lista 3

Marta Hałas

2025-05-20

Spis treści

1 Zadanie 2

1.1 a) Algorytmy klasyfikacji

1.1.1 O czym będzie analiza

Analiza dotyczy zastosowania algorytmów klasyfikacji:

metoda k-najbliższych sąsiadów (k-Nearest Neighbors),

drzewa klasyfikacyjne (classification trees),

naiwny klasyfikator bayesowski (naïve Bayes classifier)

i porównania ich dokładności w zależności od podzbiorów, parmaetrów czy schematów oceny dokładności.

1.1.2 Opis analizy

Dla spójności analizy i poprawności wniosków, ustawiłam ziarno, aby przy generowaniu na nowo pliku rmd obliczane wyniki nie zmieniły swojej wartości.

Analiza zawiera:

- porównanie rozkładu klas, dyspropocji poszczególnych grup, zmienności, zdolności dyskryminacyjnych.
- porównanie dokładności klasyfikacji algorytmów przy użyciu różnych schematów oceny dokładności.
- wpływ różnych parametrów algorytmów oraz podzbiorów cech na dokładność klasyfikacji.

1.2 b) Wybór, przygotowanie i zapoznanie się z danymi

1.2.1 Wybór danych

Dane: PimaIndiansDiabetes2

Do porównania metod klasyfikacji wybrałam zbiór danych PimaIndiansDiabetes2.

1.2.2 Zapoznanie się z danymi

Rozmiar danych: liczba wierszy to 768 , liczba kolumn to 9.

Typy poszczególnych cech:

```
##          names.d. sapply.d..class.
## pregnant pregnant          numeric
## glucose   glucose          numeric
## pressure pressure          numeric
## triceps   triceps          numeric
## insulin   insulin          numeric
## mass         mass          numeric
## pedigree pedigree          numeric
## age           age          numeric
## diabetes diabetes           factor
```

Opis zmiennych

- pregnant - liczba przebytych ciąż
- glucose - stężenie glukozy w osoczu
- pressure - ciśnienie rozkurczowe (mm Hg)
- triceps - grubość fałdu skórnego tricepsa (mm)
- insulin - insulina w surowicy 2-godzinna (mu U/ml)
- mass - maksymalny indeks ciała
- pedigree - funkcja rodowodu cukrzycy
- age - wiek (lata)
- diabetes - zmienna klasowa (test na cukrzycę)

Zbiór danych PimaIndiansDiabetes2 posiada dwie klasy: 0 - brak cukrzycy 1 - obecność cukrzycy

Za informacje o przynależności obkietu do konkretnej klasy odpowiada zmienna ***diabetes*** i zawiera wartości "pos" (positive – cukrzyca) oraz "neg" (negative – brak cukrzycy).

Zmienne ciągłe

```
## [1] "pregnant" "glucose"  "pressure" "triceps"  "insulin"  "mass"     "pedigree"
## [8] "age"
```

Zmienne dyskretne

```
## [1] "diabetes"
```

Wszystkie zmienne mają prawidłowo przypisane typy. Zmienna zawierająca etykietki klas jest zmienną typu factor. Reszta zmiennych jest typu numeric, co jest zgodne z intuicją.

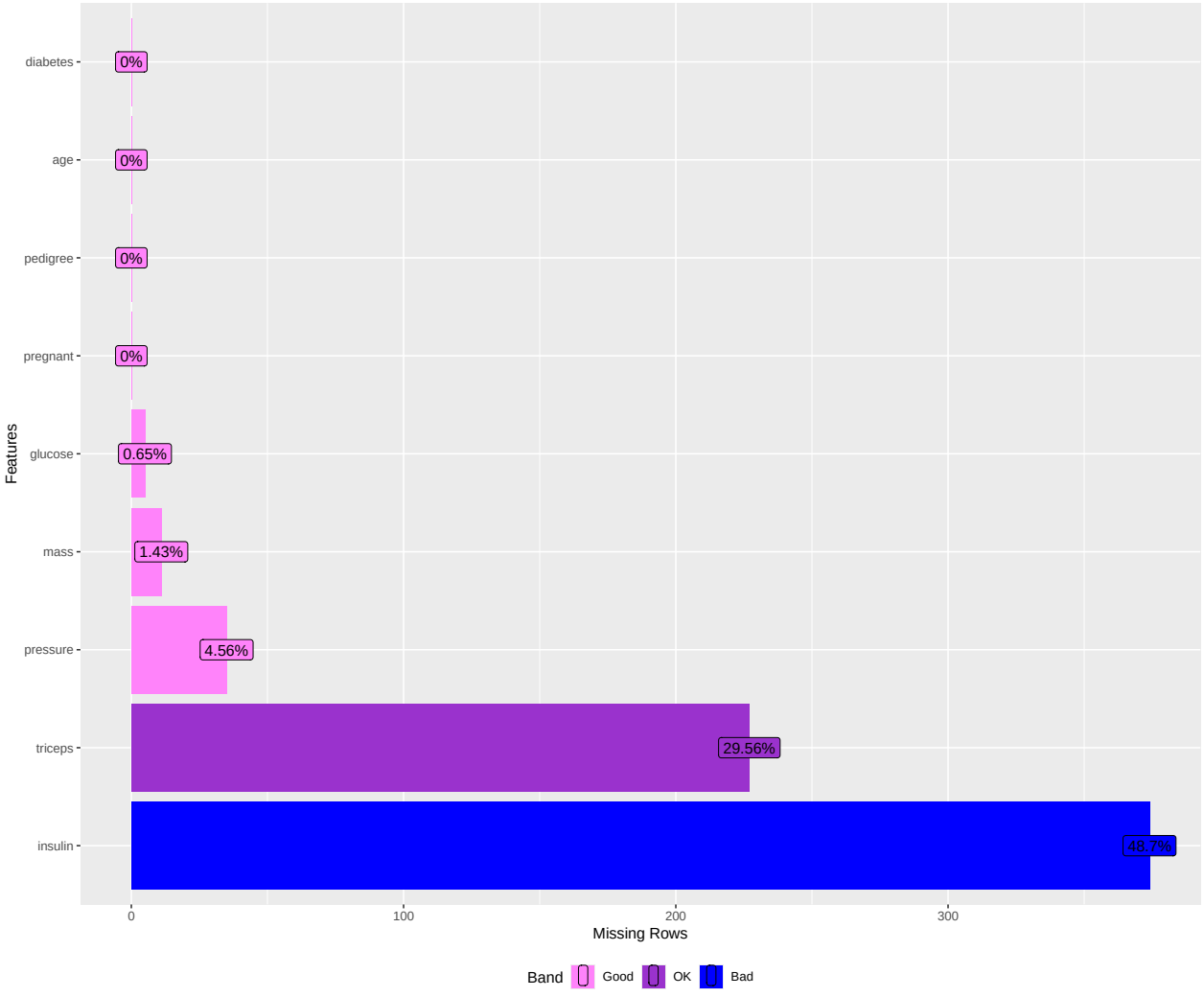
Przydatność w analizie: wszytskie zmienne są przydatne w analizie. Wykorzytsam je do wysunięcia ciekawych wniosków.

1.2.3 Brakujące obserwacje - nietypowe własności danych

Liczba NA - brakujących obserwacji - Liczba braków danych kodowanych za pomocą "NA" wynosi 652.

	Ilość brakujących danych
pregnant	0
glucose	5
pressure	35
triceps	227
insulin	374
mass	11
pedigree	0
age	0
diabetes	0

Brakujące dane są tylko dla zmiennych typu numeric.

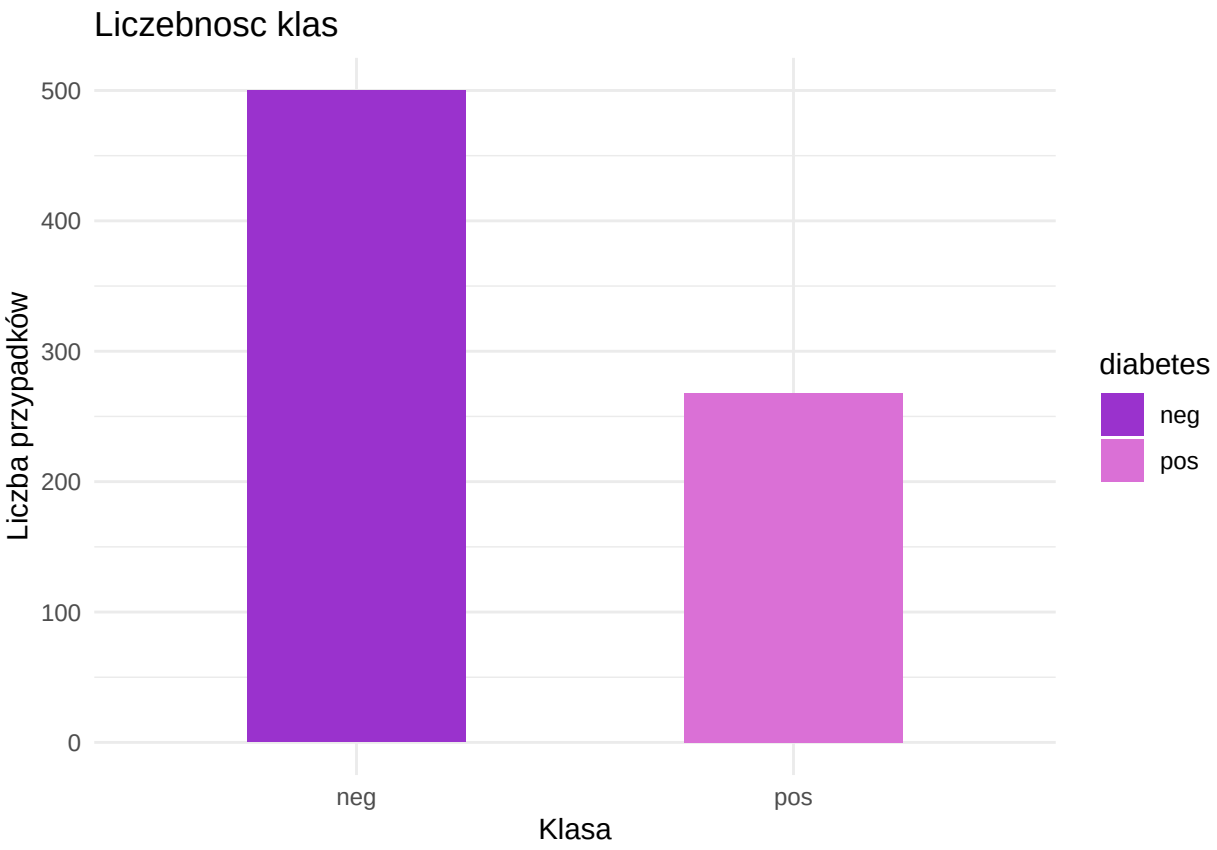


Do dalszej analizy potrzebne jest pominięcie wartości NA lub zastąpienie ich wartościami liczbowymi. Usunięcie (czyli pominięcie) wierszy zawierających NA, powoduje utratę wielu obserwacji. Zatem zastosujemy metodę "KNN" - dla 5 sąsiadów.

1.3 c) Wstępna analiza danych

1.3.1 Rozkład klas - dysproporcje, błąd klasyfikacji

Etykiety klas	Liczebność	Liczebność w %
neg	500	65.1
pos	268	34.9

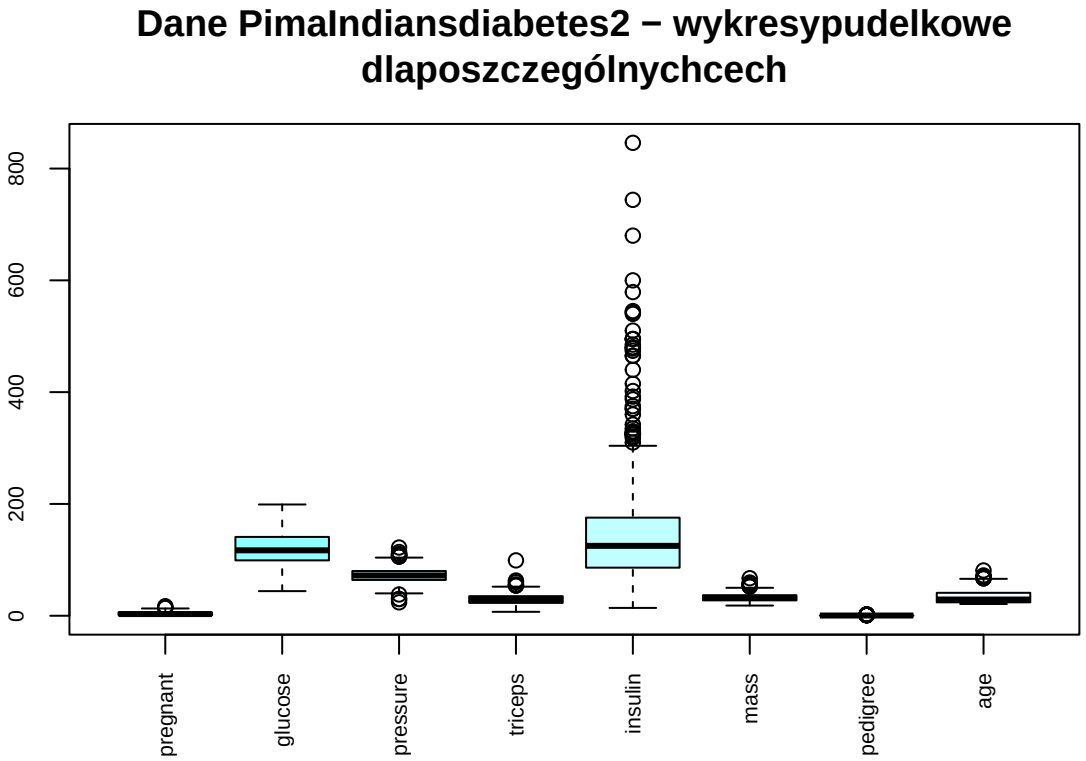


Zauważamy dysproporcje klas, jest ona jednak umiarkowana około 65% (klasa “neg”) do około 35% (klas “pos”). Dużą dysproporcją byłą by róznica na poziomie 95% do 5%.

Najczęściej występująca klasą jest klasa “neg”.

Bład klasyfikacji przypsiania wszytskich obiektów do klasy “neg” jest na poziomie 34.9% co odpowiada procentowej liczebności klasy “pos”.

1.3.2 Zmienność poszczególnych cech



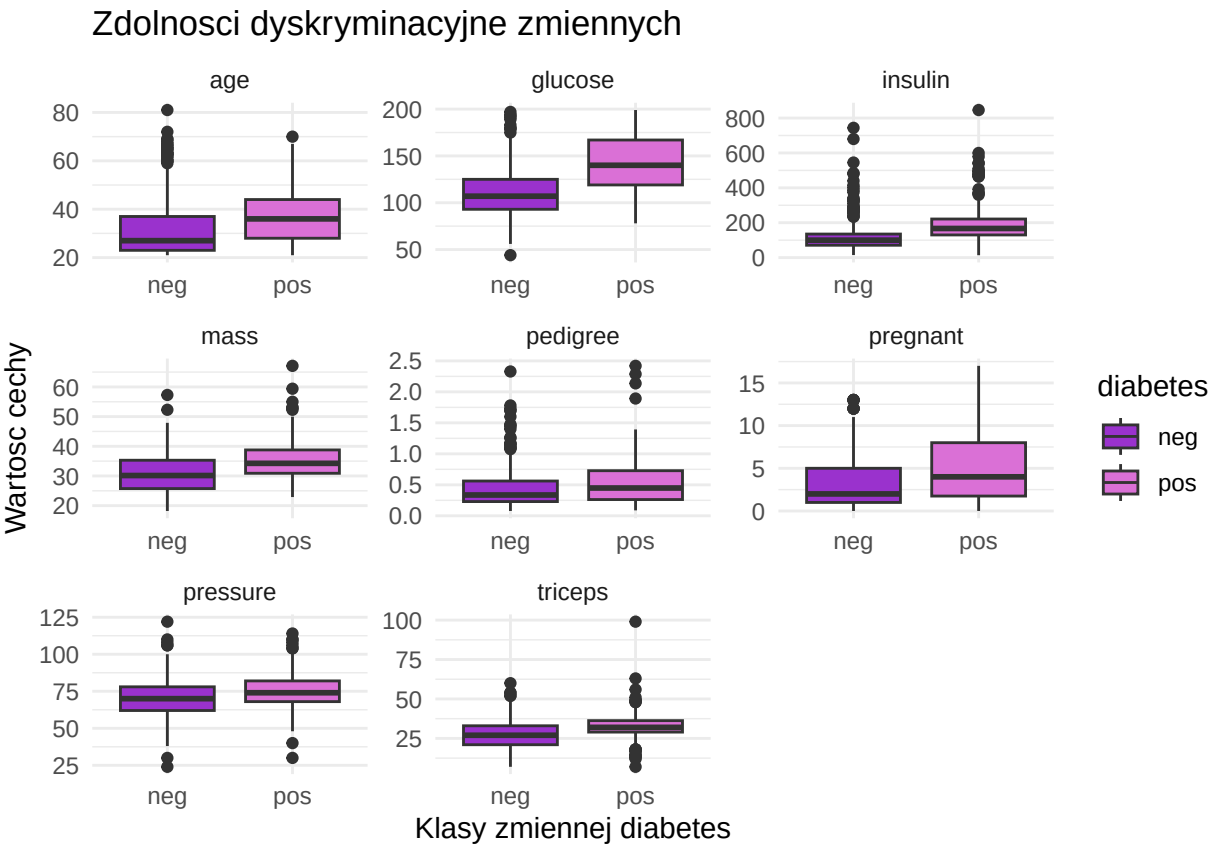
Na powyższym wykresie zauważamy dużą rozbieżność miedzy zakresami zmienności poszczególnych cech.

	wariancje zmiennych
pregnant	11.35
glucose	929.78
pressure	150.25
triceps	85.90
insulin	9357.00
mass	47.40
pedigree	0.11
age	138.30

Zmienna glucose oraz insulin posiadają wysoką wariancję (odpowiednio 9357, 929.78) w porównaniu z niewielką zmiennością cechy pedigree - 0.11 czy z wariancją zmiennej pregnant - 11.35.

Konieczne będzie zastosowanie standaryzacji dla alogorytmu KNN.

1.3.3 Zdolności dyskryminacyjne



Cecha ***glucose*** i ***insulin*** charaktaryzują się najlepszymi zdolnościami predykcyjnimi na tle innych zmiennych.

Warto wspomieć, że żadna z cech nie osiąga 100% zdolności do dyskryminacji.

Najgorszą zdolność mają cechy takie jak ***pressure***, ***triceps***, ***pedigree***, czy ***pregnat***.

1.4 d) Prównanie algorytmów - dokładność klasyfikacji na zbiorze uczącym i testowym

1.4.1 Zbiór uczący i testowy

Zbiór uczący i testowy

Tworzymy ***zbiór uczący*** i ***testowy***.

Liczba przypadków w zbiorze uczącym to 512, a w zbiorze testowym 759.

1.4.2 Algorytm KNN

Algorytm KNN

Algorytm ten jak wspomniałam wyżej wymaga standaryzacji danych.

Zastosuję algorytm KNN dla wszystkich cech ze zbioru danych PimaIndiansdiabetes2, przyjmując k=5.

Zbiór testowy

Macierz pomyłek

Tablica 4: Macierz pomyłek dla zbioru testowego

	rzeczywisty neg	rzeczywisty pos
prognozowany neg	136	26
prognozowany pos	35	59

Błąd klasyfikacji

Błąd klasyfikacji na zbiorze testowym wynosi 0.2383, czyli 23.83% próbek było sklasyfikowanych błędnie.

Zbiór uczący

Macierz pomyłek

Tablica 5: Macierz pomyłek dla zbioru uczącego

	rzeczywisty neg	rzeczywisty pos
prognozowany neg	295	39
prognozowany pos	34	144

Błąd klasyfikacji

Błąd klasyfikacji na zbiorze uczącym wynosi 0.1426, czyli 14.26%próbek było sklasyfikowanych błędnie.

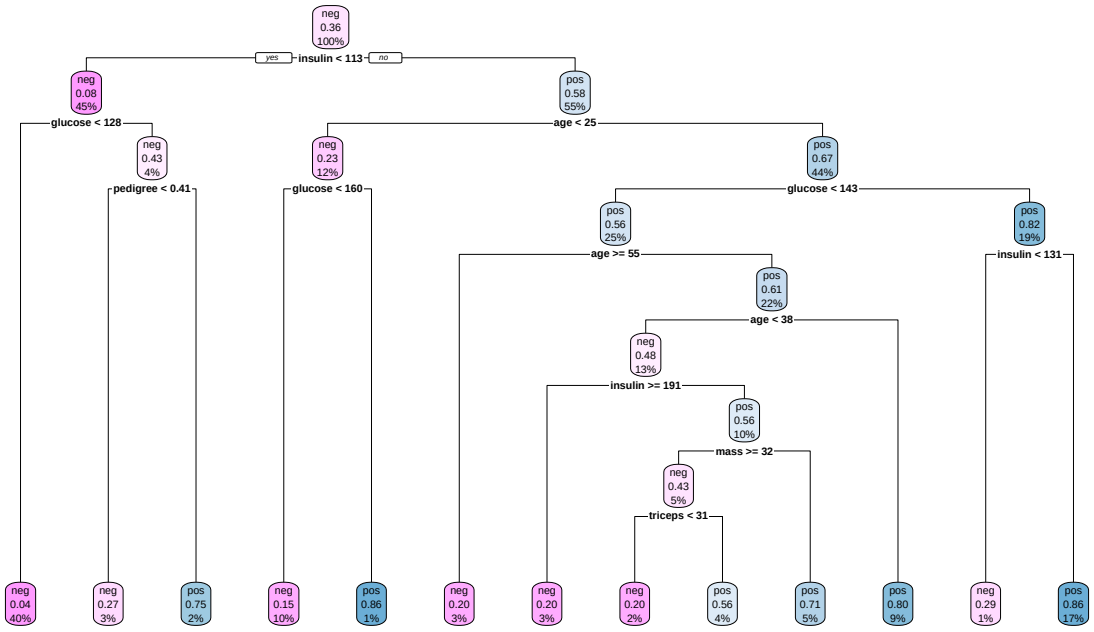
Wnioski

Błąd klasyfikacji na zbiorze uczącym jest mniejszy niż na zbiorze testowym, co jest zgodne z oczekiwaniami, ponieważ model, który był trenowany na zbiorze uczącym można powiedzieć, że "nauczył" się lepszego rozwiązania dla konkretnych wcześniej mu znanych danych.

1.4.3 Algorytm drzewa klasyfikacyjnego

Konstrukacja drzewa klasyfikacyjnego

Drzewo klasyfikacyjne dla danych PimaIndiansdiabetes2



Zbiór testowy

Macierz pomyłek

Tablica 6: Macierz pomyłek dla zbioru testowego

	rzeczywisty neg	rzeczywisty pos
prognozowany neg	136	28
prognozowany pos	35	57

Błąd klasyfikacji

Błąd klasyfikacji na zbiorze testowym wynosi 0.2461, czyli 24.61% próbek było sklasyfikowanych błędnie.

Zbiór uczący

Macierz pomyłek

Tablica 7: Macierz pomyłek dla zbioru uczącego

	rzeczywisty neg	rzeczywisty pos
prognozowany neg	290	31
prognozowany pos	39	152

Błąd klasyfikacji

Błąd klasyfikacji na zbiorze uczącym wynosi 0.1367, czyli 13.67% próbek było sklasyfikowanych błędnie.

Wnioski

Błąd klasyfikacji na zbiorze uczącym jest mniejszy niż na zbiorze testowym, co jest zgodne z oczekiwaniami, ponieważ model, który był trenowany na zbiorze uczącym można powiedzieć, że "nauczył" się lepszego rozwiązania dla konkretnych wcześniej mu znanych danych.

1.4.4 Algorytm naiwnnego klasyfikatora bayesowskiego

Budujemy model korzystając z funkcji naiveBayes.

Zbiór testowy

Macierz pomyłek

Tablica 8: Macierz pomyłek dla zbioru testowego

	rzeczywisty neg	rzeczywisty pos
prognozowany neg	139	30
prognozowany pos	32	55

Błąd klasyfikacji

Błąd klasyfikacji na zbiorze testowym wynosi 0.2422, czyli 24.22% próbek było sklasyfikowanych błędnie.

Zbiór uczący

Macierz pomyłek

Tablica 9: Macierz pomyłek dla zbioru uczącego

	rzeczywisty neg	rzeczywisty pos
prognozowany neg	271	64
prognozowany pos	58	119

Błąd klasyfikacji

Błąd klasyfikacji na zbiorze uczącym wynosi 0.2383, czyli 23.83% próbek było sklasyfikowanych błędnie.

Wnioski

Błąd klasyfikacji na zbiorze uczącym jest mniejszy niż na zbiorze testowym, co jest zgodne z oczekiwaniami, ponieważ model, który był trenowany na zbiorze uczącym można powiedzieć, że "nauczył" się lepszego rozwiązania dla konkretnych wcześniej mu znanych danych. Zauważamy jednak, że różnica między poszczególnymi błędami jest niewielka, o wiele mniejsza niż w przypadku algorytmu KNN czy drzew klasyfikacyjnych.

1.4.5 Wnioski

Tablica 10: Błąd klasyfikacyjny na zbiorze testowym i uczącym odpowiednich algorytmów

	zbiór testowy	zbiór uczący
KNN	23.83%	14.26%
drzewa klasyfikacyjne	24.61%	13.67%
naiwny klasyfikator bayesowski	24.22%	23.83%

Największą różnicę między błędem klasyfikacyjnym na zbiorze uczącym, a testowym widzimy dla algorytmu drzew klasyfikacyjnych. Podobnie zarówno na zbiorze testowym jak i uczącym poradził sobie naiwny klasyfikator bayesowski. Algorytm KNN daje wyniki na obu zbiorach zbliżone odpowiedno do tych otrzymany po zastosowaniu alkorytmu drzew klasyfikacyjnych.

1.5 d') Prównanie algorytmów - dokładność klasyfikacji zaawansowanych metod

1.5.1 Zaawansowany schemat oceny dokładności

Przygotowuje "wrapper" dostosowując funkcję predict dla modelu do standardu wymaganego przez funkcję "errorest".

1.5.2 Algorytm KNN

Algorytm KNN

Konieczne jest zastosowanie standrayzacji danych dla algorytmu KNN.

Metoda cross-validation

```
##
## Call:
## errorest.data.frame(formula = diabetes ~ ., data = datas, model = my.ipredknn,
##     predict = my.predict, estimator = "cv", est.para = control.errorest(k = 10),
##     ile.sasiadow = 5)
##
##   10-fold cross-validation estimator of misclassification error
##
## Misclassification error:  0.2318
```

Schemat typu bootstrap

```
##
## Call:
## errorest.data.frame(formula = diabetes ~ ., data = datas, model = my.ipredknn,
##     predict = my.predict, estimator = "boot", est.para = control.errorest(nboot = 50),
##     ile.sasiadow = 5)
##
##   Bootstrap estimator of misclassification error
##   with 50 bootstrap replications
##
## Misclassification error:  0.2578
## Standard deviation: 0.0033
```

Schemat "632+"

```
##
## Call:
## errorest.data.frame(formula = diabetes ~ ., data = datas, model = my.ipredknn,
##     predict = my.predict, estimator = "632plus", est.para = control.errorest(nboot = 50),
##     ile.sasiadow = 5)
##
```

```
##   .632+ Bootstrap estimator of misclassification error
##   with 50 bootstrap replications
##
## Misclassification error:  0.2285
```

Wnioski - algorytm KNN

Tablica 11: Błąd klasyfikacyjny algorytmu KNN

	wartość w %
cross-validation	23.18%
bootstrap	25.78%
632+	22.85%

Największy błąd klasyfikacyjny dla algorytmu KNN otrzymujemy stosując schemat typu bootstrap, najmniejszy błąd, czyli najlepsze dopasowanie danych dla schematu 632+.

1.5.3 Algorytm drzew klasyfikacyjnych

Algorytm drzew klasyfikacyjnych

metoda cross-validation

```
##
## Call:
## errorest.data.frame(formula = diabetes ~ ., data = data, model = my.tree,
##     predict = my.predict, estimator = "cv", est.para = control.errorest(k = 10))
##
##   10-fold cross-validation estimator of misclassification error
##
## Misclassification error:  0.1953
```

Schemat typu bootstrap

```
##
## Call:
## errorest.data.frame(formula = diabetes ~ ., data = data, model = my.tree,
##     predict = my.predict, estimator = "boot", est.para = control.errorest(nboot = 50))
##
##   Bootstrap estimator of misclassification error
##   with 50 bootstrap replications
##
## Misclassification error:  0.2336
## Standard deviation: 0.0032
```

Schemat "632+"

```
##
## Call:
## errorest.data.frame(formula = diabetes ~ ., data = data, model = my.tree,
##     predict = my.predict, estimator = "632plus", est.para = control.errorest(nboot = 50))
```

```
##
##   .632+ Bootstrap estimator of misclassification error
##   with 50 bootstrap replications
##
## Misclassification error:  0.2112
```

Wnioski - algorytm drzew klasyfikacyjnych

Tablica 12: Błąd klasyfikacyjny algorytmu drzew klasyfikacyjnych

	wartość w %
cross-validation	19.53%
bootstrap	23.36%
632+	21.12%

Najlepiej sprawdzającym się schematem dla algorytmu drzew klasyfikacyjnych jest cross-validation - otrzymujemy najmniejszy błąd klasyfikacyjny. Największy błąd otrzymujemy przy zastosowaniu metody bootstrap, drugie miejsce zajmuje schemat 632+.

1.5.4 Algorytm naiwnnego klasyfikatora bayesowskiego

Algorytm naiwnnego klasyfikatora bayesowskiego

Metoda cross-validation

```
##
## Call:
## errorest.data.frame(formula = diabetes ~ ., data = data, model = my.naiveBayes,
##     predict = my.predict, estimator = "cv", est.para = control.errorest(k = 10))
##
##   10-fold cross-validation estimator of misclassification error
##
## Misclassification error:  0.2409
```

Schemat typu bootstrap

```
##
## Call:
## errorest.data.frame(formula = diabetes ~ ., data = data, model = my.naiveBayes,
##     predict = my.predict, estimator = "boot", est.para = control.errorest(nboot = 50))
##
##   Bootstrap estimator of misclassification error
##   with 50 bootstrap replications
##
## Misclassification error:  0.24
## Standard deviation: 0.0013
```

Schemat "632+"

```
##
## Call:
## errorest.data.frame(formula = diabetes ~ ., data = data, model = my.naiveBayes,
##     predict = my.predict, estimator = "632plus", est.para = control.errorest(nboot = 50))
##
##   .632+ Bootstrap estimator of misclassification error
##   with 50 bootstrap replications
##
## Misclassification error:  0.2416
```

Wnioski - algorytm naiwnnego klasyfikatora bayesowskiego

Tablica 13: Błąd klasyfikacyjny algorytmu naiwnnego klasyfikatora bayesowskiego

	wartość w %
cross-validation	24.09%
bootstrap	24%
632+	24.16%

Najlepszym schematem dla naiwnnego klasyfikatora bayesowskiego okazał się schemat typu bootstrap, najgorszym 632+. Trzeba jednak zwrócić uwagę na to, że każdy ze schematów zwraca zbliżone wyniki, w skrajnym przypadku różnica między błędami klasyfikacji wynosi 0.16%.

1.5.5 Wnioski

Tablica 14: Błąd klasyfikacyjny odpowiednich algorytmów

	cross-validation	bootstrap	632+
KNN	23.18%	25.78%	22.85%
drzewa klasyfikacyjne	19.53%	23.36%	21.12%
naiwny klasyfikator bayesowski	24.09%	24%	24.16%

Schemat cross-validation, bootstrap jak i 632+ okazał się najlepszy dla algorytmu drzew klasyfikacyjnych. Największy błąd klasyfikacyjny otrzymaliśmy dla algorytmu KNN i zastosowanej metody bootstrap. Schemat 632+ oraz cross-validation okazał się najgorszy dla algorytmu naiwnego klasyfikatora bayesowskiego.

1.6 d") Wnioski - podsumowanie i porównanie wyniosków otrzymanych z podpunktów d) oraz d')

Tablica 15: Błąd klasyfikacyjny odpowiednich algorytmów

	cross-validation	bootstrap	632+
KNN	23.18%	25.78%	22.85%
drzewa klasyfikacyjne	19.53%	23.36%	21.12%
naiwny klasyfikator bayesowski	24.09%	24%	24.16%

Tablica 16: Błąd klasyfikacyjny na zbiorze testowym i uczącym odpowiednich algorytmów

	zbiór testowy	zbiór uczący
KNN	23.83%	14.26%
drzewa klasyfikacyjne	24.61%	13.67%
naiwny klasyfikator bayesowski	24.22%	23.83%

Najmniejszy błądy klasyfikacyjny otrzymujemy na zbiorze uczącym dla algorytmu drzew klasyfikacyjnych. Również dla przetestowanych schematów: cross-validation, bootstrap, 632+ ten algorytm uplasował się najlepiej tzn. otrzymaliśmy najmniejsze błędy klasyfikacyjne. Największe błędy notujemy dla naiwnego klasyfikatora bayesowskiego zarówno na zbiorze uczącym jak i używając schematu cross-validation, 632+. Algorytm KNN osiągnał najmniejszy błąd klasyfikacji na zbiorze testowym, a największy uzyskaliśmy testując algorytm drzew klasyfikacyjnych. Stosując dla algorytmu KNN schemat typu bootstrap otrzymujemy największy możliwy błąd klasyfikacji z porównywanych błędów otrzymanych przy tym schemacie.Wyniki dla naiwnego klasyfikatora bayesowskiego są bardzo do siebie podobne, te otrzymane na zbiorze testowym, uczącym do tych uzyskanych przy użyciu schematów bootstrap, cross-validation, 632+. Oscylują one między 23.83%-24.16%.

1.7 e) Porównanie dokładości klasyfikacji uwzględniając różne podzbiory cech

1.7.1 Algorytm KNN

1) Podzbiór zawierający wyszystkie zmienie

Wykorzystam wyniki z podpunktu d') do porównania, aby nie powielać identycznych obliczeń.

Tablica 17: Błąd klasyfikacyjny odpowiednich algorytmów

	cross-validation	bootstrap	632+
KNN	23.18%	25.78%	22.85%

2) Podzbiór zawierający zmienne glucose i insulin

Konieczne jest zastosowanie standrayzacji danych dla algorytmu KNN.

Schemat cross-validation

```
##
## Call:
## errorest.data.frame(formula = diabetes ~ ., data = datas, model = my.ipredknn,
##     predict = my.predict, estimator = "cv", est.para = control.errorest(k = 10),
##     ile.sasiadow = 5)
##
##   10-fold cross-validation estimator of misclassification error
##
## Misclassification error:  0.2682
```

Schemat bootstrap

```
##
## Call:
## errorest.data.frame(formula = diabetes ~ ., data = datas, model = my.ipredknn,
##     predict = my.predict, estimator = "boot", est.para = control.errorest(nboot = 50),
##     ile.sasiadow = 5)
##
##   Bootstrap estimator of misclassification error
##   with 50 bootstrap replications
##
## Misclassification error:  0.2889
## Standard deviation: 0.0027
```

Schemat "632+"

```
##
## Call:
## errorest.data.frame(formula = diabetes ~ ., data = datas, model = my.ipredknn,
##     predict = my.predict, estimator = "632plus", est.para = control.errorest(nboot = 50),
##     ile.sasiadow = 5)
##
##   .632+ Bootstrap estimator of misclassification error
##   with 50 bootstrap replications
##
## Misclassification error:  0.2558
```

Wnioski

Tablica 18: Błąd klasyfikacyjny odpowiednich algorytmów dla podzbioru: insulin, glucose

	cross-validation	bootstrap	632+
KNN	26.82%	28.89%	25.58%

Dla algorytmu KNN najlepiej wypadł schemat 632+ - najmniejszy błąd klasyfikacyjny, największy błąd klasyfikacyjny notujemy dla schematu typu bootstrap.

3) Podzbiór niezawierający zmiennych triceps pressure, pregnant oraz pedigree. (z podpunktu c otrzyamliśmy, że te zmienne najbardziej wypływają - zwiększają błąd klasywikacyjny)

Schemat cross-validation

```
##
## Call:
## errorest.data.frame(formula = diabetes ~ ., data = datas, model = my.ipredknn,
##     predict = my.predict, estimator = "cv", est.para = control.errorest(k = 10),
##     ile.sasiadow = 5)
##
##   10-fold cross-validation estimator of misclassification error
##
## Misclassification error:  0.2161
```

Schemat typu bootstrap

```
##
## Call:
## errorest.data.frame(formula = diabetes ~ ., data = datas, model = my.ipredknn,
##     predict = my.predict, estimator = "boot", est.para = control.errorest(nboot = 50),
##     ile.sasiadow = 5)
##
##   Bootstrap estimator of misclassification error
##   with 50 bootstrap replications
##
## Misclassification error:  0.2465
## Standard deviation: 0.003
```

Schemat "632+"

```
##
## Call:
## errorest.data.frame(formula = diabetes ~ ., data = datas, model = my.ipredknn,
##     predict = my.predict, estimator = "632plus", est.para = control.errorest(nboot = 50),
##     ile.sasiadow = 5)
##
##   .632+ Bootstrap estimator of misclassification error
##   with 50 bootstrap replications
##
## Misclassification error:  0.2178
```

Wnioski

Tablica 19: Błąd klasyfikacyjny odpowiednich algorytmów dla podzbioru nie zawierającego zmiennych: triceps, pressure, pedigree, pregnant

	cross-validation	bootstrap	632+
KNN	21.61%	24.65%	21.78%

Najlepiej dla KNN tego podzbioru danych wypadł schemat cross-validation - najmniejszy błąd klasyfikacyjny. Największy błąd klasyfikacyjny notujemy dla schematu typu bootstrap.

1.7.2 Algorytm drzew klasyfikacyjnych

1) Podzbiór zawierający wyszystkie zmienie

Wykorzystam wyniki z podpunktu d') do porównania, aby nie powielać identycznych obliczeń.

Tablica 20: Błąd klasyfikacyjny odpowiednich algorytmów

	cross-validation	bootstrap	632+
drzewa klasyfikacyjne	19.53%	23.36%	21.12%

2) Podzbiór zawierający zmienne glucose i insulin

Schemat cross-validation

```
##
## Call:
## errorest.data.frame(formula = diabetes ~ ., data = data[, c(2,
##     5, 9)], model = my.tree, predict = my.predict, estimator = "cv",
##     est.para = control.errorest(k = 10))
##
##   10-fold cross-validation estimator of misclassification error
##
## Misclassification error:  0.2305
```

Schemat typu bootstrap

```
##
## Call:
## errorest.data.frame(formula = diabetes ~ ., data = data[, c(2,
##     5, 9)], model = my.tree, predict = my.predict, estimator = "boot",
##     est.para = control.errorest(nboot = 50))
##
##   Bootstrap estimator of misclassification error
##   with 50 bootstrap replications
##
## Misclassification error:  0.2569
## Standard deviation: 0.0034
```

Schemat "632+"

```
##
## Call:
## errorest.data.frame(formula = diabetes ~ ., data = data[, c(2,
##     5, 9)], model = my.tree, predict = my.predict, estimator = "632plus",
##     est.para = control.errorest(nboot = 50))
##
##   .632+ Bootstrap estimator of misclassification error
##   with 50 bootstrap replications
##
## Misclassification error:  0.2356
```

Wnioski

Tablica 21: Błąd klasyfikacyjny odpowiednich algorytmów dla podzbioru: insulin, glucose

	cross-validation	bootstrap	632+
drzewa klasyfikacyjne	23.05%	25.69%	23.56%

Dla algorytmu drzew klasyfikacyjnych najlepiej (dla tego podzbioru danych) wypadł schemat cross-validation - najmniejszy błąd klasyfikacyjny. Największy błąd klasyfikacyjny notujemy dla schematu typu bootstrap.

3) Podzbiór niezawierający zmiennych triceps pressure, pregnant oraz pedigree. (z podpunktu c otrzyamliśmy, że te zmienne najbardziej wypływają - zwiększają błąd klasywikacyjny)

Schemat cross-validation

```
##
## Call:
## errorest.data.frame(formula = diabetes ~ ., data = data[, -c(1,
##     3, 4, 7)], model = my.tree, predict = my.predict, estimator = "cv",
##     est.para = control.errorest(k = 10))
##
##   10-fold cross-validation estimator of misclassification error
##
## Misclassification error:  0.2331
```

Schemat typu bootstrap

```
##
## Call:
## errorest.data.frame(formula = diabetes ~ ., data = data[, -c(1,
##     3, 4, 7)], model = my.tree, predict = my.predict, estimator = "boot",
##     est.para = control.errorest(nboot = 50))
##
##   Bootstrap estimator of misclassification error
##   with 50 bootstrap replications
##
## Misclassification error:  0.2166
## Standard deviation: 0.0028
```

Schemat "632+"

```
##
## Call:
## errorest.data.frame(formula = diabetes ~ ., data = data[, -c(1,
##     3, 4, 7)], model = my.tree, predict = my.predict, estimator = "632plus",
##     est.para = control.errorest(nboot = 50))
##
##   .632+ Bootstrap estimator of misclassification error
##   with 50 bootstrap replications
##
## Misclassification error:  0.2047
```

Wnioski

Tablica 22: Błąd klasyfikacyjny odpowiednich algorytmów dla podzbioru nie zawierającego zmiennych: triceps, pressure, pedigree, pregnant

	cross-validation	bootstrap	632+
drzewa klasyfikacyjne	23.31%	21.66%	20.47%

Najlepiej dla tego podzbioru danych i algorytmu drzew klasyfikacyjnych wypadł schemat 632+ - najmniejszy błąd klasyfikacyjny. Największy błąd klasyfikacyjny notujemy dla schematu typu cross-validation.

1.7.3 Algorytm naiwnnego klasyfikatora bayesowskiego

1) Podzbiór zawierający wyszystkie zmienie

Wykorzystam wyniki z podpunktu d') do porównania, aby nie powielać identycznych obliczeń.

Tablica 23: Błąd klasyfikacyjny odpowiednich algorytmów

	cross-validation	bootstrap	632+
naiwny klasyfikator bayesowski	24.09%	24%	24.16%

2) Podzbiór zawierający zmienne glucose i insulin

Schemat cross-validation

```
##
## Call:
## errorest.data.frame(formula = diabetes ~ ., data = data[, c(2,
##     5, 9)], model = my.naiveBayes, predict = my.predict, estimator = "cv",
##     est.para = control.errorest(k = 10))
##
##   10-fold cross-validation estimator of misclassification error
##
## Misclassification error:  0.2552
```

Schemat typu bootstrap

```
##
## Call:
## errorest.data.frame(formula = diabetes ~ ., data = data[, c(2,
##     5, 9)], model = my.naiveBayes, predict = my.predict, estimator = "boot",
##     est.para = control.errorest(nboot = 50))
##
##   Bootstrap estimator of misclassification error
##   with 50 bootstrap replications
##
## Misclassification error:  0.2597
## Standard deviation: 0.001
```

Schemat "632+"

```
##
## Call:
## errorest.data.frame(formula = diabetes ~ ., data = data[, c(2,
##     5, 9)], model = my.naiveBayes, predict = my.predict, estimator = "632plus",
##     est.para = control.errorest(nboot = 50))
##
##   .632+ Bootstrap estimator of misclassification error
##   with 50 bootstrap replications
##
## Misclassification error:  0.2591
```

Wnioski

Tablica 24: Błąd klasyfikacyjny odpowiednich algorytmów dla podzbioru: insulin, glucose

	cross-validation	bootstrap	632+
naiwny klasyfikator bayesowski	25.52%	25.97%	25.91%

Najlepiej dla tego podzbioru danych i naiwnego algorytmu bayesowskiego wypadł schemat cross-validation - najmniejszy błąd klasyfikacyjny. Największy błąd klasyfikacyjny notujemy dla schematu typu bootstrap. Trzeba jednak zwrócić uwagę, że dla wszytskich schematów wynik jest niemalże identyczny. Różnica między błędem klasyfikacji dla schematu typu bootstrap, a schematu 632+ jest na poziomie 0.06%.

3) Podzbiór niezawierający zmiennych triceps pressure, pregnant oraz pedigree. (z podpunktu c otrzyamliśmy, że te zmienne najbardziej wypływają - zwiększają błąd klasywikacyjny)

Schemat cross-validation

```
##
## Call:
## errorest.data.frame(formula = diabetes ~ ., data = data[, -c(1,
##     3, 4, 7)], model = my.naiveBayes, predict = my.predict, estimator = "cv",
##     est.para = control.errorest(k = 10))
##
##   10-fold cross-validation estimator of misclassification error
##
## Misclassification error:  0.237
```

Schemat typu bootstrap

```
##
## Call:
## errorest.data.frame(formula = diabetes ~ ., data = data[, -c(1,
##     3, 4, 7)], model = my.naiveBayes, predict = my.predict, estimator = "boot",
##     est.para = control.errorest(nboot = 50))
##
##   Bootstrap estimator of misclassification error
##   with 50 bootstrap replications
##
## Misclassification error:  0.2364
## Standard deviation: 0.0011
```

Schemat "632+"

```
##
## Call:
## errorest.data.frame(formula = diabetes ~ ., data = data[, -c(1,
##     3, 4, 7)], model = my.naiveBayes, predict = my.predict, estimator = "632plus",
##     est.para = control.errorest(nboot = 50))
##
##   .632+ Bootstrap estimator of misclassification error
##   with 50 bootstrap replications
##
## Misclassification error:  0.2331
```

Wnioski

Tablica 25: Błąd klasyfikacyjny odpowiednich algorytmów dla podzbioru nie zawierającego zmiennych: triceps, pressure, pedigree, pregnant

	cross-validation	bootstrap	632+
naiwny klasyfikator bayesowski	23.7%	23.64%	23.31%

Najlepiej dla tego podzbioru danych i naiwnego algorytmu bayesowskiego wypadł schemat 632+ - najmniejszy błąd klasyfikacyjny. Największy błąd klasyfikacyjny notujemy dla schematu cross-validation. Trzeba jednak zwrócić uwagę, że dla wszytskich schematów wynik jest niemalże identyczny. Różnica między błędem klasyfikacji dla schematu typu bootstrap, a schematu cross-validation jest na poziomie 0.06%.

1.7.4 Wnioski

Tablica 26: Błąd klasyfikacyjny odpowiednich algorytmów dla podzbioru nie zawierającego zmiennych: triceps, pressure, pedigree, pregnant

zbiór	schemat	cross-validation	bootstrap	623+
Cały zbiór	KNN	23.18%	25.78%	22.85%
	drzewa klasyfikacyjne	19.53%	23.36%	21.12%
	naiwny klasyfikator bayesowski	24.09%	24%	24.16%
Podzbiór ze zmiennymi insuline,glucose	KNN	26.82%	28.89%	25.58%
	drzewa klasyfikacyjne	23.05%	25.69%	23.56%
	naiwny klasyfikator bayesowski	25.52%	25.97%	25.91%
Podzbiór bez zmiennych tricpes,pedigree,pregnant,pressure	KNN	21.61%	24.65%	21.78%
	drzewa klasyfikacyjne	23.31%	21.66%	20.47%
	naiwny klasyfikator bayesowski	23.7%	23.64%	23.31%

Na całym zbiorze zmiennych oraz na podzbiorze zawierającym zmienne insulinę i glucose najmniejszy błąd klasyfikacyjny otrzymaliśmy dla algorytmu drzew klasyfikacyjnych stosując schemat cross-validation. Największy błąd klasyfikacji na tych zbiorach należy do algorytmu KNN i schematu bootstrap. Na zbiorze danych bez zmiennych triceps, pedigree, pregnant, pressure najlepszy okazał się algorytm drzew klasyfikacyjnych dla schematu 632+, drugi pod względem najmniejszego błędu klasyfikacji w tym zbiorze jest algorytm KNN dla schematu cross-validation. Tym razem również KNN i schemat bootstrap jest odpowiedzialny za największy błąd klasyfikacyjny na badanym zbiorze, Największe, zauważalne błędy klasyfikacji notujemy dla zbioru zawierającego tylko zmienne glucose oraz insulin. W tym zbiorze błąd klasyfikacji każdego z algorytmów niezależnie od schematu uległ znacznemu pogorszeniu tzn. wzrósł. Zbiorem dla którego notujemy najmniejsze błędy klasyfikacyjne dla badanych algorytmów i schematów jest zbiór bez zmiennych triceps, pedigree, pregnant oraz pressure. Ciekawą obserwacja jest również, że nie zależnie od wybranego podzbioru czy zbioru wyniki uzyskane dla naiwnego algorytmu bayseowskiego bez względu na wybrany schemat są prawie identyczne : cały zbiór - około 24%, podzbiór ze zmienną insuline,glucose - około 26%, podzbiór bez zmiennych tricpes, pedigree, pregnant, pressure - około 23.5%.

1.8 e') Nietuzinkowy dobór parametrów dla poszczególnych metod

Do porównania metod wykorzystam podzbiór, dla którego błąd klasyfikacyjny dla 3 algorytmów okazał się najmniejszy.

1.8.1 Algorytm KNN

Algorytm KNN

Schemat cross-validation

Dla 3 sąsiadów

```
##
## Call:
## errorest.data.frame(formula = diabetes ~ ., data = datas, model = my.ipredknn,
##     predict = my.predict, estimator = "cv", est.para = control.errorest(k = 10),
##     ile.sasiadow = 3)
##
##   10-fold cross-validation estimator of misclassification error
##
## Misclassification error:  0.2344
```

Dla 7 sąsiadów

```
##
## Call:
## errorest.data.frame(formula = diabetes ~ ., data = datas, model = my.ipredknn,
##     predict = my.predict, estimator = "cv", est.para = control.errorest(k = 10),
##     ile.sasiadow = 7)
##
##   10-fold cross-validation estimator of misclassification error
##
## Misclassification error:  0.2135
```

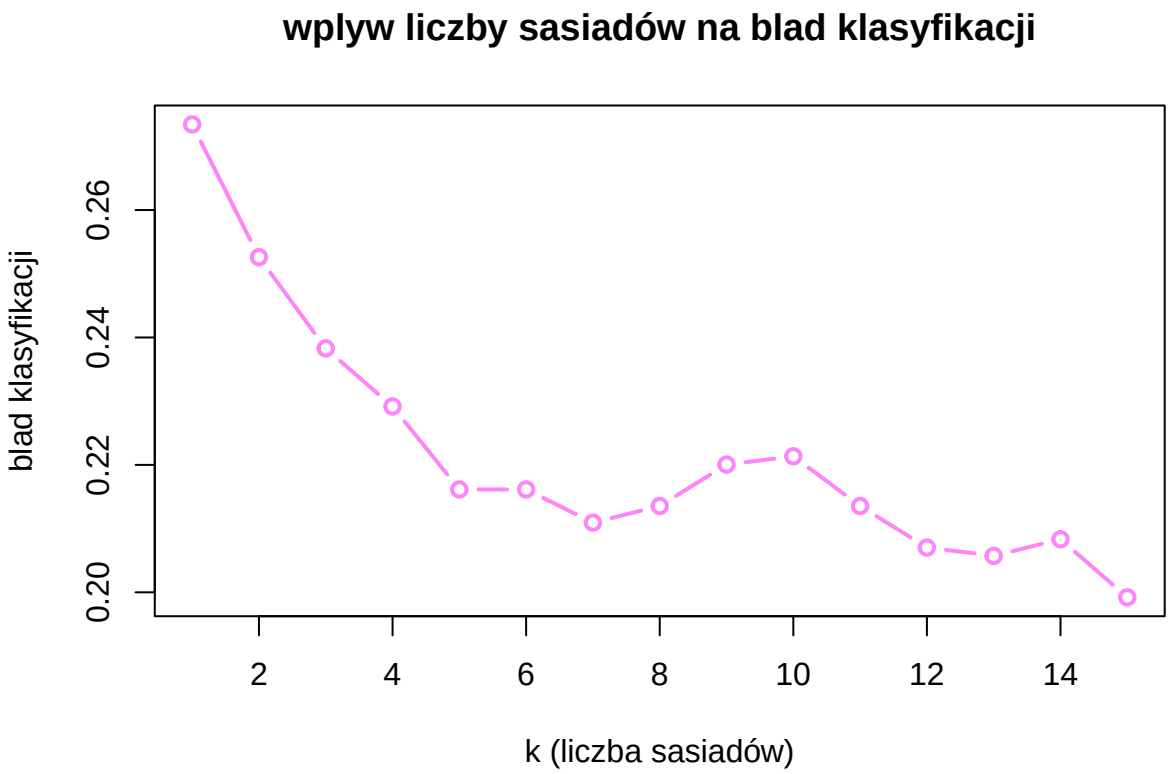
Dla 11 sąsiadów

```
##
## Call:
## errorest.data.frame(formula = diabetes ~ ., data = datas, model = my.ipredknn,
##     predict = my.predict, estimator = "cv", est.para = control.errorest(k = 10),
##     ile.sasiadow = 10)
##
##   10-fold cross-validation estimator of misclassification error
##
## Misclassification error:  0.207
```

Dla 15 sąsiadów

```
##
## Call:
## errorest.data.frame(formula = diabetes ~ ., data = datas, model = my.ipredknn,
```

```
##     predict = my.predict, estimator = "cv", est.para = control.errorest(k = 10),
##     ile.sasiadow = 15)
##
##   10-fold cross-validation estimator of misclassification error
##
## Misclassification error:  0.1966
```



Zauważamy dużą różnicę w wartości błędu klasyfikacji między małą liczbą sąsiadów, a coraz to większą. Im większa liczba (3-10) sąsiadów tym mniejszy błąd klasyfikacji.

Schemat typu bootstrap

Dla 3 sąsiadów

```
##
## Call:
## errorest.data.frame(formula = diabetes ~ ., data = datas, model = my.ipredknn,
##     predict = my.predict, estimator = "boot", est.para = control.errorest(nboot = 50),
##     ile.sasiadow = 3)
##
##   Bootstrap estimator of misclassification error
##   with 50 bootstrap replications
##
## Misclassification error:  0.2593
## Standard deviation: 0.0031
```

Dla 7 sąsiadów

```
##
## Call:
```

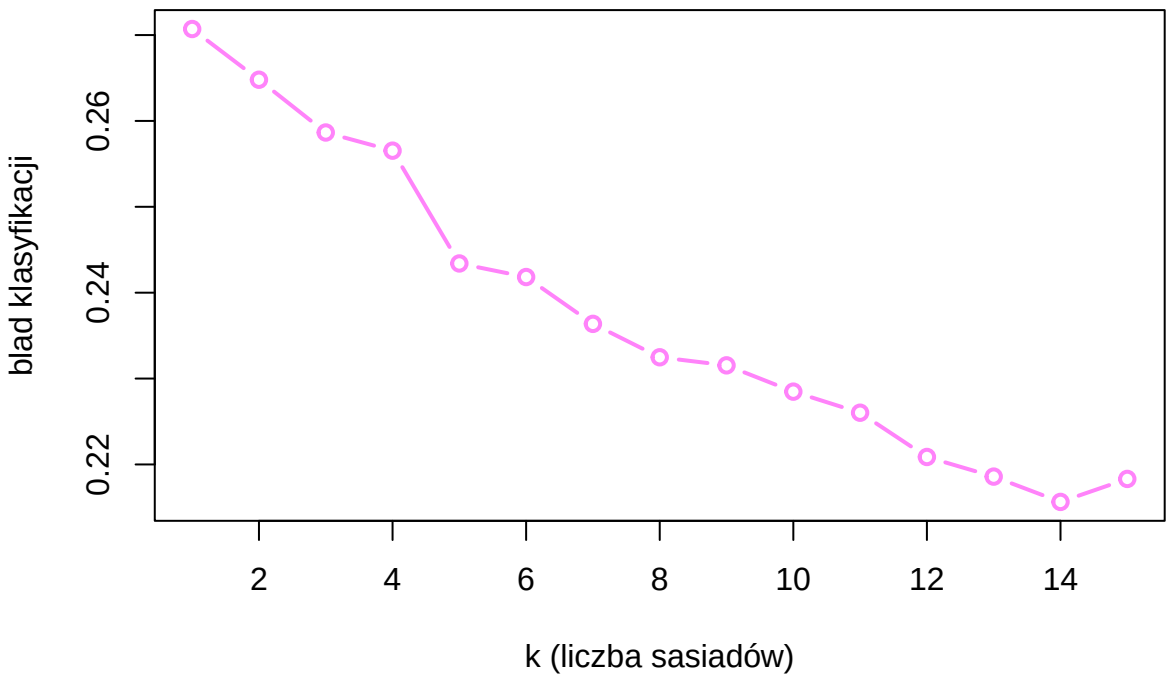
```
## errorest.data.frame(formula = diabetes ~ ., data = datas, model = my.ipredknn,
##     predict = my.predict, estimator = "boot", est.para = control.errorest(nboot = 50),
##     ile.sasiadow = 7)
##
##   Bootstrap estimator of misclassification error
##   with 50 bootstrap replications
##
## Misclassification error:  0.2312
## Standard deviation: 0.0026
```

Dla 11 sąsiadów

```
##
## Call:
## errorest.data.frame(formula = diabetes ~ ., data = datas, model = my.ipredknn,
##     predict = my.predict, estimator = "boot", est.para = control.errorest(nboot = 50),
##     ile.sasiadow = 11)
##
##   Bootstrap estimator of misclassification error
##   with 50 bootstrap replications
##
## Misclassification error:  0.2247
## Standard deviation: 0.0026
```

Dla 15 sąsiadów

```
##
## Call:
## errorest.data.frame(formula = diabetes ~ ., data = datas, model = my.ipredknn,
##     predict = my.predict, estimator = "boot", est.para = control.errorest(nboot = 50),
##     ile.sasiadow = 15)
##
##   Bootstrap estimator of misclassification error
##   with 50 bootstrap replications
##
## Misclassification error:  0.2155
## Standard deviation: 0.0024
```



Odnotowujemy spadek wartości błędu klasyfikacji ze wsrostem liczby sąsiadów, od liczby sąsiadów 14 nasz wykres przybiera tendencję rosnącą co może znaczyć, że dla dużej liczby sąsiadów 35-50 błąd klasyfikacji ponownie wzrośnie.

Schemat "632+"

Dla 3 sąsiadów

```
##
## Call:
## errorest.data.frame(formula = diabetes ~ ., data = datas, model = my.ipredknn,
##     predict = my.predict, estimator = "632plus", est.para = control.errorest(nboot = 50),
##     ile.sasiadow = 3)
##
##   .632+ Bootstrap estimator of misclassification error
##   with 50 bootstrap replications
##
## Misclassification error:  0.2282
```

Dla 7 sąsiadów

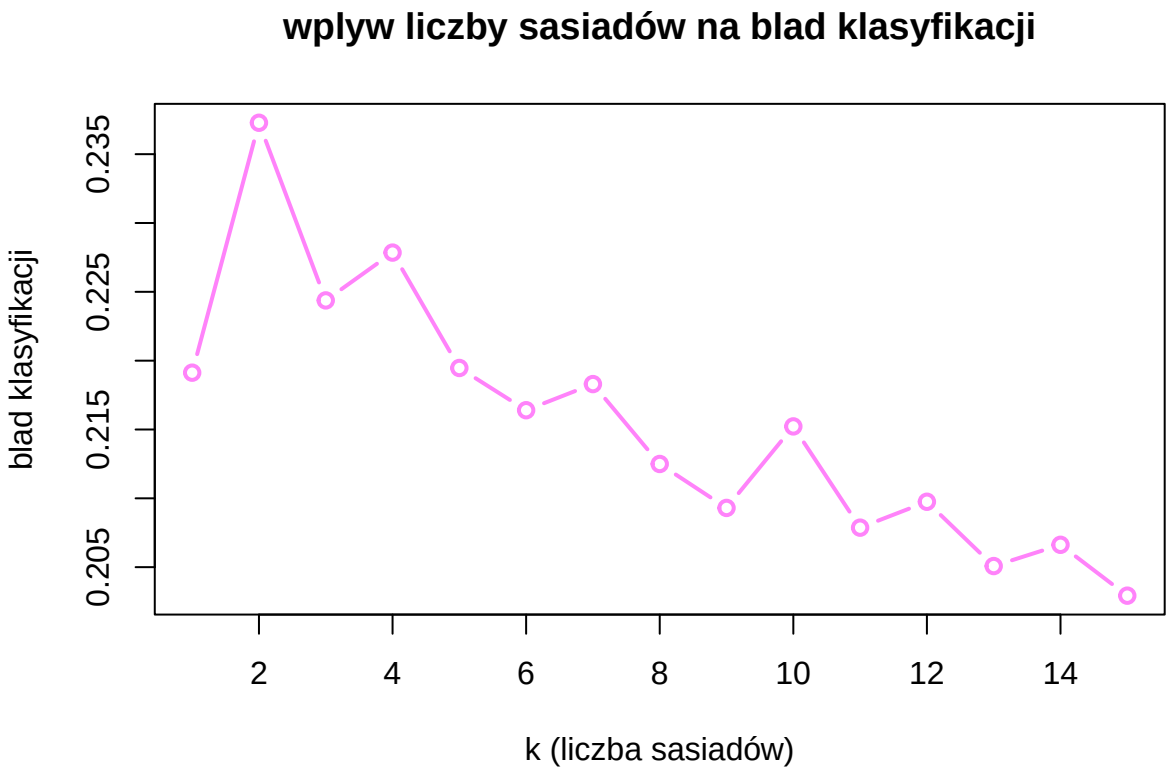
```
##
## Call:
## errorest.data.frame(formula = diabetes ~ ., data = datas, model = my.ipredknn,
##     predict = my.predict, estimator = "632plus", est.para = control.errorest(nboot = 50),
##     ile.sasiadow = 7)
##
##   .632+ Bootstrap estimator of misclassification error
##   with 50 bootstrap replications
##
## Misclassification error:  0.2165
```

Dla 11 sąsiadów

```
##
## Call:
## errorest.data.frame(formula = diabetes ~ ., data = datas, model = my.ipredknn,
##     predict = my.predict, estimator = "632plus", est.para = control.errorest(nboot = 50),
##     ile.sasiadow = 11)
##
##   .632+ Bootstrap estimator of misclassification error
##   with 50 bootstrap replications
##
## Misclassification error:  0.2083
```

Dla 15 sąsiadów

```
##
## Call:
## errorest.data.frame(formula = diabetes ~ ., data = datas, model = my.ipredknn,
##     predict = my.predict, estimator = "632plus", est.para = control.errorest(nboot = 50),
##     ile.sasiadow = 15)
##
##   .632+ Bootstrap estimator of misclassification error
##   with 50 bootstrap replications
##
## Misclassification error:  0.2036
```



Najgorsze wyniki tzn. największy błąd klasyfikacji ponownie zauważamy dla niewielkich liczb sąsiadów.

Wnioski

Tablica 27: Błąd klasyfikacyjny algorytmu KNN w zależności od liczby sąsiadów

Liczba sąsiadów	cross-validation	bootstrap	632+
3	23.44%	25.93%	22.82%
7	21.35%	23.12%	21.65%
11	20.7%	22.47%	20.83%
15	19.66%	21.55%	20.36%

Dla porównywanych parametrów najlepszy wynik - tzn. najmnijeszy błąd klasyfikacji otrzymujemy dla liczby sąsiadów równej 15 i shcmeatu cross-validation. Największy natomiast dla liczby sąsiadów równej 3 i schematu typu bootstrap. Interesującym wnioskiem jest też, że patrząc na kolumny danych schematów odnotowujemy spadki wartości błędów klasyfikacyjnych (w badanym zakresie liczby sąsiadów 1-15, dla wartości 35-50 ten błąd ponownie być może by wzrastał) im wyższa jest liczba sąsiadów

1.8.2 Algorytm drzew klasyfikacyjnych w zależności od parametrów cp, minsplit oraz maxdepth

Algorytm drzew klasyfikacyjnych

Będę porównywać parametry cp, minsplit oraz maxdepth

Schemat cross-validation

cp=0.001 minsplit=2 maxdepth=30

```
##
## Call:
## errorest.data.frame(formula = diabetes ~ ., data = data[, -c(1,
##     3, 4, 7)], model = function(formula, data) {
##     rpart(formula = formula, data = data, method = "class", control = rpart.control(minsplit = 2,
##         cp = 0.001, maxdepth = 30))
## }, predict = my.predict, estimator = "cv", est.para = control.errorest(k = 10))
##
##   10-fold cross-validation estimator of misclassification error
##
## Misclassification error:  0.2279
```

cp=0.01 minsplit=20 maxdepth=5

```
##
## Call:
## errorest.data.frame(formula = diabetes ~ ., data = data[, -c(1,
##     3, 4, 7)], model = function(formula, data) {
##     rpart(formula = formula, data = data, method = "class", control = rpart.control(minsplit = 20,
##         cp = 0.01, maxdepth = 5))
## }, predict = my.predict, estimator = "cv", est.para = control.errorest(k = 10))
##
##   10-fold cross-validation estimator of misclassification error
##
## Misclassification error:  0.2135
```

cp=0.1 minsplit=60 maxdepth=2

```
##
## Call:
## errorest.data.frame(formula = diabetes ~ ., data = data[, -c(1,
##     3, 4, 7)], model = function(formula, data) {
##     rpart(formula = formula, data = data, method = "class", control = rpart.control(minsplit = 60,
##         cp = 0.1, maxdepth = 2))
## }, predict = my.predict, estimator = "cv", est.para = control.errorest(k = 10))
##
##   10-fold cross-validation estimator of misclassification error
##
## Misclassification error:  0.2318
```

Schemat typu bootstrap

cp=0.001 minsplit=2 maxdepth=30

```
##
## Call:
## errorest.data.frame(formula = diabetes ~ ., data = data[, -c(1,
##     3, 4, 7)], model = function(formula, data) {
##     rpart(formula = formula, data = data, method = "class", control = rpart.control(minsplit = 2,
##         cp = 0.001, maxdepth = 30))
## }, predict = my.predict, estimator = "boot", est.para = control.errorest(nboot = 50))
##
##   Bootstrap estimator of misclassification error
##   with 50 bootstrap replications
##
## Misclassification error:  0.2368
## Standard deviation: 0.0031
```

cp=0.01 minsplit=20 maxdepth=5

```
##
## Call:
## errorest.data.frame(formula = diabetes ~ ., data = data[, -c(1,
##     3, 4, 7)], model = function(formula, data) {
##     rpart(formula = formula, data = data, method = "class", control = rpart.control(minsplit = 20,
##         cp = 0.01, maxdepth = 5))
## }, predict = my.predict, estimator = "boot", est.para = control.errorest(nboot = 50))
##
##   Bootstrap estimator of misclassification error
##   with 50 bootstrap replications
##
## Misclassification error:  0.2176
## Standard deviation: 0.0027
```

cp=0.1 minsplit=60 maxdepth=2

```
##
## Call:
## errorest.data.frame(formula = diabetes ~ ., data = data[, -c(1,
##     3, 4, 7)], model = function(formula, data) {
##     rpart(formula = formula, data = data, method = "class", control = rpart.control(minsplit = 60,
```

```
##         cp = 0.1, maxdepth = 2))
## }, predict = my.predict, estimator = "boot", est.para = control.errorest(nboot = 50))
##
##   Bootstrap estimator of misclassification error
##   with 50 bootstrap replications
##
## Misclassification error:  0.2465
## Standard deviation: 0.0039
```

Schemat "632+"

cp=0.001 minsplit=2 maxdepth=30

```
##
## Call:
## errorest.data.frame(formula = diabetes ~ ., data = data[, -c(1,
##     3, 4, 7)], model = function(formula, data) {
##     rpart(formula = formula, data = data, method = "class", control = rpart.control(minsplit = 2,
##         cp = 0.001, maxdepth = 30))
## }, predict = my.predict, estimator = "632plus", est.para = control.errorest(nboot = 50))
##
##   .632+ Bootstrap estimator of misclassification error
##   with 50 bootstrap replications
##
## Misclassification error:  0.1861
```

cp=0.01 minsplit=20 maxdepth=5

```
##
## Call:
## errorest.data.frame(formula = diabetes ~ ., data = data[, -c(1,
##     3, 4, 7)], model = function(formula, data) {
##     rpart(formula = formula, data = data, method = "class", control = rpart.control(minsplit = 20,
##         cp = 0.01, maxdepth = 5))
## }, predict = my.predict, estimator = "632plus", est.para = control.errorest(nboot = 50))
##
##   .632+ Bootstrap estimator of misclassification error
##   with 50 bootstrap replications
##
## Misclassification error:  0.2054
```

cp=0.1 minsplit=60 maxdepth=2

```
##
## Call:
## errorest.data.frame(formula = diabetes ~ ., data = data[, -c(1,
##     3, 4, 7)], model = function(formula, data) {
##     rpart(formula = formula, data = data, method = "class", control = rpart.control(minsplit = 60,
##         cp = 0.1, maxdepth = 2))
## }, predict = my.predict, estimator = "632plus", est.para = control.errorest(nboot = 50))
##
##   .632+ Bootstrap estimator of misclassification error
##   with 50 bootstrap replications
##
## Misclassification error:  0.2378
```

Wnioski

Tablica 28: Błąd klasyfikacyjny algorytmu drzew klasyfikacyjnych w zależności od cp, maxdepth, minsplit

parametry	cross-validation	bootstrap	632+
cp=0.001, minsplit=2, maxdepth=30	22.79%	23.68%	18.61%
cp=0.01, minsplit=20, maxdepth=5	21.35%	21.76%	20.54%
cp=0.1, minsplit=60, maxdepth=2	23.18%	24.65%	23.78%

Dla parametrów cp=0.001, minsplit=2, maxdepth=30 najlepszy wynik otrzymujemy używając schematu 632+. Dla parametrów cp=0.01, minsplit=20, maxdepth=5 najlepszy wynik otrzymujemy używając schematu 632+. Dla parametrów cp=0.1, minsplit=60, maxdepth=2 najlepszy wynik otrzymujemy używając schematu cross-validation.

Ogólnie możemy wysunąć wnioski, że najlepszy dobór parametrów to cp=0.001, minsplit=2, maxdepth=30 w połączeniu ze schematem 632+. Najgorszym możliwym połączeniem jest schemat typu bootstrap i parametry cp=0.1, minsplit=60, maxdepth=2.

1.8.3 Algorytm naiwnnego klasyfikatora bayesowskiego w zależności od laplace i

Algorytm naiwnnego klasyfikatora bayesowskiego

Będę porównywać parametry laplace, usekernel, adjust

Schemat cross-validation

laplace=0

usekernel=FALSE

adjust=1

```
##
## Call:
## errorest.data.frame(formula = diabetes ~ ., data = data[, -c(1,
##     3, 4, 7)], model = function(formula, data) {
##     naiveBayes(formula = formula, data = data, laplace = 0, usekernel = FALSE,
##         adjust = 1)
## }, predict = my.predict, estimator = "cv", est.para = control.errorest(k = 10))
##
##   10-fold cross-validation estimator of misclassification error
##
## Misclassification error:  0.2357
```

laplace=1

usekernel=FALSE

adjust=1

```
##
## Call:
## errorest.data.frame(formula = diabetes ~ ., data = data[, -c(1,
##     3, 4, 7)], model = function(formula, data) {
```

```
##     naiveBayes(formula = formula, data = data, laplace = 1, usekernel = FALSE,
##         adjust = 1)
## }, predict = my.predict, estimator = "cv", est.para = control.errorest(k = 10))
##
##   10-fold cross-validation estimator of misclassification error
##
## Misclassification error:  0.2357
```

laplace=1

usekernel=TRUE

adjust=1.5

```
##
## Call:
## errorest.data.frame(formula = diabetes ~ ., data = data[, -c(1,
##     3, 4, 7)], model = function(formula, data) {
##     naiveBayes(formula = formula, data = data, laplace = 1, usekernel = TRUE,
##         adjust = 1.5)
## }, predict = my.predict, estimator = "cv", est.para = control.errorest(k = 10))
##
##   10-fold cross-validation estimator of misclassification error
##
## Misclassification error:  0.2318
```

Schemat typu bootstrap

laplace=0

usekernel=FALSE

adjust=1

```
##
## Call:
## errorest.data.frame(formula = diabetes ~ ., data = data[, -c(1,
##     3, 4, 7)], model = function(formula, data) {
##     naiveBayes(formula = formula, data = data, laplace = 0, usekernel = FALSE,
##         adjust = 1)
## }, predict = my.predict, estimator = "boot", est.para = control.errorest(nboot = 50))
##
##   Bootstrap estimator of misclassification error
##   with 50 bootstrap replications
##
## Misclassification error:  0.2383
## Standard deviation: 0.0012
```

laplace=1

usekernel=FALSE

adjust=1

```
##
## Call:
```

```
## errorest.data.frame(formula = diabetes ~ ., data = data[, -c(1,
##     3, 4, 7)], model = function(formula, data) {
##     naiveBayes(formula = formula, data = data, laplace = 1, usekernel = FALSE,
##         adjust = 1)
## }, predict = my.predict, estimator = "boot", est.para = control.errorest(nboot = 50))
##
##   Bootstrap estimator of misclassification error
##   with 50 bootstrap replications
##
## Misclassification error:  0.2357
## Standard deviation: 0.0013
```

laplace=1

usekernel=TRUE

adjust=1.5

```
##
## Call:
## errorest.data.frame(formula = diabetes ~ ., data = data[, -c(1,
##     3, 4, 7)], model = function(formula, data) {
##     naiveBayes(formula = formula, data = data, laplace = 1, usekernel = TRUE,
##         adjust = 1.5)
## }, predict = my.predict, estimator = "boot", est.para = control.errorest(nboot = 50))
##
##   Bootstrap estimator of misclassification error
##   with 50 bootstrap replications
##
## Misclassification error:  0.2358
## Standard deviation: 0.001
```

Schemat "632+"

laplace=0

usekernel=FALSE

adjust=1

```
##
## Call:
## errorest.data.frame(formula = diabetes ~ ., data = data[, -c(1,
##     3, 4, 7)], model = function(formula, data) {
##     naiveBayes(formula = formula, data = data, laplace = 0, usekernel = FALSE,
##         adjust = 1)
## }, predict = my.predict, estimator = "632plus", est.para = control.errorest(nboot = 50))
##
##   .632+ Bootstrap estimator of misclassification error
##   with 50 bootstrap replications
##
## Misclassification error:  0.2333
```

laplace=1

usekernel=FALSE

adjust=1

```
##
## Call:
## errorest.data.frame(formula = diabetes ~ ., data = data[, -c(1,
##     3, 4, 7)], model = function(formula, data) {
##     naiveBayes(formula = formula, data = data, laplace = 1, usekernel = FALSE,
##         adjust = 1)
## }, predict = my.predict, estimator = "632plus", est.para = control.errorest(nboot = 50))
##
##   .632+ Bootstrap estimator of misclassification error
##   with 50 bootstrap replications
##
## Misclassification error:  0.2344
```

laplace=1

usekernel=TRUE

adjust=1.5

```
##
## Call:
## errorest.data.frame(formula = diabetes ~ ., data = data[, -c(1,
##     3, 4, 7)], model = function(formula, data) {
##     naiveBayes(formula = formula, data = data, laplace = 1, usekernel = TRUE,
##         adjust = 1.5)
## }, predict = my.predict, estimator = "632plus", est.para = control.errorest(nboot = 50))
##
##   .632+ Bootstrap estimator of misclassification error
##   with 50 bootstrap replications
##
## Misclassification error:  0.2329
```

Wnioski

Tablica 29: Błąd klasyfikacyjny algorytmu naiwnego klasyfikatora bayesowskiego zależności od laplace, usekernel, adjust

parametry	cross-validation	bootstrap	632+
laplace=0 ,usekernel=FALSE, adjust=1	23.57%	23.83%	23.33%
laplace=1 ,usekernel=FALSE, adjust=1	23.57%	23.57%	23.44%
laplace=1 ,usekernel=TRUE, adjust=1.5	23.18%	23.58%	23.29%

Dla parametrów laplace=0 ,usekernel=FALSE, adjust=1 najlepszy wynik otrzymujemy używając schematu 632+. Dla parametrów laplace=1 ,usekernel=FALSE, adjust=1, maxdepth=5 najlepszy wynik otrzymujemy używając schematu 632+. Warto podkreślić, że dla metody cross-validation i bootstrap przy zadanych parametrach otrzymujemy identyczne miejsca co do zaokrągleń do dwóch miejsc po przecinku. Dla parametrów laplace=1 ,usekernel=TRUE, adjust=1.5 najlepszy wynik otrzymujemy używając schematu cross-validation.

Ciekawą oberwacją jest, że niezależnie odoboru parametrów oraz schematów otrzymujemy wyniki z zakresu wartości od 23.18%-23.83%. Z czego 3 wyniki są identyczne co do 2 cyfr znaczących po przecinku:

laplace=0 ,usekernel=FALSE, adjust=1 cross-validation laplace=1 ,usekernel=FALSE, adjust=1 cross-validation laplace=1 ,usekernel=FALSE, adjust=1 bootstrap

1.8.4 Wnioski

1.9 f) Końcowe wnioski - podsumowanie analizy porównawczej metod klasyfikacji

- Algorytm KNN:

najlepszym parametrem k (liczby sąsiadów) okazała się liczba 15 dla podzbioru nie zawierającego triceps, pedigree, pregnant, pressure.

- Algorytm drzew klasyfikacyjnych:

najlepszy wynik otrzymaliścy przy parametrach domyślnych i testując cały zbiór danych na schemacie cross-validation.

- Naiwyny algorytm bayesowski:

najlepszym doborem parametrów okazało się laplace=1 ,usekernel=TRUE, adjust=1.5 oraz schemat cross-validation dla podzbioru nie zawierającego triceps, pedigree, pregnant, pressure.

- Ogólniej:

Najlepiej radził sobie schemat 632+ oraz cross-validation niezależnie od dobranego algorytmu.

Najmniejszą zmienność jeśli chodzi o wyniki otrzymaliśmy testując algorytm naiwnego bayesa bez względu na schemat. Nie były to najlepsze z możliwych wyników - błąd klasyfikacyjny był dość wysoki w porównaiu z innymi wartościami.

Najlepiej wypadł algorytm drzew klasyfikacyjnych na całym zbiorze, dla każdego schematu. Na zbiorze uczącym zwrócił najmniejszy błąd klasyfikacyjny. Na zbiorze nie zawierającym zmiennych triceps, pedigree, pregnant, pressure również osiągnał możliwie najmniejsze błedy dla każdego z schematów: cross-validation, bootstrap, 632+.