Physico-Chemical Institute of NAS of Ukraine

Machine learning methods, 25 July 2014

Decision trees, Random Forest, Gradient Boosting

Dr. Pavel Polishchuk

pavel_polishchuk@ukr.net

Physico-Chemical Institute of National Academy of Sciences of Ukraine

Decision trees

Деревья решений: алгоритм

Задача

Определить пол посетителя магазина по двум параметрам – времени посещения магазина и длительности пребывания.

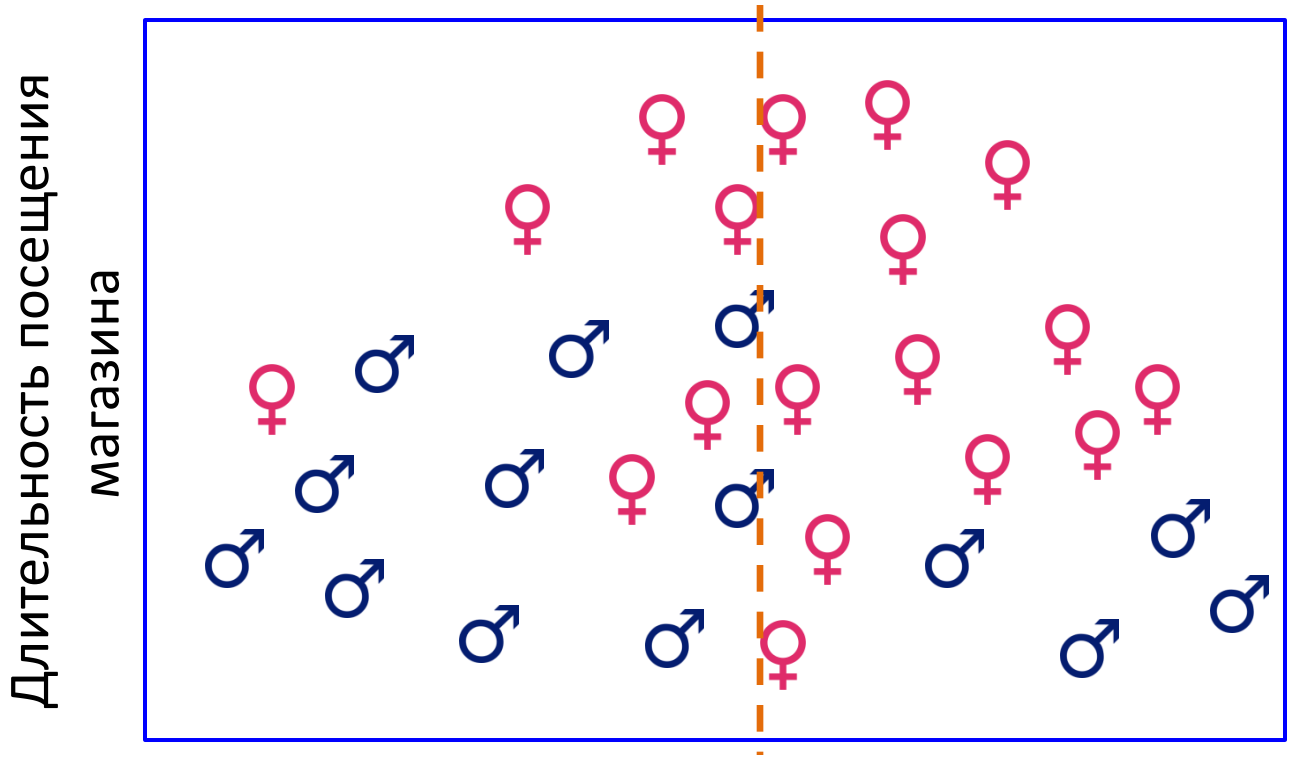
Деревья решений: алгоритм

Датасет

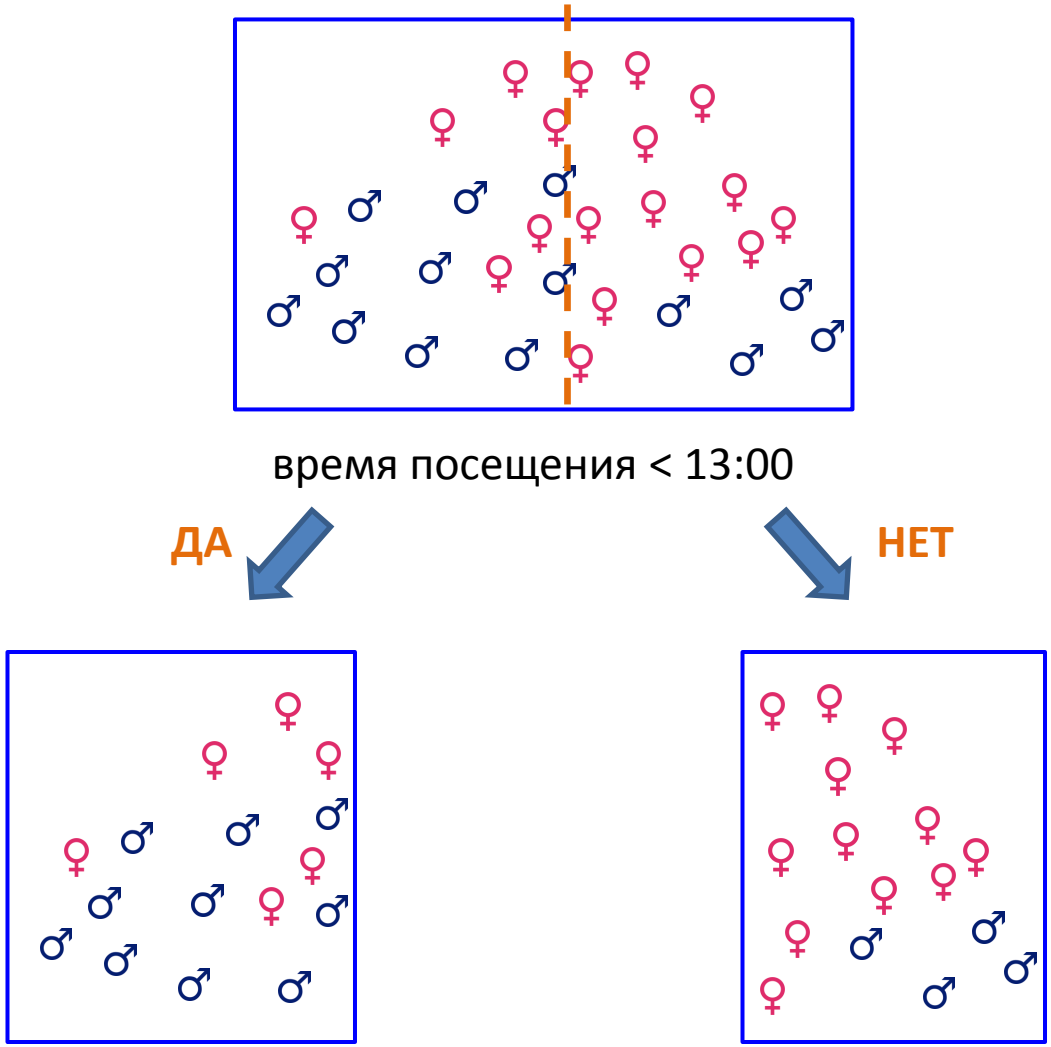
Пол	Время посещения магазина	Длительность посещения магазина
М	00:01	2 мин.
М	00:05	11 мин.
М	01:32	20 мин.
Ж	02:45	8 мин.
Ж	06:12	16 мин.
Ж	08:20	32 мин.
М	08:25	2 мин.
Ж	09:10	45 мин.
…	…	…
Ж	23:20	26 мин.

Деревья решений: алгоритм

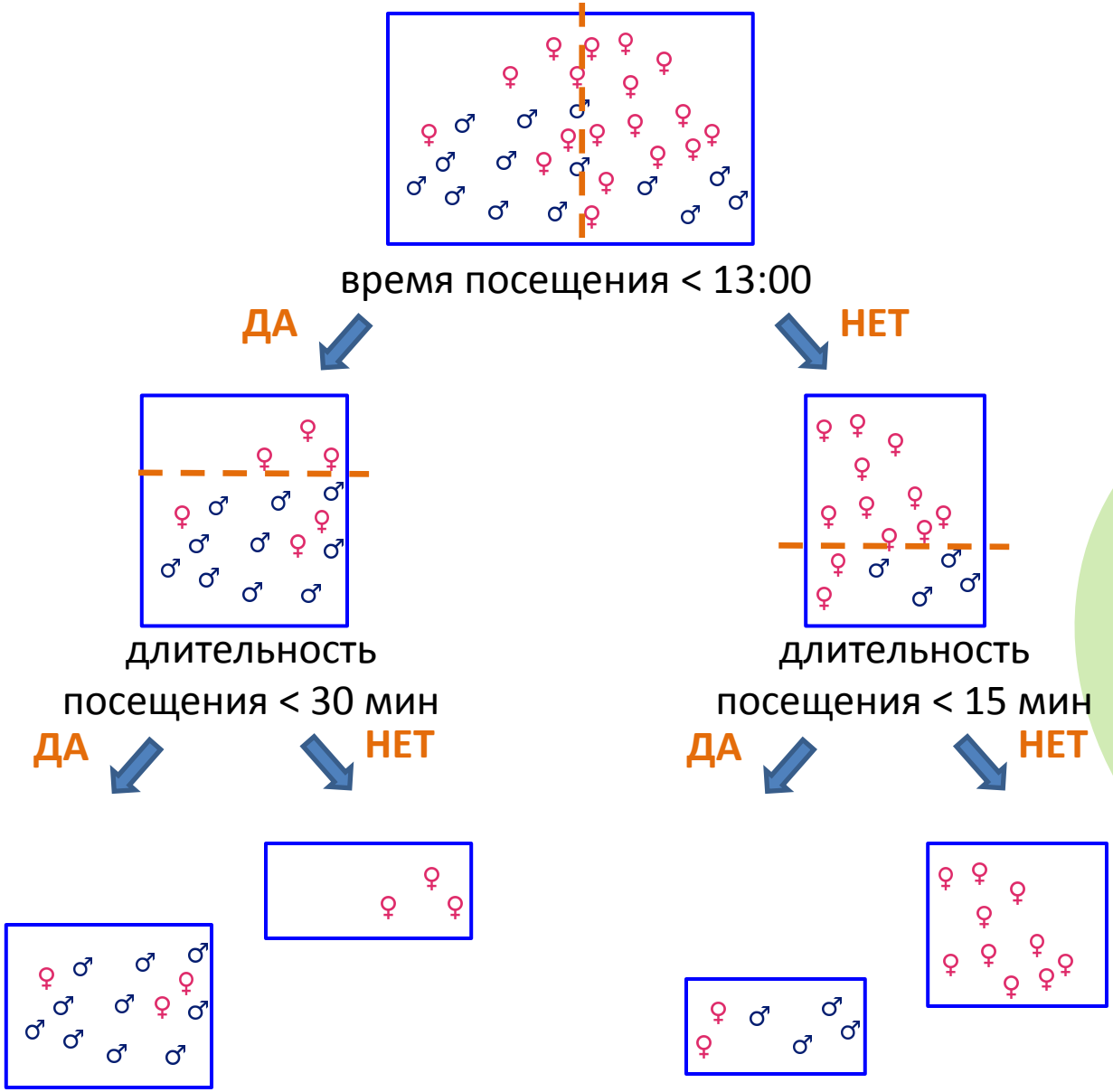
Датасет (графическое представление)



Деревья решений: алгоритм

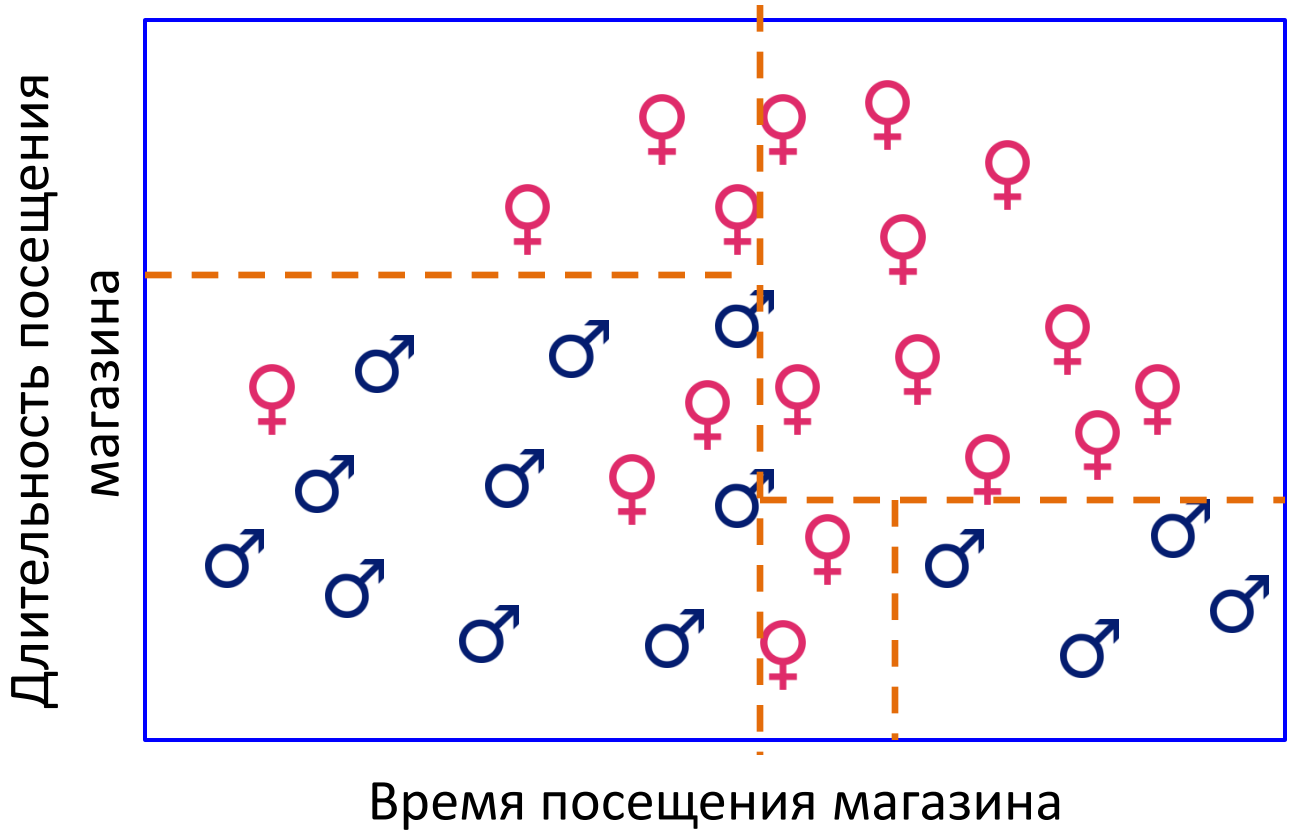


Деревья решений: алгоритм



ЕСЛИ время посещения >= 13:00 И длительность посещения >= 15 минут , ТО посетитель женщина

Деревья решений: алгоритм



Методы построения деревьев решений

1. Chi-squared Automatic Interaction Detection (CHAID) - Kass. G.V. (1980)
2. Classification and regression trees (CART) - Breiman L., Friedman J. H., Olshen R. A., & Stone C. J. (1984)
3. ID3 - Quinlan, J. R. (1986)
4. C4.5 - Quinlan, J. R. (1993)

Classification and regression trees (CART)

CART: classification

Классификационные задачи

Gini impurity measure

$$Gini = 1 - \sum_{i=1}^{m} p_i^2 = 1 - \sum_{i=1}^{m} \left(\frac{N_i}{N} \right)^2$$

p_i – доля объектов i-го класса
m – общее число классов
N_i – число объектов i-го класса
N – общее число объектов

Оптимизирующая функция

$$Gini_{split} = \frac{N_{left}}{N} Gini_{left} + \frac{N_{right}}{N} Gini_{right} \to \min$$

N_{left}, N_{right} – число объектов в левом и правом дочернем узле
N – общее количество объектов в родительском узле

CART: classification

```
gini <- function(classes) {
  tb <- table(classes)
  1 - sum(tb ^ 2 / length(classes) ^ 2)
}

gini_split <- function(x, y, threshold) {
  gini_left <- gini(y[x <= threshold])
  gini_right <- gini(y[x > threshold])
  (sum(x <= threshold) * gini_left +
     sum(x > threshold) * gini_right) /
length(x)
}
```

CART: classification

Y labels	1	0	0	0	1	1	0	1	1	1	1
Variable value	3	3	3	3	4	4	4	5	5	5	5

```
df <- data.frame(x=c(3,3,3,3,4,4,4,5,5,5,5),
                 y=c(1,0,0,0,1,1,0,1,1,1,1))

> gini_split(df$x, df$y, 3.5)
[1] 0.2922078

> gini_split(df$x, df$y, 4.5)
[1] 0.3116883
```

optimal threshold = 3.5

CART: classification

Y labels	1	0	0	0	1	1	0	1	1	1	1
Variable value	3	3	3	3	4	4	4	5	5	5	5

Variable <= 3.5

Variable > 3.5

1	0	0	0
3	3	3	3

1	1	0	1	1	1	1
4	4	4	5	5	5	5

CART: classification

```
G_opt = max_int
Var_opt = 0
Threshold_opt = 0

for i = each variable
    for j = each possible threshold
        G = Gini_split(var_i, y, threshold_j)
        if G < G_opt then
            G_opt = G
            Var_opt = var_i
            Threshold_opt = threshold_j
```

CART: regression

Регрессионные задачи

Среднеквадратичное отклонение

$$MSE = \frac{1}{N}\sum_{i=1}^{N}(x_i - \bar{x})^2$$

N – количество объектов

x_i – наблюдаемое значение для i-го объекта

$\bar{x}_i$ – среднее значение для всех объектов

Оптимизирующая функция

$$MSE_{split} = MSE_L + MSE_R \rightarrow \min$$

минимизация суммы среднеквадратичных ошибок для левого и правого дочернего узлов

CART: regression

```
mse <- function(y) {
  sum((y - mean(y)) ^ 2) / length(y)
}

mse_split <- function(x, y, threshold) {
  mse_left <- mse(y[x <= threshold])
  mse_right <- mse(y[x > threshold])
  mse_left + mse_right
}
```

CART: regression

Y values	1	0	0	0	1	1	0	1	1	1	1
Variable value	3	3	3	3	4	4	4	5	5	5	5

```
df <- data.frame(x=c(3,3,3,3,4,4,4,5,5,5,5),
                 y=c(1,0,0,0,1,1,0,1,1,1,1))

> mse_split(df$x, df$y, 3.5)
[1] 0.309949

> mse_split(df$x, df$y, 4.5)
[1] 0.244898
```

optimal threshold = 4.5

CART: regression

Y values	1	0	0	0	1	1	0	1	1	1	1
Variable value	3	3	3	3	4	4	4	5	5	5	5

Variable <= 4.5

Variable > 4.5

1	0	0	0	1	1	0
3	3	3	3	4	4	4

1	1	1	1
5	5	5	5

ID3

ID3

Решает только классификационные задачи

Переменные могут быть только номинальными

Может работать с переменными с частично отсутствующими значениями

ID3

Классификационные задачи

Энтропия

$$E = -\sum_{i=1}^{m} p_i \log_2 p_i = -\sum_{i=1}^{m} \frac{N_i}{N} \log_2 \frac{N_i}{N}$$

p_i – доля объектов i-го класса

m – общее число классов

N_i –число объектов i-го класса

N – общее число объектов

Оптимизирующая функция

$$E_{split} = \sum_{j=1}^{M} \frac{N_j}{N} E(A_j) \to \min$$

M – число уникальных значений переменной A

N_j – число объектов, для которых переменная A принимает j-е значение

N – общее число объектов

A_j – объекты, для которых переменная A принимает j-тое значение

ID3

```
entropy <- function(classes) {
  tb <- table(classes)
  n <- length(classes)
  -sum(sapply(tb, function(i) i / n * log2(i/n)))
}

entropy_split <- function(x, y) {
  lst <- split(y, x)
  e <- sapply(lst, entropy)
  n <- sapply(lst, length)
  sum(e / n)
}
```

ID3

Y labels	1	0	0	0	1	1	0	1	1	1	1
Variable 1 value	3	3	3	3	4	4	4	5	5	5	5

Y labels	1	1	1	0	1	0	0	0	1	1	1
Variable 2 value	2	2	2	2	2	3	3	3	3	5	5

```
df <- data.frame(x1=c(3,3,3,3,4,4,4,5,5,5,5),
                 x2=c(3,2,3,3,2,2,3,5,2,5,5),
                 y=c(1,0,0,0,1,1,0,1,1,1,1))

> entropy_split(df$x1, df$y)
[1] 0.5089181
> entropy_split(df$x2, df$y)
[1] 0.3472052
```

optimal variable = 2

ID3

Y labels	1	1	1	0	1	0	0	0	1	1	1
Variable 2 value	2	2	2	2	2	3	3	3	3	5	5

Variable 2 = 2

1	1	1	0	1
2	2	2	2	2

Variable 2 = 3

0	0	0	1
3	3	3	3

Variable 2 = 5

1	1
5	5

C4.5

C4.5

Решает только классификационные задачи

Численные переменные приводятся к ранговой шкале

Может работать с переменными с частично отсутствующими значениями

Классификационные задачи

Оптимизирующая функция

$$Gain\ ratio_{split} = \frac{E(y) - E_{split}(x, y)}{E(x)} \to \max$$

E(y) – значение энтропии для исходного набора значений y

E(x) – значение энтропии для исходного набора значений x

E_{split}(x,y) – значение энтропии при разделении набора значений y по значениям переменной x.

C4.5

```
gain_ratio <- function(x, y) {
  (entropy(y) - entropy_split(x,y)) / entropy(x)
}
```

Y labels	1	0	0	0	1	1	0	1	1	1	1
Variable 1 value	3	3	3	3	4	4	4	5	5	5	5

Y labels	1	1	1	0	1	0	0	0	1	1	1
Variable 2 value	2	2	2	2	2	3	3	3	3	5	5

```
> gain_ratio(df$x1, df$y)
[1] 0.2777156
> gain_ratio(df$x2, df$y)
[1] 0.4003262
```

optimal variable = 2

C4.5

Y labels	1	1	1	0	1	0	0	0	1	1	1
Variable 2 value	2	2	2	2	2	3	3	3	3	5	5

Variable 2 = 2

1	1	1	0	1
2	2	2	2	2

Variable 2 = 3

0	0	0	1
3	3	3	3

Variable 2 = 5

1	1
5	5

Деревья решений: преимущества и недостатки

Преимущества:

+ нелинейный метод
+ простые в понимании и интерпретации модели
+ непараметрический метод – не зависит от вида распределения рассматриваемого свойства
+ не требует отбора переменных – алгоритм сам выбирает значимые переменные
+ способны работать с численными и номинальными переменными

Недостатки:

– неустойчивость моделей, при незначительном изменении входных данных модель может изменится кардинальным образом
– переобучение моделей (можно бороться, используя обрезку дерева – pruning, или предварительную остановку)
– длительный процесс обучения (необходимость перебора всех переменных в каждом узле)
– невозможность экстраполяции

Random Forest

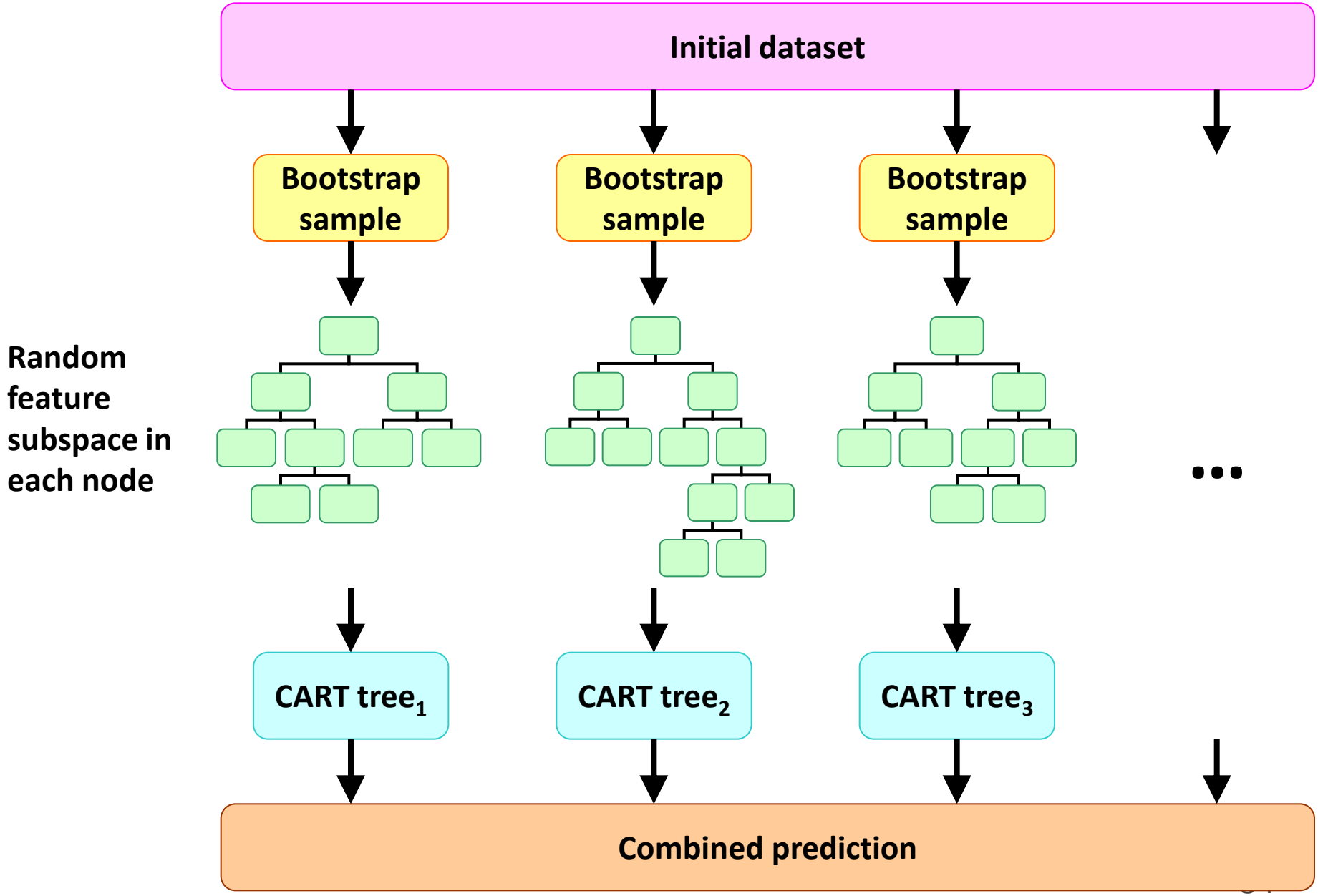
Random Forest

Random Forest – ансамбль деревьев решений построенных по особым правилам (L. Breiman, 2001).

Этапы обучения модели:

1. Выбор из исходного обучающего набора объектов случайного поднабора с повторами, число объектов равно числу объектов в исходном наборе (бутстреп). Этот поднабор используется для обучения отдельно взятого дерева решений.
2. Построение дерева решений алгоритмом CART, но на каждом шаге рассматриваются не все переменные, а только небольшая случайно выбранная их часть (mtry – настроечный параметр модели).
3. Строится N деревьев до максимальной глубины.

Random Forest



Random Forest

Рекомендации.

Поскольку модели устойчивы к переобучению, то добавление деревьев в ансамбль не снижает прогнозирующую способность. В большинстве случаев 500 деревьев достаточно.

Выбор числа переменных, которые рассматриваются при построении каждого узла дерева, в наибольшей степени влияет на конечную модель. Эмпирически Бриманом было предложено что для регрессионных задач оптимальное число переменных $N/3$, для классификационных $\mathrm{sqrt}(N)$.

Важность переменных

$$Importance = Error_{OOB}^{(i)} - Error_{OOB}$$

$Error_{OOB}$ – ошибка прогноза модели для OOB выборки
$Error^{(i)}_{OOB}$ – ошибка прогноза модели для OOB выборки, при условии предварительного случайного перемешивания значений i-й переменной

Под ошибкой модели следует понимать ошибку классификации или среднеквадратичную ошибку.

Random Forest

Преимущества (унаследованные от деревьев решений):

+ нелинейный непараметрический метод
+ не требует отбора переменных
+ способен работать с числовыми и номинальными переменными

Преимущества (собственные):

+ устойчивость к переобучению
+ устойчивость к изменениям во входных данных
+ быстрый процесс обучения

Недостатки:

− невозможность экстраполяции
− сложность, а зачастую невозможность интерпретации

Gradient Boosting

Бустинг – создание ансамбля слабых моделей, путем их последовательного добавления к существующему набору. При этом каждая последующая модель уточняет прогноз всех предыдущих моделей.

Основная идея – создание на каждом шаге модели, прогноз которой максимально коррелирует с отрицательным градиентом функции потерь существующего ансамбля моделей.

Постановка задачи

Имеется исходный набор данных $\{x,y\}^N_{i=1}$ необходимо построить модель, которая как можно точнее предсказывала новые объекты.

Пусть имеется некоторое множество базовых моделей H, каждый элемент которого $h(x,a)$ определяется некоторым вектором параметров a.

Будем искать конечную модель в виде линейной комбинации

$$F_M(x) = \sum_{i=1}^{M} b_m h(x, a_m)$$

Упростим задачу

$$F_m(x) = F_{m-1} + b_m h(x, a_m)$$

Необходимо подобрать оптимальные параметры b_m и a_m. Для чего вводится функция потерь $L(y_i, F_m(x_i))$, которая показывает насколько сильно прогноз $F_m(x_i)$ отличается от наблюдаемого значения y_i.

$$(a_m, b_m) = \arg\min_{a,b} \sum_{i=1}^{N} L(y_i, F_{m-1}(x_i)) + b_m h(x_i, a_m)$$

Т.о. предсказанные моделью значения, должны оптимальным образом компенсировать ошибки предыдущего шага.

Чтобы упростить задачу, предлагается искать новую модель $h(x,a_m)$, которая бы максимально коррелировала с отрицательным градиентом функции потерь

$$g_m(x) = \left[\frac{\partial L(y_i, F_{m-1})}{\partial F_{m-1}}(x_i) \right]_{i=1}^{N}$$

Тогда поиск оптимальных параметров сводится к классической минимизации наименьших квадратов

$$(a_m, b_m) = \arg\min_{a,b} \sum_{i=1}^{N} [-g_m(x_i) + b_m h(x_i, a_m)]^2$$

Конечный вид зависит от выбранной функции потерь и базовых моделей

Алгоритм

Inputs:

input data $(x, y)^N_{i=1}$

number of iterations M

choice of the loss-function L(y, f)

choice of the base-learner model h(x, θ)

Algorithm:

1: initialize F_0 with a constant

2: **for** t = 1 to M **do**

3: compute the negative gradient $g_m(x)$

4: fit a new base-learner function $h(x, a_m)$

5: find the best gradient descent step-size b_m

6: update the function estimate:

$$F_m \leftarrow F_{m-1} + b_m h(x, a_m)$$

7: **end for**

Выбор функции потерь

Регрессионные задачи $y \in R$

Gaussian L2 loss function $$L(y,F)_{L2} = \frac{1}{2}(y-F)^2$$

Laplace L1 loss function $$L(y,F)_{L1} = |y-F|$$

Huber loss function $$L(y,F)_{Huber,\delta} = \begin{cases} 1/2(y-F)^2 & |y-F| \le \delta \\ \delta((y-F)-\delta/2) & |y-F| > \delta \end{cases}$$

Quantile loss function $$L(y,F)_{\alpha} = \begin{cases} (1-\alpha)|y-F| & y-F \le \alpha \\ \alpha|y-F| & y-F > \alpha \end{cases}$$

Выбор функции потерь

Классификационные задачи $y \in \{-1,1\}$

Binomial loss function (Bernoulli)

$$L(y,F)_{Bern} = \log(1+e^{-2yF})$$

Adaboost lost function (exponential)

$$L(y,F)_{Ada} = e^{-yF}$$

Выбор базового алгоритма

1. Linear models:

- Ordinary linear regression
- Ridge penalized linear regression
- Random effects

2. Smooth models:

- P-splines
- Radial basis functions

3. Decision trees

- Decision tree stumps
- Decision trees with arbitrary interaction depth

4. Other models:

- Markov Random Fields
- Wavelets
- Custom base-learner functions

Decision tree base-learner

Decision tree stumps – модели содержащие только одно разветвление (interaction depth = 1). Результатом является аддитивная модель.

Задавая interaction depth > 1, можно учесть взаимодействия переменных.

Рекомендации. Из практических соображений это значение не должно быть очень большим (более 20). Обычно достаточно 5-7.

Regularization: subsampling

Одним из способов избежать переобучения модели является обучение каждой модели в ансамбле не на всем наборе исходных объектов, а не некоторой случайно выбранной их части (bag fraction).

Преимущество: скорость обучения отдельной модели возрастает, т.к. используется меньший набор обучающих объектов.

Недостаток: требуется строить ансамбли из значительно большего числа моделей.

Рекомендации. Чем больше число объектов в обучающем наборе, тем меньше можно задавать bag fraction. Для большинства задач значение 0.5 будет удовлетворительным.

Regularization: shrinkage

Одним из способов контроля сложности модели является использование коэффициента сжатия (shrinkage, λ).

Его суть заключается в снижении вклада отдельной модели в итоговый ансамбль моделей.

$$F_m(x) = F_{m-1} + \lambda b_m h(x, a_m)$$

Преимущество: чем меньше значение λ, тем меньшее влияние имеют модели с большой ошибкой.

Недостаток: необходимо пропорционально увеличивать число моделей в ансамбле.

Рекомендации. Выбор оптимального значения λ может быть сделан на основе out-of-bag выборки, кросс-валидации или тестовой выборки. Обычно λ = 0.1-0.01.

Interpretation: relative influence

В отдельной модели дерева t с числом разветвлений P влияние j-й переменной можно определить как сумму квадратов уточнений в нетерминальных узлах (P-1), в которых использовалась переменная j.

$$Influence_j(t) = \sum_{i=1}^{P-1} I^2 1(S_i = j)$$

Для определения влияние переменной в ансамбле моделей, ее влияние усредняется по всем деревьям в ансамбле.

$$Influence_j = \frac{1}{T} \sum_{i=1}^{T} Influence_j(T_i)$$

Преимущества:

+ высокая точность прогноза
+ очень гибкий подход, позволяющий варьировать многие параметры создаваемой модели
+ может работать с выборками, для которых часть значений отдельных переменных неизвестна (NA).

Недостатки:

− медленная скорость обучения из-за необходимости последовательного построения отдельных базовых моделей
− возможность переобучения модели

R Examples

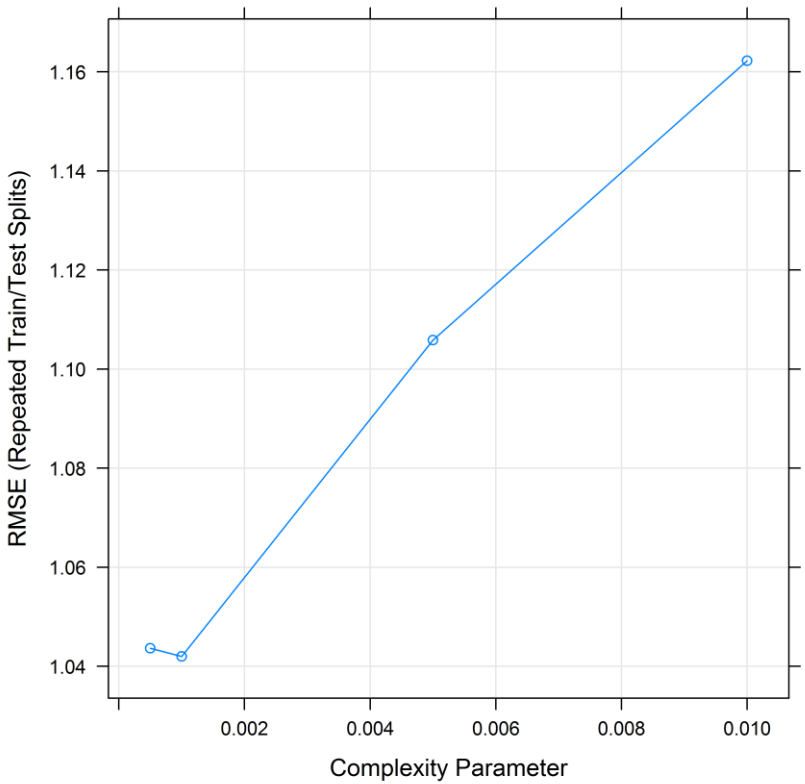
Регрессионная задача (растворимость)

Обучающая выборка: 800 соединений × 4058 дескрипторов
Тестовая выборка: 233 соединения × 3518 дескрипторов

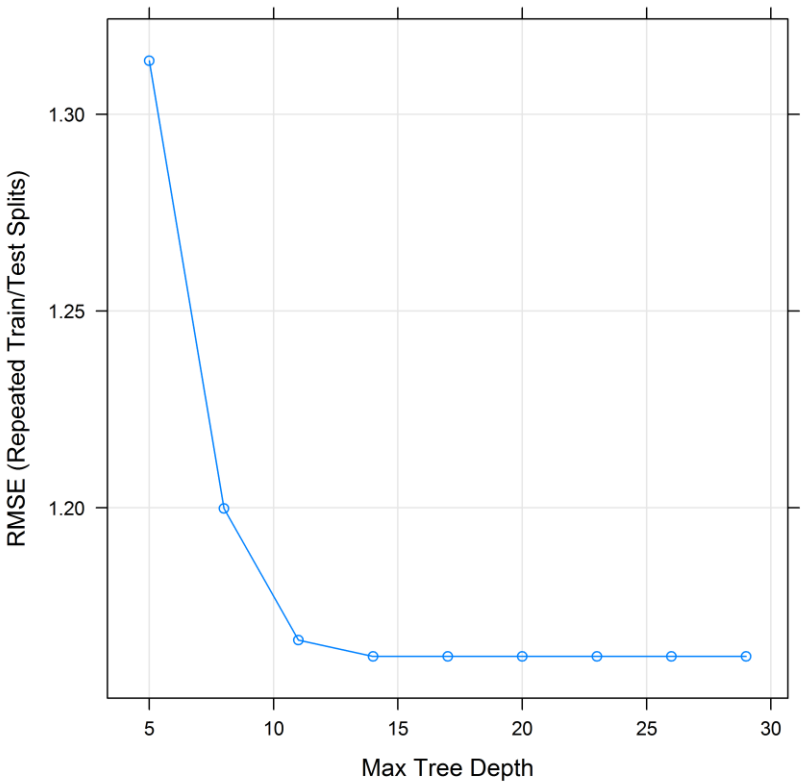
```
set.seed(42)

cv <- createFolds(y, 5, returnTrain=TRUE)

trControl <- trainControl(method="LGOCV“, index=cv,
  savePredictions=TRUE)

cart1 <- train(x, y, method="rpart", trControl=trControl,
               tuneGrid=data.frame(cp=c(0.0005, 0.001, 0.005, 0.01)))

cart2 <- train(x, y, method="rpart2", trControl=trControl,
               tuneGrid=data.frame(maxdepth=seq(5,30,3)))

rf <- train(x, y, method="rf", trControl=trControl,
            tuneGrid=data.frame(mtry=c(100,500,1000)))

gbm <- train(x, y, method="gbm", trControl=trControl,
             tuneGrid=expand.grid(n.trees=c(50,100,150,200,500),
                                  interaction.depth=1:5,
                                  shrinkage=c(0.001,0.01,0.1)))
```

Регрессионная задача (растворимость)

CART1

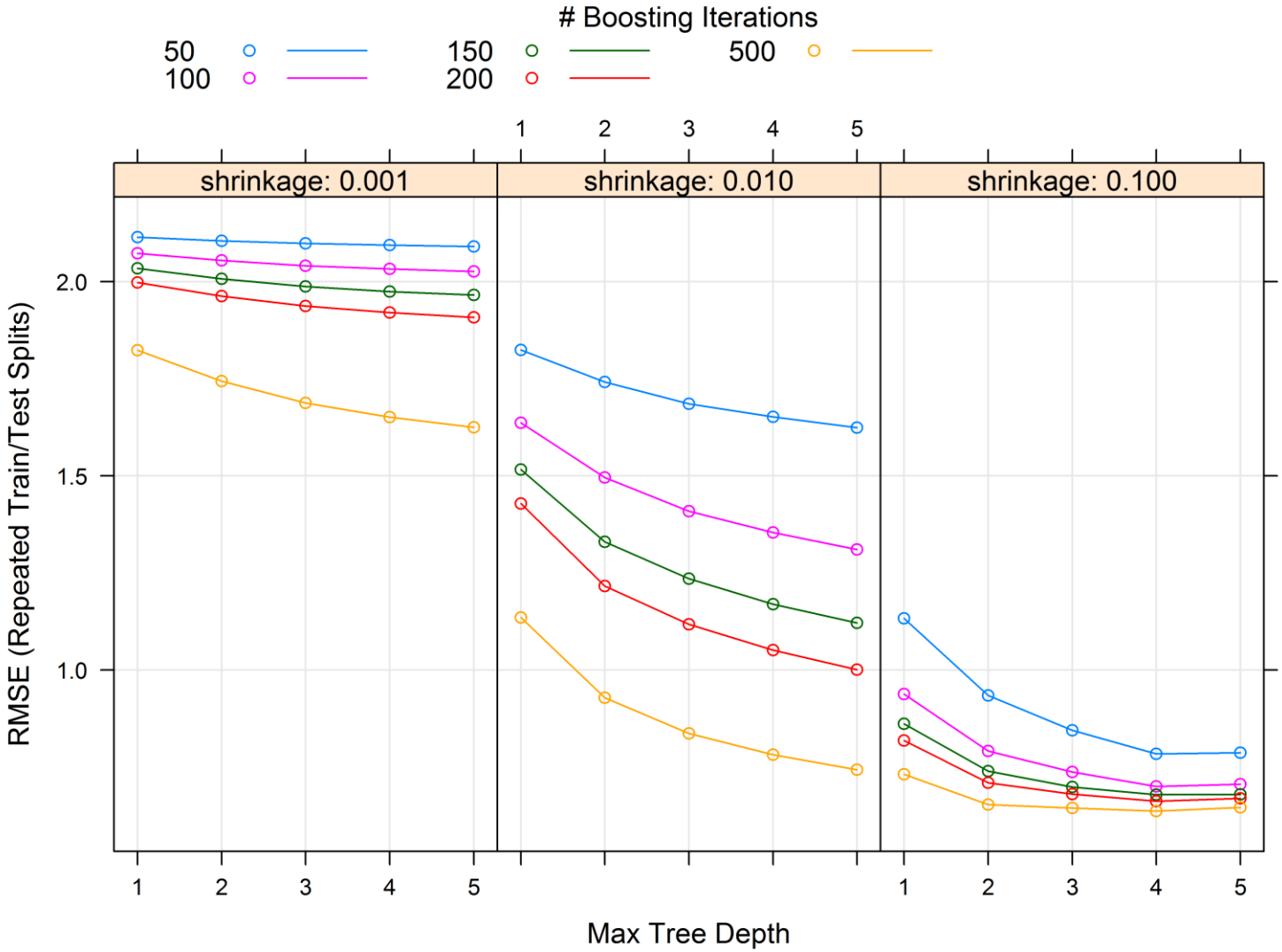


CART2

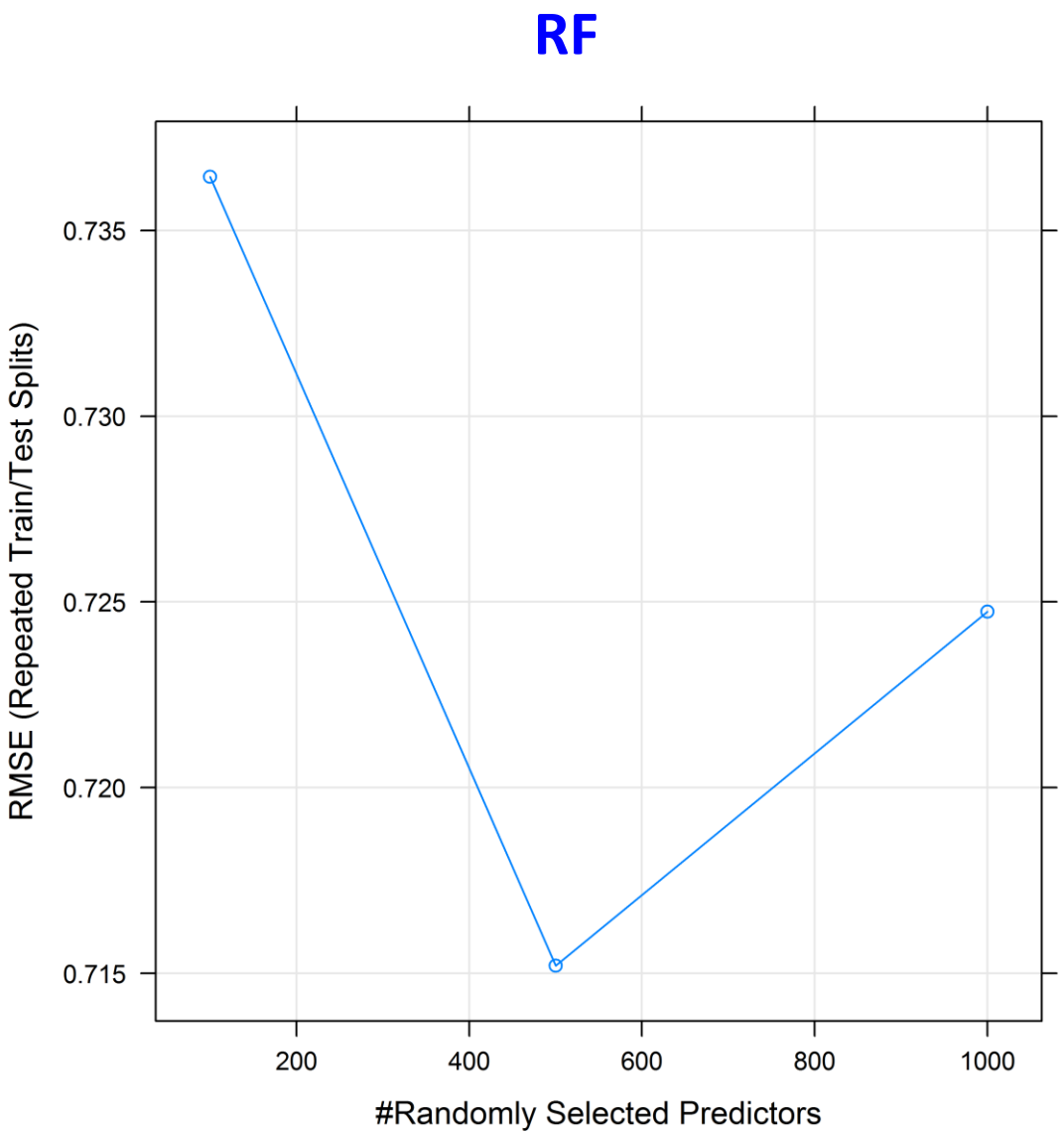


Регрессионная задача (растворимость)

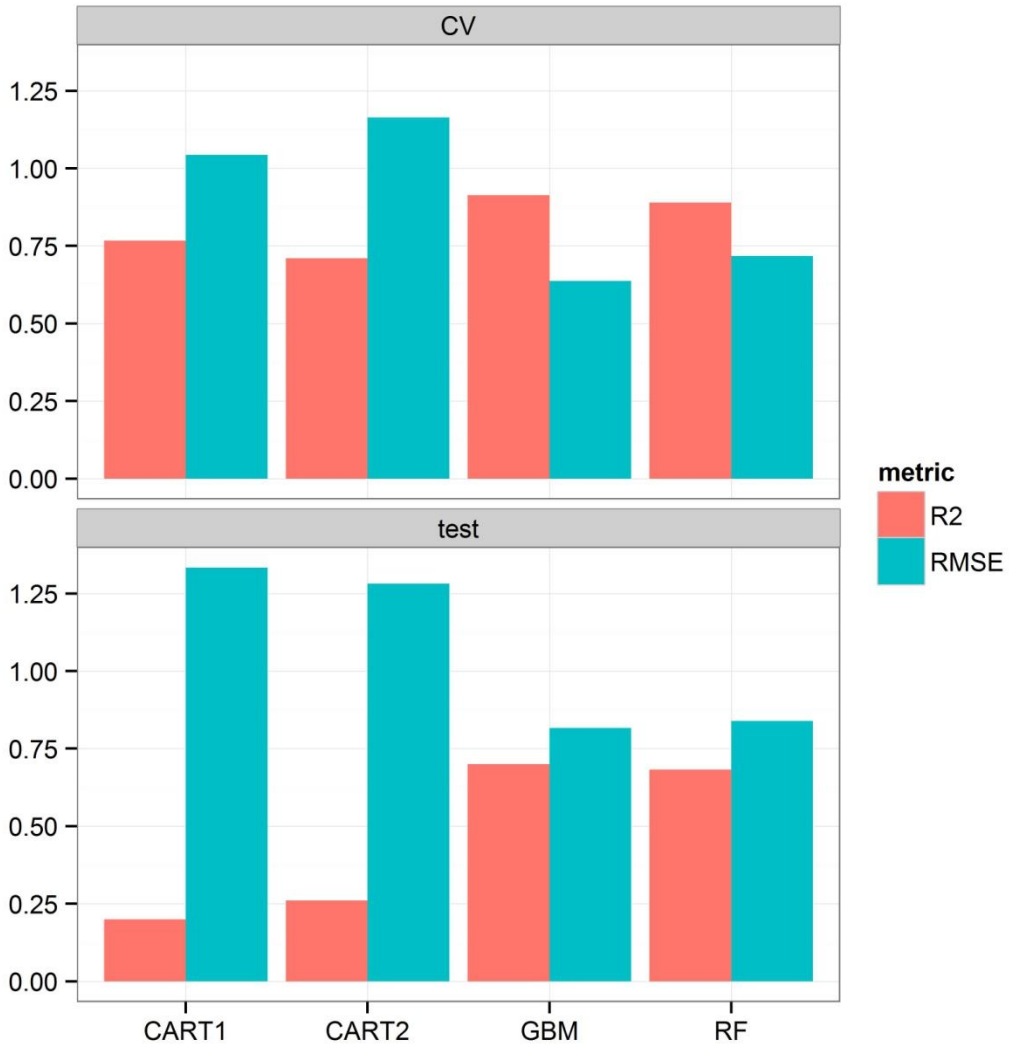
GBM



Регрессионная задача (растворимость)



Регрессионная задача (растворимость)



Классификационная задача (мутагенность)

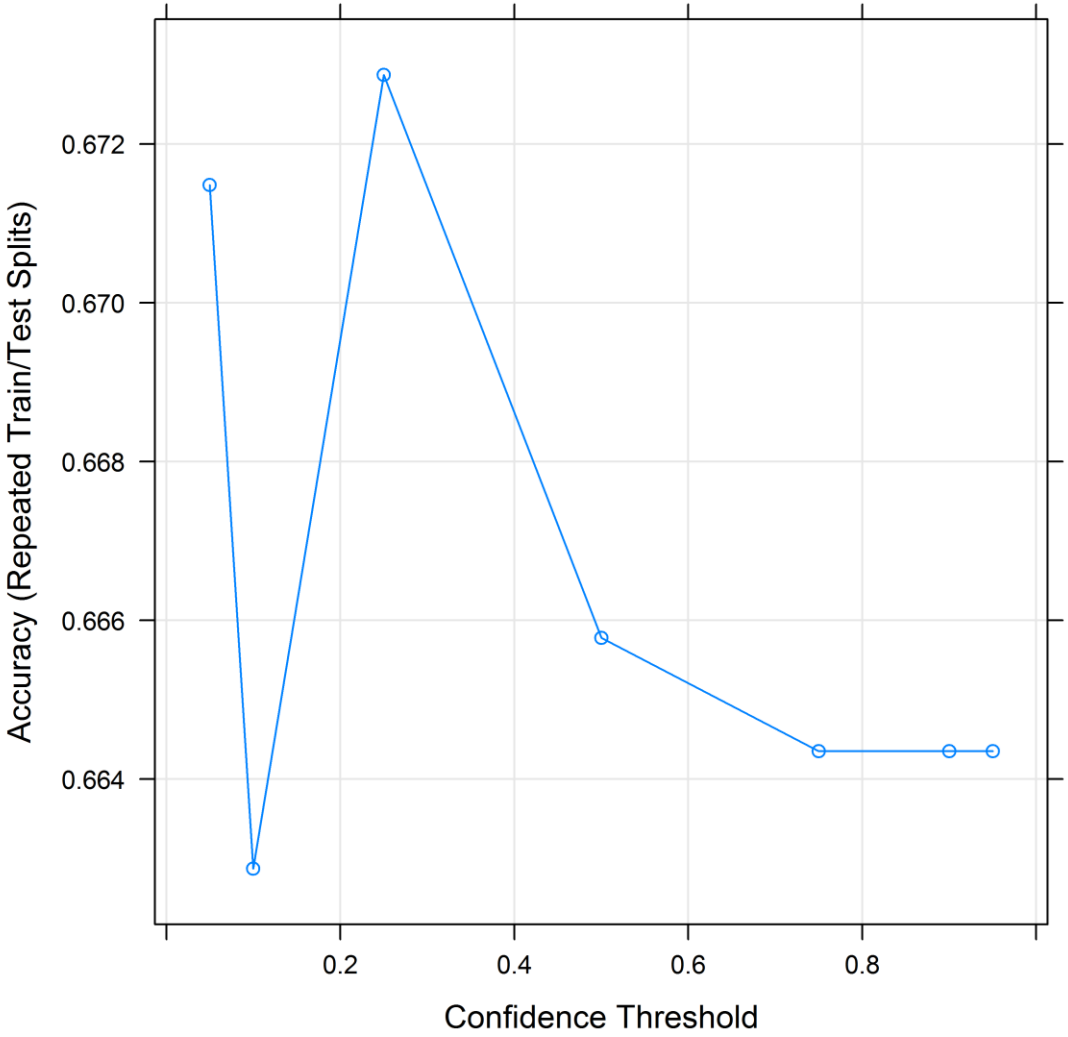
Обучающая выборка: 700 соединений × 12034 дескриптора

Тестовая выборка: 700 соединений × 3518 дескрипторов

```
set.seed(42)

cv <- createFolds(y, 5, returnTrain=TRUE)

trControl <- trainControl(method="LGOCV", index=cv,
                          savePredictions=TRUE)

c45_1 <- train(x, y, method="J48", trControl=trControl,
               tuneGrid=data.frame(C=c(0.05,0.1,0.25,0.5,0.75,0.9,0.95)))

cart1 <- train(x, y, method="rpart", trControl=trControl,
               tuneGrid=data.frame(cp=(1:5)/100))

cart2 <- train(x, y, method="rpart2", trControl=trControl,
               tuneGrid=data.frame(maxdepth=c(2,4,5,6,7)))

rf <- train(x, y, method="rf", trControl=trControl,
            tuneGrid=data.frame(mtry=c(100,500,1000,2000)))

gbm <- train(x, y, method="gbm", trControl=trControl,
             tuneGrid=expand.grid(n.trees=c(50,100,150,200,500),
                                  interaction.depth=1:5,
                                  shrinkage=c(0.001,0.01,0.1)))
```

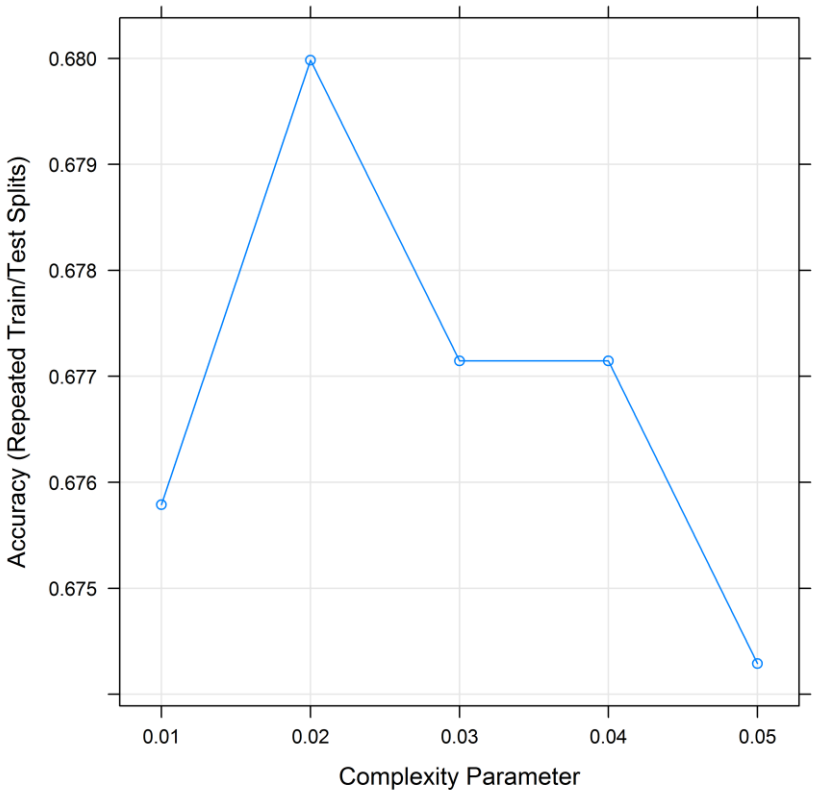
Классификационная задача (мутагенность)

C4.5

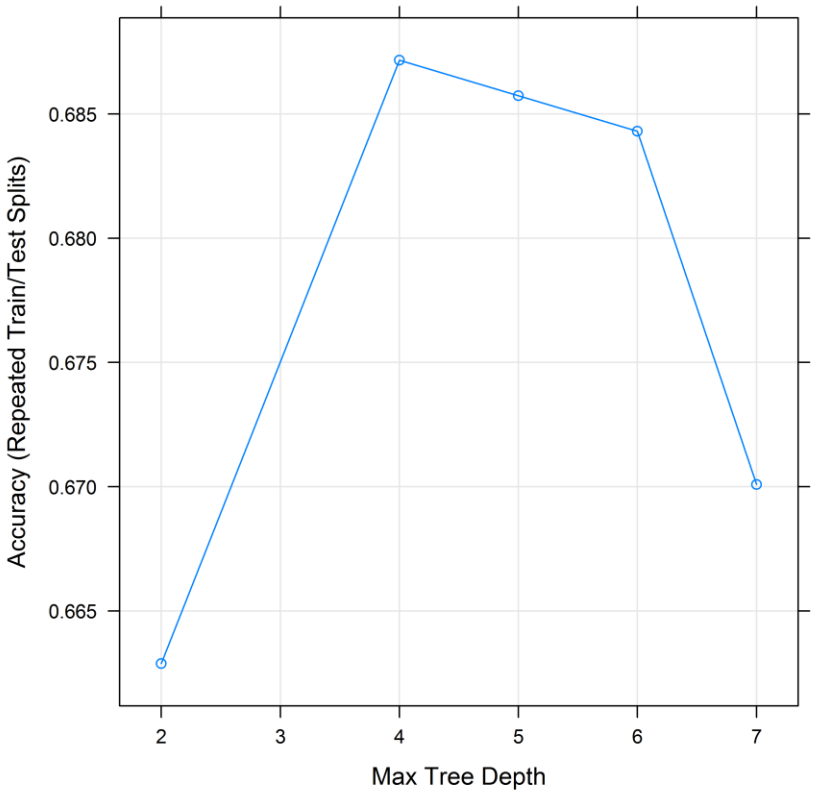


Классификационная задача (мутагенность)

CART1

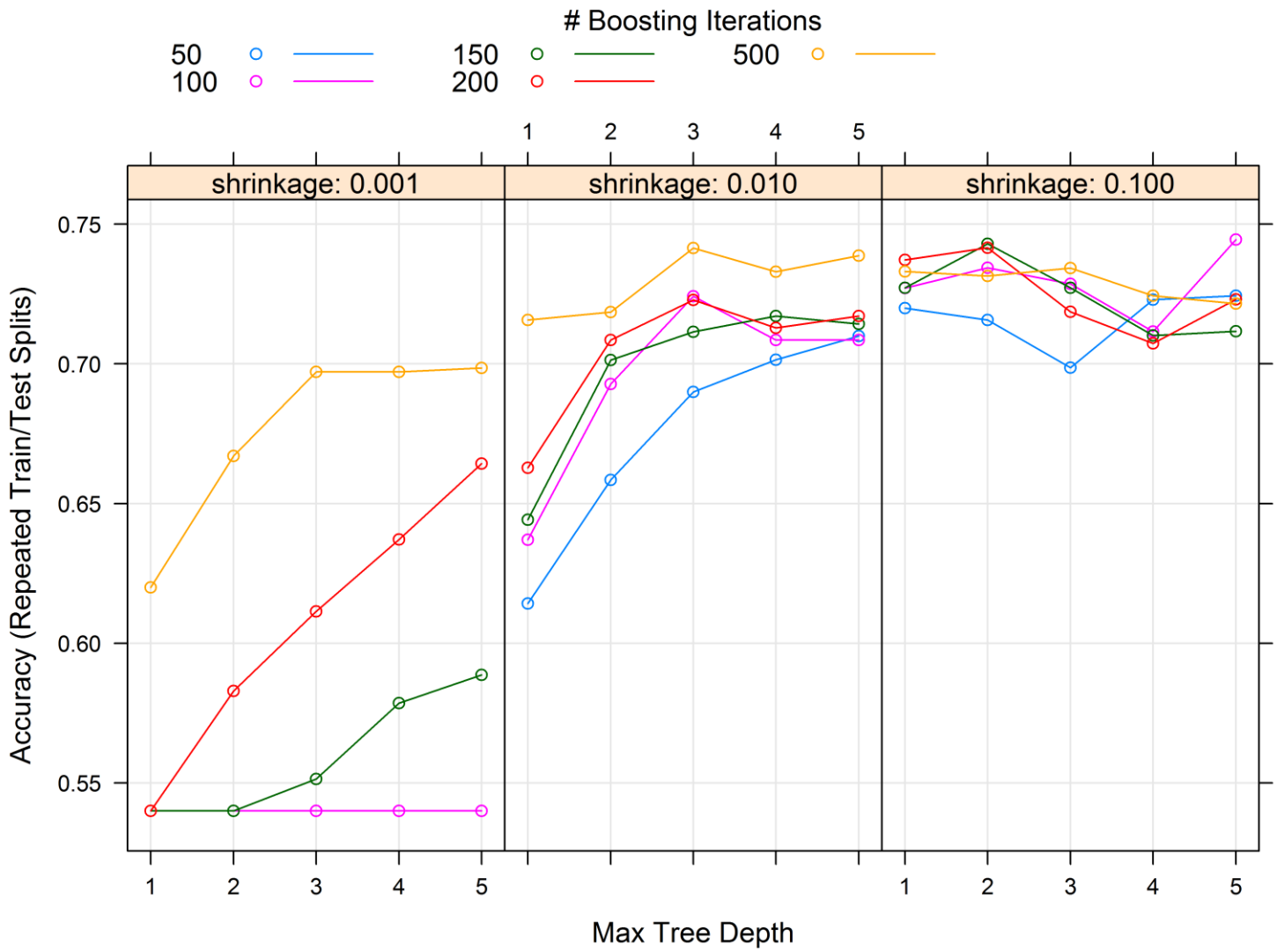


CART2

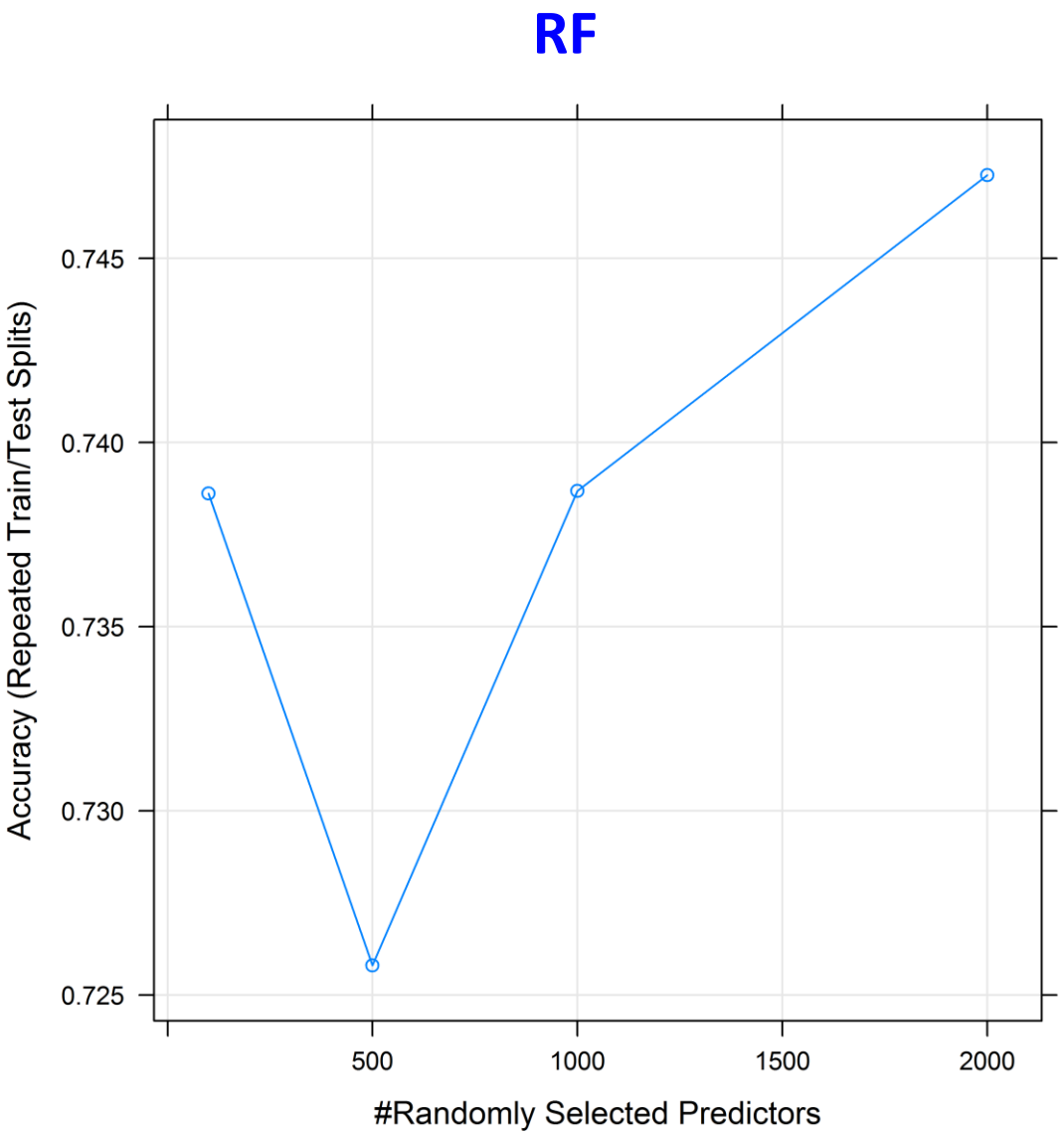


Классификационная задача (мутагенность)

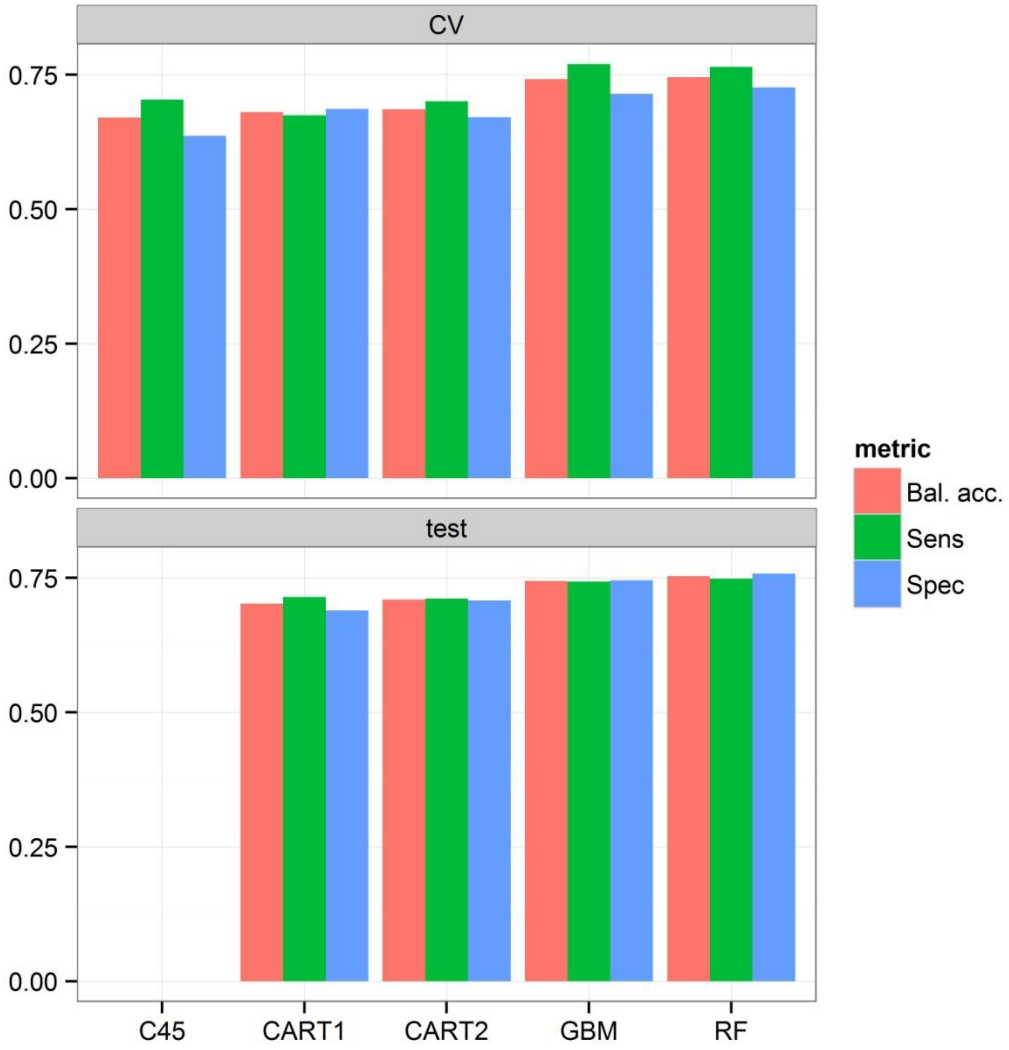
GBM



Классификационная задача (мутагенность)



Классификационная задача (мутагенность)



Recommended literature

ID3 Quinlan, J. R. 1986. Induction of Decision Trees. Mach. Learn. 1, 1 (Mar. 1986), 81-106

C4.5 Quinlan, J. R. C4.5: Programs for Machine Learning. Morgan Kaufmann Publishers, 1993

CART Leo Breiman, Jerome Friedman, Charles J. Stone, R.A. Olshen, Classification and Regression Trees, p. 368

Decision trees F. Berzal, J. Cubero, F. Cuenca, M.J. Martín-Bautista, On the quest for easy-to-understand splitting rules, Data & Knowledge Engineering, Vol. 44, № 1, 2003, 31-48, http://dx.doi.org/10.1016/S0169-023X(02)00062-9

Random Forest Breiman L. Random Forests. Machine Learning, 45(1):5-32, 2001

Gradient boosting

Friedman, J. H. "Greedy Function Approximation: A Gradient Boosting Machine."
Friedman, J. H. "Stochastic Gradient Boosting."
Natekin and Knoll "Gradient boosting machines, a tutorial",
doi: 10.3389/fnbot.2013.00021

Замечания и пожелания можно присылать автору на почту

pavel_polishchuk@ukr.net