Прикладные задачи анализа данных

Случайные леса

Дьяконов А.Г.

Московский государственный университет имени М.В. Ломоносова (Москва, Россия)

Универсальные методы

1. **Основанные на близости (~kNN)**

Недостатки: проблема перебора,
настройки параметров,
выбора метрики

2. **SVM**

Недостатки: узкая группа задач,
нужны однородные признаки в одной шкале

3. **Случайные леса**

Недостатки: плохо работает
С линейными закономерностями

Нет универсальных методов! Всё идёт от задачи…

Случайные леса

+ наиболее универсальный
(~75% задач машинного обучения)

+ все типы задач
(классификация, регрессия, кластеризация)

+ настраивается сразу под все функционалы
(или можно преобразовать – не всегда надо)

+ нечувствителен к монотонным преобразованиям признаков
не совсем так…

+ легко реализуется
(лучшие реализации: R и Python)

Ансамблирование –

построения множества классификаторов и усреднение их результатов

- бэггинг (bootstrap aggregating)

бутстреп выборки и независимое построение алгоритмов

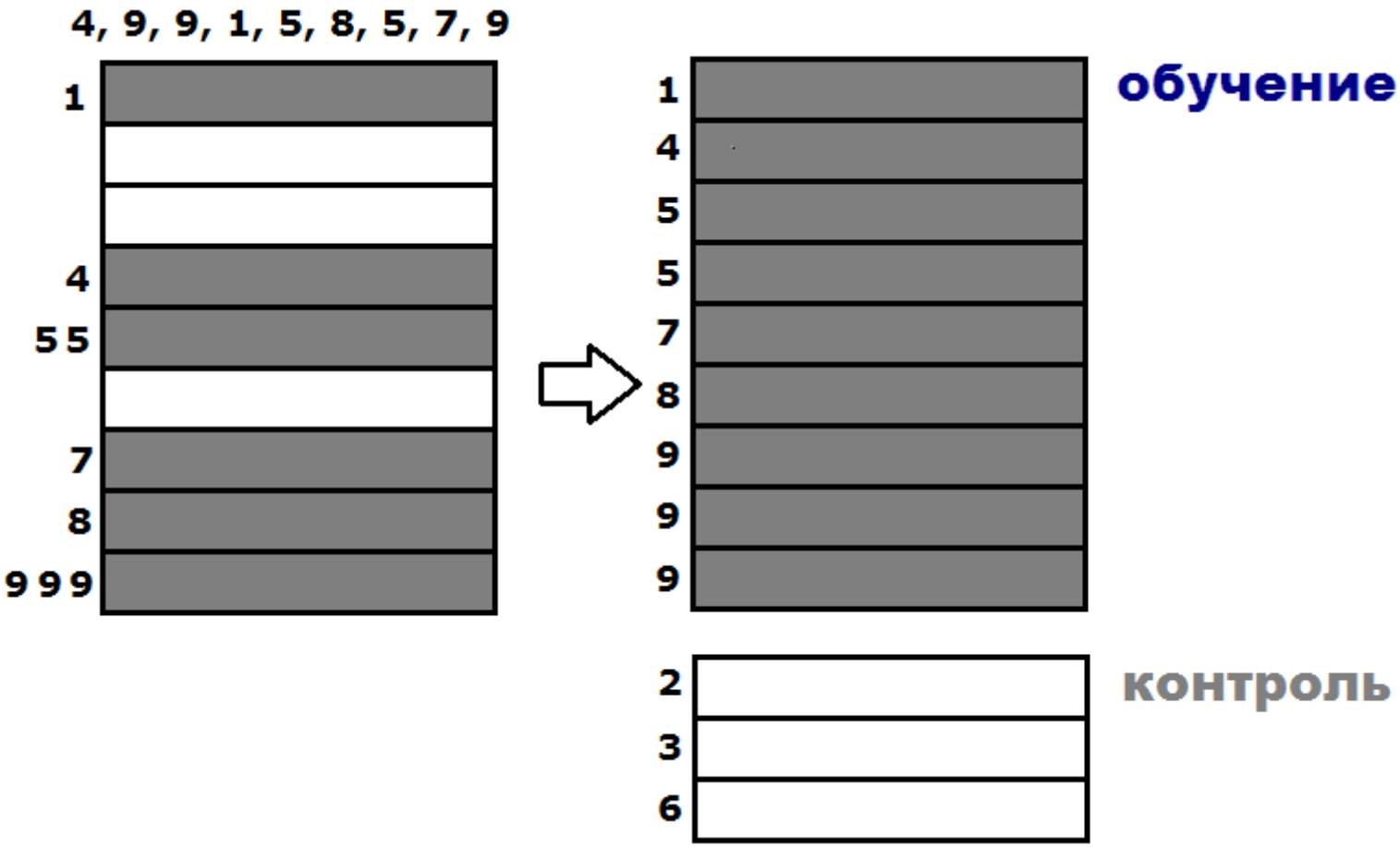
- бустинг

весовые схемы для объектов, объекты, на которых происходят ошибки получают больший вес, в дальнейшем обучение концентрируется на них

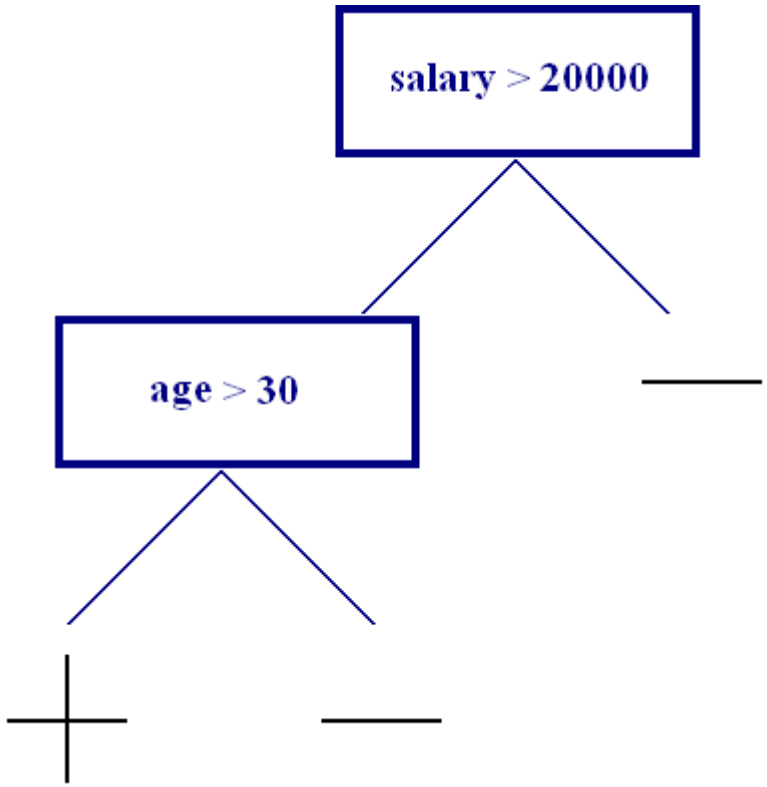
случайные леса =

бэггинг + специальное построение деревьев (подмножество признаков при расщеплении)

Бутстреп



Что такое решающее дерево?

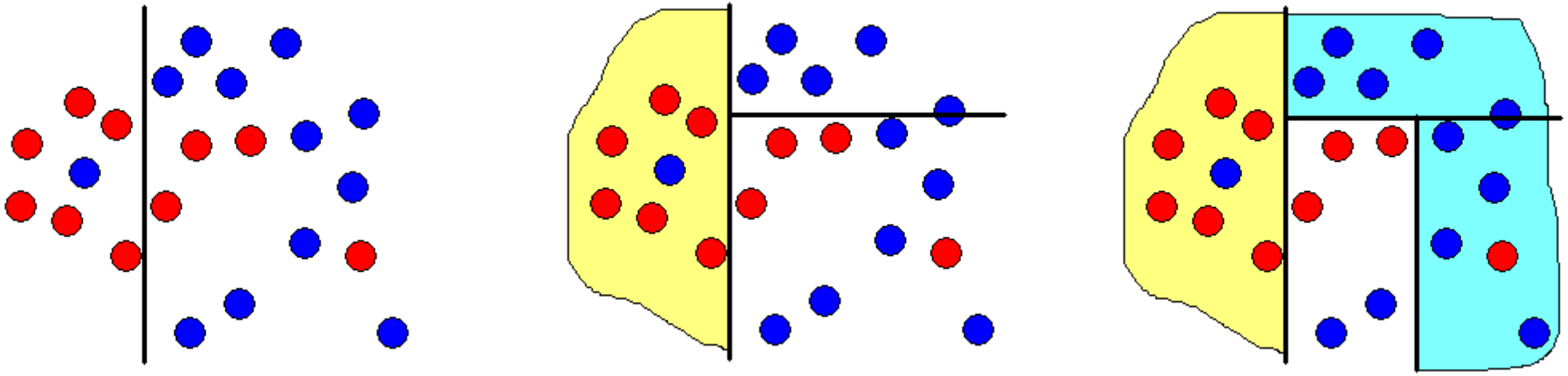


Построение одного дерева

Последовательные дихотомии:
Выбор признака и порога расщепления

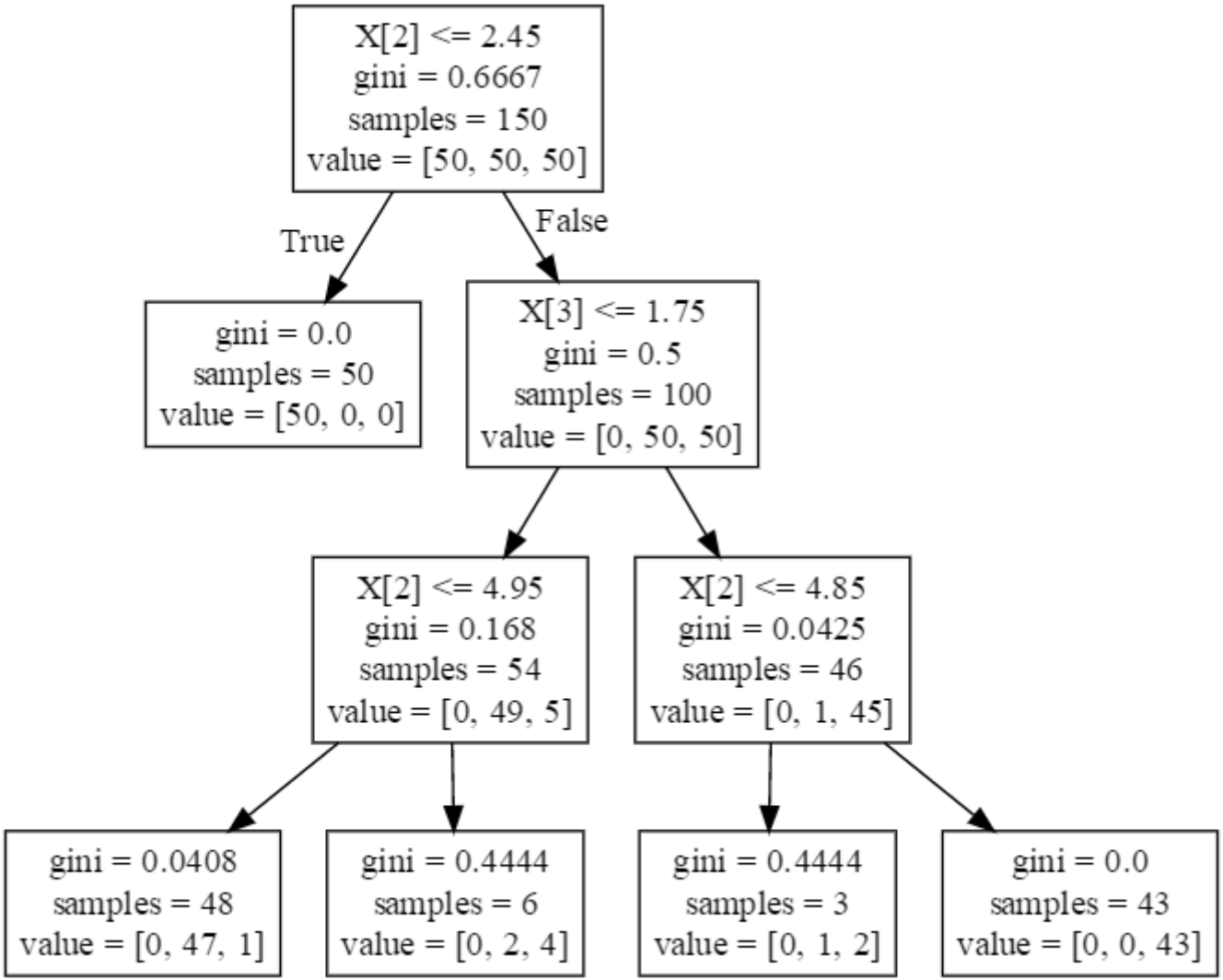
Критерии

Качество одного дерева очень низкое!
Случайный лес улучшает его, как правило, на 10%



может быть остановка дихотомии из-за `maxnodesize`...

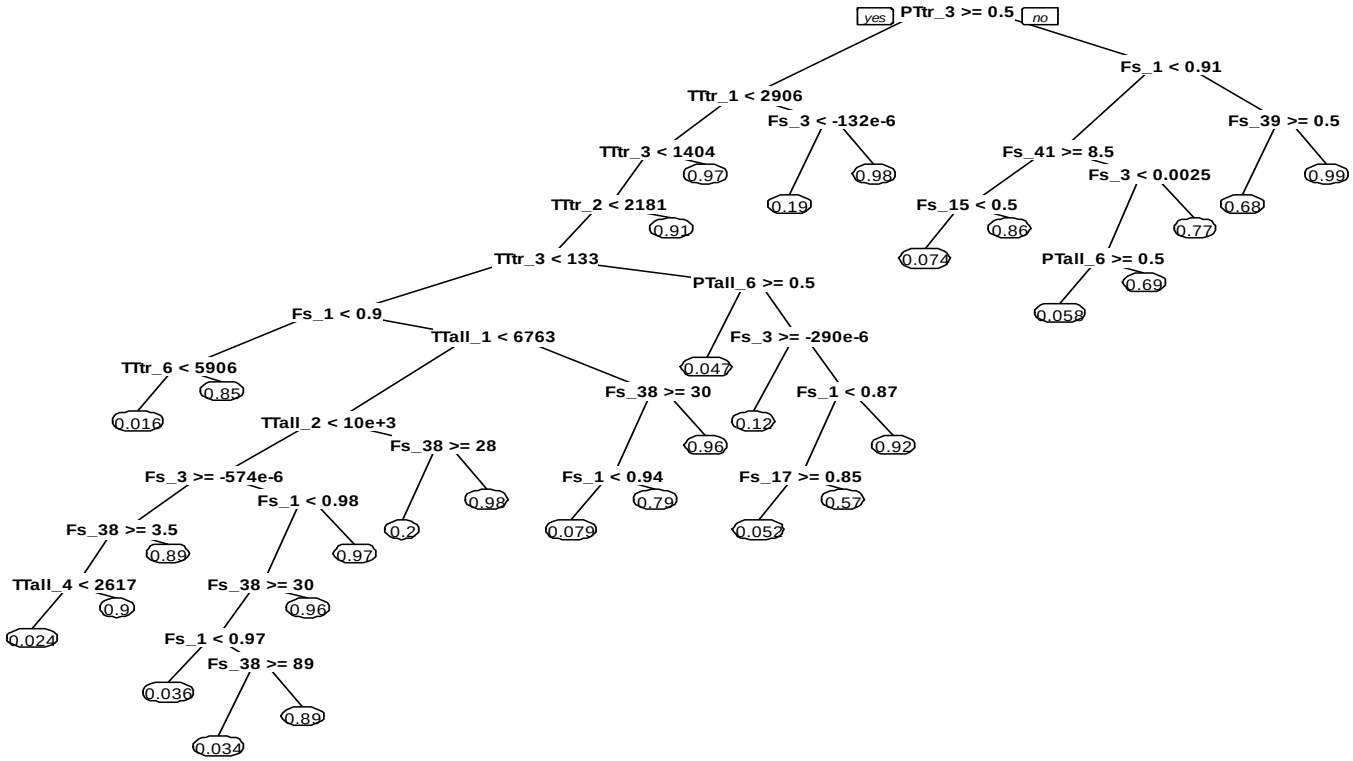
Построение одного дерева




```
from sklearn.datasets import load_iris
from sklearn import tree
clf =
tree.DecisionTreeClassifier(max_depth=3)
iris = load_iris()
clf = clf.fit(iris.data, iris.target)
tree.export_graphviz(clf,
out_file='tree.dot')
```

http://dreampuf.github.io/GraphvizOnline/

Построение одного дерева (задача Wikimart)




```
library('rpart')
model <- rpart(V1~. , T, control=rpart.control(minsplit=30, cp=0.001) )
a = predict(model, T2)
colAUC(a, T2[, 1], plotROC=FALSE)
0.9803839
plot(model)
library("rpart.plot", lib.loc="C:/R-3.0.1/library")
prp(model)
```

Построение одного дерева (задача Wikimart)

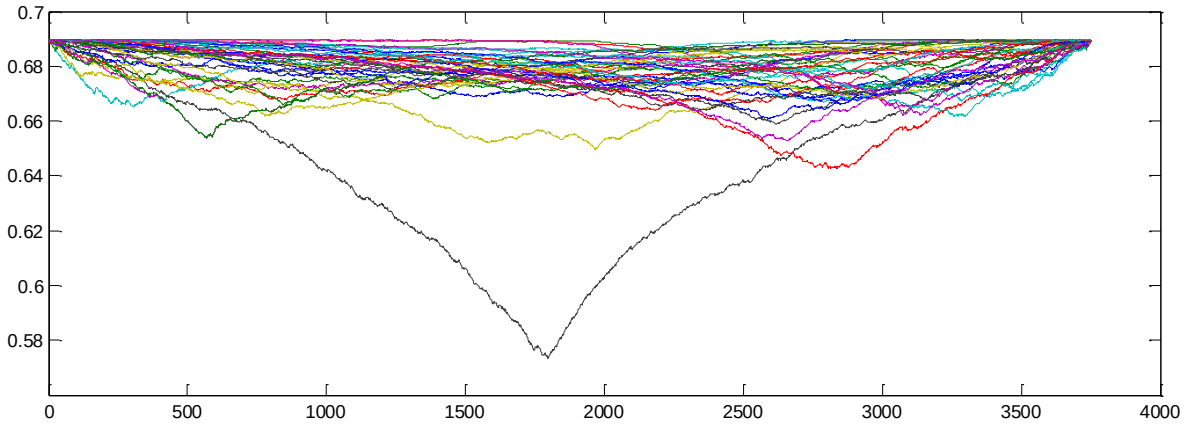
```
library('tree')
model <- tree(V1~., T)
a = predict(model, T2)
colAUC(a, T2[, 1], plotROC=FALSE)
model
node), split, n, deviance, yval
      * denotes terminal node

 1) root 51791 11650.00 0.34180
   2) PTtr_3 < 0.5 14519  1047.00 0.92180
     4) Fs_1 < 0.913665 1452   332.00 0.35400 *
     5) Fs_1 > 0.913665 13067   195.00 0.98480 *
   3) PTtr_3 > 0.5 37272  3820.00 0.11590
     6) TTtr_1 < 2905.9 35995  2821.00 0.08571
      12) TTtr_3 < 1403.75 34914  1917.00 0.05831
        24) TTtr_2 < 2181.03 34736  1773.00 0.05395 *
        25) TTtr_2 > 2181.03 178    14.56 0.91010 *
      13) TTtr_3 > 1403.75 1081    31.05 0.97040 *
     7) TTtr_1 > 2905.9 1277    39.68 0.96790 *
a = predict(model, T2)
colAUC(a, T2[, 1], plotROC=FALSE)
0.9408987
```

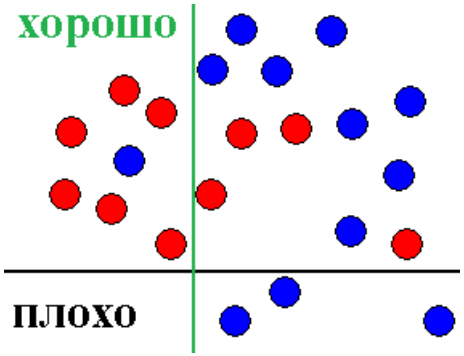
Простейшее дерево – высокое качество!

Но: это большая редкость (хорошие признаки).

Самостоятельная реализация деревьев



энтропии разбиений по всем признакам



Не хочется отщепления маленьких кусков данных…

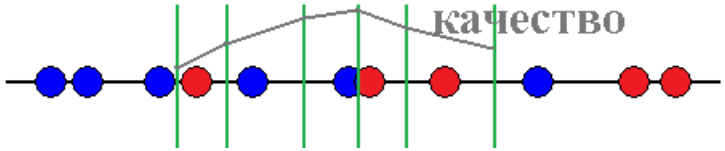
Вопрос: что делать?

Ответ:

классика

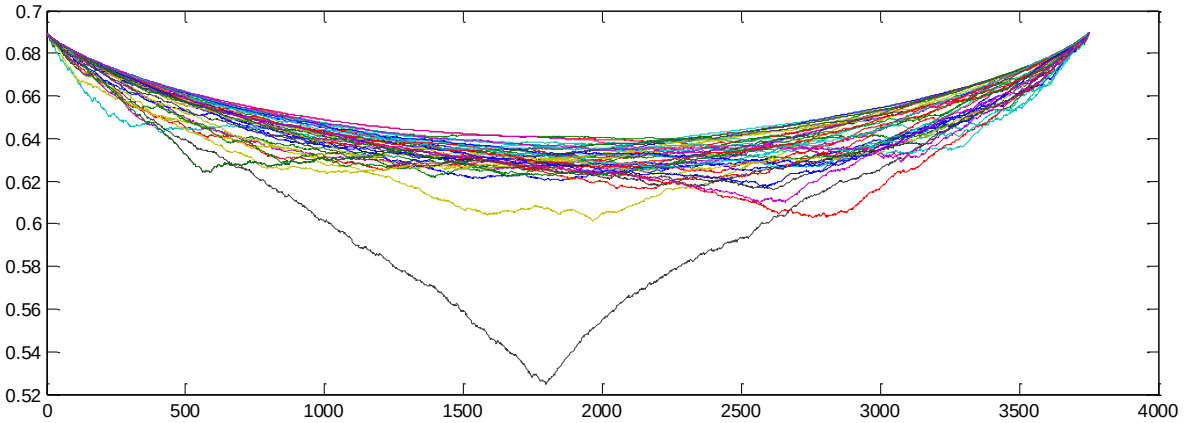
- минимальное число объектов в листе

- минимальное число, при котором возможно разбиение



добавление специальной функции, которая наказывает за края

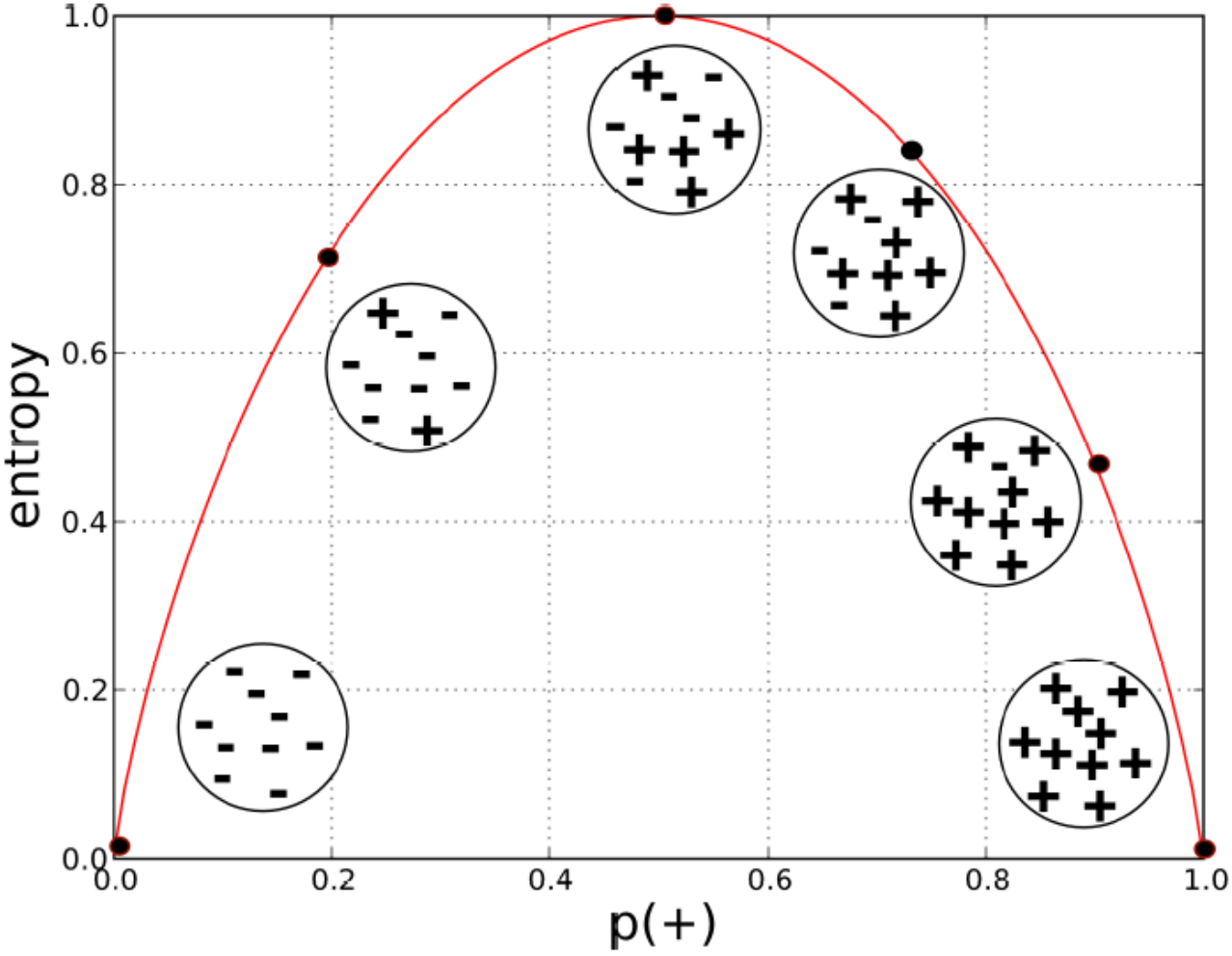
какой?



+ 0.07*штраф за «серединность»

Критерии расщепления

Что такое (информационная) энтропия…



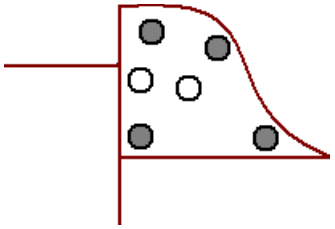
$$-p\log p-(1-p)\log(1-p)$$

Как приписать значение листу

Пусть функция потерь: Logarithmic Loss (LogLoss)

$$-y_i \log a_i - (1-y_i)\log(1-a_i) = -\begin{cases} \log a_i, & y_i = 1, \\ \log(1-a_i), & y_i = 0. \end{cases}$$

(~функция правдоподобия распределения Бернулли)



$$\sum_i \begin{cases} \log p, & y_i = 1, \\ \log(1-p), & y_i = 0. \end{cases} \to \max$$

$$m_1[\log p] + m_0[\log(1-p)] \to \max$$

$$G(p) = \frac{m_1}{m_1+m_0}\log p + \frac{m_0}{m_1+m_0}\log(1-p) \to \max$$

$p^* = \frac{m_1}{m_1+m_0}$ **т.е. оптимальная константа – оценка вероятности!**

$$G(p^*) = p^* \log p^* + (1-p^*)\log(1-p^*)$$

т.е. потеря при оптимальной константе ~ энтропия

Правда, мир устроен примитивно…

$$m_1[p-1]^2 + m_0[p]^2 \to \min$$

тогда тоже

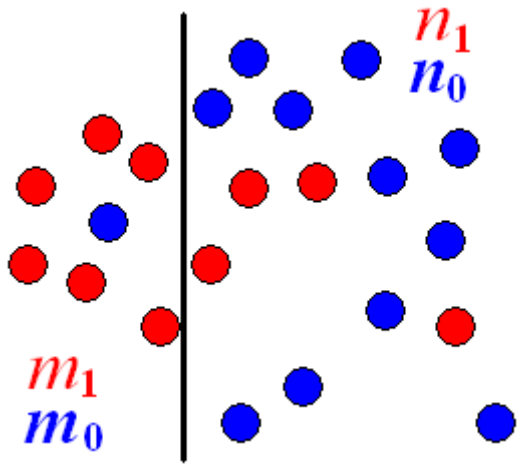
$$p^* = \frac{m_1}{m_1+m_0}$$

Критерии расщепления

Слагаемое «серединности»

Слагаемое «качества разделения»

Информационный критерий (идея)



$$\left(-\frac{m_1+m_0}{m_1+m_0+n_1+n_0}\log\frac{m_1+m_0}{m_1+m_0+n_1+n_0} - \frac{n_1+n_0}{m_1+m_0+n_1+n_0}\log\frac{n_1+n_0}{m_1+m_0+n_1+n_0} \right)$$

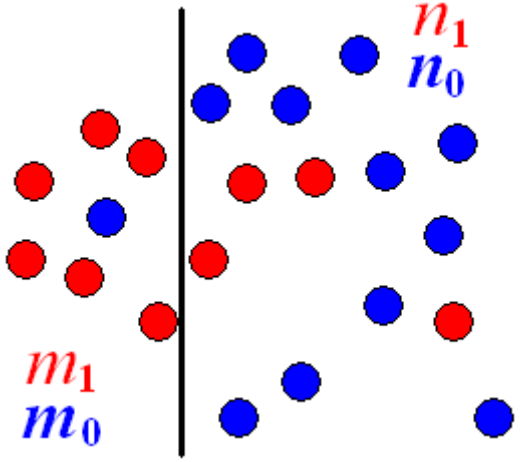
$$-$$

$$\frac{m_1+m_0}{m_1+m_0+n_1+n_0}\left(-\frac{m_1}{m_1+m_0}\log\frac{m_1}{m_1+m_0} - \frac{m_0}{m_1+m_0}\log\frac{m_0}{m_1+m_0} \right) +$$

$$\frac{n_1+n_0}{m_1+m_0+n_1+n_0}\left(-\frac{n_1}{n_1+n_0}\log\frac{n_1}{n_1+n_0} - \frac{n_0}{n_1+n_0}\log\frac{n_0}{n_1+n_0} \right)$$

Видно, как эффективно реализовывать…

Gini (идея)



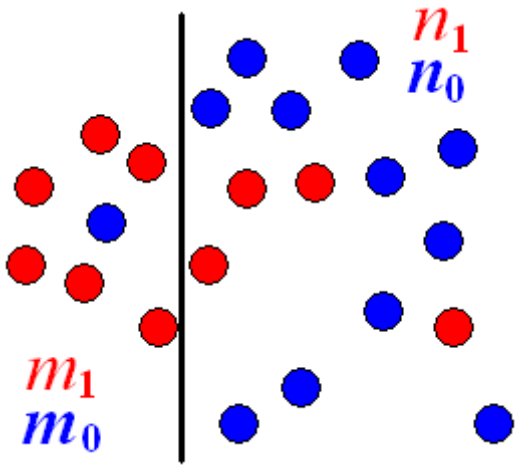
$$\left(1-\left(\frac{m_1+m_0}{m_1+m_0+n_1+n_0}\right)^2-\left(\frac{n_1+n_0}{m_1+m_0+n_1+n_0}\right)^2\right)$$

$$-$$

$$\frac{m_1+m_0}{m_1+m_0+n_1+n_0}\left(1-\left(\frac{m_1}{m_1+m_0}\right)^2-\left(\frac{m_0}{m_1+m_0}\right)^2\right)+$$

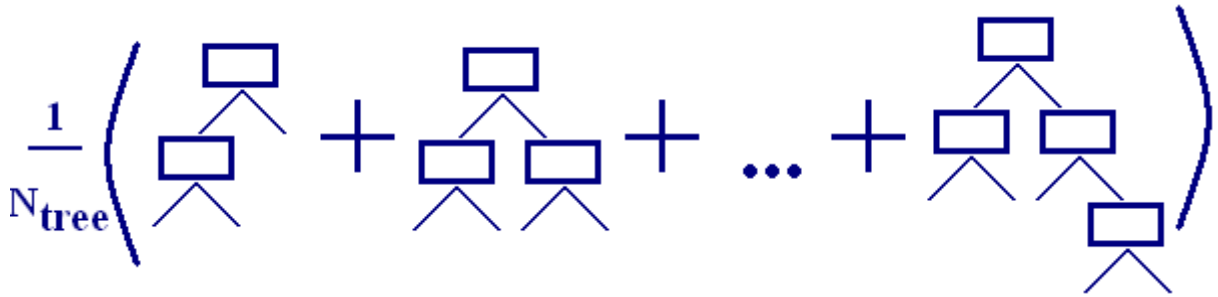
$$\frac{n_1+n_0}{m_1+m_0+n_1+n_0}\left(1-\left(\frac{n_1}{n_1+n_0}\right)^2-\left(\frac{n_0}{n_1+n_0}\right)^2\right)$$

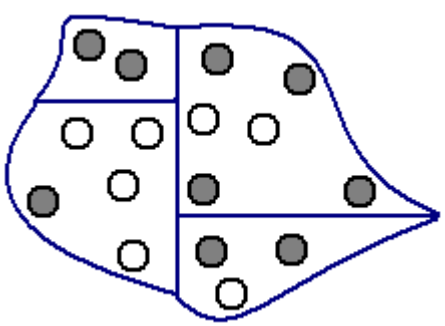
Twoing (идея)

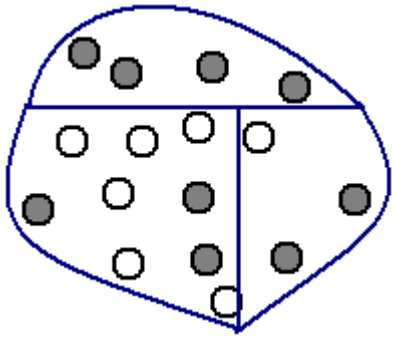


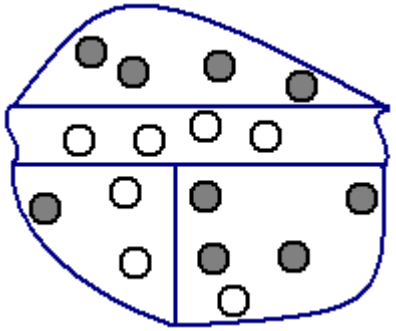
$$\frac{1}{4}\left(\frac{m_1+m_0}{m_1+m_0+n_1+n_0}\right)\left(\frac{n_1+n_0}{m_1+m_0+n_1+n_0}\right)(|\,m_1-n_1\,|+|\,m_0-n_0\,|)$$

Что такое случайный лес?

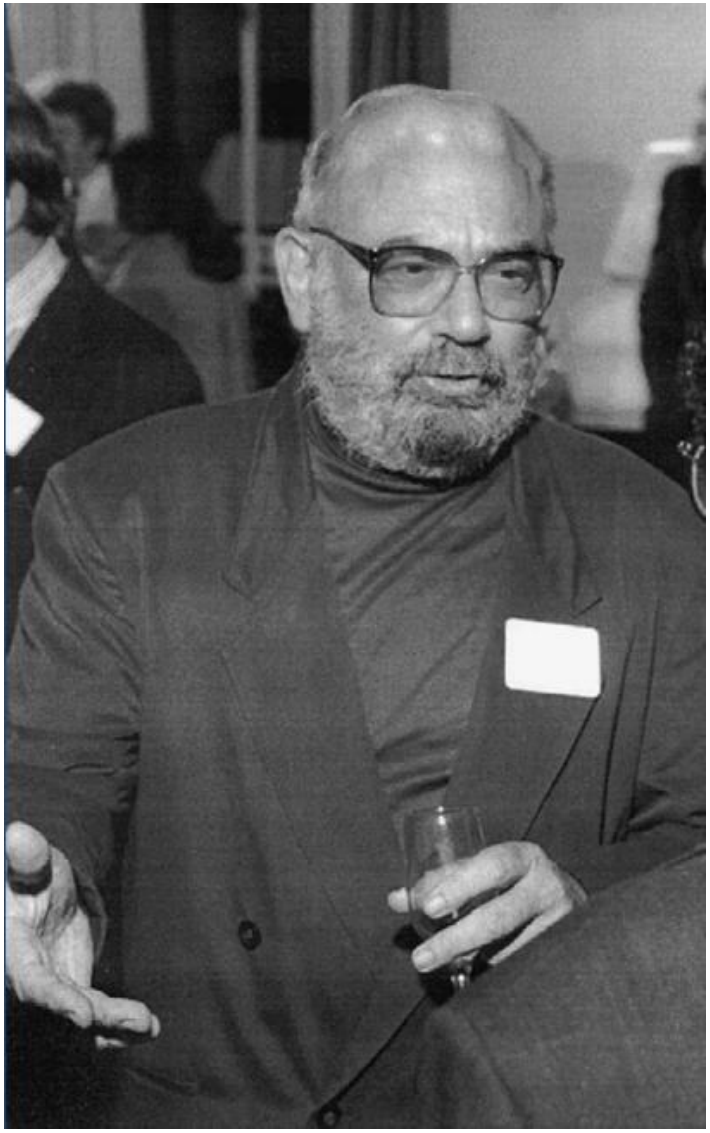








Лео Брейман, 1928 – 2005



Построение случайного леса

1. Выбирается подвыборка `samplesize` (м.б. с повторением) – на ней строится дерево

2. Строим дерево

2.1. Для построения каждого расщепления просматриваем `mtry` / `max_features` случайных признаков

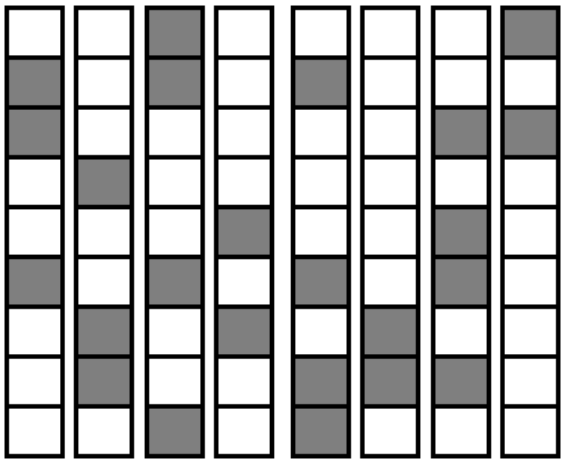
2.2. Как правило, дерево строится до исчерпания выборки (без прунинга)

Ответ леса: по большинству (в задачах классификации), среднее арифметическое (в задачах регрессии)

Автоматически: рейтинг признаков – `importance(model)` / `.feature_importances_`

Бэггинг и OOB (out of bag)

Выбор объектов для обучения (с помощью бутстрепа), остальные – локальный контроль…



Ответы разных деревьев – можно усреднить и вычислить качество

Параметры случайного леса

```
class
sklearn.ensemble.RandomForestClassifier
(n_estimators=10,
criterion='gini',
max_depth=None,
min_samples_split=2,
min_samples_leaf=1,
min_weight_fraction_leaf=0.0,
max_features='auto',
max_leaf_nodes=None,
bootstrap=True,
oob_score=False,
n_jobs=1,
random_state=None,
verbose=0,
warm_start=False,
class_weight=None)
```

```
{randomForest} randomForest(
x, y, xtest, ytest,
ntree=500,
mtry=if (!is.null(y) &&
!is.factor(y))
max(floor(ncol(x)/3), 1) else
floor(sqrt(ncol(x))),
replace=TRUE,
classwt=NULL,
cutoff,
strata,
sampsize = if (replace) nrow(x)
else ceiling(.632*nrow(x)),
nodesize = if (!is.null(y) &&
!is.factor(y)) 5 else 1,
maxnodes = NULL,
importance=FALSE,
localImp=FALSE,
nPerm=1,
proximity, oob.prox=proximity)
```

Настройка параметров: размер подвыборки `sampsize`

1. Определиться с типом выбора

Частное мнение: без возврата

2. Настройка по объёму

– не в первую очередь,

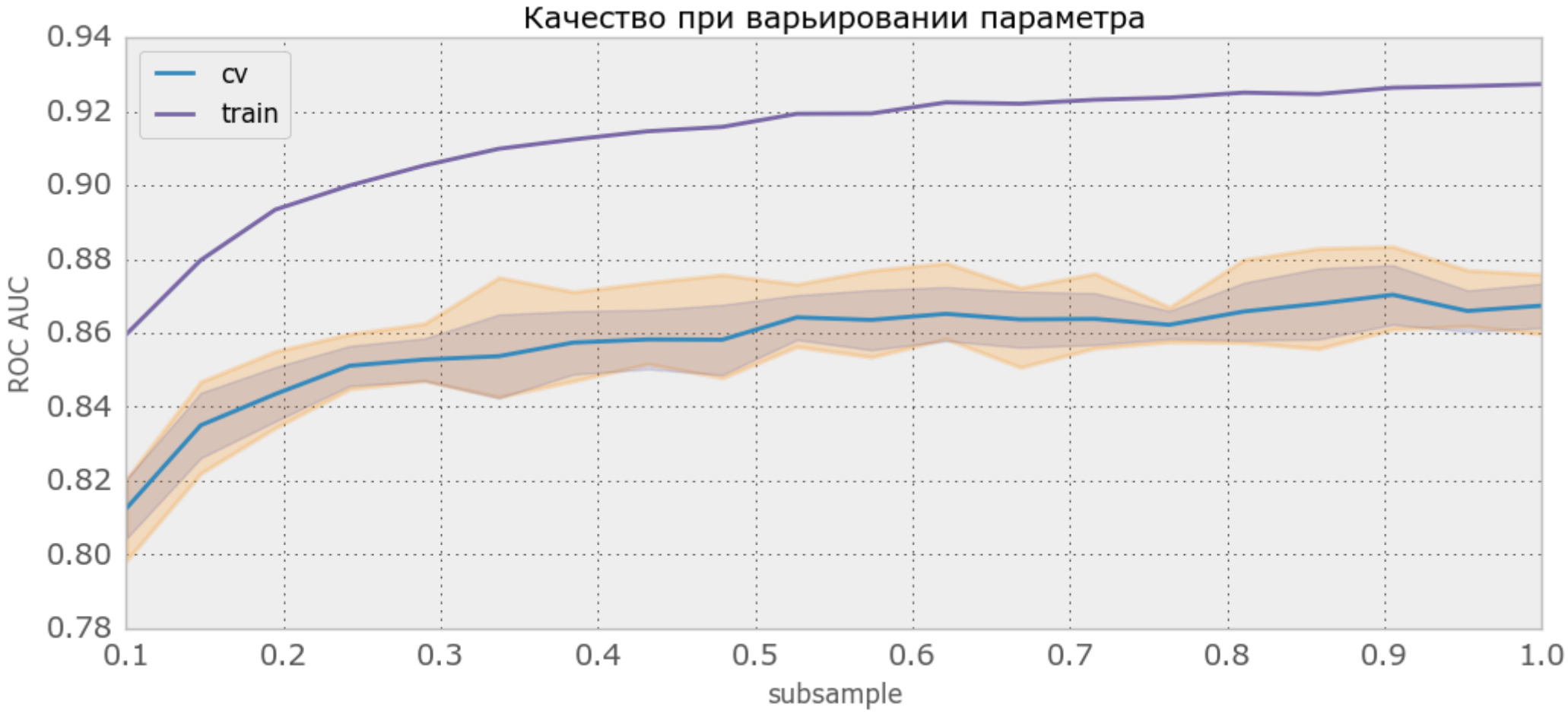
часто «нужны все объекты»

часто зависимости нет

Чем больше – тем однотипнее деревья

Что из этого следует?

Настройка параметров: размер подвыборки `sampsize` (СберБанк)



Всю выборку надо использовать по максимуму!

Настройка параметров: число признаков `mtry / max_features`

Самый серьёзный параметр

умолчание: $\sqrt{n}$ **– классификация,** $n/3$ **– регрессия**

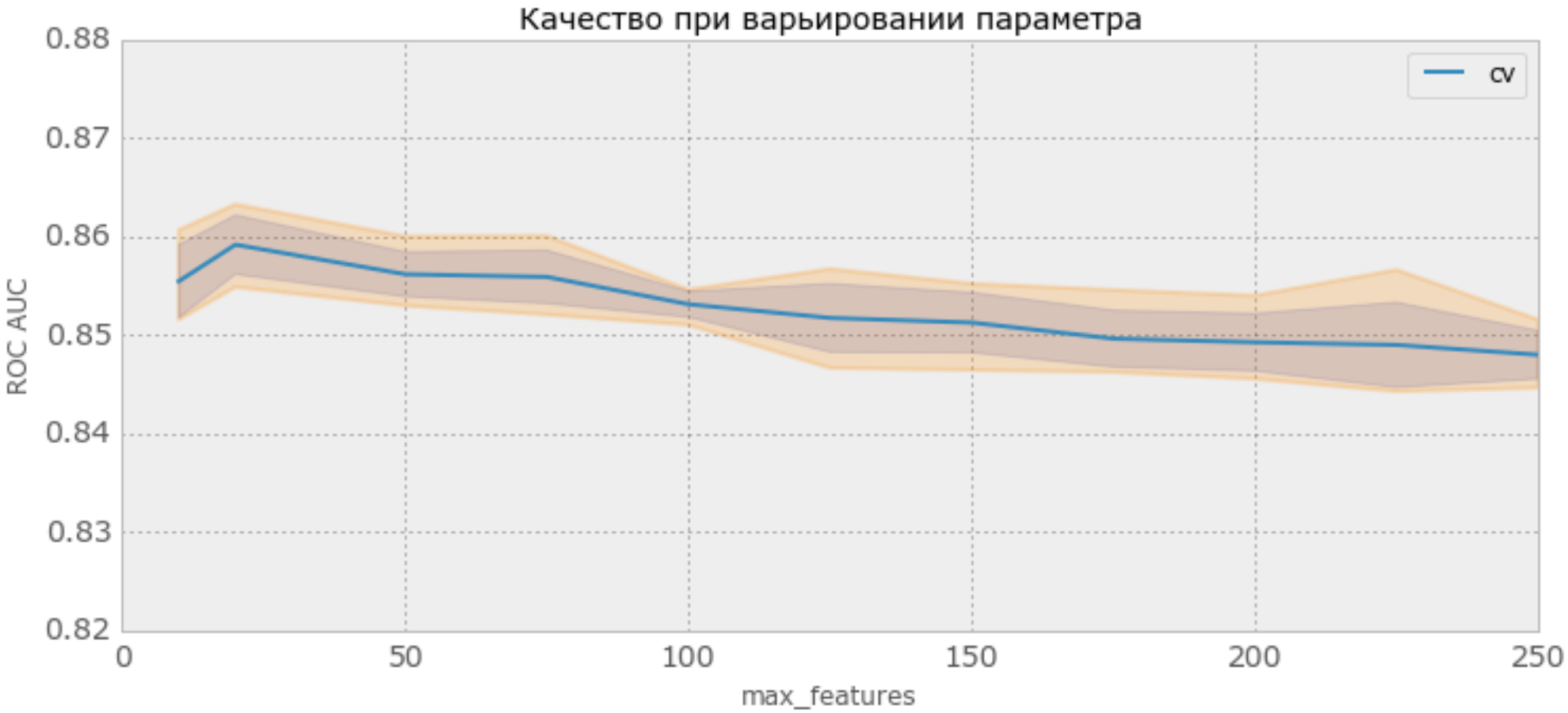
Зависимость унимодальная
Настраивается в первую очередь

Зависит от числа шумовых признаков
Надо перенастраивать при добавлении новых признаков

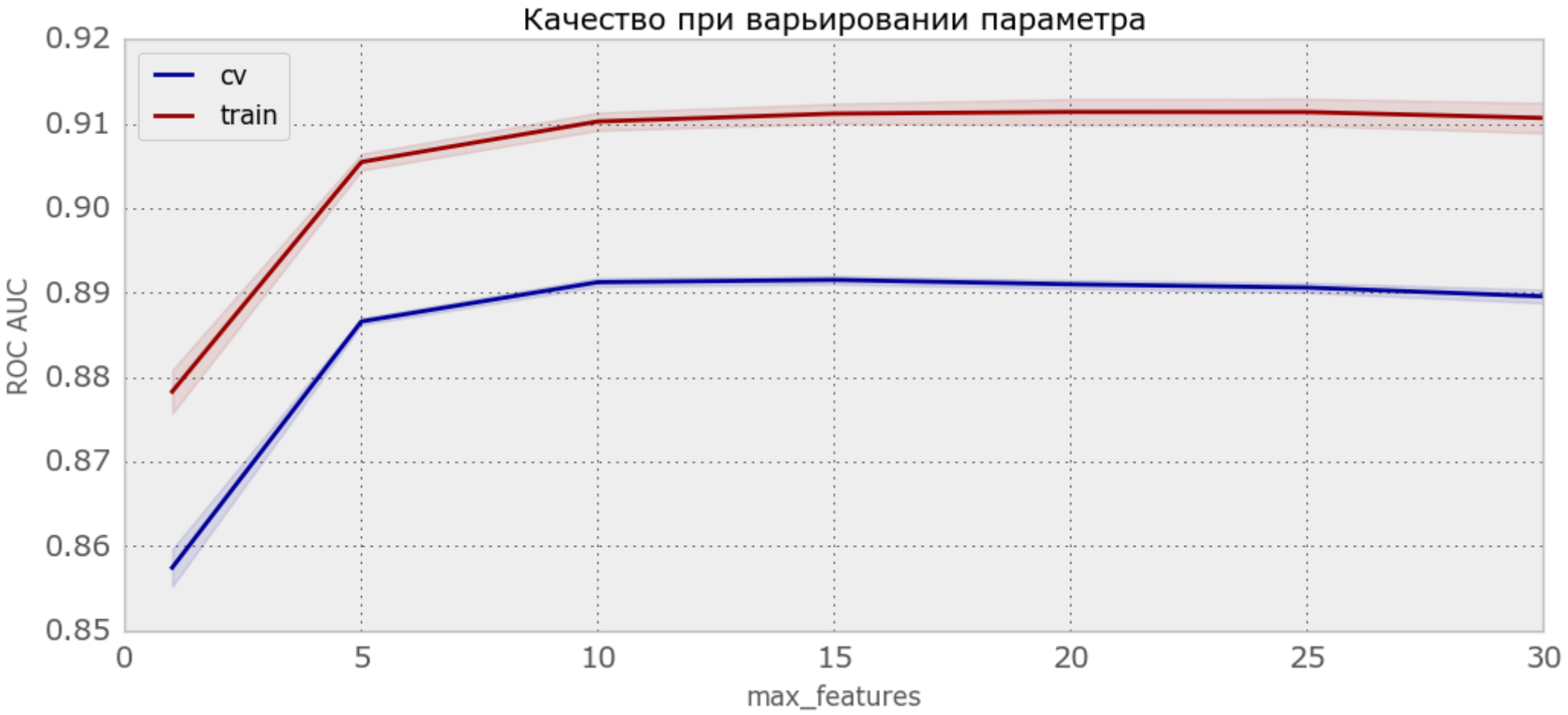
Чем больше – тем однотипнее деревья.
Чем больше – тем медленнее настройка!

Kaggle: часто суммируют алгоритмы с разными mtry.

Настройка `mtry / max_features` (СберБанк)

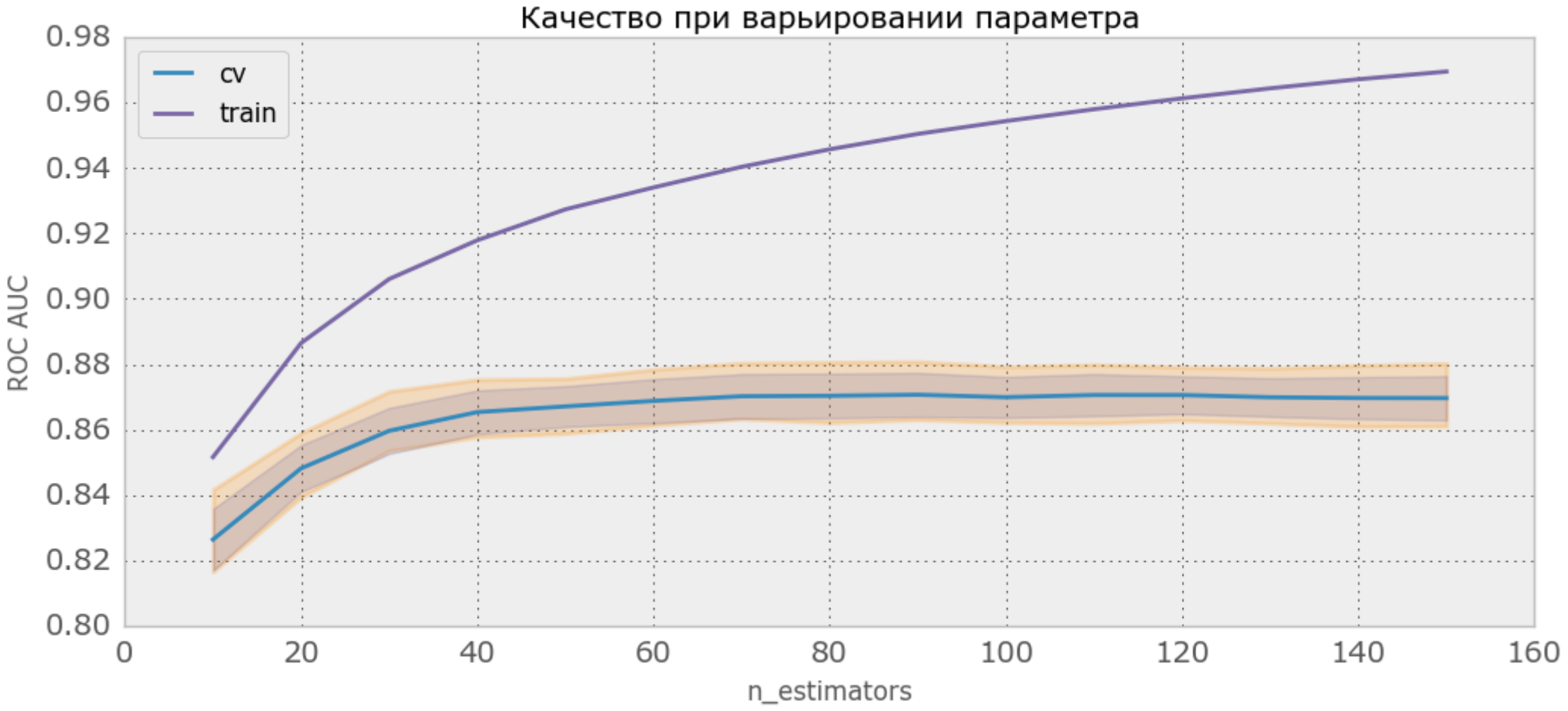


Настройка mtry / max_features (ed Бозон)



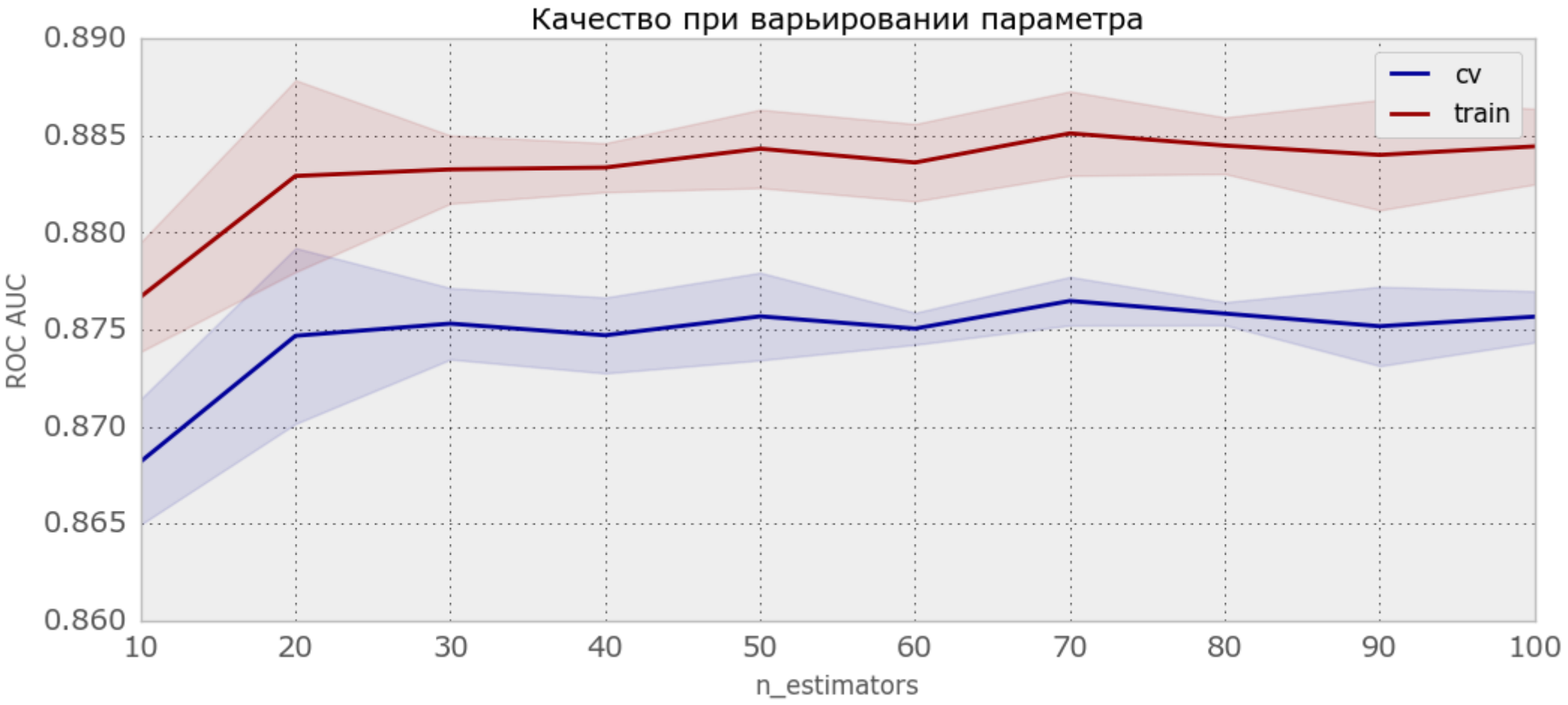
в задаче ~ 33 признака

Настройка параметров `ntree / n_estimators` (СберБанк)



Чем больше деревьев – тем лучше!

Настройка параметров `ntree / n_estimators` (ed Бозон)



Настройка параметров `ntree / n_estimators` (СберБанк)

Чем больше – тем лучше!

Проблемы:

- **как использовать при настройке параметров очень большое число деревьев**
- **что делать, если не помещаются в память… (в R)**

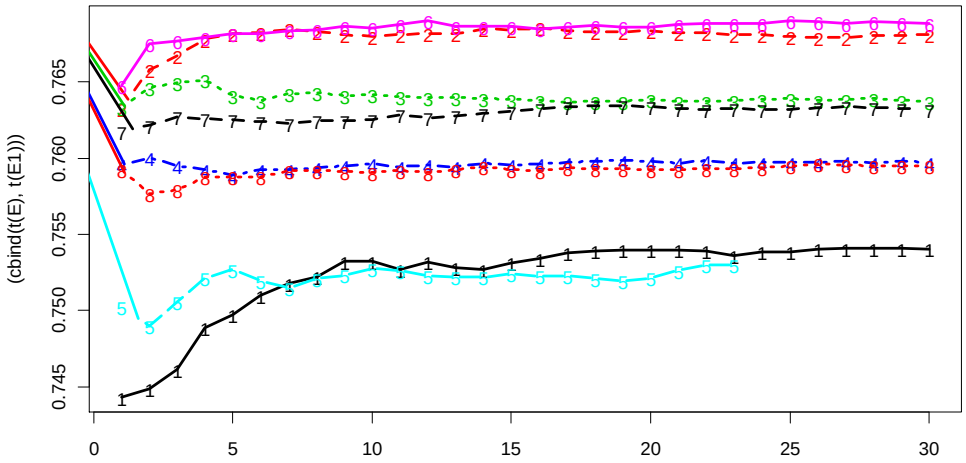
Совет

Много деревьев - много памяти.

Строим по частям (можно даже по одному...):

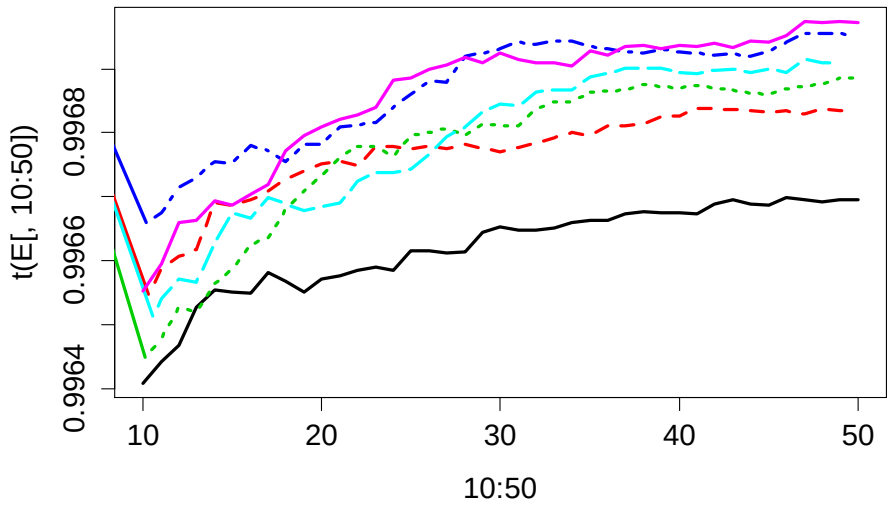
```
a=0
for (i in 1:150)
{
  model <- randomForest(T[, -1], T[, 1],
            do.trace=TRUE, # нужно ли выводить сообщения
            importance=FALSE, # нужно ли вычислять важность признаков
            ntree=10, # сколько деревьев строить
            mtry=PAR1[par1], # сколько признаков выбирать для рассмотрения
            replace=FALSE # как формировать выборку
            )
  a = a + predict(model, T2[, -1])
  e =  colAUC(a, T2[, 1], plotROC=FALSE)
  print(i)
  print(e)
}
```

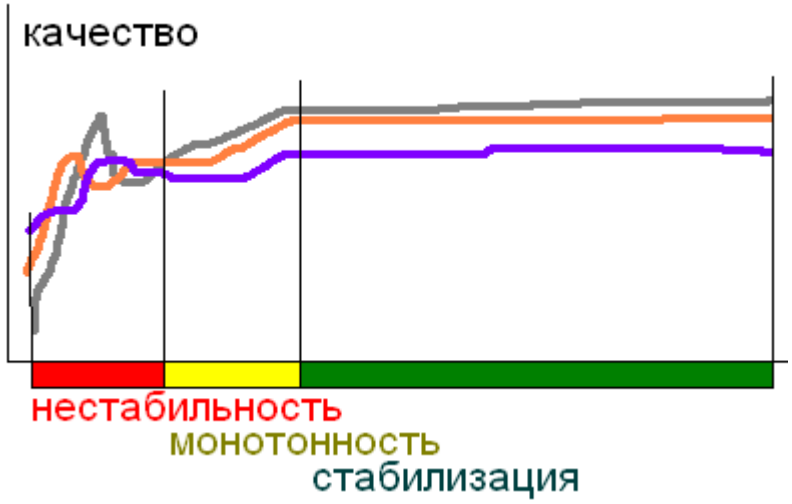
Совет по числу деревьев: область устойчивости функционала



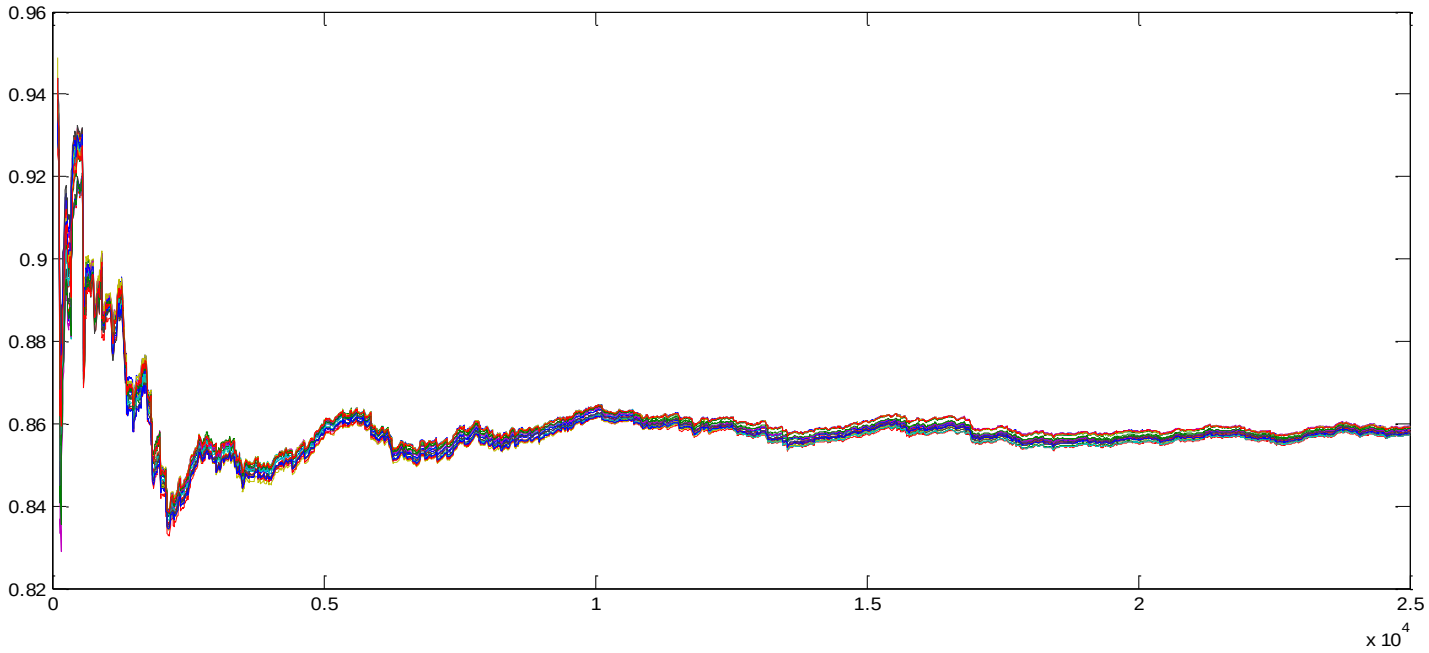
качество от числа деревьев при разных mtry (wikimart)

тут по 100 деревьев в тике

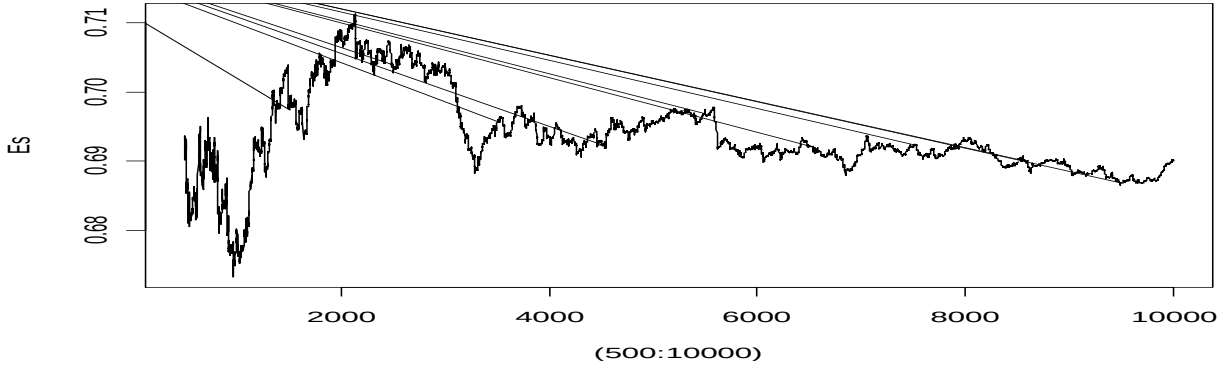


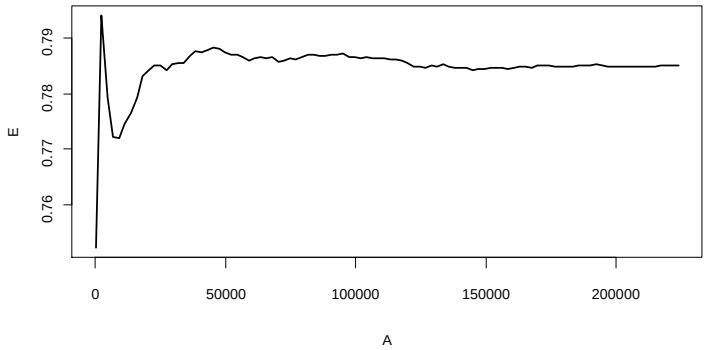


Область устойчивости функционала

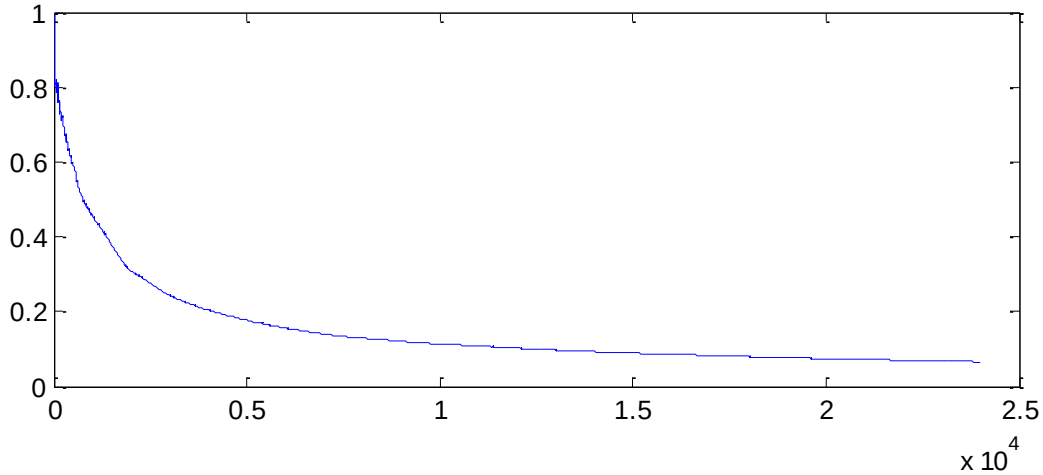


AUC в зависимости от объёма контрольной выборки

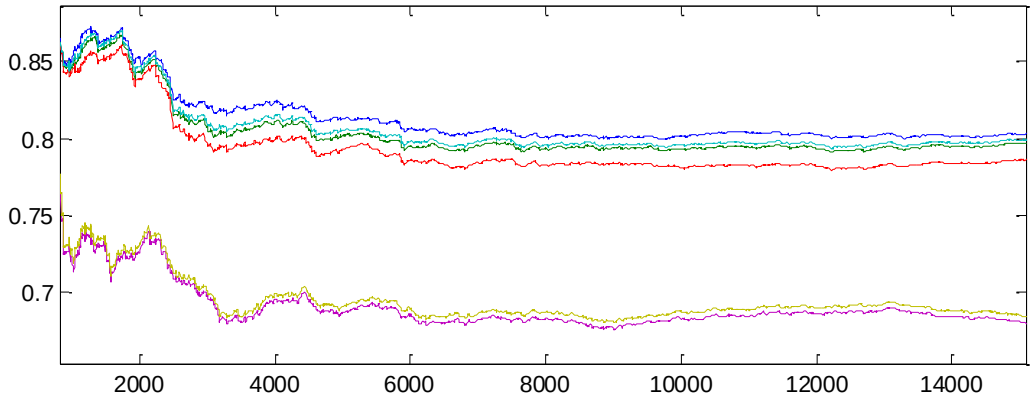




Иногда причины некорректности:
алгоритм выдаёт похожие оценки
(AUC вычисляется в среднем/худшем случае).



Число уникальных оценок на число оценок



Показатели AUC при этом…

Настройка параметров: число объектов в листе, число объектов для расщепления, максимальная глубина дерева

От параметров существенно зависит скорость построения леса

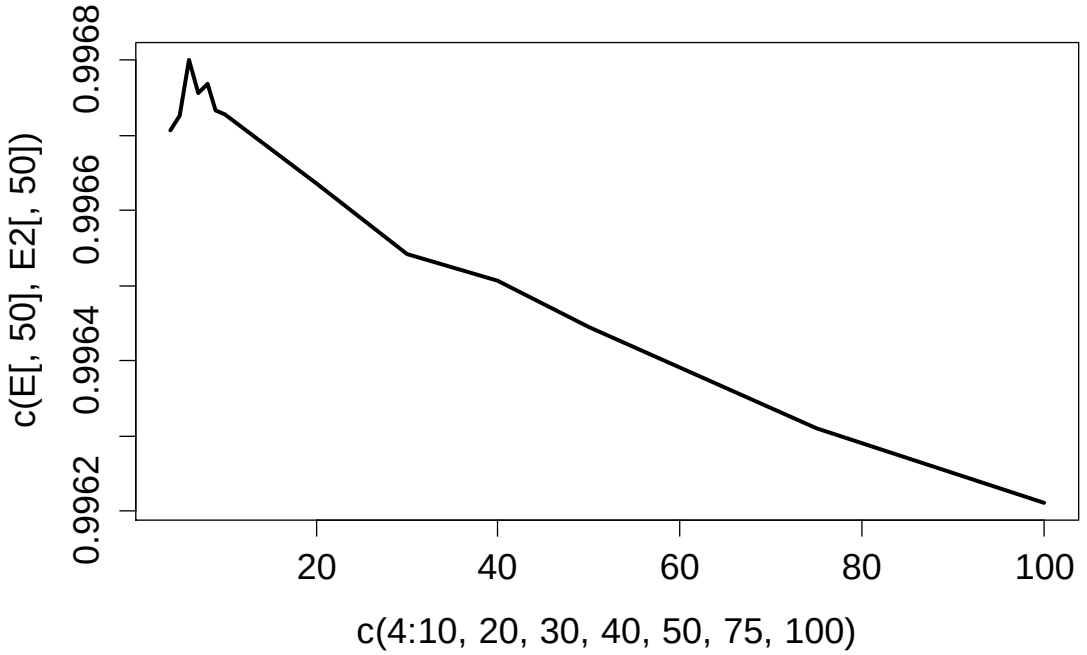
Оптимальные значения, как правило, - несколько объектов в листе.

Настраиваются не в первую очередь

В классическом случайном лесе деревья строятся до исчерпания выборки…

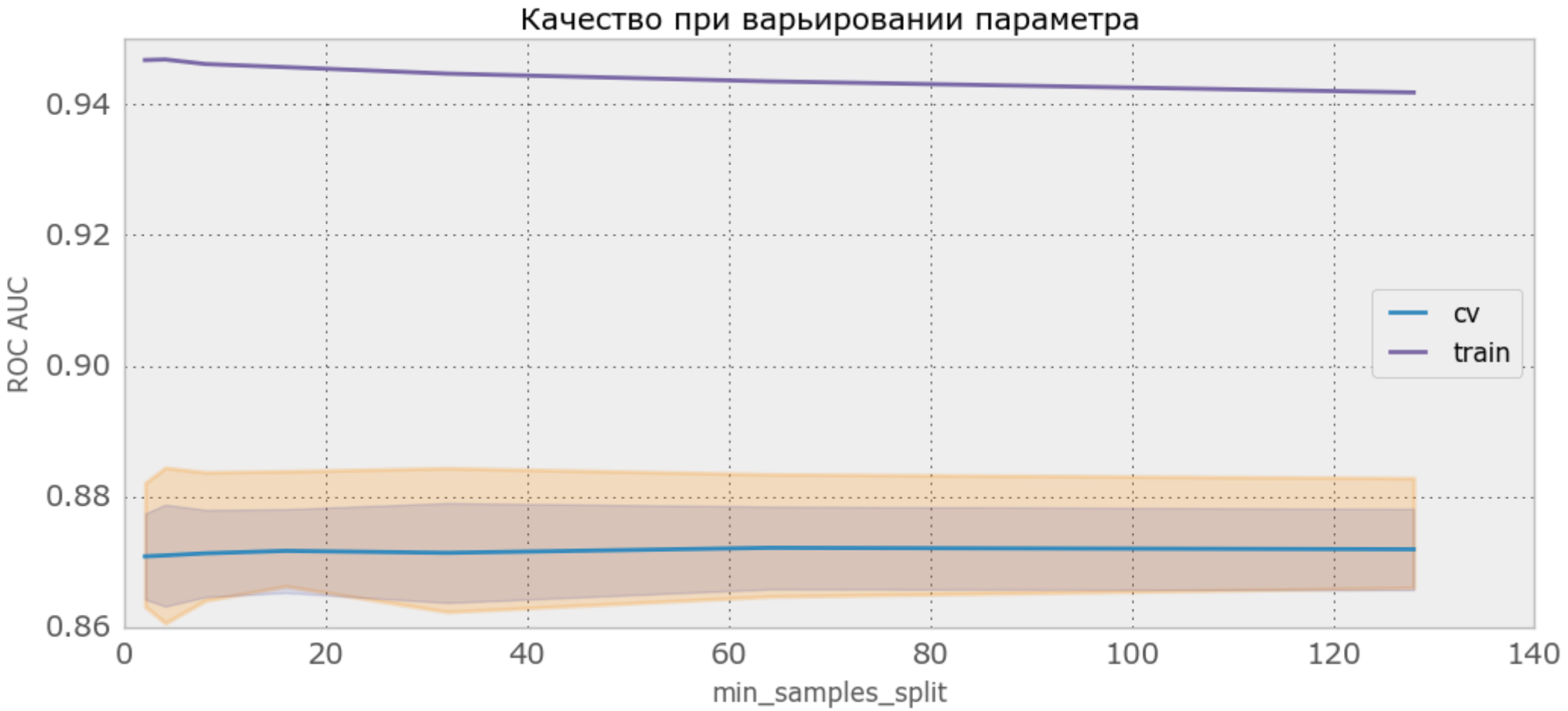
«Good results are often achieved when setting `max_depth=None` in combination with `min_samples_split=1`»

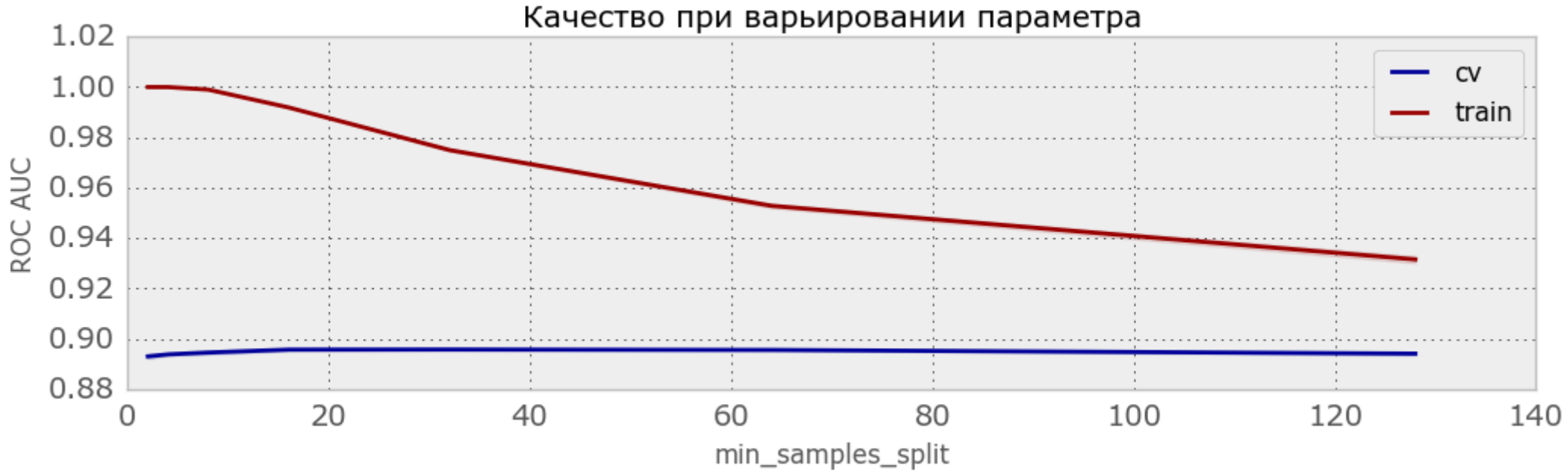
randomForest: nodesize

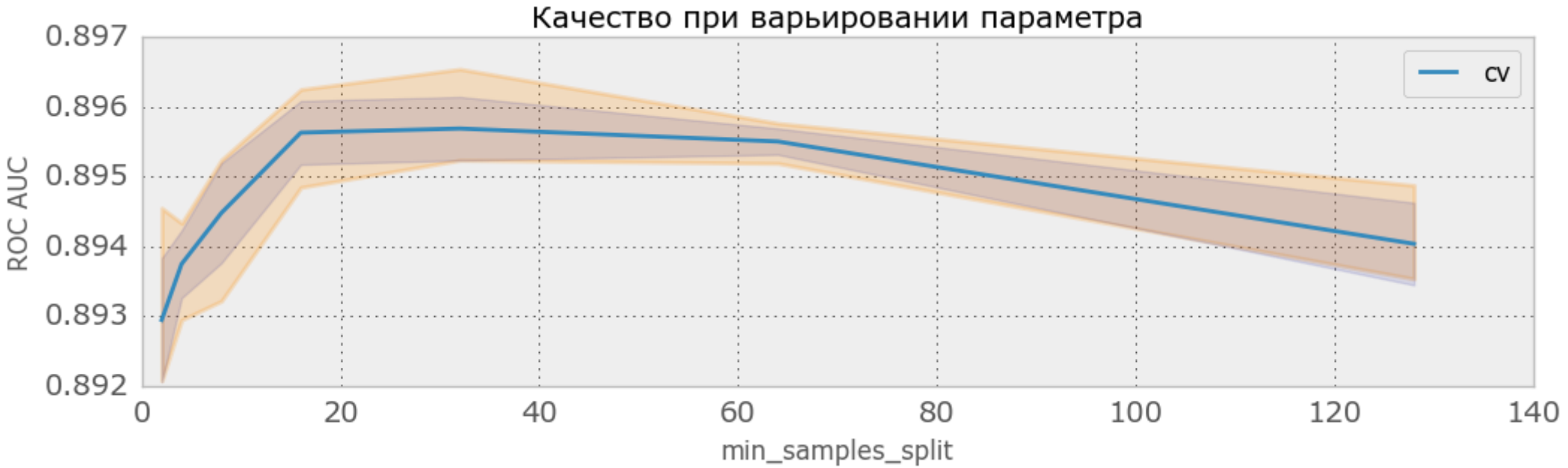


умолчание: 1 – классификация, 5 – регрессия

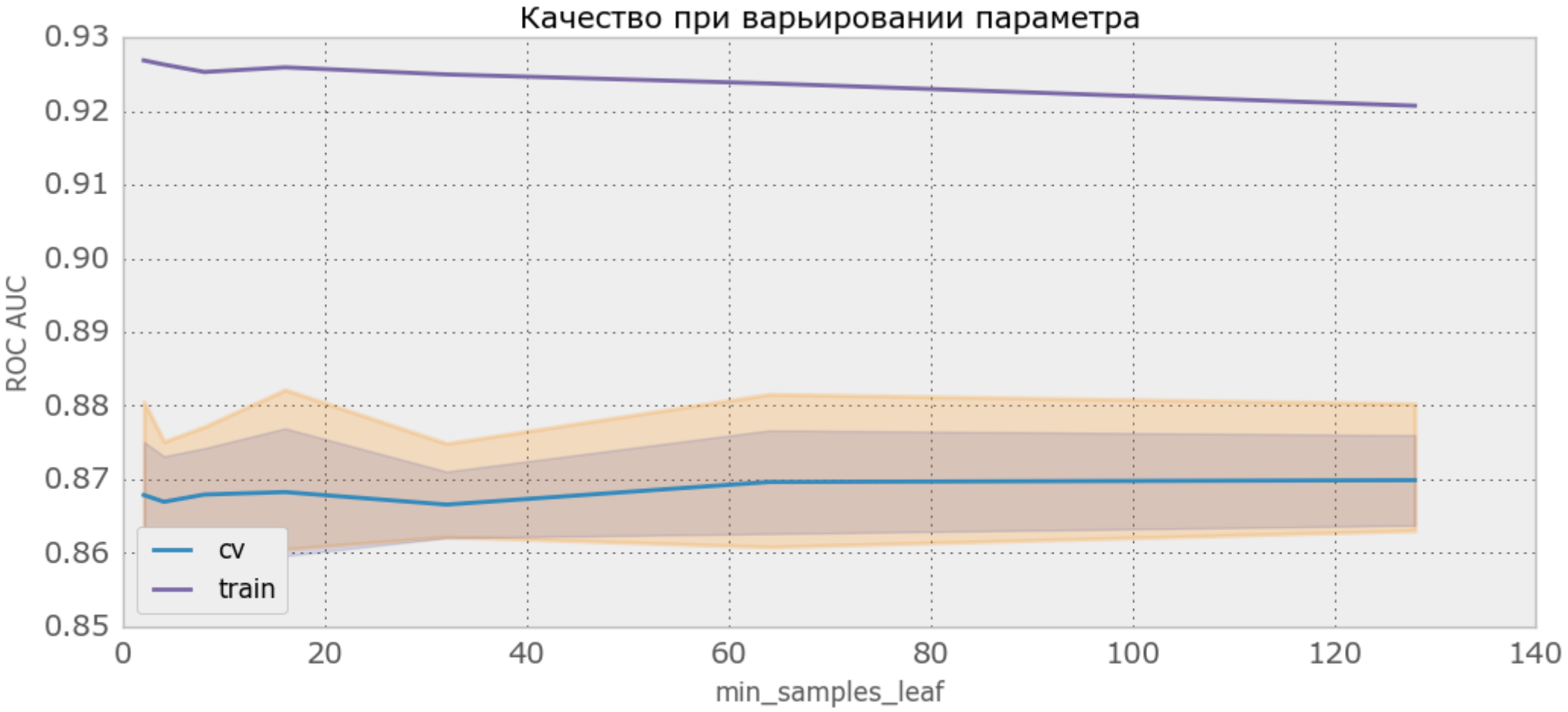
RandomForestClassifier: min_samples_split (СберБанк)

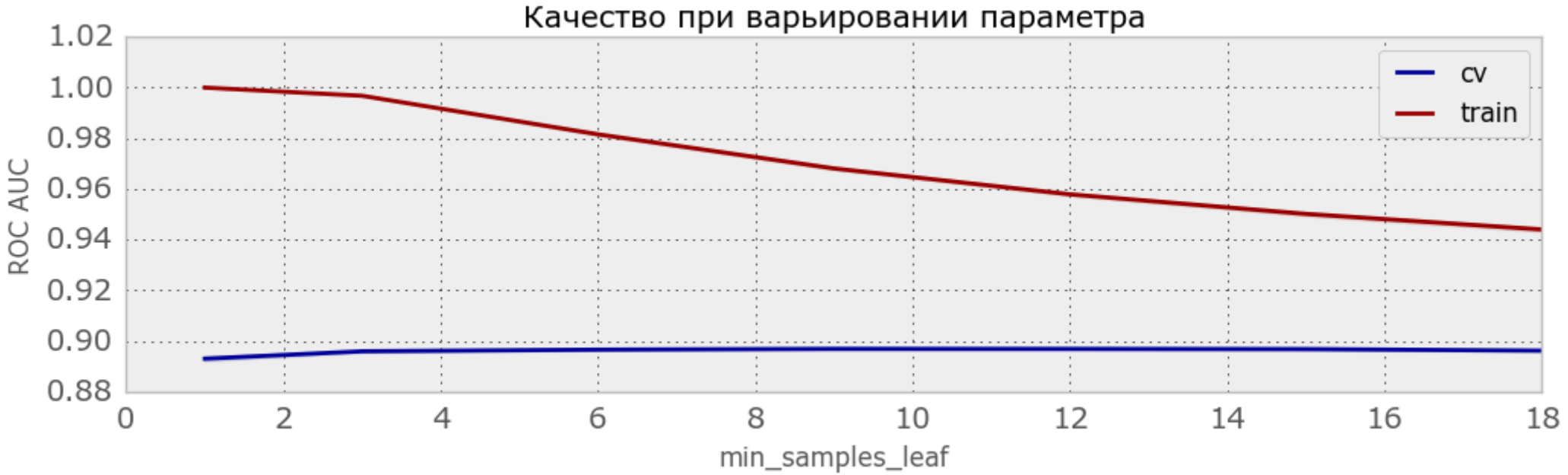


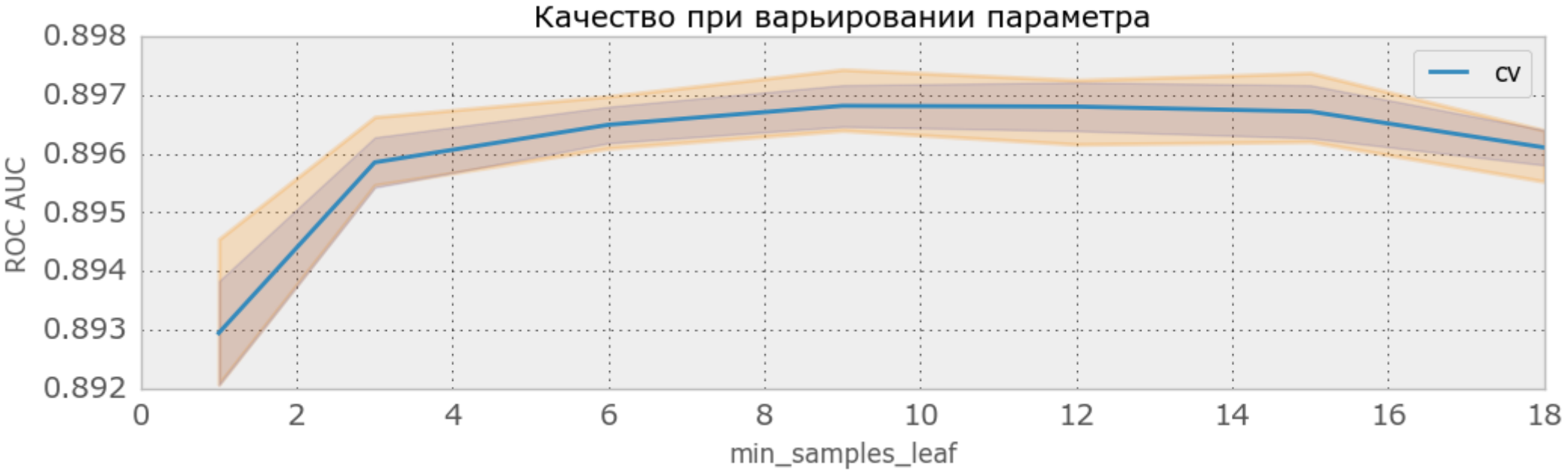

Качество при варьировании параметра
ROC AUC
1.02
1.00
0.98
0.96
0.94
0.92
0.90
0.88
0
20
40
60
80
100
120
140
min_samples_split
cv
train


Качество при варьировании параметра
ROC AUC
0.897
0.896
0.895
0.894
0.893
0.892
0
20
40
60
80
100
120
140
min_samples_split
cv

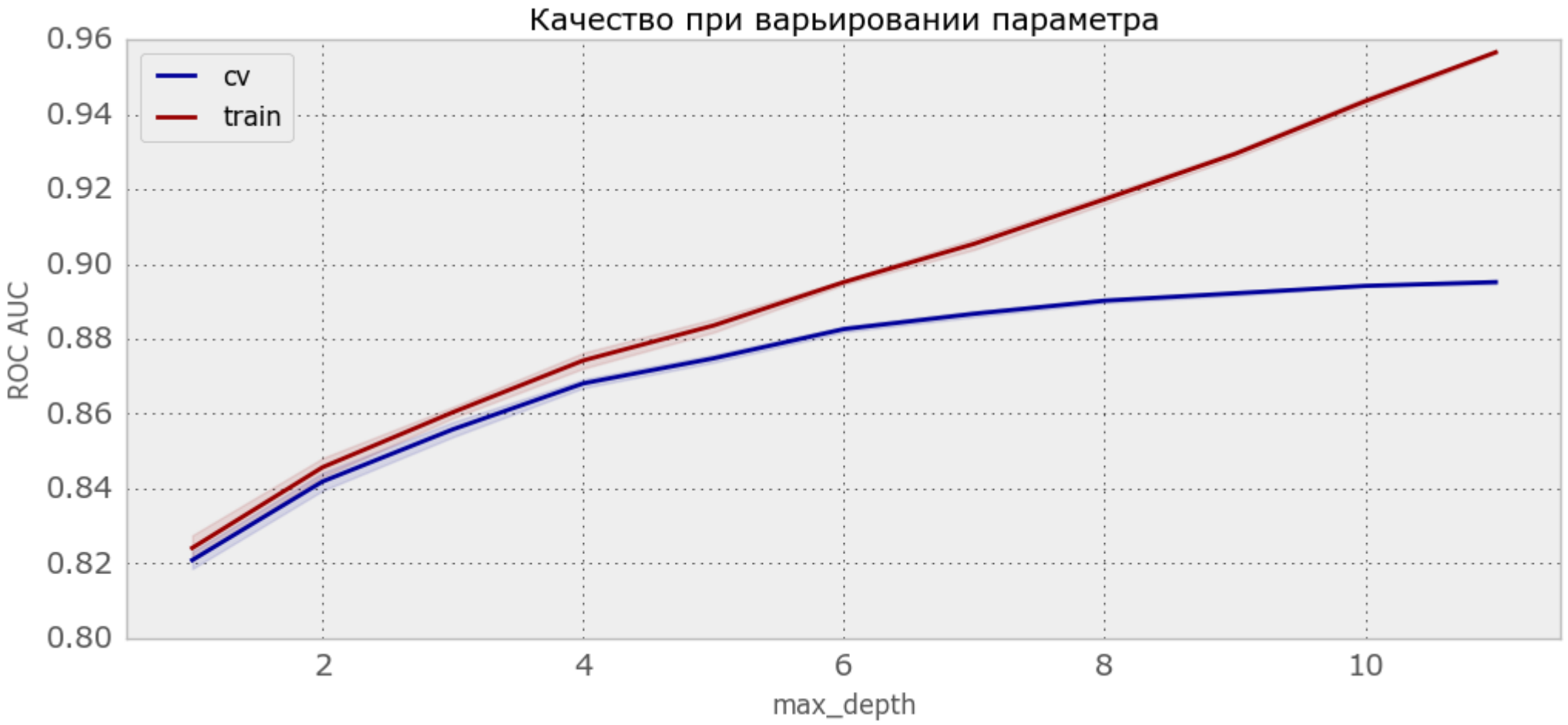
RandomForestClassifier: min_samples_leaf (СберБанк)




Качество при варьировании параметра
ROC AUC
min_samples_leaf
cv
train
1.02
1.00
0.98
0.96
0.94
0.92
0.90
0.88
0
2
4
6
8
10
12
14
16
18

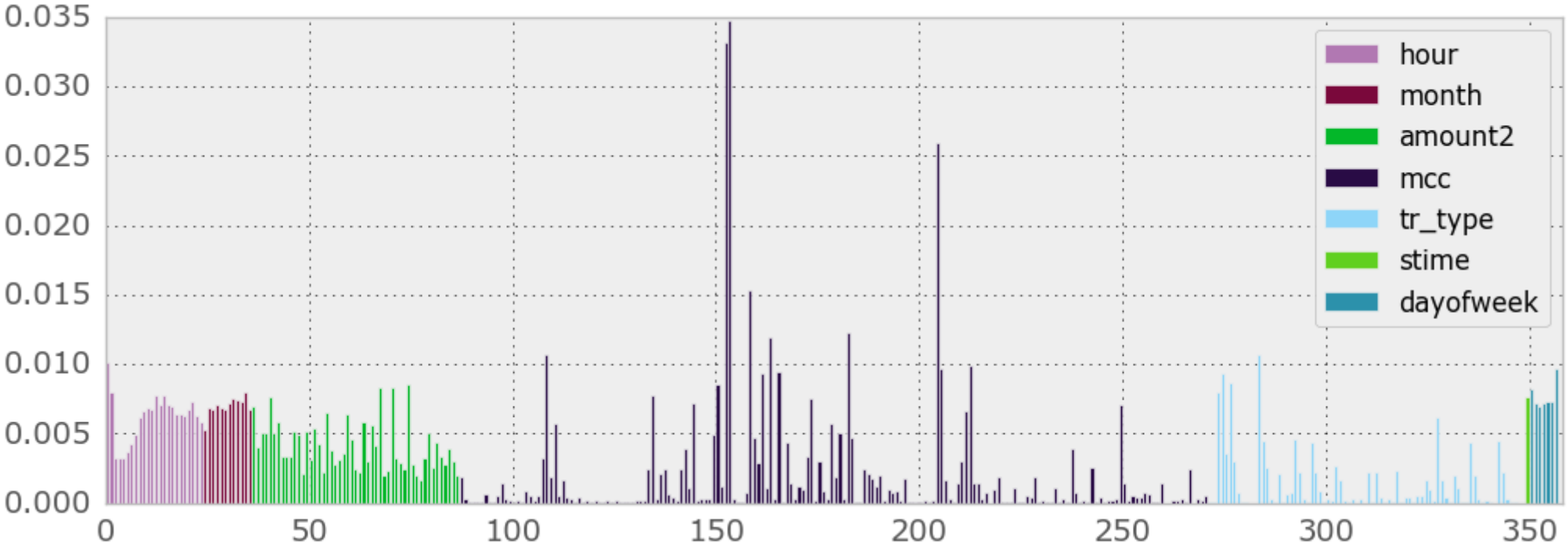

Качество при варьировании параметра
ROC AUC
min_samples_leaf
cv
0.898
0.897
0.896
0.895
0.894
0.893
0.892
0
2
4
6
8
10
12
14
16
18

Глубина дерева: max_depth (СберБанк)



Как правило, чем больше, тем лучше!

Важность признаков (СберБанк)



rf.feature_importances_

```
rf = RandomForestClassifier(n_estimators=1000, max_features=30, n_jobs=-1)
rf.fit(X, y)
plt.bar(np.arange(len(rf.feature_importances_)), rf.feature_importances_,
color='black')
```

Можно сразу увидеть важные признаки и целые группы…

Как вычислить важность?

Плохой метод – чем чаще выбирался признак, тем лучше.

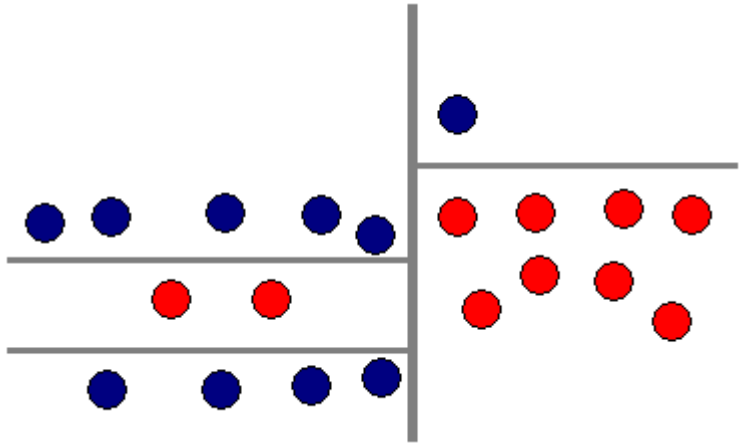
Почему?

Как использовать важность?

Не увлекаться выбрасыванием неважных признаков

Почему?

Как вычислить важность?



По хорошим признакам меньше всего расщеплений…

Как использовать важность?

не увлекаться выбрасыванием неважных признаков:

- **оценка качества признаков не всегда адекватная**
- **если много хороших коррелированных признаков, то их важность будет маленькая**
- **не рекомендуют оценивать важность и решать одним и тем же алгоритмом**

`importance(model)` в R

%IncMSE

OOB (out of bag)

1. **Вычисляем качество Q на OOB**
2. **Для *i*-го признака – делаем случайную перестановку значений, вычисляем качество Q_i на OOB**
3. **Информативность *i*-го признака =** $\max(Q - Q_i, 0)$

Запомните приём!

Вместо качества можно использовать что-то другое…

~ доля верно классифицирующих деревьев

`IncNodePurity` в R

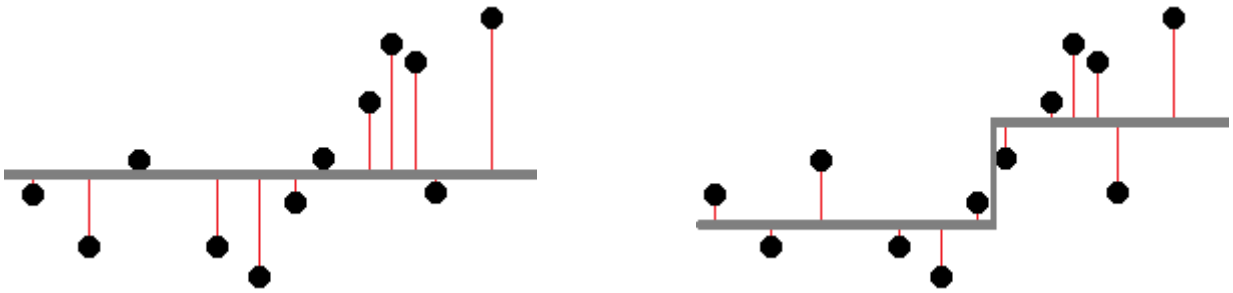
При каждом расщеплении –

$$RSS_{\text{old}} - RSS_{\text{new}}$$

Берётся сумма по всем расщеплениям для конкретной переменной, по всем деревьям.

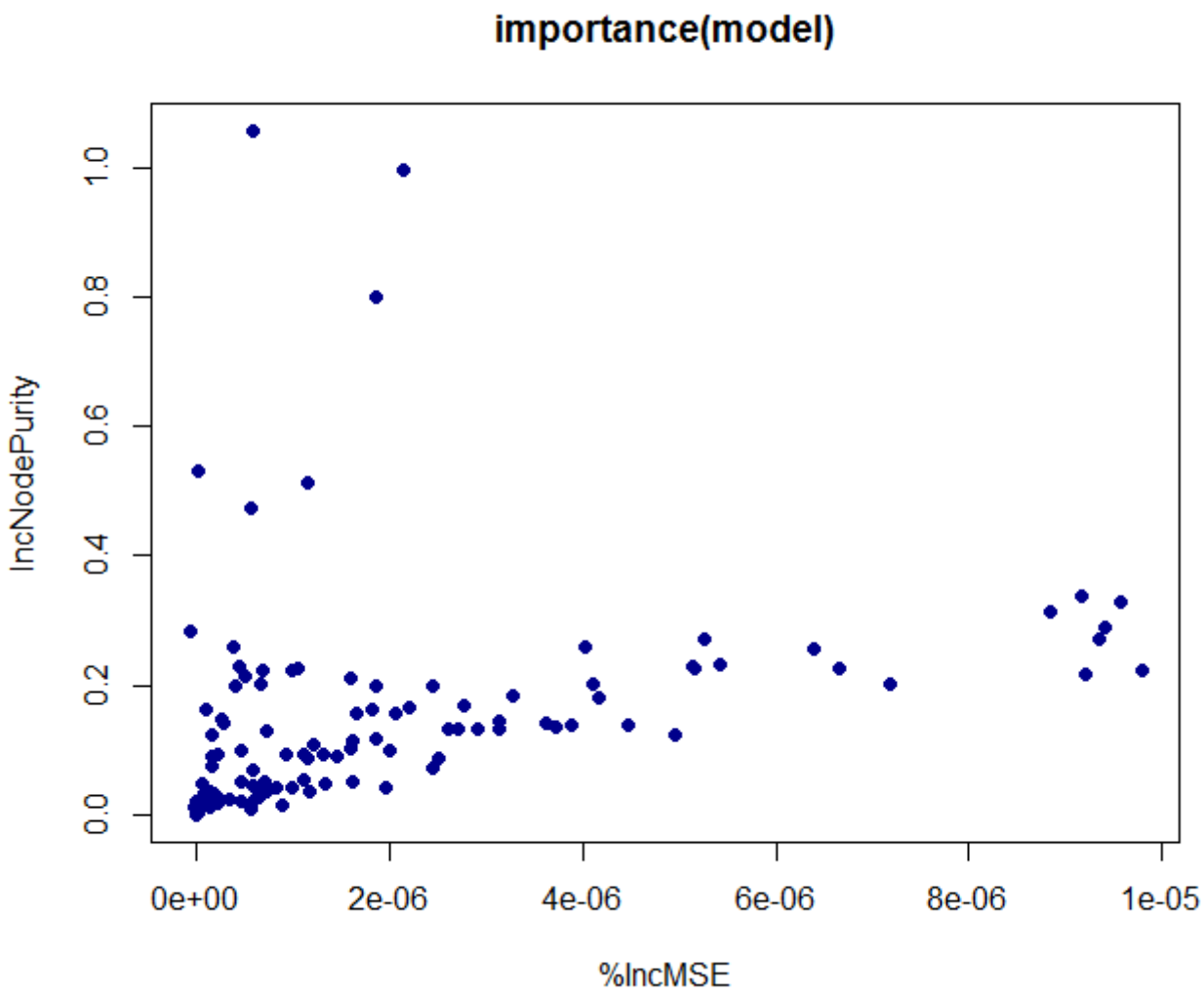
residual sum of squares (RSS)

$$\sum_{i\in\text{left}}(y_{\text{left}} - y_i)^2 + \sum_{i\in\text{right}}(y_{\text{right}} - y_i)^2$$



В sklearn (`feature_importances_`) аналогичная идея с критерием Gini

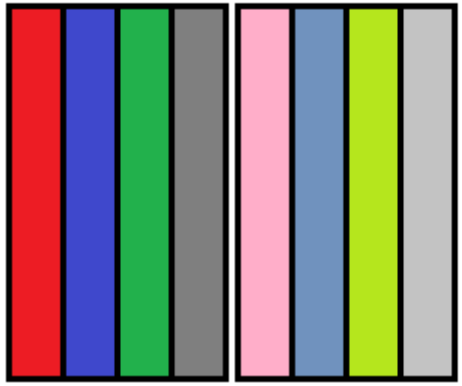
Разные важности (скоринг)



Boruta (идея)

1. Добавить к исходным признакам их перемешанные (shuffle) копии (shadow features / признаки-призраки)

признаки перемешали



2. Запустить RF – вычислить Z-меру, MSZA = max(Z-score) на перемешанных признаках

Здесь важность – потеря точности классификации, вычисляется для каждого дерева, из всех содержащих рассматриваемый признак

Приём – shuffle

Boruta (идея)

3. Запустить RF на исходных данных

Если Z-score << MSZA, то признак плохой
Если Z-score >> MSZA, то признак хороший

Можно удалить плохие признаки и повторить процедуру

Что такое Z-score

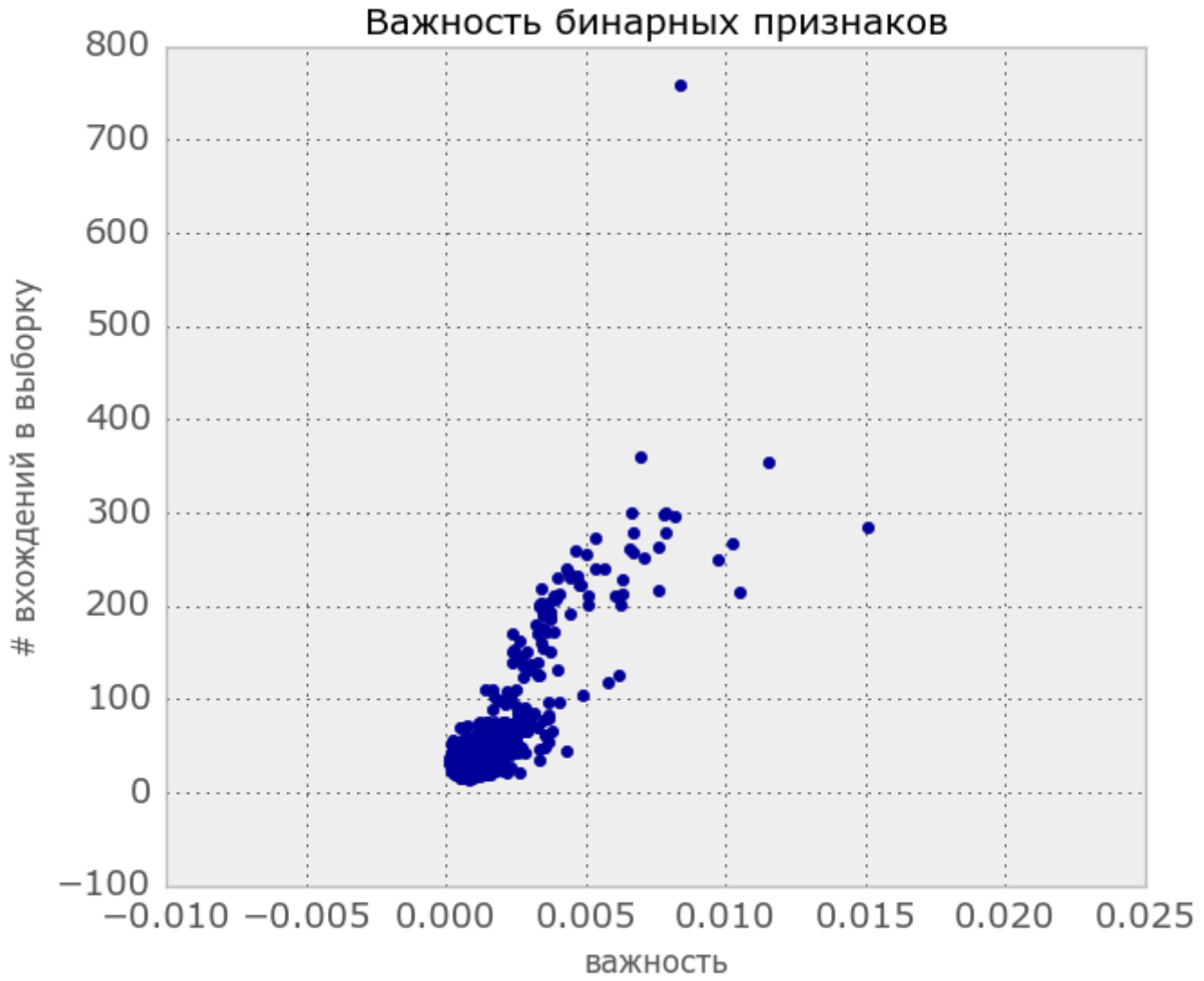
$$z = \frac{x - \mu}{\sigma}$$

здесь ~ rf_importance / дисперсия

ACE (Artificial Contrasts with Ensembles)

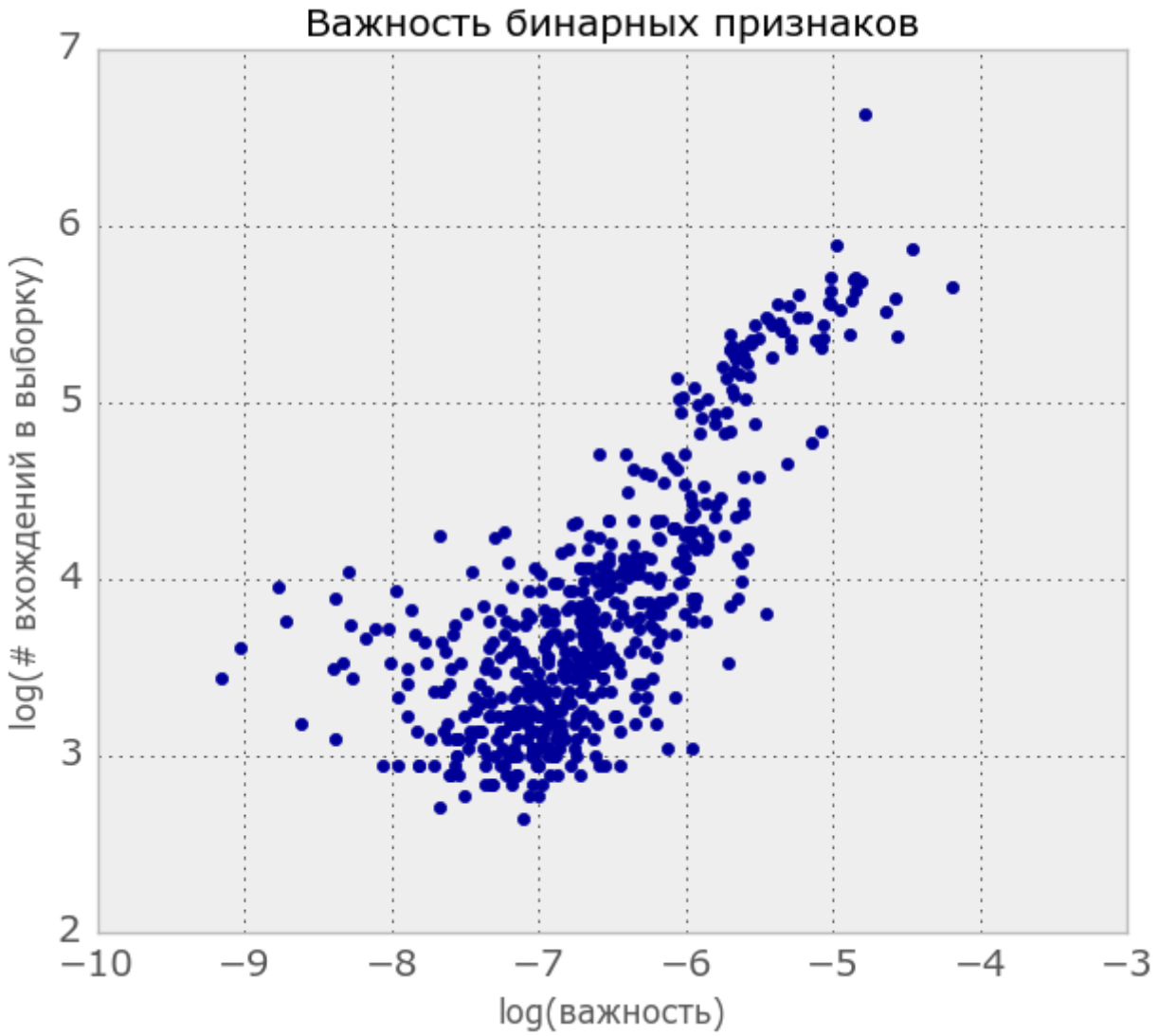
Аналогично, но удаляются хорошие признаки!

Важность признаков (СберБанк)



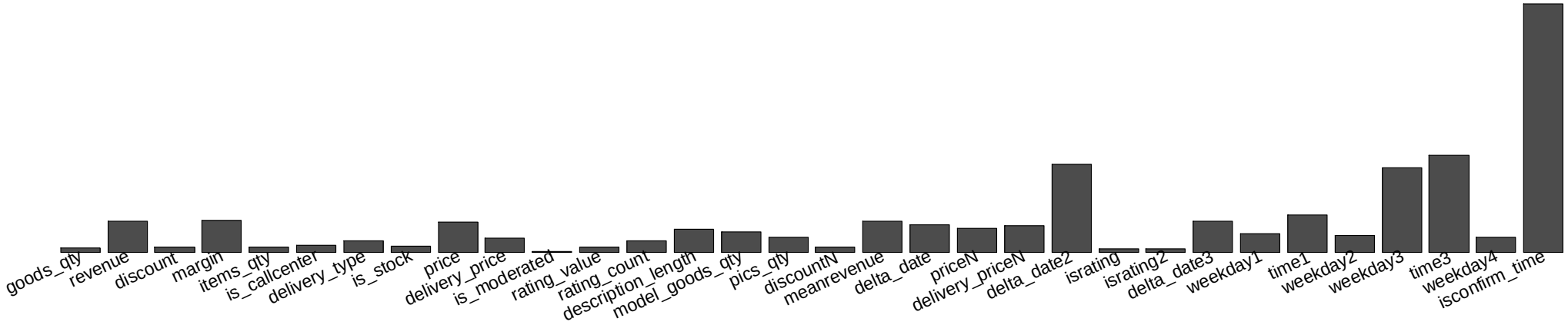
По вертикали – число ненулевых значений признака

Важность признаков (СберБанк)



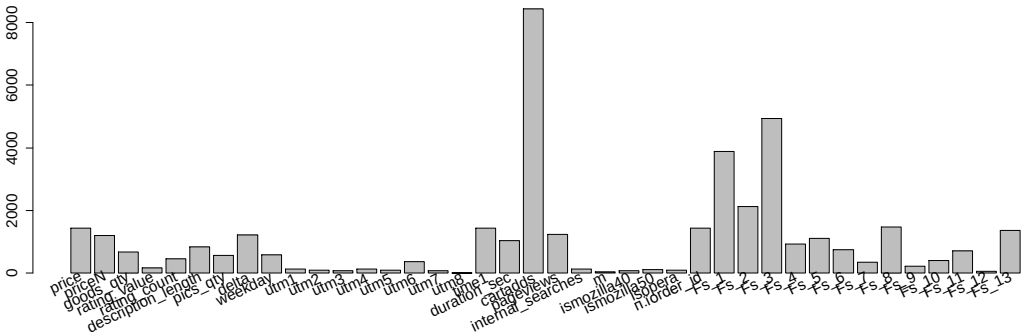
Видна группа очень неплохих признаков;)

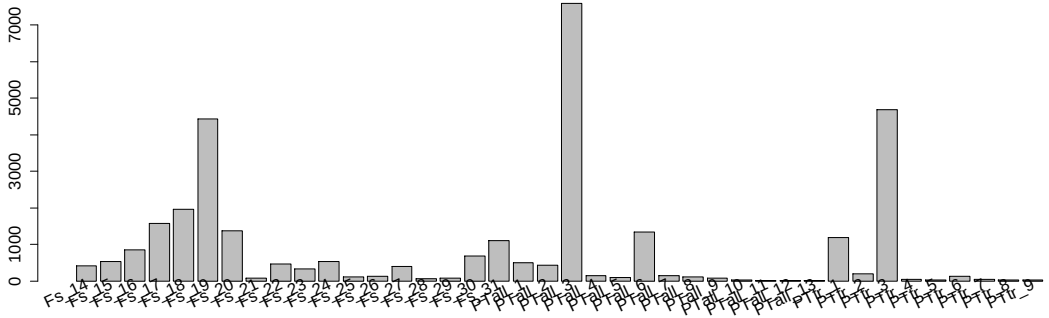
randomForest: рейтинг признаков

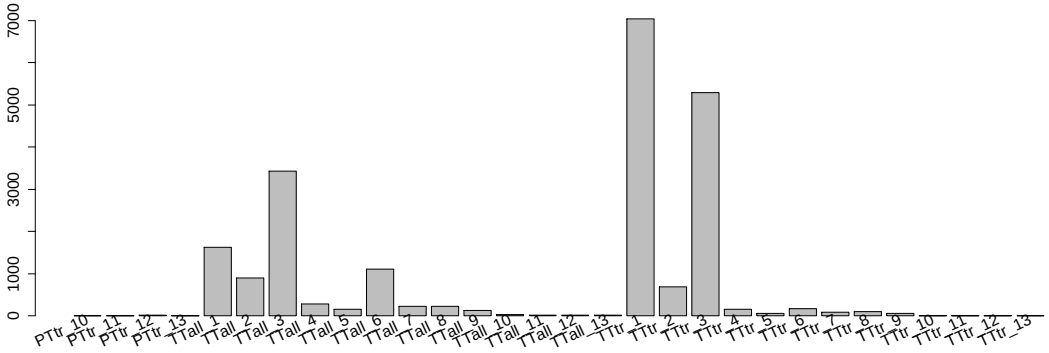


	IncNodePurity		
goods_qty	121.72230	discountN	150.23239
revenue	922.02043	meanrevenue	933.90907
discount	150.11052	delta_date	832.42437
margin	964.22036	priceN	728.81899
items_qty	164.46660	delivery_priceN	799.86912
is_callcenter	212.33560	delta_date2	2654.63597
delivery_type	345.72847	israting	105.82863
is_stock	167.53406	israting2	93.85632
price	906.85693	delta_date3	941.05108
delivery_price	426.00989	weekday1	549.10152
is_moderated	21.52787	time1	1111.65176
rating_value	154.09028	weekday2	490.17650
rating_count	329.91192	weekday3	2538.65506
description_length	698.50809	time3	2931.93715
model_goods_qty	605.08101	weekday4	436.06949
pics_qty	461.49912	isconfirm_time	7497.17935

Признаки одной задачи (wikimart)





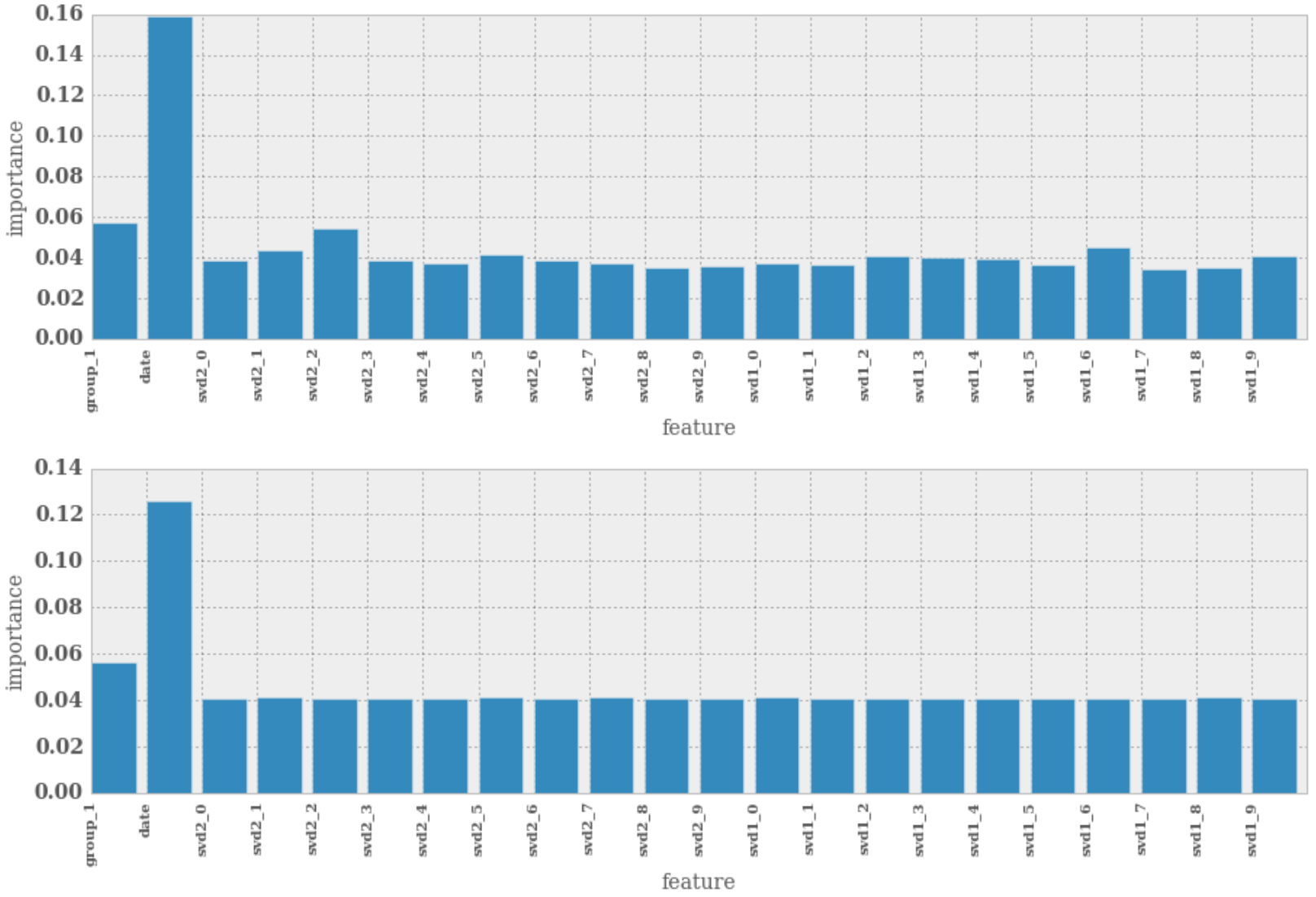


1. **Генерация признаков: Число просмотров страниц типа N, время просмотра, наличие id товара на странице (который потом купили…)**

2. **Видно выделение признаков, отвечающих просмотру определённых страниц.**

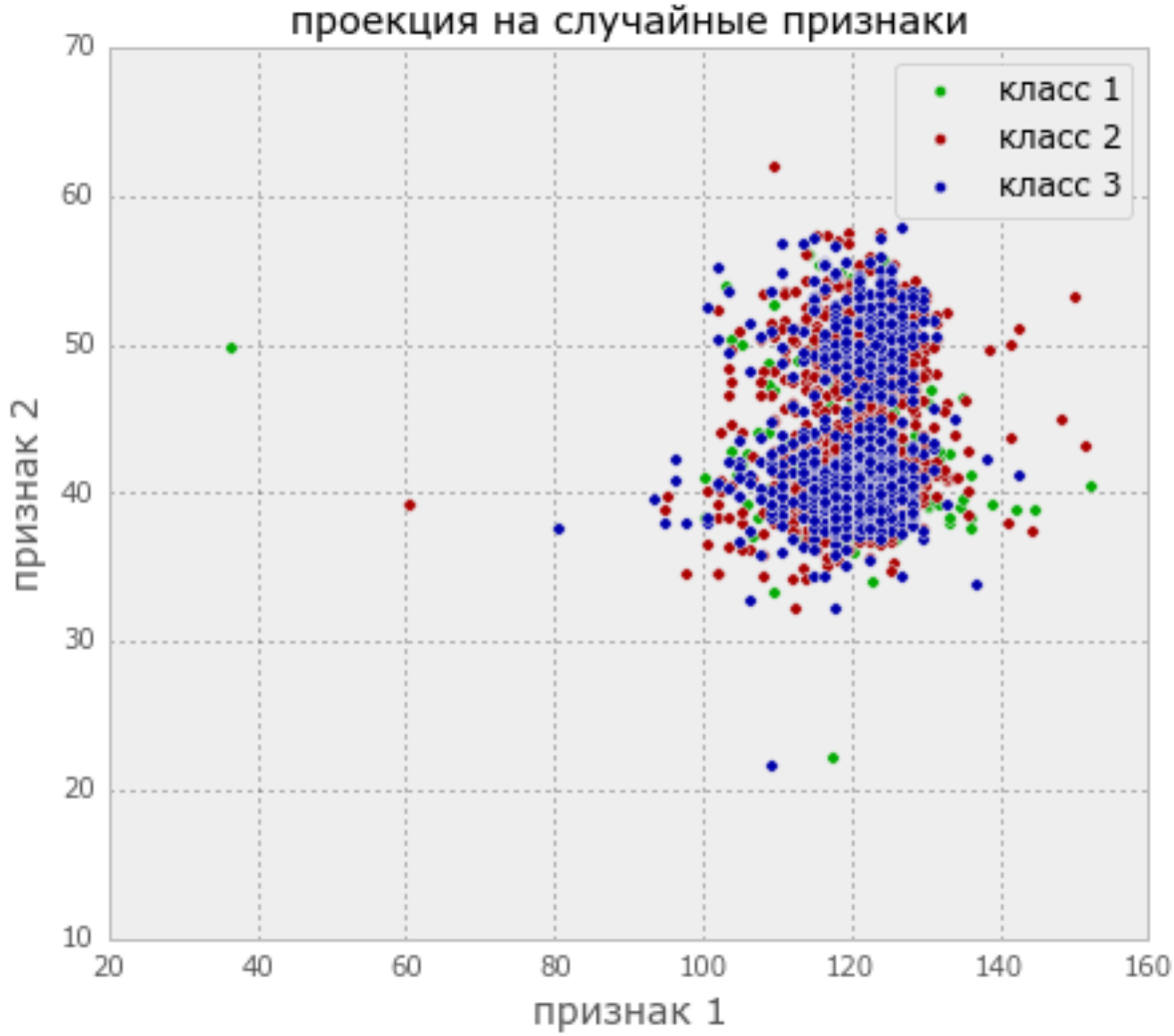
Минус: видим то, что «зашили» в признаки (не видим порядок просмотра и т.п.) – важность делится по коррелирующим признакам

Насколько верить важности



Сделать эксперимент: как отличаются важности случайных признаков (здесь признаки SVD-кодировок и случайные)

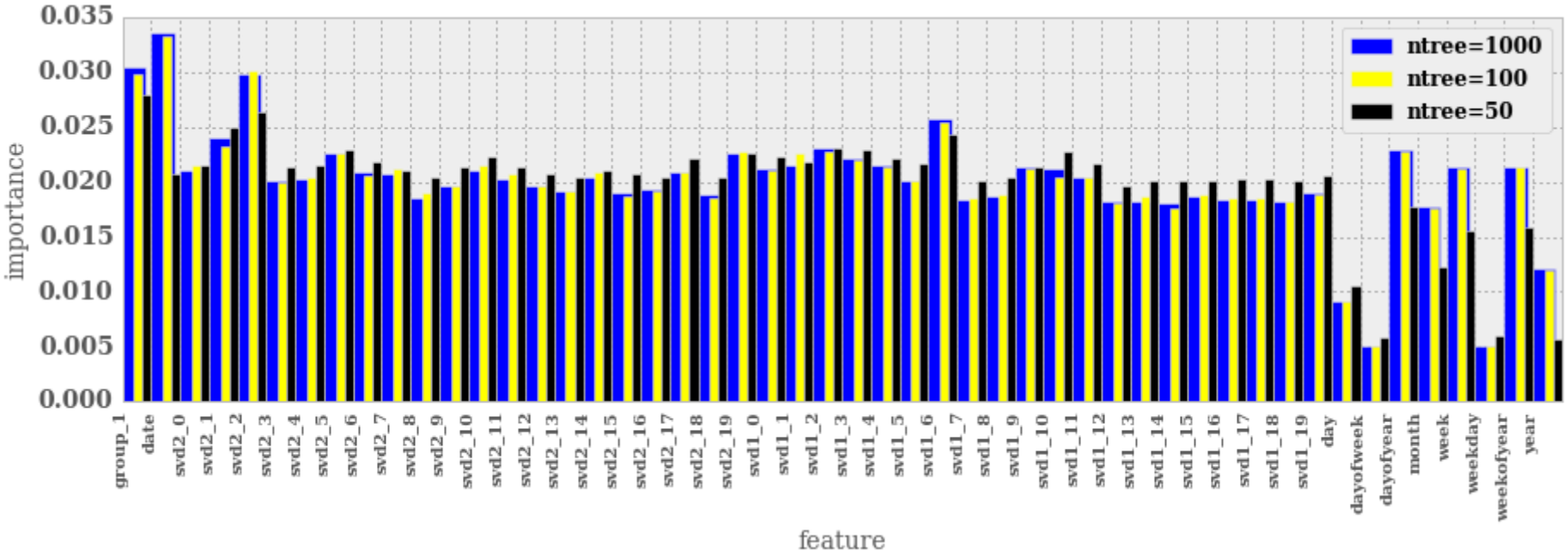
Зачем нужно находить хорошие признаки



Зачем нужно находить хорошие признаки

Быстро понять от чего зависит целевой вектор

При каких параметрах измерять важность



При увеличении числа деревьев есть сходимость!

Проблемы RF

Может долго считаться…

Вместо CV – разбиение на обучение и контроль (hold out)

```
set.seed(100) # подобрать и зафиксировать
I = sample(nrow(T));
T2 = T[I[1:10000],] # важный параметр – объём контроля => ОБУЧЕНИЯ
T = T[I[10001:length(I)],]
```

```
from sklearn.model_selection import train_test_split
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.33, random_state=42)
```

sklearn: не забывать `n_jobs`

Какие тонкости?

ответ ~ ответ на LB

не потерять важные части обучения (сохранение пропорции классов, представительности признаков и т.п.)

размер контроля = область устойчивости функционала

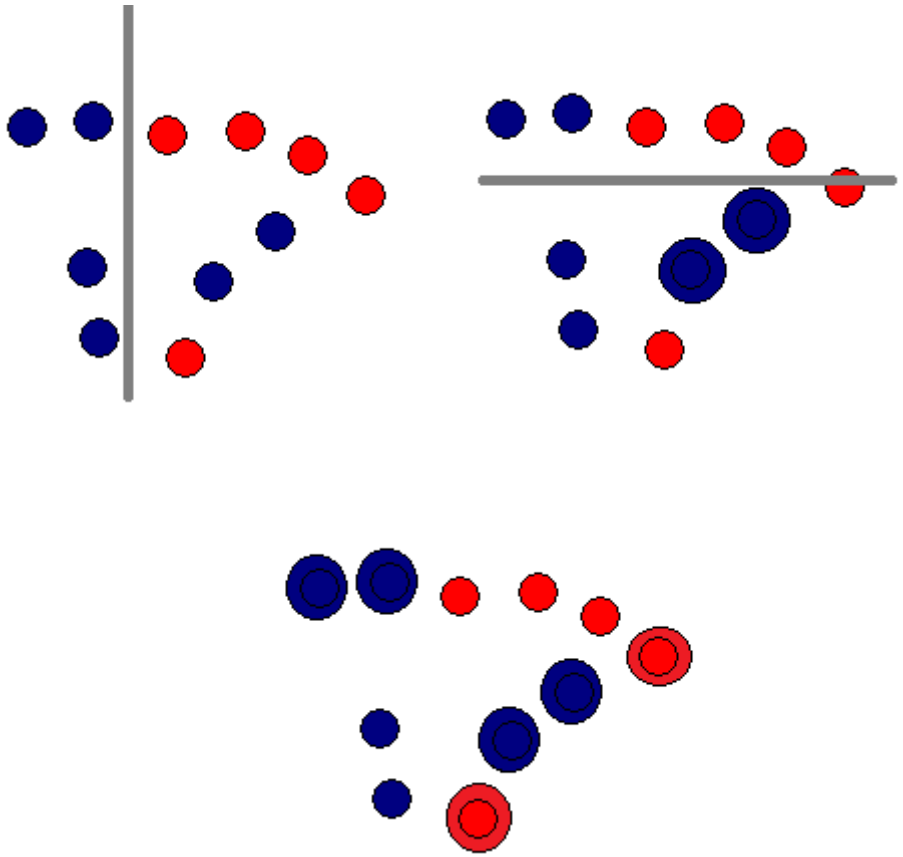
Proximity

при построении деревьев можно много чего считать…

**Чем чаще 2 объекта попадают в один лист,
тем они ближе…**

Какую метрику можно придумать?

Бустинг



GBM

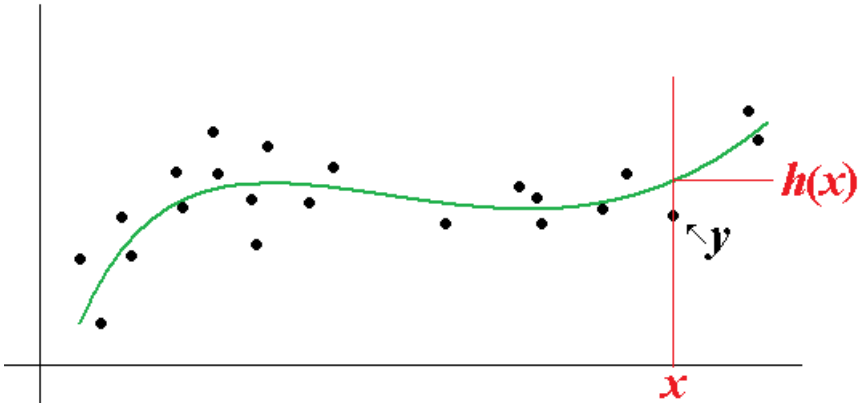
Концепция чёрного ящика

```
Pr = 0
for (j in 1:10)
{
  model <- gbm(is_client_cancel~. , # название целевой переменной
T, # as.data.frame
distribution="gaussian", # распределение... лучше всего gaussian
n.trees=ntrees, # число деревьев (лучше больше, а потом выбрать)
shrinkage=0.07, # скорость сходимости
verbose=TRUE, # вывод сообщений
interaction.depth=12 # сложность модели
)

Pr <- Pr + predict.gbm(model, T2, ntrees) # тут можно перечень числа деревьев
}
```

```
class sklearn.ensemble.GradientBoostingClassifier
(loss='deviance', # в классификации - логистическая регрессия или AdaBoost
learning_rate=0.1, , # скорость сходимости
n_estimators=100, # число деревьев
subsample=1.0,
min_samples_split=2,
min_samples_leaf=1,
min_weight_fraction_leaf=0.0,
max_depth=3, # глубина
max_features=None) # сколько признаков смотреть для расщепления
```

Что означает «распределение»



пусть ошибки распределены по нормальному закону

$$p(y \mid x) = \text{const} \cdot e^{-\frac{(y-h(x))^2}{2\sigma^2}}$$

метод максимального правдоподобия

$$\prod_i p(y_i \mid x_i) \sim \prod_i e^{-\frac{(y_i-h(x_i))^2}{2\sigma^2}} \to \max$$

$$\text{const} \cdot \sum_i (y_i - h(x_i))^2 \to \min$$

Что означает «распределение»

пусть ошибки ~ распределение Лапласа

$$p(y \mid x) = \text{const} \cdot e^{-\alpha|y-h(x)|}$$

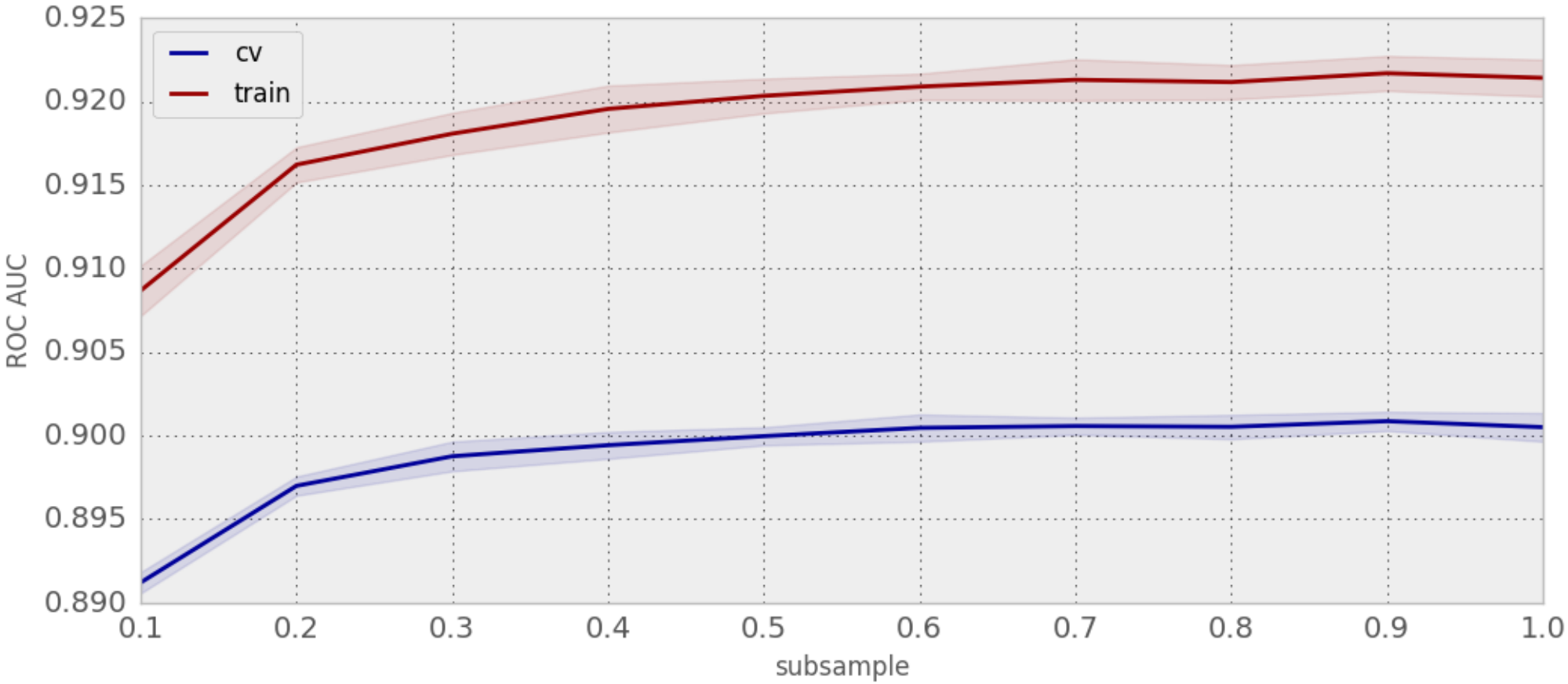
метод максимального правдоподобия

$$\prod_i p(y_i \mid x_i) \sim \prod_i e^{-\alpha|y_i-h(x_i)|} \to \max$$

это эквивалентно минимизации такой ошибки

$$\text{const} \cdot \sum_i |\, y_i - h(x_i)\,| \to \min$$

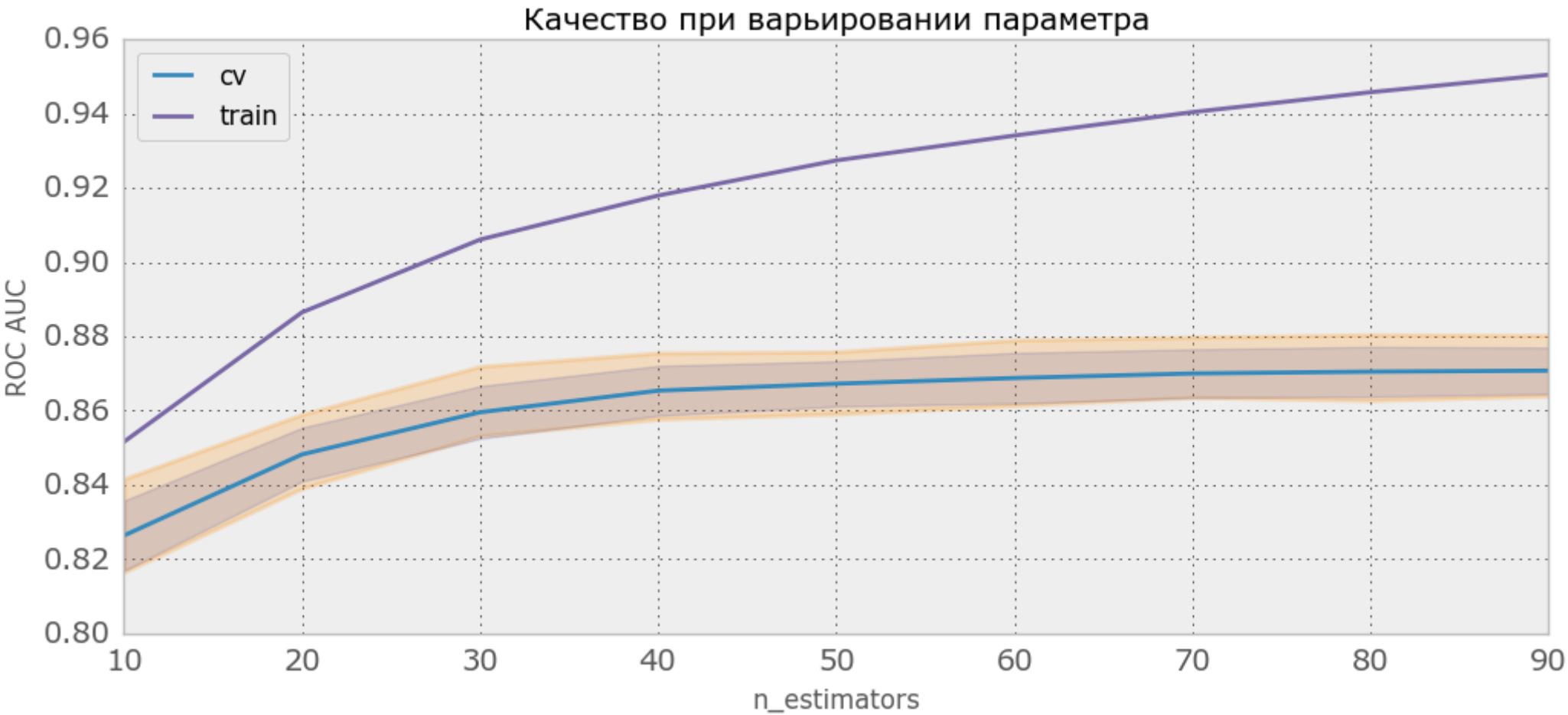
Объём выборки `subsample` (ed Бозон)



Опять, больше – лучше

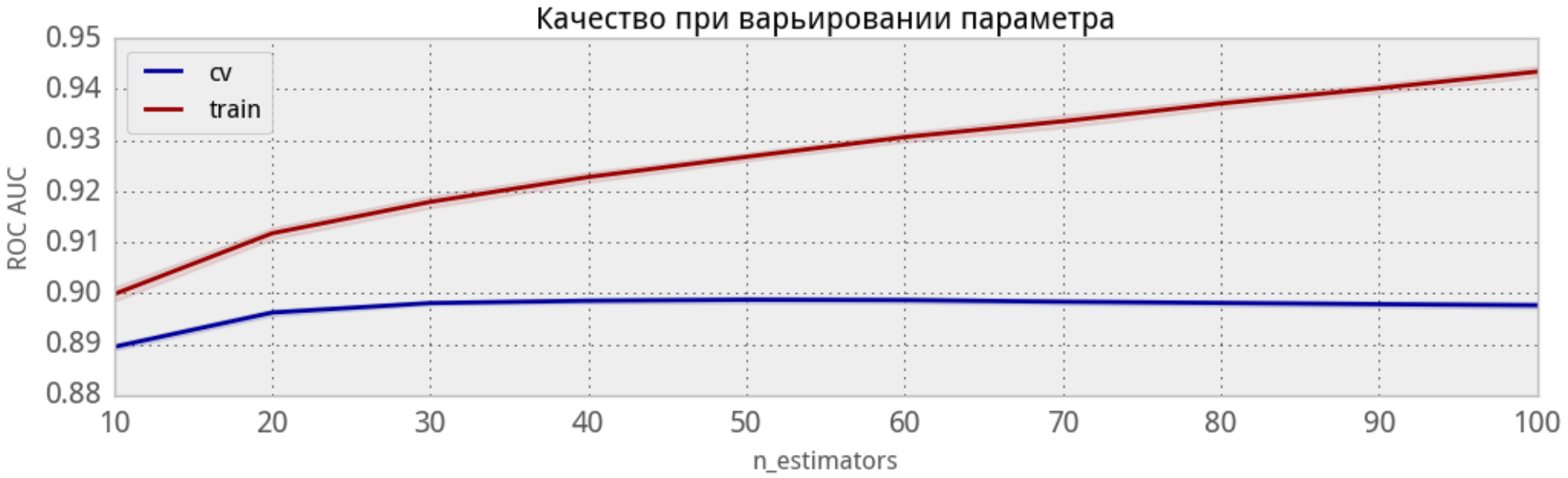
(в XGBoost это не всегда так)

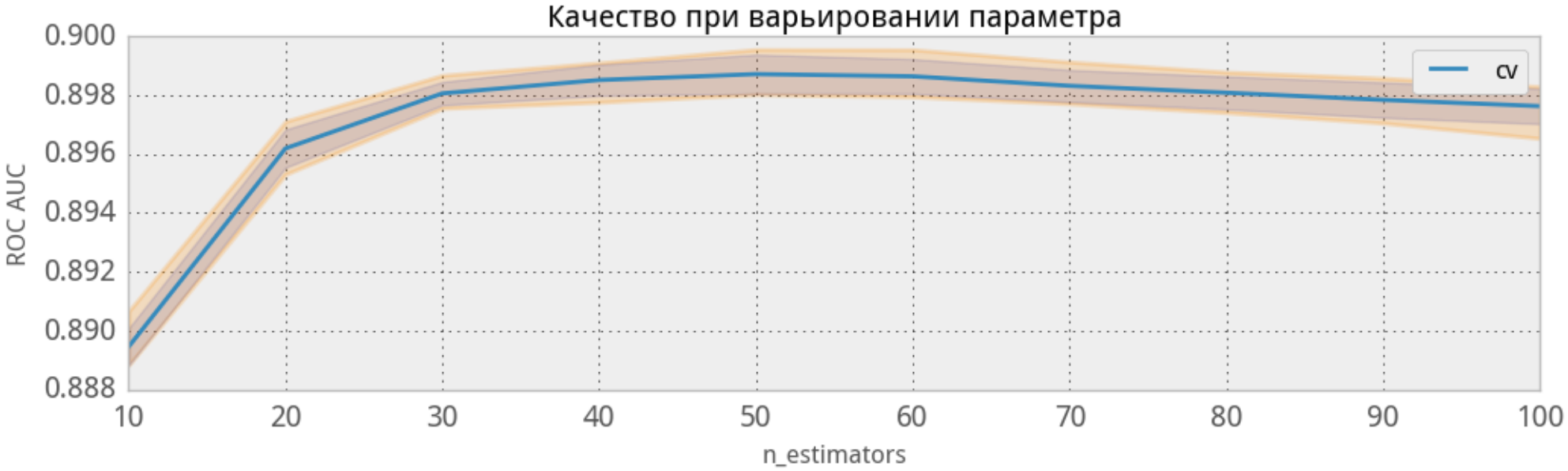
Число деревьев: n_estimators (ed Сбербанк)



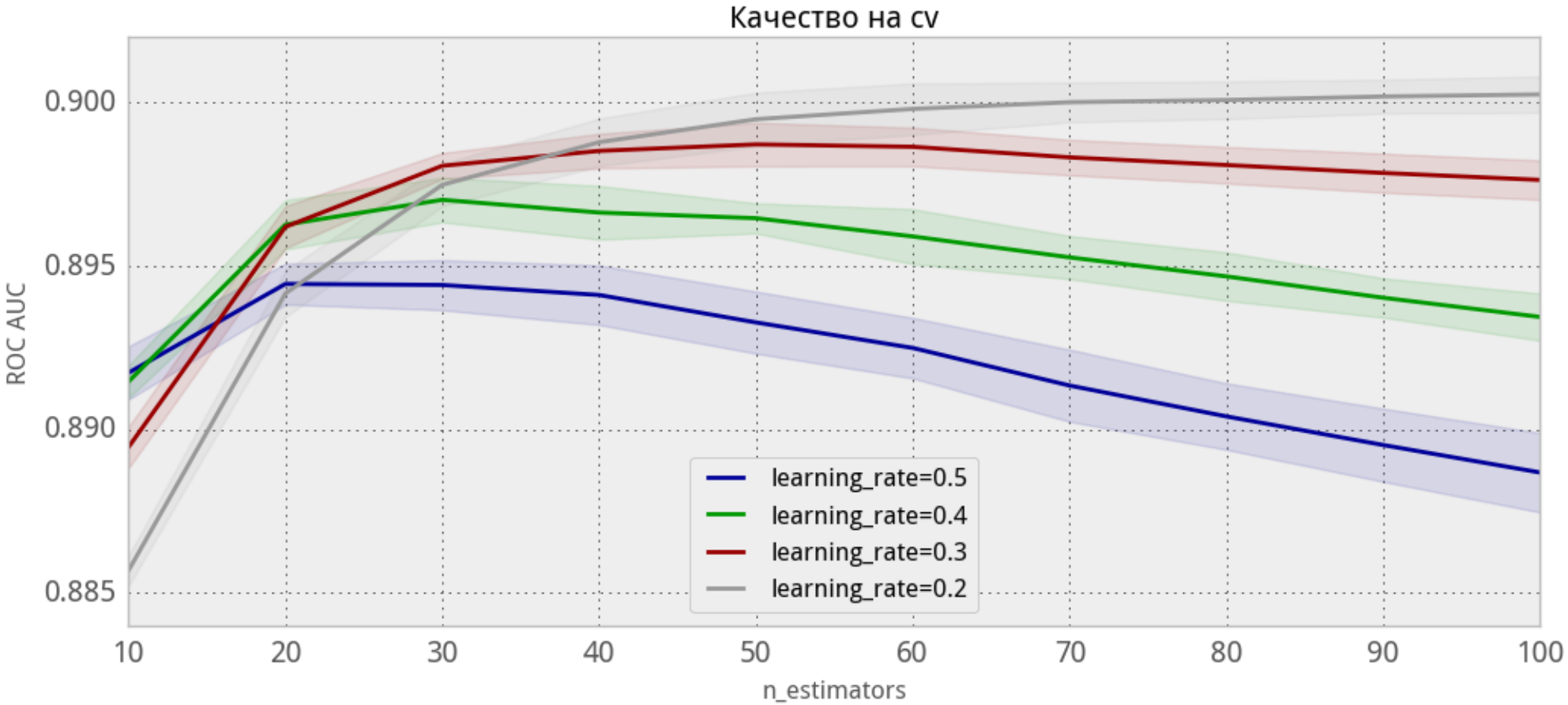
Здесь уже нет логики «чем больше, тем лучше»

Число деревьев: n_estimators (ed Бозон)





Темп обучения `learning_rate` (ed Бозон)



Темп обучения `learning_rate`

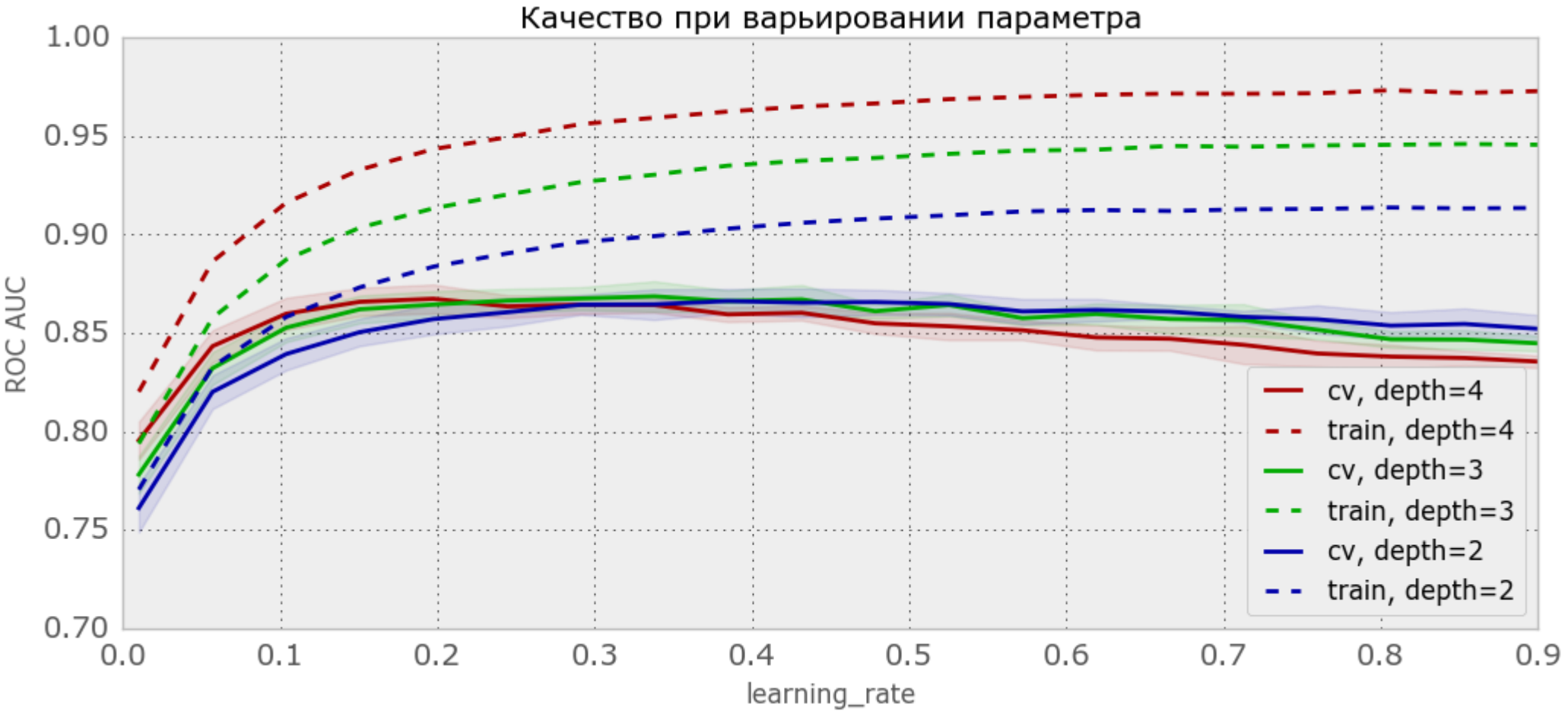
Нет логики «уменьшили темп в 2 раза – число деревьев надо увеличить в 2 раза»!

Есть стратегия – сделать очень маленький темп и очень много деревьев (но для настройки других параметров не годится)

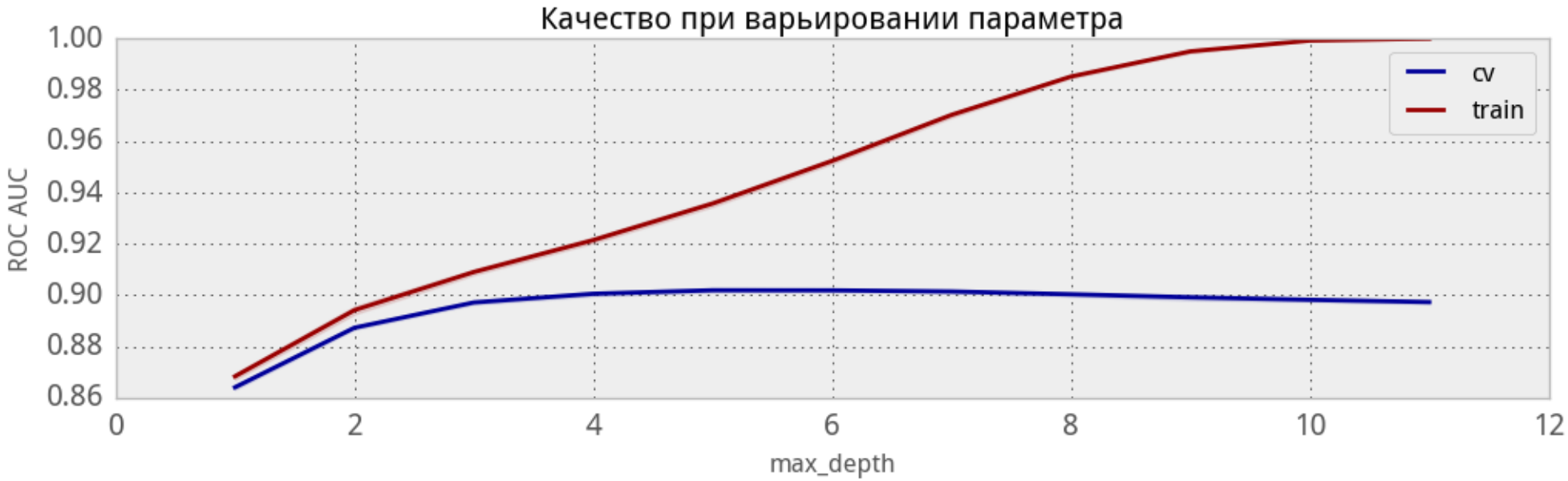
Совет:

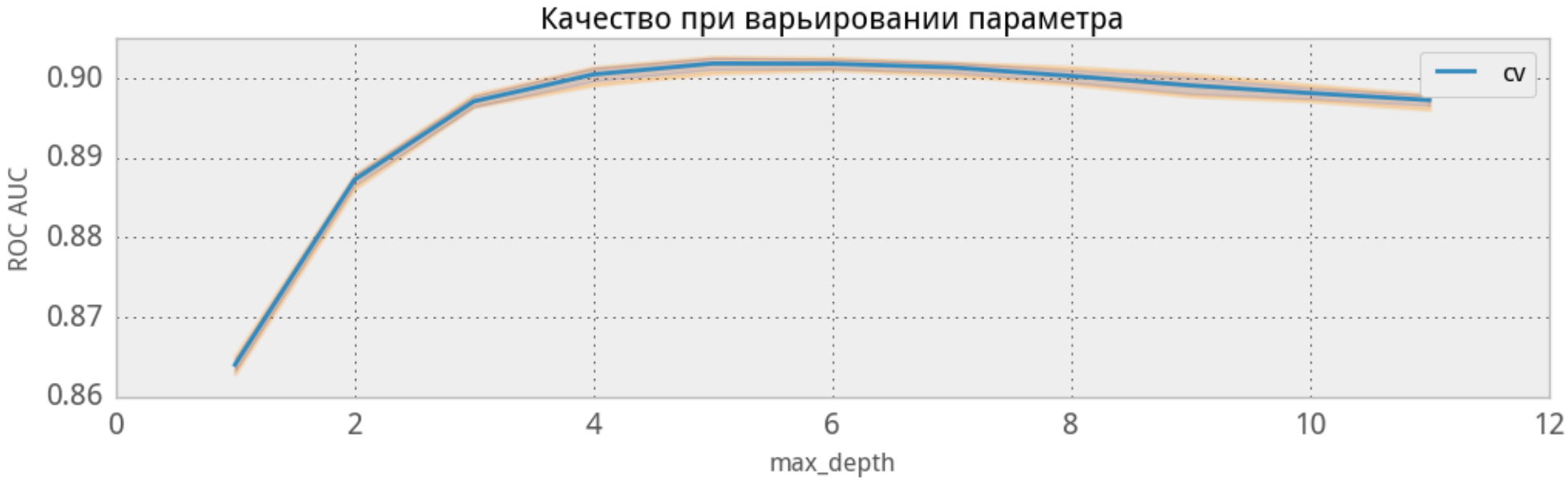
- **зафиксируйте достаточно большое число деревьев, которое ещё можно быстро построить**
- **настройте `learning_rate`**
- **настраивайте другие параметры (первым делом – глубину), но помните, что оптимальный темп может поменяться!**

Темп обучения `learning_rate`



Глубина деревьев



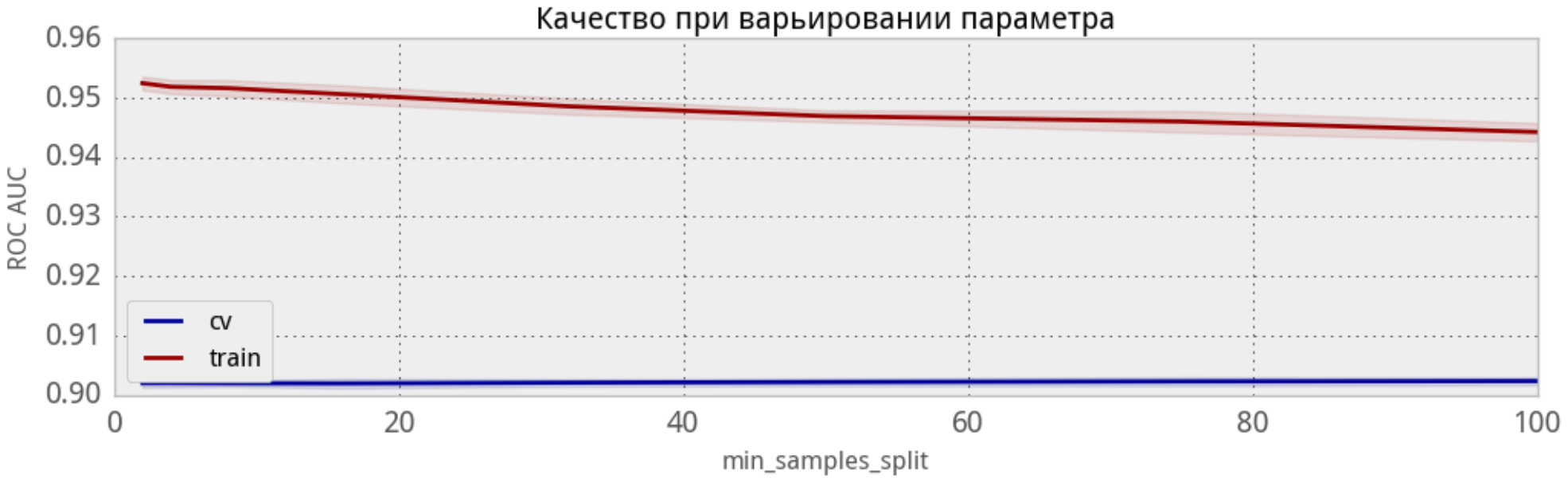


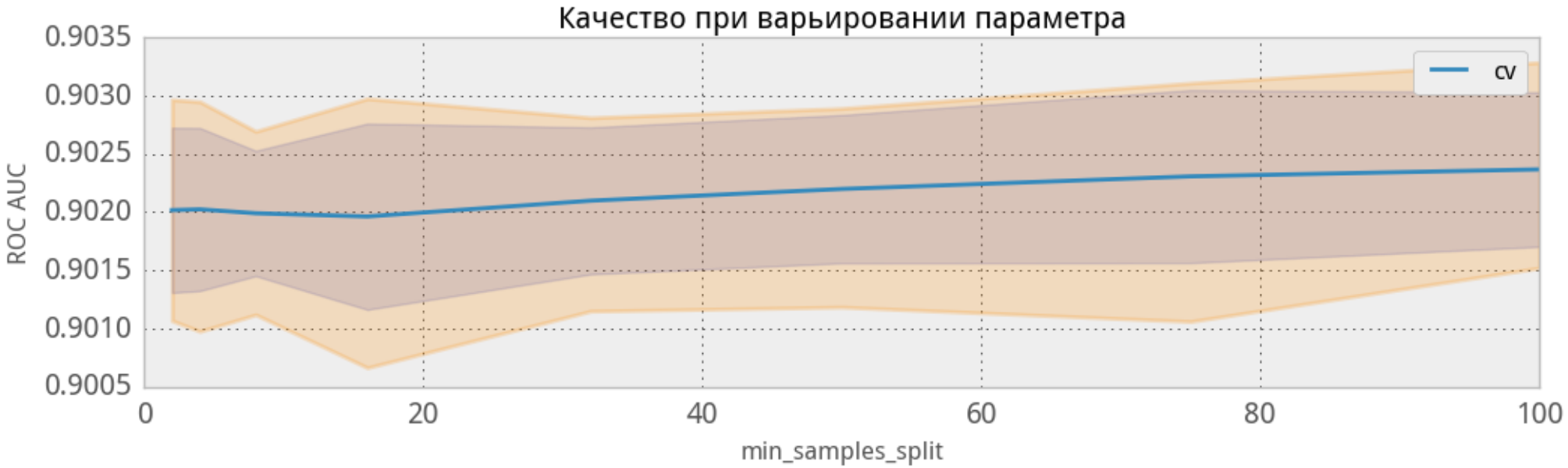
Глубина деревьев

Здесь есть понятие оптимальной глубины!

Как правило, строят неглубокие деревья (3 – 6).

Ограничение на расщепления / листья



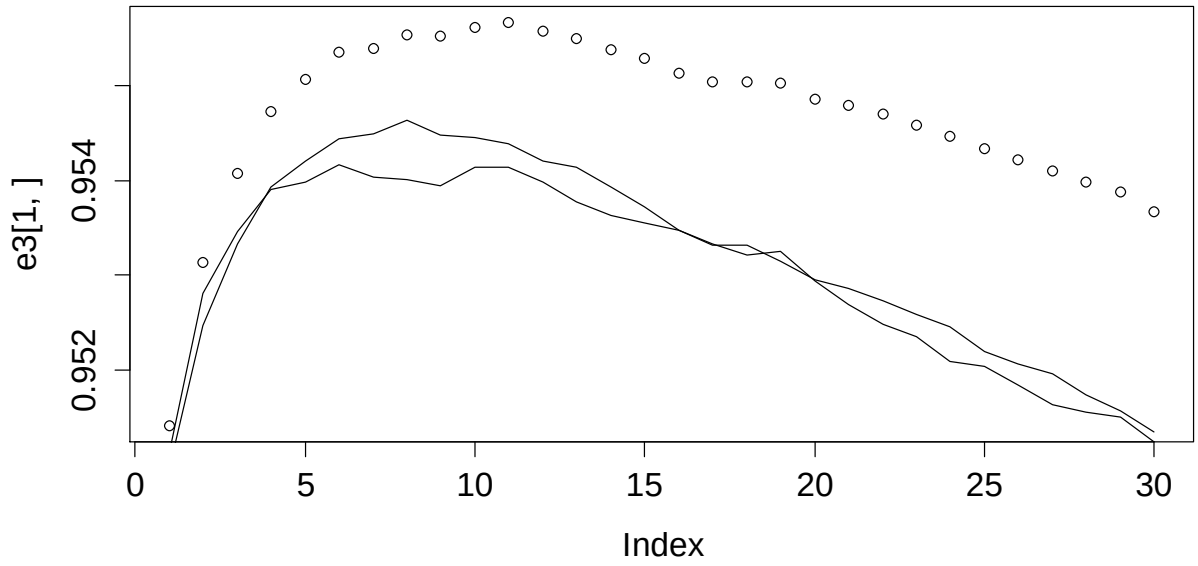


Ограничение на расщепления / листья

Здесь могут быть большие оптимальные значения (10 – 50),

но параметры менее значимые, чем другие…

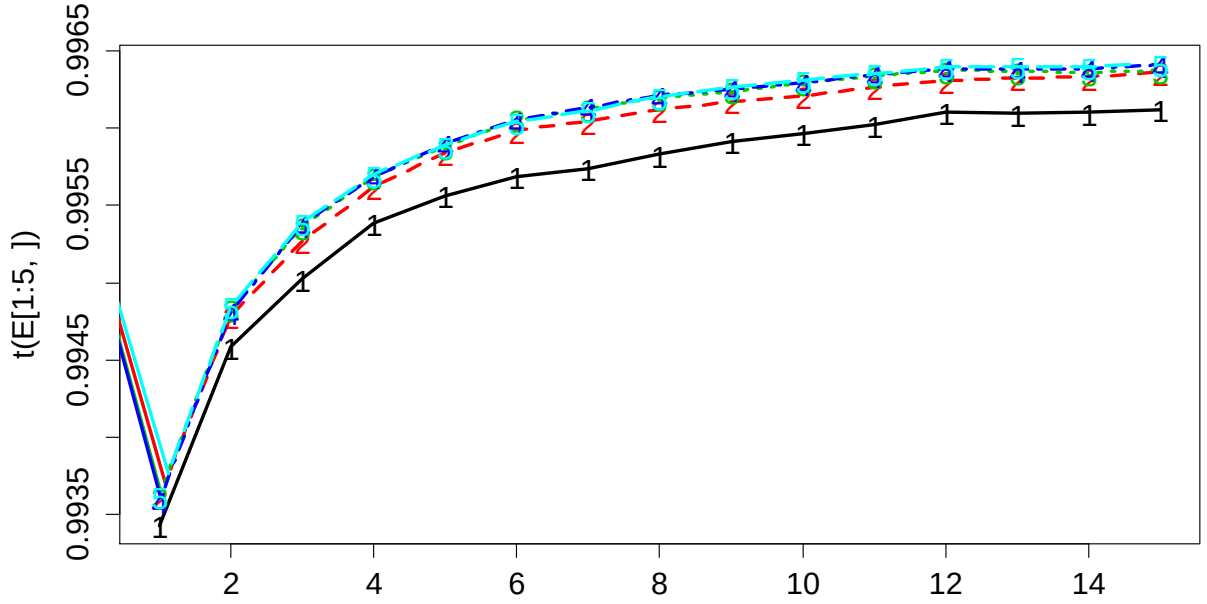
GBM можно усреднять!



Качество двух GBM и их суммы (ср. арифм.)

Для суммы GBM оптимальные параметры другие…

Суммы GBM: (wikimart)



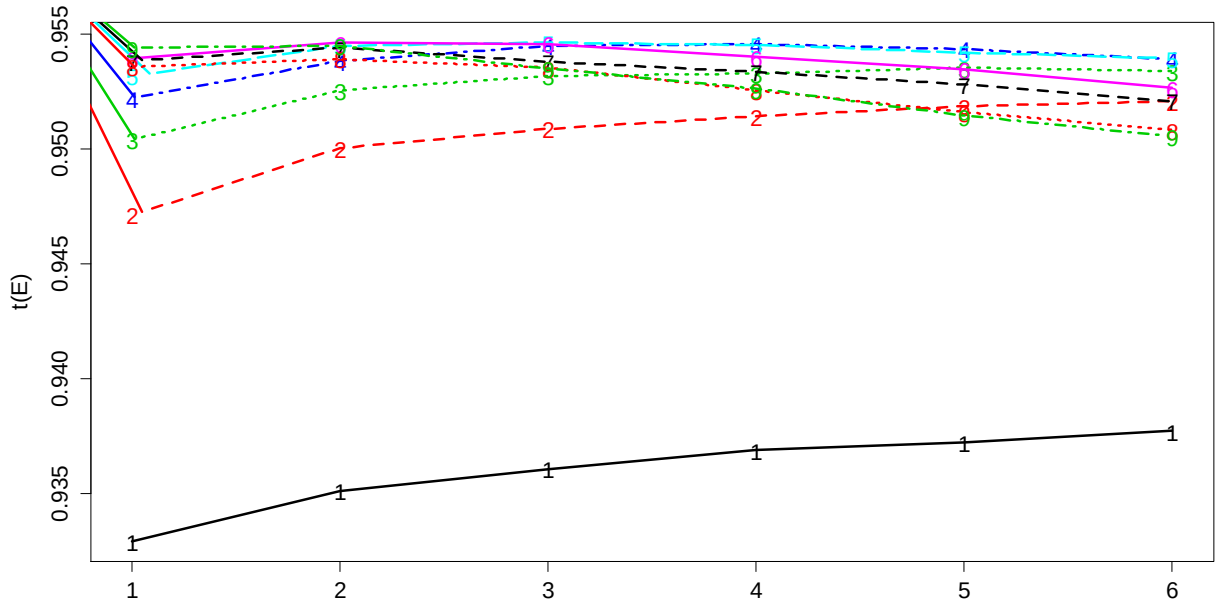

```
library('gbm')
T = as.data.frame(T)
T2 = as.data.frame(T2)
ntrees=1500
PARs = seq(from=100, to=ntrees, by=100)
E = matrix(0, 10, length(PARs))
Pr = 0

for (j in 1:10)
{
```

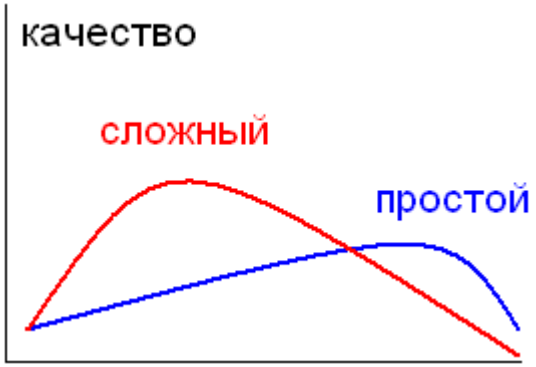
```
model <- gbm(V1~. , T,
distribution="gaussian", n.trees=ntrees,
shrinkage=0.1,verbose=TRUE,
interaction.depth=5)

  Pr <- Pr + predict.gbm(model, T2, PARs)
  e = colAUC(Pr, T2[, 1], plotROC=FALSE)
  E[j,] = e
  print(e)
  print(j)
}
```

GBM: interaction.depth в R



Качество от числа деревьев при разных interaction.depth



Чем прощё GBM – тем больше деревьев надо.

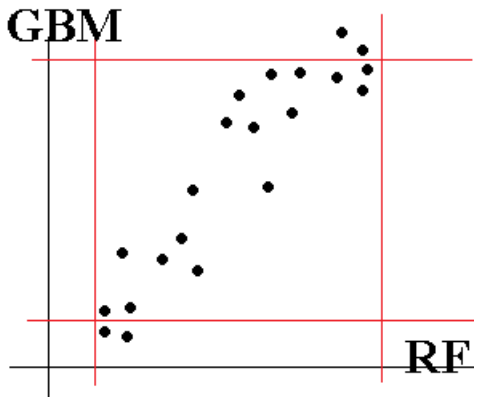
Совет

- выбрать критерий расщепления (вид бустинга) из логики
- выбрать число деревьев и темп обучения (это согласованные параметры)

для настройки можно немного деревьев

- настроить сложность деревьев (варьируя их число)
- увеличить число деревьев, взять маленький темп обучения
- использовать сумму нескольких gbm

Важно



Значения gbm могут выходить за пределы отрезка!

Вообще говоря, не важно,
как их вернуть обратно…

Задача скоринга (TKS)

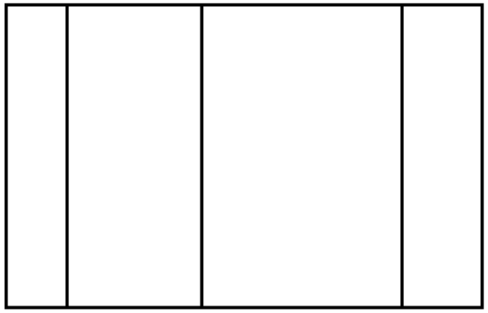
tcs_customer_id	bureau_cd	bki_request_date	inf_confirm_date	type	status	open_date	final_pmt_date	fact_close_date	credit_limit	currency	outstanding	next_pmt	curr_balar
1	2	12Aug2011	20Jul2011	99	00	13May2011	11May2012		28967	RUB	24606,00000	2743,00000	
1	1	12Aug2011	18Feb2009	99	13	27Feb2008	26Feb2009	26Feb2009	30000	RUB	0,00000		
1	1	12Aug2011	21Apr2009	99	13	28Jun2007	30Jun2008	20Apr2009	19421	RUB	0,00000		
1	1	12Aug2011	18Aug2009	9	13	15Jul2008	17Aug2009	17Aug2009	11858	RUB	0,00000		
1	1	12Aug2011	06Sep2010	99	13	09May2009	10May2010	08Sep2010	19691	RUB	0,00000		
1	1	12Aug2011	28Jul2011	7	52	07Sep2010	07Sep2040		10000	RUB			
1	1	12Aug2011	01Aug2011	9	00	31Aug2010	31Aug2015		169000	RUB			
1	1	12Aug2011	03Aug2011	9	00	04Mar2009	03Mar2014		300000	RUB			
1	3	12Aug2011	09Jul2008	9	00	28Jun2007	30Jun2008		19421	RUB	1761,00000		198
1	3	12Aug2011	19Sep2008	9	00	27Feb2008	26Feb2009		30000	RUB	15517,00000		163
1	3	12Aug2011	14Sep2010	9	13	09May2009	10May2010	06Sep2010	19691	RUB	0,00000		
1	3	12Aug2011	11Jul2011	9	00	31Aug2010	31Aug2015		169000	RUB		0,00000	433

Решение = GBM + RF + Линейная регрессия

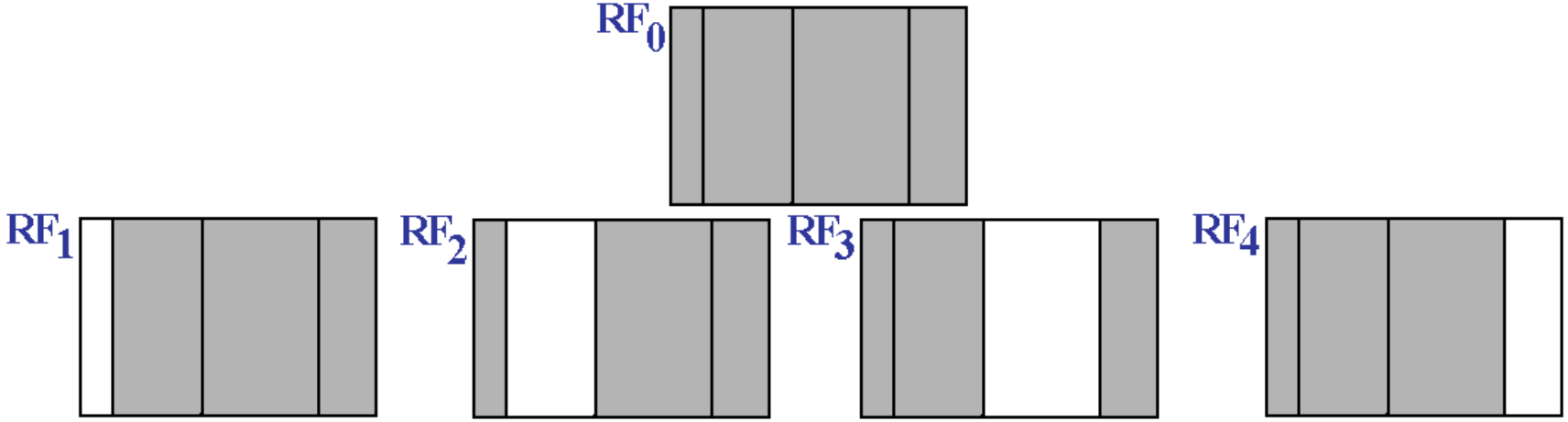
Name	Description	Type
TCS_CUSTOMER_ID	Идентификатор клиента	ID
BUREAU_CD	Код бюро, из которого получен счет	numeric
BKI_REQUEST_DATE	Дата, в которую был сделан запрос в бюро	date
CURRENCY	Валюта договора (ISO буквенный код валюты)	string
RELATIONSHIP	Тип отношения к договору	string
	1 - Физическое лицо	
	2 - Дополнительная карта/Авторизованный пользователь	
	4 - Совместный	
	5 - Поручитель	
	9 - Юридическое лицо	
OPEN_DATE	Дата открытия договора	date
FINAL_PMT_DATE	Дата финального платежа (плановая)	date
TYPE	Код типа договора	string
	1 – Кредит на автомобиль	
	4 – Лизинг. Срочные платежи за наем/пользование транспортным средством, предприятием или оборудованием и т.п.	
	6 – Ипотека – ссудные счета, имеющие отношение к домам, квартирам и прочей недвижимости. Ссуда выплачивается циклично согласно договоренности до тех пор, пока она не будет полностью выплачена или возобновлена.	
	7 – Кредитная карта	
	9 – Потребительский кредит	
	10 – Кредит на развитие бизнеса	
	11 – Кредит на пополнение оборотных средств	
	12 – Кредит на покупку оборудования	
	13 – Кредит на строительство недвижимости	
	14 – Кредит на покупку акций (например, маржинальное кредитование)	
	99 – Другой	
PMT_STRING_84M	Дисциплина (своевременность) платежей. Строка составляется из кодов состояний счета на моменты передачи банком данных по счету в бюро, первый символ - состояние на дату PMT_STRING_START, далее последовательно в порядке убывания дат.	string
	0 – Новый, оценка невозможна	
	X – Нет информации	
	1 – Оплата без просрочек	
	A – Просрочка от 1 до 29 дней	

Одна из технологий решения задач

1. Генерация групп признаков

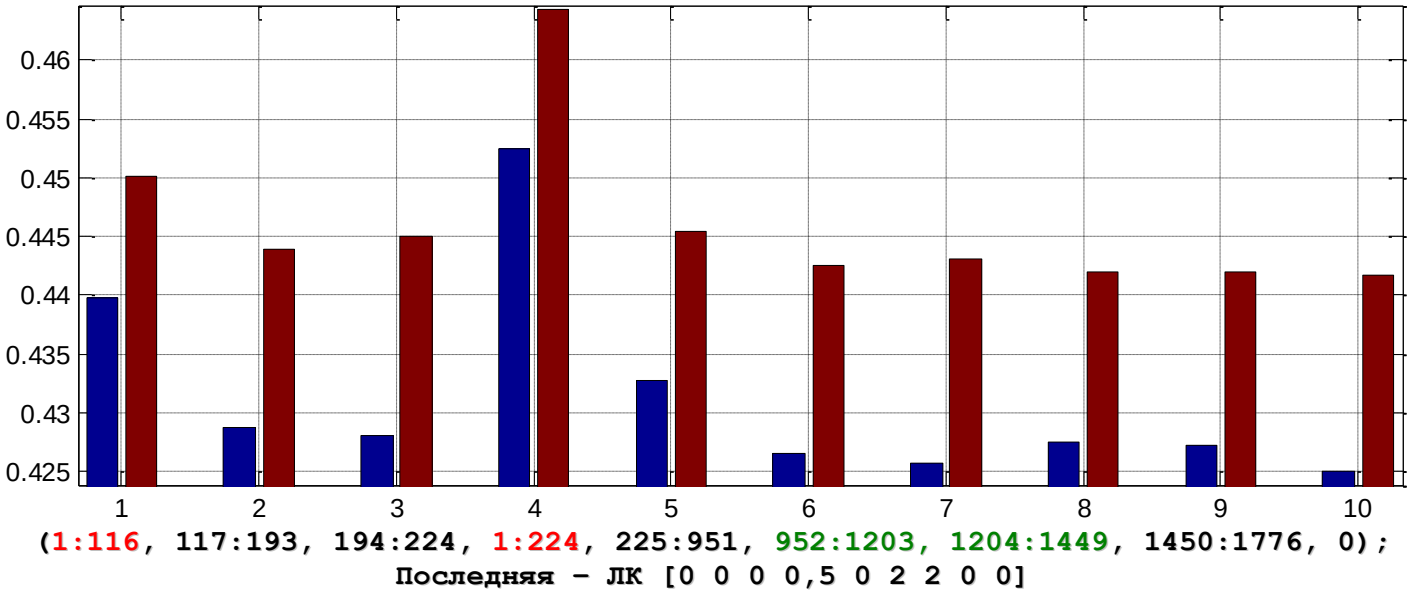


2. Обучение RF на всех группах и на данных без некоторых групп



3. Комбинация полученных алгоритмов

Biological Response



Реальная задача (Photo)

$$\sqrt{c_1 \cdot (\mathrm{rf}_1)^2 + \ldots + c_{24} \cdot (\mathrm{rf}_{24})^2 + c_{25} (\mathrm{knn})^2}$$

поиск решения в таком виде, где разные RF настроены на разных признаковых пространствах

Очень хорошо смешивать разнотипные алгоритмы!

Калибровка RF

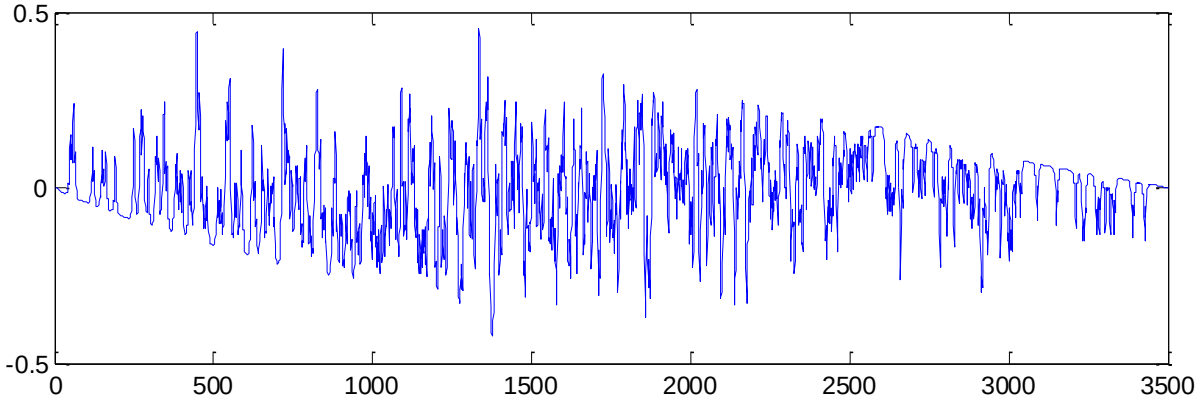
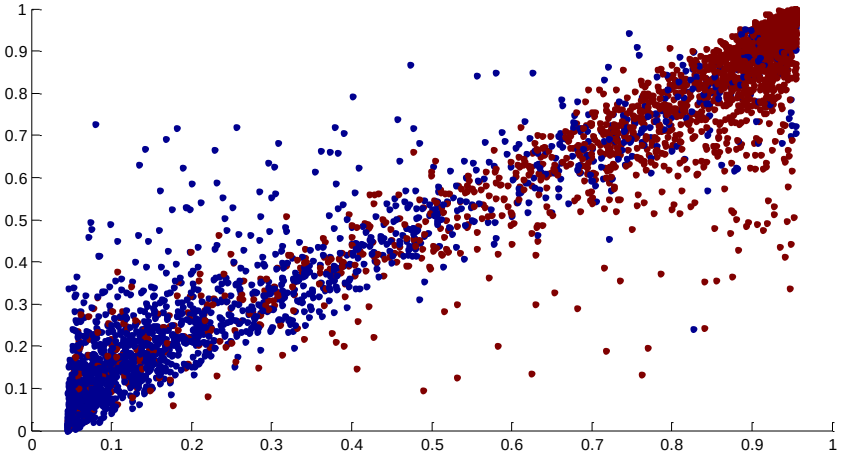
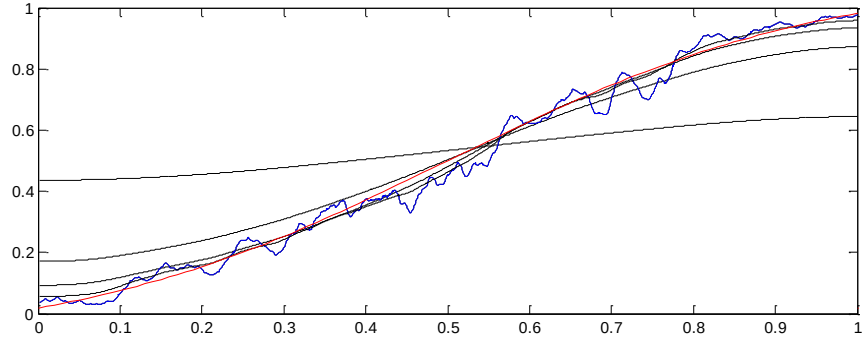
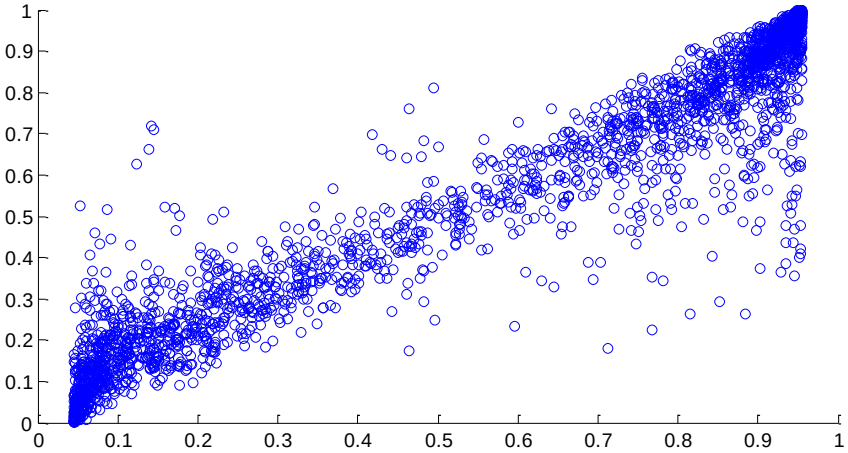


График [прогноз, истина-прогноз]

Уже было...







Деформация ответов с помощью оценки плотности

Предсказание правильности ответов студентов на вопросы тестов

**Разработать алгоритм,
который предсказывает правильность ответа
на вопросы теста.**

Зачем?

для рекомендательной системы (алгоритм решает за студента тест и сообщает ему «потенциально неприятные для него» вопросы).

GMAT, SAT, ACT

Победитель – LibFM

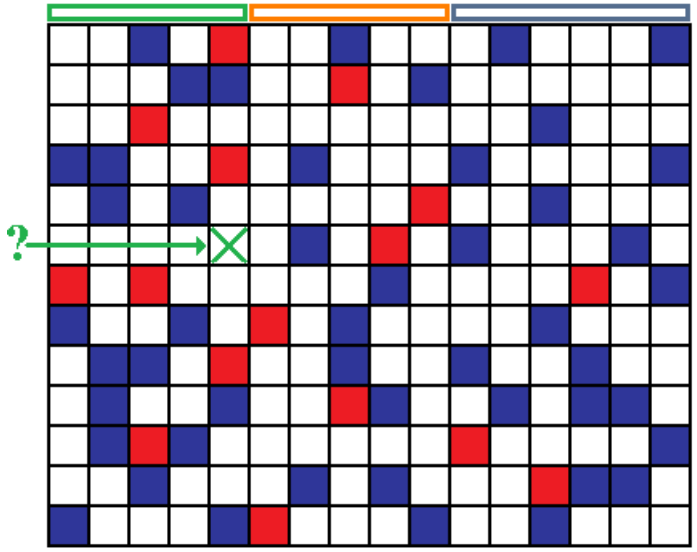
#	Team Name	$5,000 • 241 teams	Score	Entries
1	Steffen Rendle *		0.24598	16
2	Alexander D'yakonov *		0.24729	38
3	ekla *		0.24745	87
4	PlanetThanet & Birutas		0.24772	51

Обычно:

- контекстная рекомендация
- коллаборативная фильтрация

Наша идея:

свести задачу о рекомендациях
к задаче классификации (регрессии)



пара «студент–вопрос» ~ признаковое описание

генерация кучи признаков

Примеры признаков

Пусть ответы студента:

“correct, incorrect, correct, correct, correct, incorrect”

$$\text{IQ} \sim \frac{+1-1+1+1+1-1}{6}$$

$$\text{weighted IQ} \sim \frac{+1w_1 - 1w_2 + 1w_3 + 1w_4 + 1w_5 - 1w_6}{w_1 + w_2 + w_3 + w_4 + w_5 + w_6}$$

веса измеряют «похожесть вопросов»

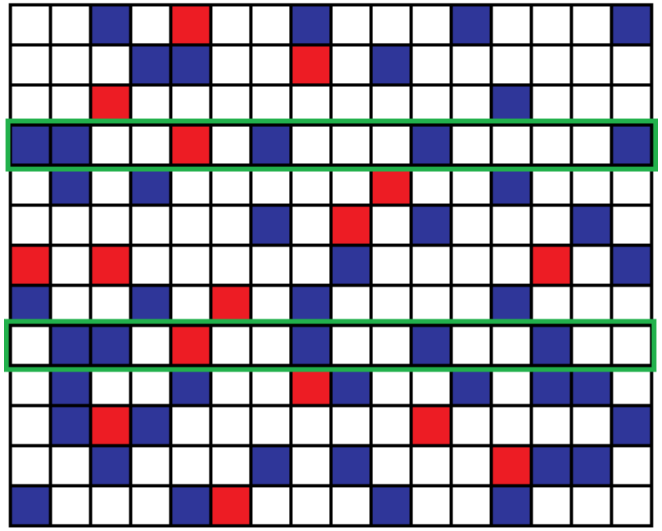
$$w_j = \frac{2}{1+|t-t_j|^{0.3}} - 1 \text{ или } w_j = 1 - \sqrt{|t-t_j|}$$

t_j **– время ответа на j-й вопрос,**

t **– время ответа на этот вопрос.**

Ещё веса:

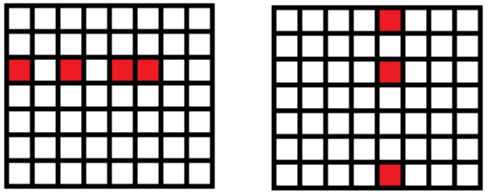
Корреляция столбцов матрицы «студент–ответ»



Аналогично:

Признак «сложность вопроса»

(здесь усредняются ответы на данный вопрос)



Простые признаки:

- время ответа
- 1/(число ответов всего)

SVD-признаки

Восстановление матрицы с помощью SVD-преобразования (даже по подматрице)

Решение

gbm + glm + NN (CLOP)

Опять: хорошо «смешиваются» разные алгоритмы…

Как настраивать – чуть позже…

Литература

A. Liaw, M. Wiener Classification and Regression by randomForest // R News (2002) Vol. 2/3 p. 18.
http://www.bios.unc.edu/~dzeng/BIOS740/randomforest.pdf

И. Генрихов О критериях ветвления, используемых при синтезе решающих деревьев // Машинное обучение и анализ данных, 2014, Т.1, №8, С.988-1017
http://jmlda.org/papers/doc/2014/no8/Genrikhov2014Criteria.pdf

A. Natekin, A. Knoll Gradient boosting machines, a tutorial // Front Neurorobot. 2013; 7: 21.
http://www.ncbi.nlm.nih.gov/pmc/articles/PMC3885826/