**Wstęp**

Wybrana przeze mnie baza danych to wyniki studentów ze wstępnego egzaminu.

Zawiera ona 9 kolumn:

* gender - płeć badanego
* race/ethnicity - pochodzenie etniczne
* parental level of education(klasa) - wykształcenie rodziców
* lunch - nie bardzo wiem, więc klasa została usunięta
* test preparation course - czy student podszedł do kursu przygotowawczego
* math score - wynik z matematyki
* writing score - wynik z pisania
* reading score - wynik z czytania

Celem projektu będzie sprawdzenie czy za pomocą względnie losowych danych można przewidywać odpowiednie klasy. W tym wypadku będę starał się nauczyć mój model przewidywać czy rodzice studentów mają wyższe czy niższe wykształcenie.

**Przetwarzanie i obróbka danych**

Zamieniam edukację na dwie klasy wyzsza/srednia

for (i in 1:nrow(db[3])) {

if (db[3][i, ] == uniqueEducationValues[1] ||

db[3][i, ] == uniqueEducationValues[2] ||

db[3][i, ] == uniqueEducationValues[3] ||

db[3][i, ] == uniqueEducationValues[4]) {

db[3][i, ] = "higher"

} else {

db[3][i, ] = "high"

}

}

Zamieniam stringi na number:

db$gender <-

sapply(as.character(db$gender),

switch,

'male' = 0,

'female' = 1)

db$`race/ethnicity` <-

sapply(

as.character(db$`race/ethnicity`),

switch,

'group A' = 0,

'group B' = 1,

'group C' = 2,

'group D' = 3,

'group E' = 4

)

db$`test preparation course` <-

sapply(

as.character(db$`test preparation course`),

switch,

'none' = 0,

'completed' = 1

)

Usuwam kolumnę lunch, bo nie jest mi ona potrzebna do moich badań

db$`lunch` <- NULL

Zmieniam nazwę kolumny, bo uważam, że jest łatwiejsza do zrozumienia i krótsza

colnames(db)[which(names(db) == "parental level of education")] <-

"education"

Zmieniam kolejność kolumn, tak aby edukacja rodziców była na końcu

db <- db[, c(1, 2, 4, 5, 6, 7, 3)]

db <- as.data.frame(db)

**Klasyfikatory**

Zastosowałem cztery klasyfikatory:

1. **Knn**

knn.3 <- knn(db.train,

db.test,

cl = train\_label,

k = 3,

prob = FALSE)

knn.predicted <- knn.3

1. **Naive Bayes**

nbayes <- naiveBayes(db.train, as.factor(train\_label))

nbayes.predicted <- predict(nbayes, db.test)

1. **Conditional Tree**

myFormula <-

factor(train\_label) ~ gender + race.ethnicity + test.preparation.course + math.score + reading.score + writing.score

db\_ctree <- ctree(myFormula, data = db.train)

tree.predicted <-

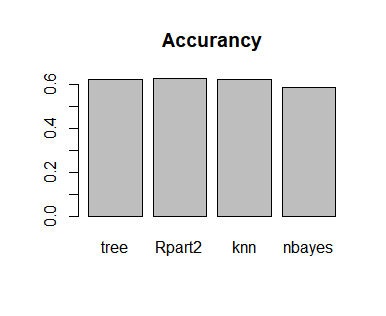
predict(db\_ctree, newdata = db.test, type = "response")

1. **Rpart2**

model\_rpart2 <- train(db.train, train\_label, method = 'rpart2')

cart.predictions <- predict(object = model\_rpart2, db.test)

Sprawdzalność klasyfikatorów:



Sprawdzalność klasyfikatorów oscyluje w granicach 60%.

Macierz błędu:

True-Positive (TP - prawdziwie pozytywna): Przewidziano klasę high w rzeczywistości też jest high.

True-Negative (TN - prawdziwie negatywna): Przewidziano klasę negatywną, w tym wypadku higher został sklasyfikowany jako higher.

False-Positive (FP - fałszywie pozytywna): Przewidziano klasę higher, a realna wartość była high.

False-Negative (FN - fałszywie negatywna): Przewidziano klasę high,a realna wartość była higher.

SE = TPR 1 – SE = FNR

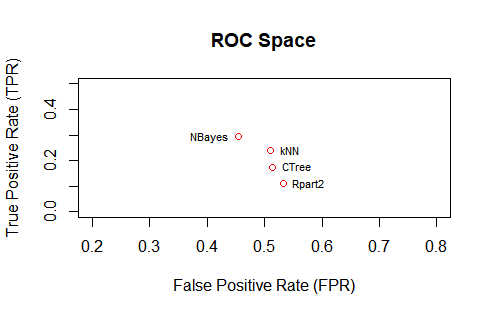
SP = TNR 1‐SP = FPR

Błąd I rodzaju to sytuacja gdy przewidujemy klasę higher gdy w rzeczywistości klasa powinna być high.

Błąd II rodzaju to sytuacja gdy przewidujemy klasę high gdy w rzeczywistości klasa powinna być higher.

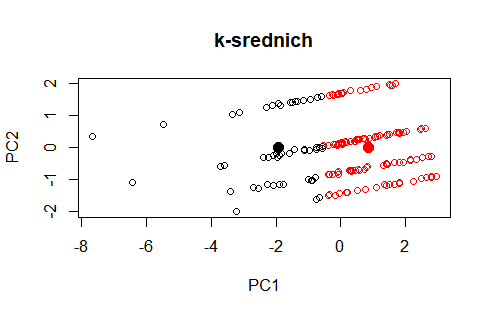
Im więcej błędów I rodzaju tym większe jest FPR. Natomiast im większy błąd II rodzaju tym większe jest TPR.

W moim projekcie nie ma znaczenia, który błąd częściej wystąpi(Nie ma różnicy czy sklasyfikujemy rodziców jako ludzie z wyższym czy niższym wykształceniem). W moim badaniu ważniejsza jest sprawdzalność.



Najbliżej idealnego klasyfikatora jest Nbayes, poniewaz jest najbliżej pkt(0,1)

**Grupowanie**



kod:

db.log <- log(db\_normalized)

db.log <- db.log[is.finite(rowSums(db.log)),]

db.log$gender <- NULL

db.log$test.preparation.course <- NULL

db.scale <- scale(db.log, center=TRUE)

db.pca <- prcomp(db.scale)

db.final <- predict(db.pca)

db.final <- db.final[,1:4]

db.kmeans <- kmeans(db.final, 2)

table(db.kmeans[["cluster"]])

plot(db.final, col = db.kmeans[["cluster"]],

main="k-srednich")

points(db.kmeans[["centers"]], col = 1:2, pch = 16, cex=1.5)

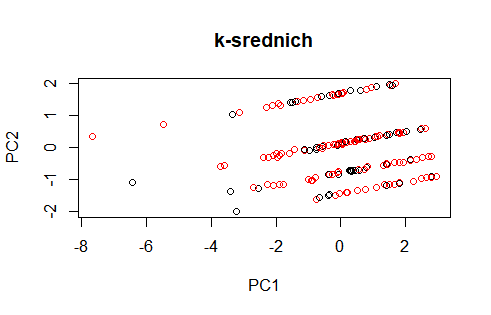
Algorytm dobrze wydzielił dwie grupy. Centroidy zostały wyliczone w punktach:

PC1 PC2 PC3 PC4

1 0.8138903 0.01571815 -0.02633775 -0.02202207

2 -2.0430307 -0.03945577 0.06611313 0.05527990

Klasy rzeczywiste pokrywają się w pewnym stopniu z klasami wydzielonnymi



**Asocjacje**

Kod szukający :

rules <- apriori(raDB, parameter = list(support = 0.1,conf = 0.7,target = "rules"))

Można zmieniać support i conf.

Ciekawe Asocjacje:

Support: 0.1

Confidence: 0.71

{race/ethnicity=4} => {education=higher}

Support: 0.4

Confidence: 0.62

{test preparation course=0} => {education=higher}

Support: 0.33

Confidence: 0.52

{test preparation course=0} => {gender=1}

Wszystkie wypisane asocjację są bardzo ciekawe, jednak w pierwszej nasz support wynosi tylko 0.1 więc występowalność jest mała.

**Wnioski**

Badanie przeszło moje oczekiwania. Względnie losowe dane pozwalają nam z 64% skutecznością zaklasyfikować elementy. W moim przypadku - pochodzenie etniczne, wyniki z matematyki, pisania, czytania, płeć oraz to czy osoba przystąpiła do kursu przygotowawczego pozwala przewidzieć czy rodzice tej osoby mają wyższe czy średnie wykształcenie, ze sprawdzalnością 64%. Najlepiej sprawdziły się klasyfikatory knn oraz cTree. Oprócz tego, że zaskoczyła mnie względnie wysoka sprawdzalność, to reguły asocjacyjne, które można znaleźć w sprawozdaniu też są bardzo ciekawe. Chociażby to, że często dzieci ludzi z wyższym wykształceniem nie podchodzili do kursu przygotowawczego.