Projekt 1

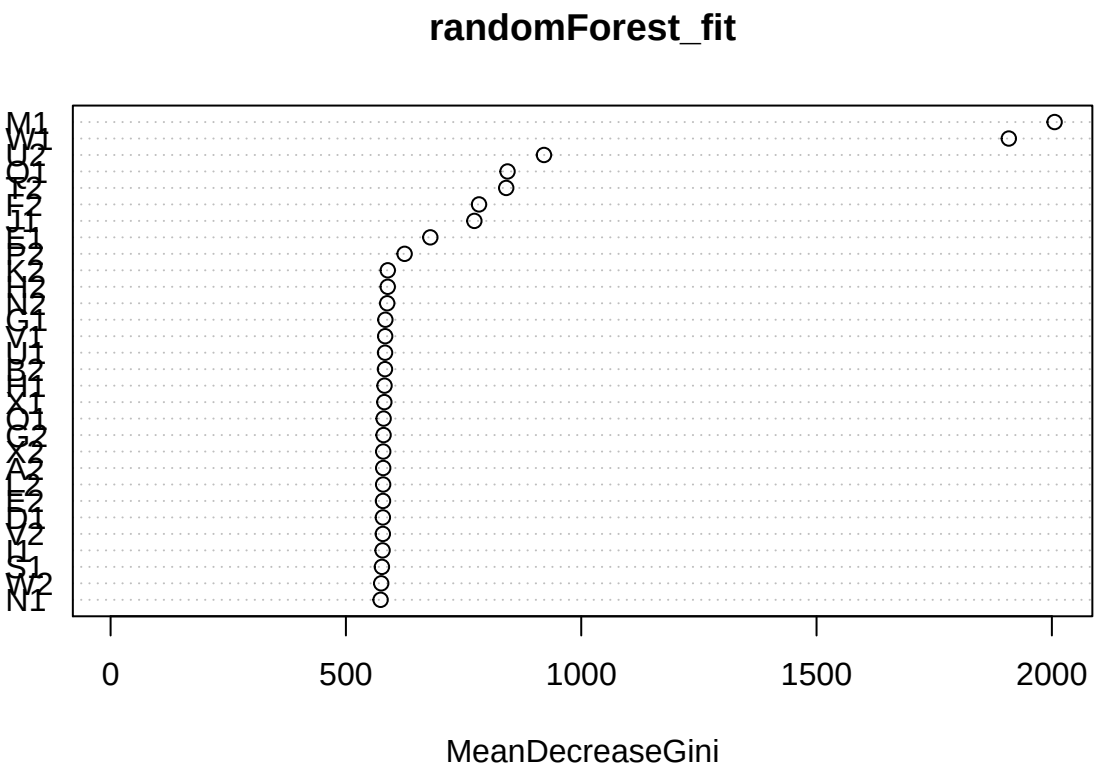
Kamil Romaszko

14 października 2017

Analiza zmiennych

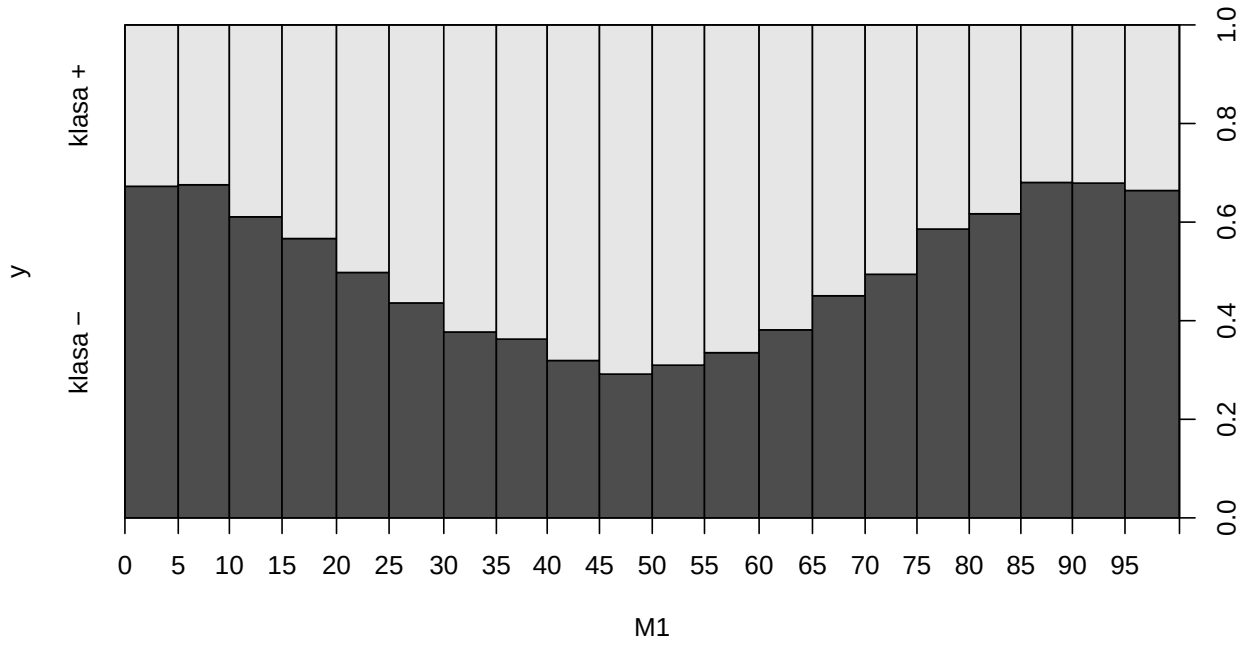
Analizowane dane zawierają 50 zmiennych niezależnych oraz 1 zmienną zależną od reszty - informację o przynależności do klasy "+" lub "-". Co ważne, klasy są zrównoważone. Moim pierwszym krokiem była selekcja najbadziej istotnych zmiennych objaśniających. W tym celu użyłem metody drzew losowych oraz fuknicji *importance* wywołanej na wygenerowanym modelu. Zaletą użytej metody jest to, że jest ona w stanie wykryć nieliniowe zależności pomiędzy zmiennymi.

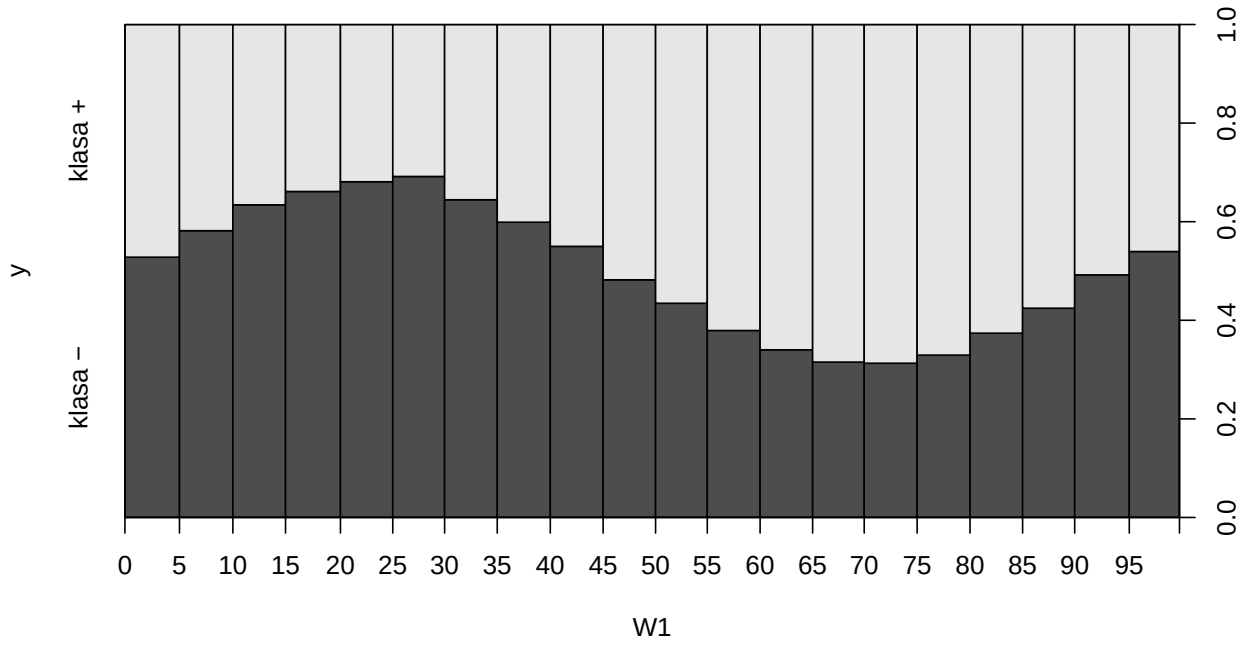
```
randomForest_fit <- randomForest(y~., data=data_train)
varImpPlot(randomForest_fit)
```




```
importance_fit <- importance(randomForest_fit)
```

Powyższy wykres przedstawia istotność zmiennych. Na jego podstawie możemy stwierdzić, że tylko kilka pierwszych zmiennych jest istotna. Możemy zatem uprościć dane - ja do dalszych rozważać wybrałem 10 najbardziej istotnych zmiennych.





Powyższy wykres przedstawia zależność zmiennej objaśnienaj od najistotniejszej zmiennych "*M1*" i "*W1*". Widać, że zależność nie jest liniowa.

Testowanie modeli

Po analizie danych przetestowałem, które metody najlepiej się sprawdzają. Aby przetestować wybrane metody, dane podzieliłem na zbiór treningowy i testowy w stosunku 4:1.

Każdy model wyznacza prawdopodobieństwo przynależności do klasy "+". Do oceny modelu, dokonujemy predykcji na zbiorze testowym, a następnie wybieramy 20% najabrdziej prawdopodobnych obserwacji i mierzymy procent obserwacji, które faktycznie należą do klasy "+". Im więcej poprawnie przydzielonych obserwacji tym lepiej.

Model xgboost

Wybrana przeze mnie metoda klasyfikacji to eXtreme Gradient Boosting. Polega ona budowaniu sekwencji drzew optymalizowanych na klasyfikacje przypadków z którymi nie radziły sobie wcześniejsze drzewa. Metoda ta, tak jak inne modele oparte na drzewach, dobrze nadaje się do wykrywania nieliniowych zależności, dlatego postanowiłej jej użyć dla zadanego problemu.

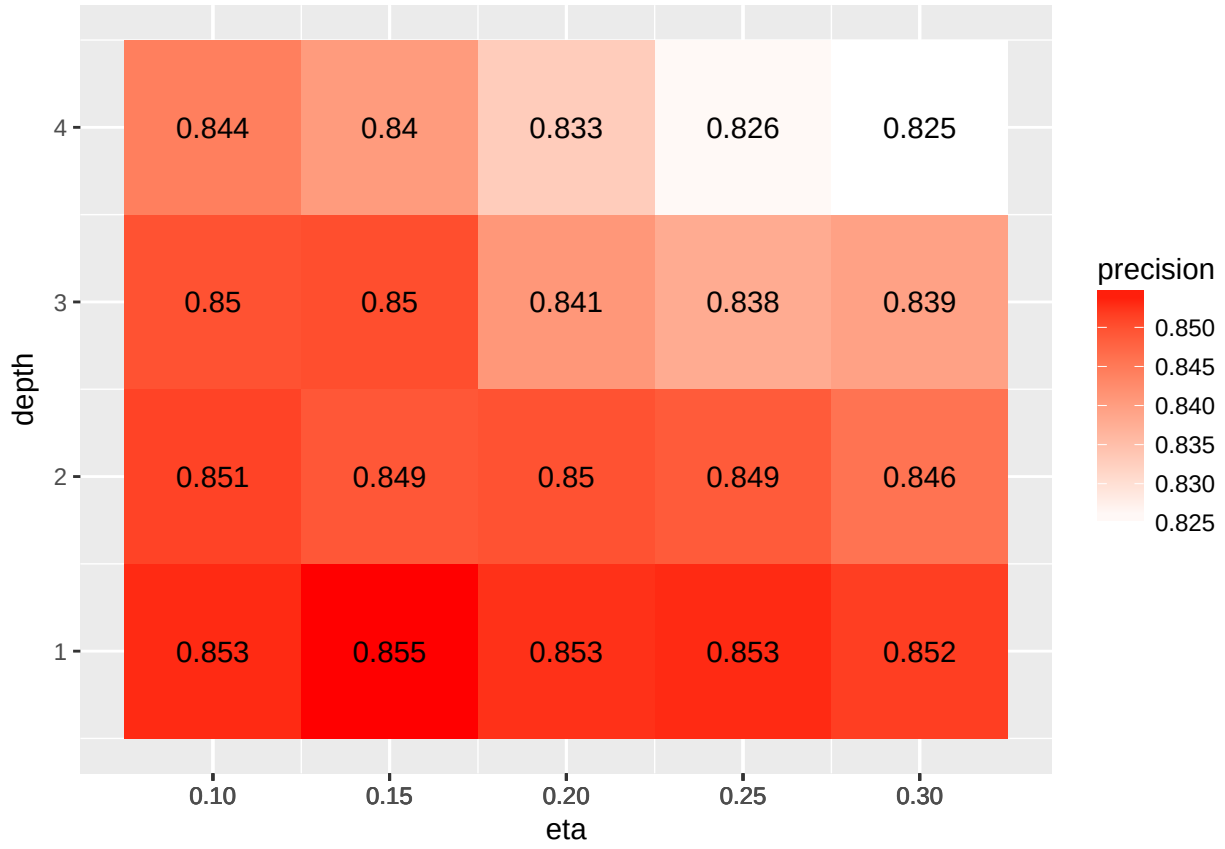
Funkcja budująca model posiada jeden wymagany parametr *nround* który określa maksymalną liczbę iteracji. W moich testach przyjąłem *nround=500* co daje dostatecznie złożony model, a jednocześnie sprawia, że wyliczany jest on stosunkowo szybko.

Metoda przyjmuje także opcjonalnie parametry od których zależy dokładność modelu. Postanowiłem zbadać jak wpływają one na dokładność.

max_depth - określa maksymalną głębokość drzewa. Im większa głębokość, tym bardziej złożony będzie model.

eta - określa krzywą uczenia. Stosowany, żeby zapobiegać przeuczeniu modelu. Im mniejszy, tym model będzie bardziej uogólniony.

Dobór parametrów



Dla wybranych wartości parametrów wygenerowałem modele oraz wyznaczyłem ich dokładność. Powyższa heatmapa pokazuje zależność jakości modelu od przyjętych parametrów. Na jego podstawie wybrałem *max_depth=1* oraz *eta=0.15*. Dla tak wybranych parametrów model osiągnął wynik 85.5% co daje najlepszy z uzyskanych wyników.

Podsumowanie

Dla tak wybranych parametrów dokonałem predykcji dla zbioru testowego.

```
boost_model <- xgboost(data=matrix_train[,-1], label=matrix_train[,1],
                       objective="binary:logistic", nrounds = 500,
                       max_depth=1, eta = 0.15, verbose=0)
score <- predict(boost_model, matrix_test)
ordered <- order(score, decreasing = TRUE)
result <- cbind(data_test, score)
write.table(result, file = "kamil_romaszko.txt", sep=";")
```