### Исследование временных рядов с помощью среды R. Нелинейная модель: решающие деревья

Деревья принятия решений (также называются деревьями классификации или регрессии) — являются одним из основных методов в области анализа данных. Структура дерева включает в себя «листья» и «ветки». Назначение веток состоит в определении правил, согласно которым происходит разбиение пространства. Листья соответствуют подмножествам, которые образуются в результате разбиения.

Для использования решающих деревьев в R используется пакет rpart. Полное название данного пакета – «Recursive Partitioning and Regression Trees». Загрузка и подключение пакета производится стандартным образом. Подробную документацию можно найти по ссылке: <http://cran.r-project.org/web/packages/rpart/rpart.pdf> .

|  |
| --- |
| *install.packages("rpart")*  *library(rpart)* |

Код для обучения модели:

*object <- rpart(tg ~ ., method="class", data=A)*

Табл. 4. Параметры функции *rpart*

|  |  |
| --- | --- |
| tg~. | Формула, показывающая, что ищется зависимость поля Survived от полей, указанных в правой части. |
| data=train | Указываем используемые для обучения данные. |
| method=”class” | Если нужно получить непрерывное значение, например, возраст, можно использовать метод ”anova”. Он генерирует десятичные числа, но так как сейчас нужно получить 0 или 1, уместно использовать метод “class”. |

|  |
| --- |
| object <- rpart(A\_train$SeriousDlqin2yrs~., method="class", data=A\_train) |

Для вывода информации о дереве есть несколько способов:

* *plot(object)* – представляет дерево в виде наглядного изображения. Для более корректного отображения необходимо выполнить так же команду *text(object, use.n=TRUE, all=TRUE, cex=1.4)* – она дополнит дерево цифровой информацией.

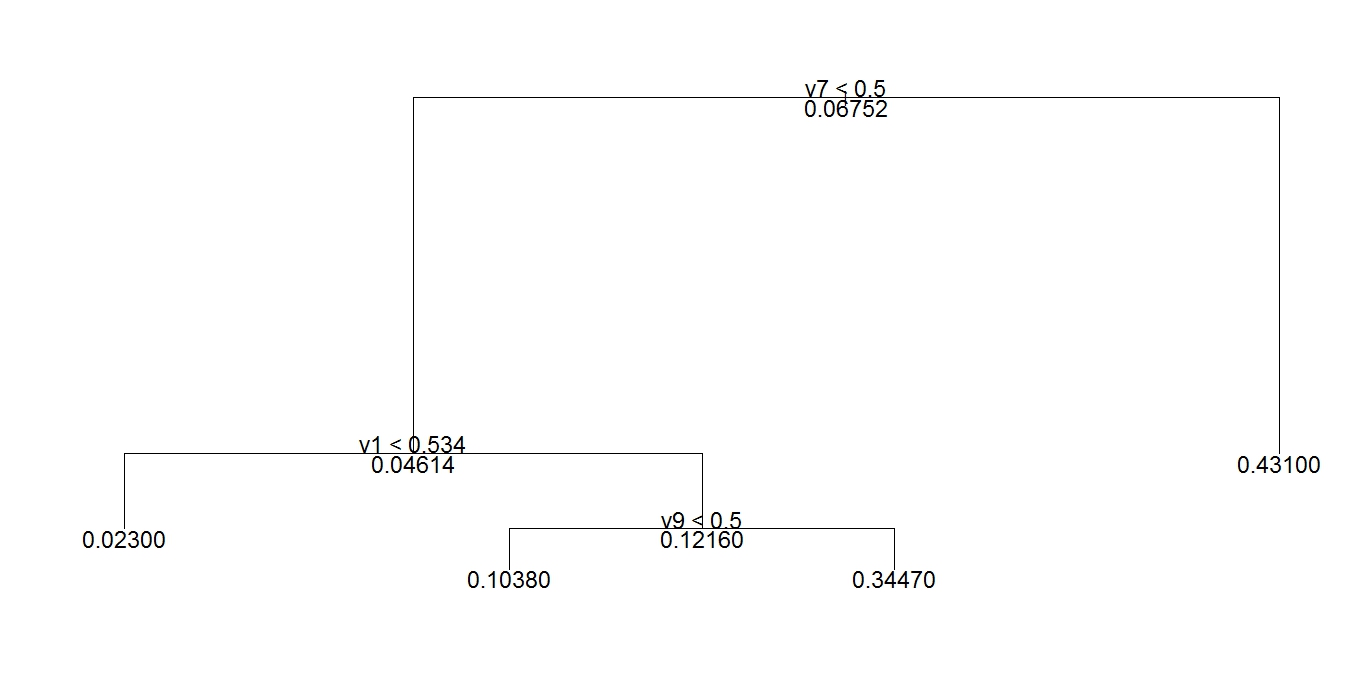


Рис. 3. Дерево принятия решений

|  |
| --- |
| plot(object)  text(object, use.n=TRUE, all=TRUE, cex=0.8) |

* *print(object)* – печать информации в консоль. Результат представлен ниже.

n= 80000

node), split, n, loss, yval, (yprob)

\* denotes terminal node

1) root 80000 5305 0 (0.93368750 0.06631250)

2) NumberOfTimes90DaysLate< 0.5 75517 3449 0 (0.95432816 0.04567184) \*

3) NumberOfTimes90DaysLate>=0.5 4483 1856 0 (0.58599152 0.41400848)

6) NumberOfTimes90DaysLate< 1.5 2819 937 0 (0.66761263 0.33238737) \*

7) NumberOfTimes90DaysLate>=1.5 1664 745 1 (0.44771635 0.55228365) \*

|  |
| --- |
| sol\_trn <- predict(object,newdata=A\_train)  sol\_vld<-predict(object,newdata=A\_validate)  S <- auc(A\_train[,2],sol\_trn[,2])  Z <- auc(A\_validate[,2],sol\_vld[,2]) |

Area under the curve: 0.6586

### Бустинг и пакет GBM

Бустинг (англ. boosting — поддержка) — это процедура последовательного построения композиции алгоритмов машинного обучения, когда каждый следующий алгоритм стремится компенсировать недостатки композиции всех предыдущих алгоритмов.

Бустинг при использовании решающих деревьев в качестве элементарных классификаторов считается одним из наиболее эффективных методов.

В качестве примера можно рассмотреть пакет *gbm*. Полное название данного пакета – «Generalized Boosted Regression Models». Загрузка и подключение пакета производится стандартным образом. Подробную документацию можно найти по ссылке: <http://cran.r-project.org/web/packages/gbm/gbm.pdf> .

Базовая функция пакета *GBM* включает в себя довольно много регулирующих параметров:

*object <- gbm(tg~., data=A,distribution="bernoulli", n.trees=400, shrinkage=0.012,interaction.depth=12,n.minobsinnode = 15, verbose=TRUE)*

Табл. 5. Параметры функции *gbm*

|  |  |
| --- | --- |
| tg~. | Формула, показывающая, что ищется зависимость метки tg от остальных признаков. |
| data=A | Данные для тренировки. |
| distribution="bernoulli" | Выбор используемой функции потерь. |
| n.trees | Число решающих деревьев. Этот параметр имеет большое влияние на скорость работы алгоритма и на качество модели. |
| shrinkage=0.012,  interaction.depth=12 | Другие параметры алгоритма. |
| bag.fraction = 0.5,  train.fraction = 0.5,  n.minobsinnode = 10,  cv.folds = 5, | Параметры валидации. |
| verbose=TRUE | Параметр, показывающий, нужна ли печать в консоль результатов  процесса настройки модели. |

|  |
| --- |
| object <- gbm(A\_train$SeriousDlqin2yrs~., data=A\_train,distribution="bernoulli", n.trees=200, shrinkage=0.012,interaction.depth=12,n.minobsinnode = 15, verbose=TRUE, cv.folds=2) |

Итог работы функции выводится в консоль. По ней можно видеть результат на некоторых итерациях кроссвалидации и его зависимость от количества деревьев, а также общий результат по обучению модели. Можно заметить, что при увеличении числа деревьев с определённого значения результат перестаёт улучшаться.

Теперь посмотрим, какой результат даёт полученная модель на этапе валидации:

|  |
| --- |
| S <- auc(A\_train[,2],sol\_trn)  print(S)  plot(roc(A\_train[,2],sol\_trn)) |

Area under the curve: 0.8707

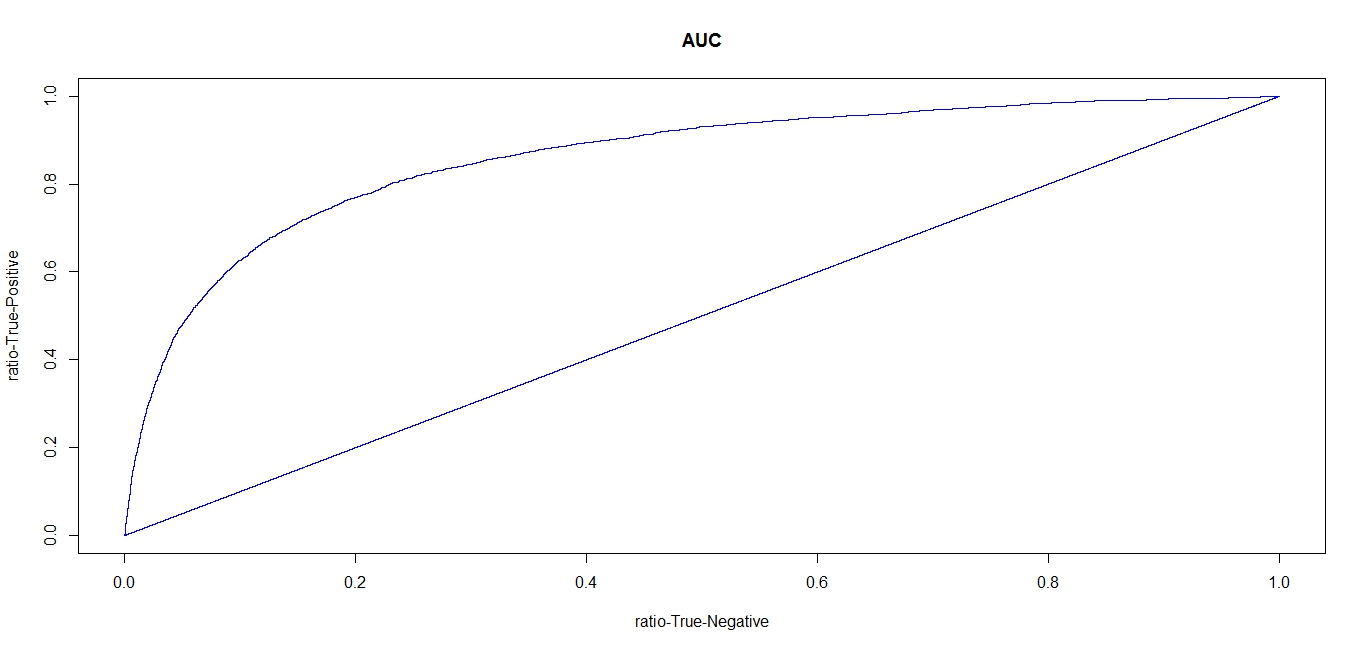
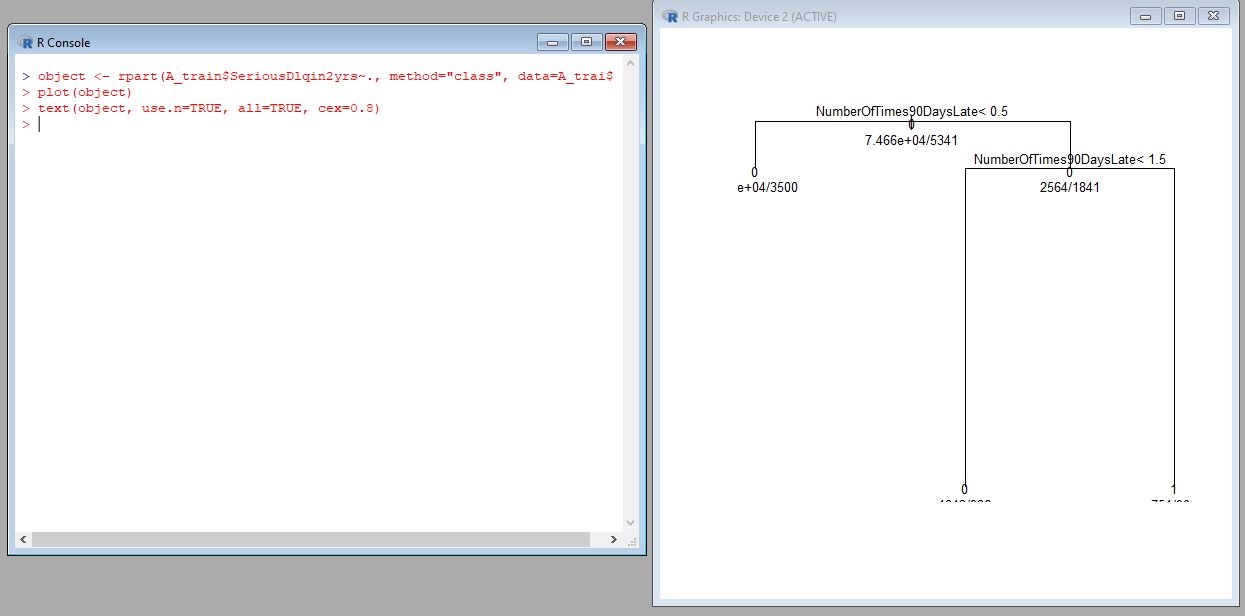


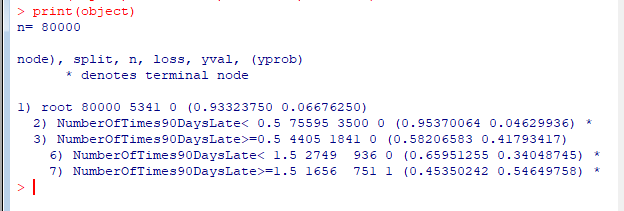
Рис. 4. ROC-кривая в случае модели GBM, AUC = 0.87

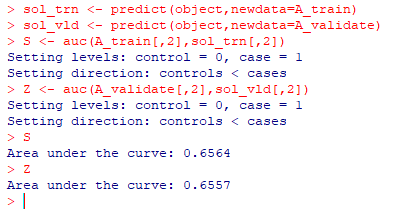
Видно, что результат выше, чем тот, который был получен, используя предыдущие модели.

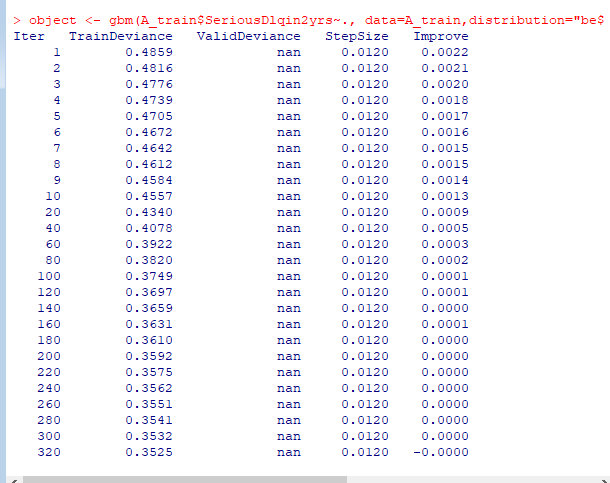
##### Задание

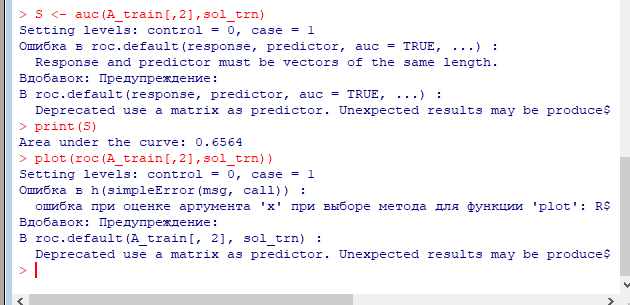
1) Построить описанные выше модели с использованием данных из предыдущей работы.











|  |
| --- |
| A <- read.table("cs-test.csv",header=TRUE,sep=",")  **>** A\_test<- read.table("test1.csv",header=**TRUE**,sep=";")  **>**  **>** A\_train<- A[sample(**1**:nrow(A), **80000**, replace=**FALSE**),]  **>** A\_validate<- A[sample(**1**:nrow(A), **30000**, replace=**FALSE**),]  **>**  **>** object <- lm(A\_train$SeriousDlqin2yrs~.,data=A\_train)  **>** sol\_trn <- predict(object,newdata=A\_train)  **>** sol\_vld<-predict(object,newdata=A\_validate)  **>**  **>** S <- auc(A\_train[,**2**],sol\_trn)  Setting levels: control = 0, case = 1  Setting direction: controls < cases  **>** Z <- auc(A\_validate[,**2**],sol\_vld)  Setting levels: control = 0, case = 1  Setting direction: controls < cases  **>**  **>** print(S)  Area under the curve: 0.6864  **>** print(Z)  Area under the curve: 0.6955  **>** object <- rpart(A\_train$SeriousDlqin2yrs~., method="class", data=A\_train)  **>** plot(object)  **>** text(object, use.n=**TRUE**, all=**TRUE**, cex=**0.8**)  **>** sol\_trn <- predict(object,newdata=A\_train)  **>** sol\_vld<-predict(object,newdata=A\_validate)  **>** S <- auc(A\_train[,**2**],sol\_trn[,**2**])  Setting levels: control = 0, case = 1  Setting direction: controls < cases  **>** Z <- auc(A\_validate[,**2**],sol\_vld[,**2**])  Setting levels: control = 0, case = 1  Setting direction: controls < cases  **>**  **>** print(S)  Area under the curve: 0.6564  **>** print(Z)  Area under the curve: 0.6557  **>** S <- auc(A\_train[,**2**],sol\_trn)  Setting levels: control = 0, case = 1  Ошибка в roc.default(response, predictor, auc = TRUE, ...) :  Response and predictor must be vectors of the same length.  Вдобавок: Предупреждение:  В roc.default(response, predictor, auc = TRUE, ...) :  Deprecated use a matrix as predictor. Unexpected results may be produced, please pass a numeric vector.  **>** print(S)  Area under the curve: 0.6564  **>** plot(roc(A\_train[,**2**],sol\_trn))  Setting levels: control = 0, case = 1  Ошибка в h(simpleError(msg, call)) :  ошибка при оценке аргумента 'x' при выборе метода для функции 'plot': Response and predictor must be vectors of the same length.  Вдобавок: Предупреждение:  В roc.default(A\_train[, 2], sol\_trn) :  Deprecated use a matrix as predictor. Unexpected results may be produced, please pass a numeric vector.  **>**  **>** object <- gbm(A\_train$SeriousDlqin2yrs~., data=A\_train,distribution="bernoulli", n.trees=**400**, shrinkage=**0.012**,interaction.depth=**12**,n.minobsinnode = **15**, verbose=**TRUE**, cv.folds=**2**)  Iter TrainDeviance ValidDeviance StepSize Improve  1 0.4859 nan 0.0120 0.0022  2 0.4816 nan 0.0120 0.0021  3 0.4776 nan 0.0120 0.0020  4 0.4739 nan 0.0120 0.0018  5 0.4705 nan 0.0120 0.0017  6 0.4672 nan 0.0120 0.0016  7 0.4642 nan 0.0120 0.0015  8 0.4612 nan 0.0120 0.0015  9 0.4584 nan 0.0120 0.0014  10 0.4557 nan 0.0120 0.0013  20 0.4340 nan 0.0120 0.0009  40 0.4078 nan 0.0120 0.0005  60 0.3922 nan 0.0120 0.0003  80 0.3820 nan 0.0120 0.0002  100 0.3749 nan 0.0120 0.0001  120 0.3697 nan 0.0120 0.0001  140 0.3659 nan 0.0120 0.0000  160 0.3631 nan 0.0120 0.0001  180 0.3610 nan 0.0120 0.0000  200 0.3592 nan 0.0120 0.0000  220 0.3575 nan 0.0120 0.0000  240 0.3562 nan 0.0120 0.0000  260 0.3551 nan 0.0120 0.0000  280 0.3541 nan 0.0120 0.0000  300 0.3532 nan 0.0120 0.0000  320 0.3525 nan 0.0120 -0.0000  340 0.3517 nan 0.0120 0.0000  360 0.3510 nan 0.0120 0.0000  380 0.3504 nan 0.0120 -0.0000  400 0.3498 nan 0.0120 0.0000  **>** S <- auc(A\_train[,**2**],sol\_trn)  Setting levels: control = 0, case = 1  Ошибка в roc.default(response, predictor, auc = TRUE, ...) :  Response and predictor must be vectors of the same length.  Вдобавок: Предупреждение:  В roc.default(response, predictor, auc = TRUE, ...) :  Deprecated use a matrix as predictor. Unexpected results may be produced, please pass a numeric vector.  **>** print(S)  Area under the curve: 0.6564  **>** plot(roc(A\_train[,**2**],sol\_trn))  Setting levels: control = 0, case = 1  Ошибка в h(simpleError(msg, call)) :  ошибка при оценке аргумента 'x' при выборе метода для функции 'plot': Response and predictor must be vectors of the same length.  Вдобавок: Предупреждение:  В roc.default(A\_train[, 2], sol\_trn) :  Deprecated use a matrix as predictor. Unexpected results may be produced, please pass a numeric vector.  **>** |

2) Построить несколько моделей с различным числом решающих деревьев. Сделать выводы о влиянии количества решающих деревьев на точность модели.