clean <- text_corpus %>%
  tm_map(content_transformer(tolower)) %>%
  tm_map(removeNumbers) %>%
  tm_map(removePunctuation) %>%
  tm_map(stripWhitespace)
stopwords.pl <- readLines(".\\stopwords.txt", encoding = 'UTF-8')
text_corpus_clean <- tm_map(text_corpus_clean, removeWords, stopwords.pl)
```

Wykorzystując tak wyczyszczone korpusy dokumentów utworzono macierz pojęć, zawierającą częstotliwości występowania danego pojęcia w danym dokumencie.

```
text_dtm <- DocumentTermMatrix(text_corpus_clean)
text_dtm
```

```
## <<DocumentTermMatrix (documents: 9837, terms: 264253)>>
## Non-/sparse entries: 1524005/2597932756
## Sparsity           : 100%
## Maximal term length: 173
## Weighting           : term frequency (tf)
```

Przykładowe wizualizacje przygotowanych danych. Kategorie: "Albania", "Choroby", "Astronautyka"



Propozycja metody selekcji cech i implementacji modeli

Ze względu na fakt, iż mamy do czynienia z szukaniem zależności między dwoma kategorycznymi zmiennymi, selekcja cech zostanie przeprowadzona z wykorzystaniem testów χ^2 . Implementacja klasyfikatora Naiwnego Bayesa zostanie zaczerpnięta z biblioteki “fastNaiveBayes”. Biblioteka tak obsługuje wszystkie rodzaje klasyfikatora (Bernoulli, Gaussian, Multinomial), co może być bardzo przydatne z perspektywy przeprowadzania badań. Wykorzystana zostanie również implementacja drzewa decyzyjnego z pakietu “tree”.

Naive-Bayes

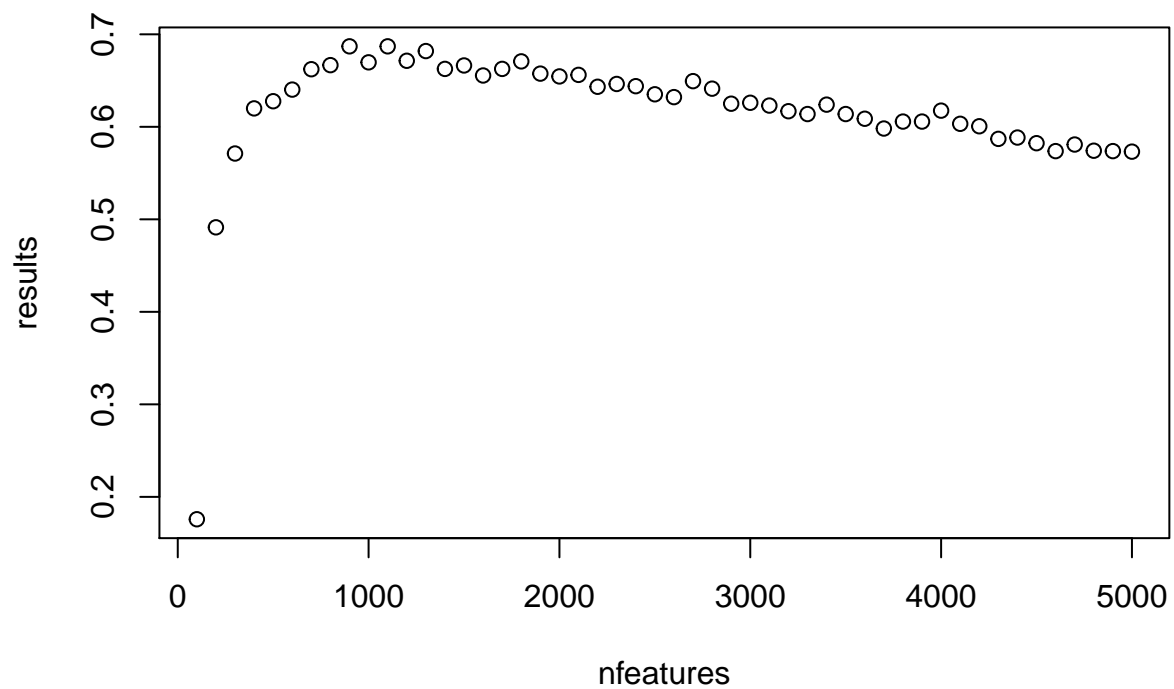
Model został zaimplementowany w następujący sposób: Wykonany trening z wykorzystaniem 10-krotnej walidacji krzyżowej, wywołując funkcję z wybranej wcześniej biblioteki fastNaiveBayes

Badanie wpływu liczby wybranych cech na wynik działania modelu

```
nfeatures = seq(100, 5000, by = 100)
results = vector()

for(i in nfeatures){
  results = c(results, performNaiveBayes(i, 0)[2])
}

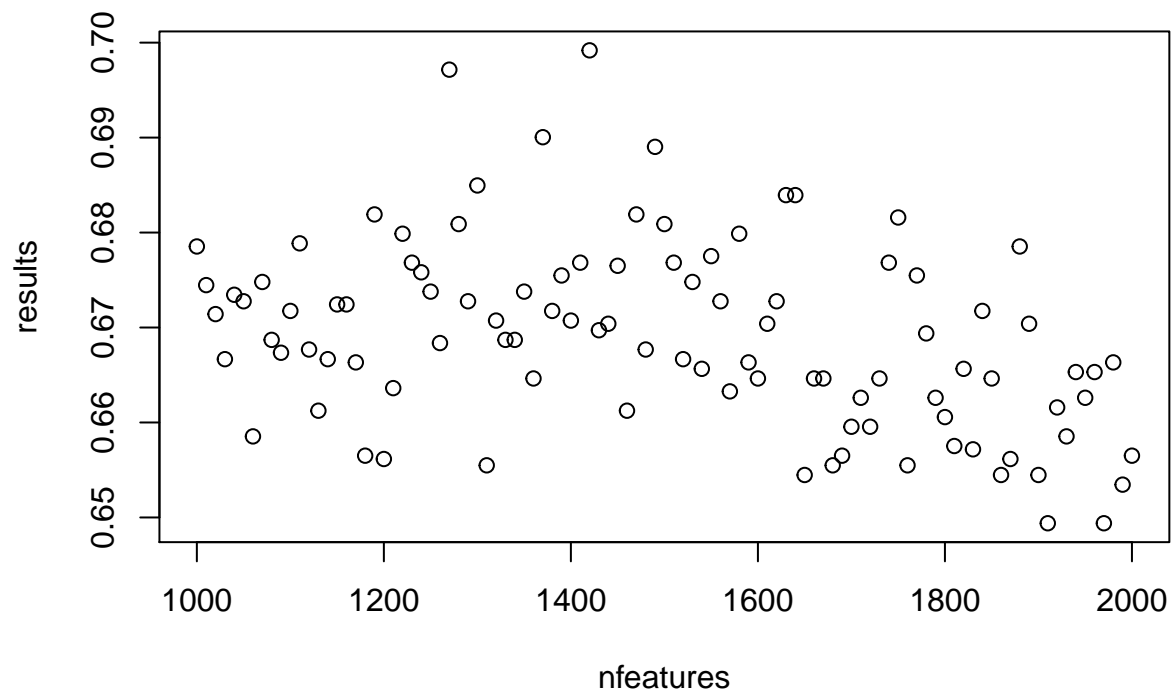
plot(nfeatures, results)
```



```
nfeatures = seq(1000, 2000, by = 10)
results = vector()

for(i in nfeatures){
  results = c(results, performNaiveBayes(i, 0)[2])
}

plot(nfeatures, results)
```

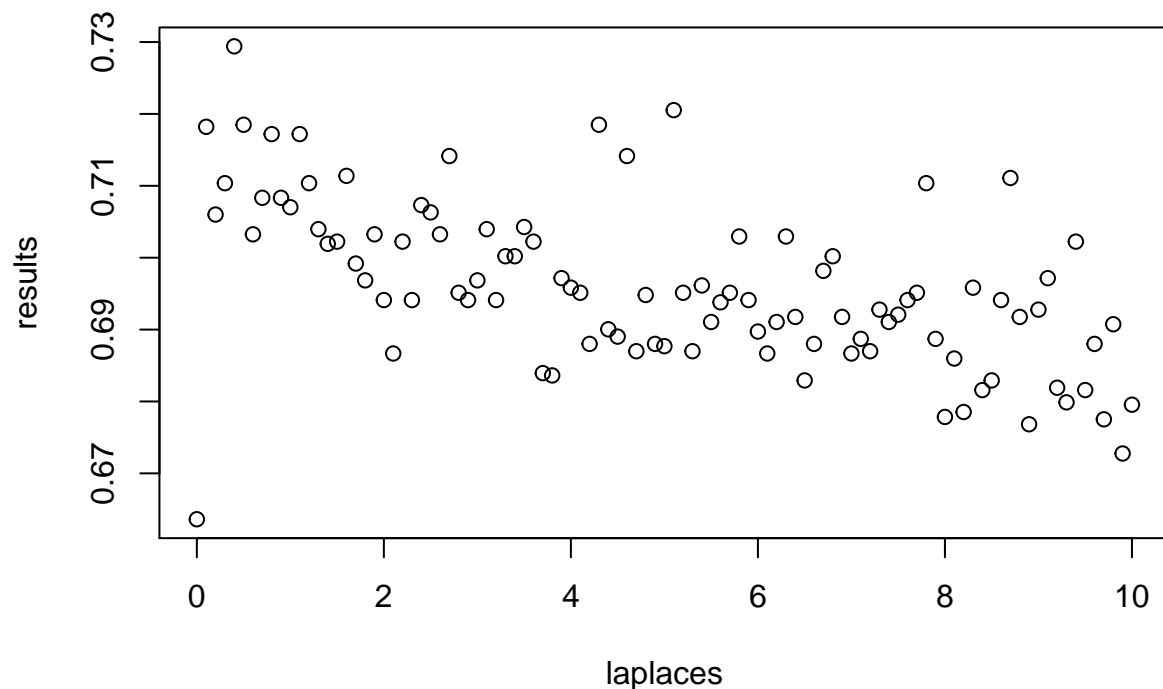


Badanie wpływu współczynnik wygładzania laplace'a na wynik działa modelu

```
laplaces = seq(0,10, by = 0.1)
results = vector()

for(i in laplaces){
  results = c(results, performNaiveBayes(1350, i)[2])
}

plot(laplaces, results)
```



Wyniki badań

Najlepsze wyniki osiągnięto dla 1350 cech oraz współczynnika 0.5

```
results = vector()
for(i in 1:10){
  results = c(results, performNaiveBayes(1350, 0.5)[2])
}

mean(unlist(results, use.names=FALSE))
```

```
## [1] 0.7043506
```

Decision tree

Wybrana biblioteka ma już wbudowaną 10-krotną walidację krzyżową. Uwaga! Niestety ze względu na nieznanne wcześniej ograniczenia wybranej biblioteki - można wykorzystać maksymalnie 32 kategorie. Odrzucono dwie kategorie oraz opisywane przez nie dokumenty.

```
chi2vals <- dfm_group(fs_dfm) %>%
  textstat_keyness(measure = "chi2")
chi2vals$chi2 <- abs(chi2vals$chi2)
chi2vals <- chi2vals[order(-chi2vals$chi2), ]
```

```

dfmTop <- dfm_select(fs_dfm, chi2vals$feature[1:1000])

texts_freq_df <- as.data.frame(dfmTop)
texts_freq_df$category <- as.factor(text_labels)
colnames(texts_freq_df) <- make.names(colnames(texts_freq_df))
texts_freq_df <- texts_freq_df[,!duplicated(colnames(texts_freq_df))]
texts_freq_df <- texts_freq_df[,!(names(texts_freq_df) == '-')]
texts_freq_df <- texts_freq_df[!(texts_freq_df$category %in% c('Albania', 'Zydzi')),]
texts_freq_df$category <- drop.levels(texts_freq_df$category)
colnames(texts_freq_df) <- make.names(colnames(texts_freq_df))

model <- tree(category ~ ., data = texts_freq_df)
summary(model)

##
## Classification tree:
## tree(formula = category ~ ., data = texts_freq_df)
## Variables actually used in tree construction:
## [1] "ur"          "samolot"     "karkonosze" "moneta"      "chemiczny"
## [6] "gry"         "czołgów"     "samochód"    "wersja"      "wyniki"
## [11] "serii"       "kot"         "szachowa"    "zm"
## Number of terminal nodes: 15
## Residual mean deviance: 4.719 = 43520 / 9223
## Misclassification error rate: 0.6941 = 6412 / 9238

pred <- predict(model, texts_freq_df[,!names(texts_freq_df) == 'category'], type = 'class')
sum(pred == texts_freq_df$category)/length(pred)

## [1] 0.3059104

```

Drzewo decyzyjne - Python

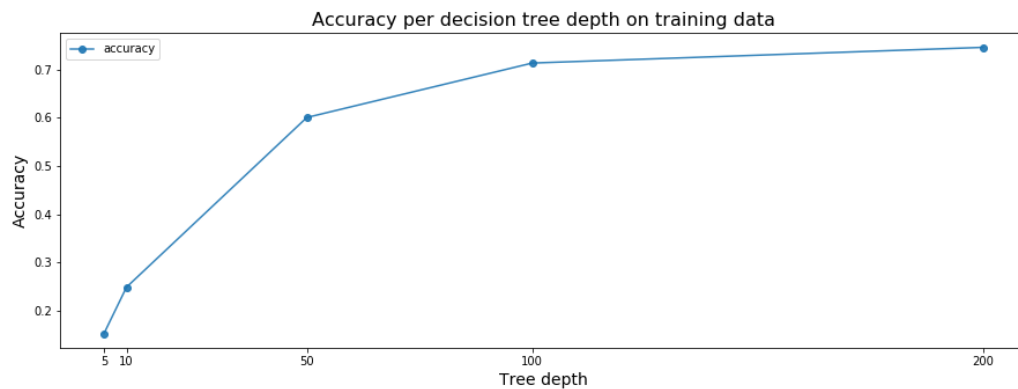
Niestety wybranie R jako środowiska pracy okazało się błędem. Implementacja modelu Naiwnego Baysa się powiodła, niestety biblioteki udostępniające Drzewo Decyzyjne okazały się mieć zbyt duże ograniczenia (ograniczona liczba kategorii, czas obliczeń, wbudowana selekcja maksymalnie kilkunastu cech). Dlatego ta część zadania została zrealizowana w języku Python. Ramkę danych przeniesiono między projektami jako plik csv, a następnie dane przekształcono w sposób analogiczny do poprzedniej części.

Próba wywołania algorytmu

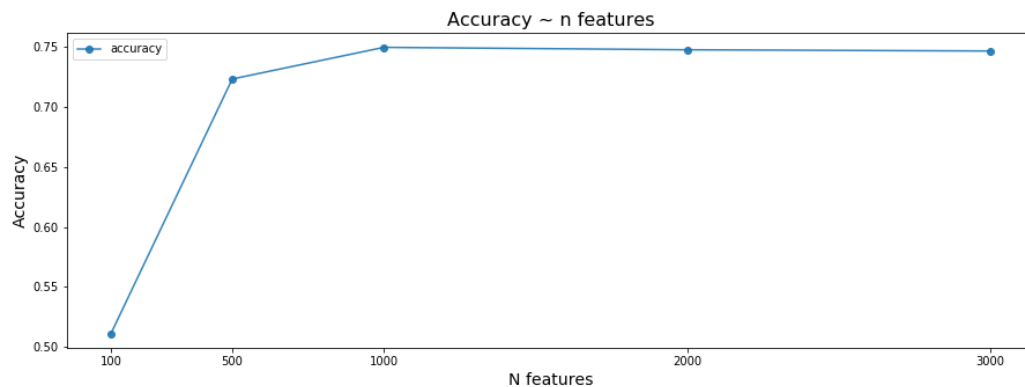
Dla próbnego wywołania predykcji z 5-krotną walidacją krzyżową, otrzymano następujące wyniki: [0.73780488 0.73831301 0.7458058 0.74682257 0.75241484]

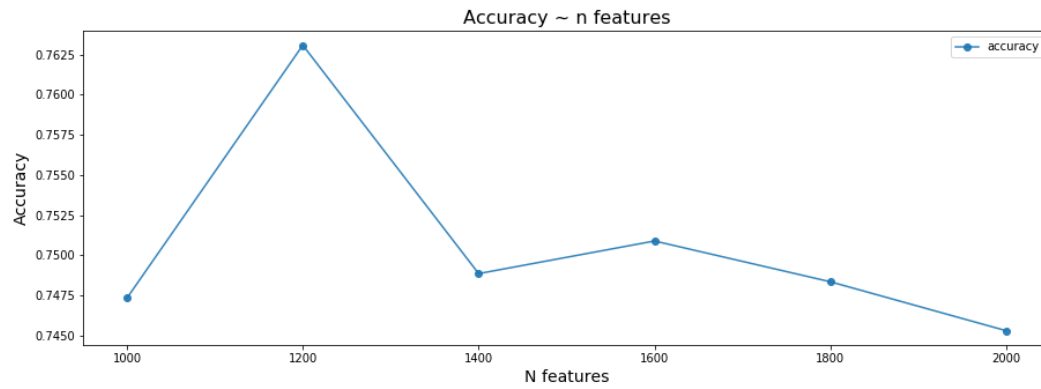
Badania

Wpływ głębokości dla stałej liczby cech, równej 1000



Wpływ liczby cech bez podania ograniczenia głębokości





Wyniki badań

Najlepsze wyniki osiągnięto dla drzewa bez podanego ograniczenia głębokości oraz liczby cech na poziomie 1000 - 1500 (wybrano 1500). Wyniki: [0.73932927 0.73577236 0.74631418 0.74885613 0.73970513]