ClassificationIMDB

Nadezhda

2023-10-23

*Предобработка данных.* *Нужно перекодировать переменную Genre (так как у одного фильма одновременно может быть указано несколько жанров): создадим новую переменную, которая будет нести информацию о первом (основном) из указанных жанров.* *Наиболее часто появляющимися жанрами в этой базе являются Drama, Comedy, Action, Crime, все остальные жанры объединим в категорию Other (объединяем в большую группу, так как присутствующее число наблюдений по этим категориям не достаточно для анализа).* *Нужно перекодировать переменную Runtime, чтобы осталось только число минут для каждого фильма.* *Несколько переменных содержат пропущенные значения, заменим их на медианные значения.* *Для анализа нам также понадобятся новые переменные: выручка больше или равная медианному значению, год релиза позже или равный медианному значению, оценка критиков больше или равная медианному значению, оценка аудитории больше или равная медианному значению.* *Основными переменными для анализа в этой базе для нас будут: год релиза, длительность фильма, жанр, рейтинг среди аудитории, рейтинг среди критиков и выручка.*

knitr::opts\_chunk$set(echo = TRUE, message = FALSE, warning = FALSE, eval = TRUE)

library(readr)  
imdb\_top\_1000 <- read\_csv("C:/Users/Nadezhda/Desktop/imdb\_top\_1000.csv")  
df <- imdb\_top\_1000[, c(3, 5:7, 9, 16)]  
head(df, n = 10)

## # A tibble: 10 × 6  
## Released\_Year Runtime Genre IMDB\_Rating Meta\_score Gross  
## <chr> <chr> <chr> <dbl> <dbl> <dbl>  
## 1 1994 142 min Drama 9.3 80 2.83e7  
## 2 1972 175 min Crime, Drama 9.2 100 1.35e8  
## 3 2008 152 min Action, Crime, Drama 9 84 5.35e8  
## 4 1974 202 min Crime, Drama 9 90 5.73e7  
## 5 1957 96 min Crime, Drama 9 96 4.36e6  
## 6 2003 201 min Action, Adventure, Drama 8.9 94 3.78e8  
## 7 1994 154 min Crime, Drama 8.9 94 1.08e8  
## 8 1993 195 min Biography, Drama, History 8.9 94 9.69e7  
## 9 2010 148 min Action, Adventure, Sci-Fi 8.8 74 2.93e8  
## 10 1999 139 min Drama 8.8 66 3.70e7

library(stringr)  
df[c('Time', 'Last')] <- str\_split\_fixed(df$Runtime, ' ', 2)   
df[c('Genre1', 'Genre2', 'Genre3')] <- str\_split\_fixed(df$Genre, ',', 3)  
sapply(df, class)

## Released\_Year Runtime Genre IMDB\_Rating Meta\_score   
## "character" "character" "character" "numeric" "numeric"   
## Gross Time Last Genre1 Genre2   
## "numeric" "character" "character" "character" "character"   
## Genre3   
## "character"

cols.num <- c('Released\_Year', 'Time')  
df[cols.num] <- sapply(df[cols.num], as.numeric)  
names(which(colSums(is.na(df))>0))

## [1] "Released\_Year" "Meta\_score" "Gross"

df$Gross[is.na(df$Gross)]<- median(df$Gross,na.rm = TRUE)  
df$Meta\_score[is.na(df$Meta\_score)]<- median(df$Meta\_score,na.rm = TRUE)  
df$Released\_Year[is.na(df$Released\_Year)]<- median(df$Released\_Year,na.rm = TRUE)  
df$Revenue <- with(df, ifelse(Gross >= 23530892, 1, 0))  
df$Rating <- with(df, ifelse(IMDB\_Rating >= 7.9, 1, 0))  
df$Year <- with(df, ifelse(Released\_Year >= 1999, 1, 0))  
df$Critics <- with(df, ifelse(Meta\_score >= 79, 1, 0))  
df$MainGenre <- with(df, ifelse(Genre1 == 'Drama' | Genre1 == 'Comedy' | Genre1 == 'Action' | Genre1 == 'Crime', Genre1, 'Other'))  
cols.char <- c('Genre1', 'Genre2', 'Genre3', 'MainGenre', 'Revenue', 'Rating', 'Year', 'Critics')  
df[cols.char] <- lapply(df[cols.char], factor)  
sapply(df, class)

## Released\_Year Runtime Genre IMDB\_Rating Meta\_score   
## "numeric" "character" "character" "numeric" "numeric"   
## Gross Time Last Genre1 Genre2   
## "numeric" "numeric" "character" "factor" "factor"   
## Genre3 Revenue Rating Year Critics   
## "factor" "factor" "factor" "factor" "factor"   
## MainGenre   
## "factor"

df <- df[, c(1, 4:7, 12:16)]

*Далее мы будем решать задачу классификации фильмов, представленных в этой базе. Эту задачу мы будем решать при помощи кластерного анализа.* *Однако до перехода к кластерному анализу, посмотрим при помощи визуального анализа на какие группы потенциально могут быть разделены фильмы.* *Для решения этой задачи по визуализации воспользуемся алгоритмом многомерного шкалирования.* *Многомерное шкалирование хорошо подходит для задач визуального отображения близости между объектами, помещенными в двумерное пространство.* *Для такой визуализации возьмем три переменных из нашей базы - рейтинг критиков, рейтинг аудитории и доход. Одной из базовых гипотез может быть предположение о том, что наша выборка в 1000 фильмов разделится на группу, включающие фильмы с высоким рейтингом и большой выручкой и группу, объединяющую фильмы с низким рейтингом и маленькой выручкой.*

df\_stand <- as.data.frame(scale(df[, 2:4])) # переменные измерены в разных шкалах, требуется стандартизация  
library(vegan)  
nmds\_result <- metaMDS(df\_stand, distance = 'euclidean', k = 2)

## Run 0 stress 0.129574   
## Run 1 stress 0.149246   
## Run 2 stress 0.1487514   
## Run 3 stress 0.1497175   
## Run 4 stress 0.1475529   
## Run 5 stress 0.1516755   
## Run 6 stress 0.1452058   
## Run 7 stress 0.1320571   
## Run 8 stress 0.1323526   
## Run 9 stress 0.1391267   
## Run 10 stress 0.1293104   
## ... New best solution  
## ... Procrustes: rmse 0.004614095 max resid 0.06883975   
## Run 11 stress 0.1295746   
## ... Procrustes: rmse 0.004616685 max resid 0.09955138   
## Run 12 stress 0.1514177   
## Run 13 stress 0.1392135   
## Run 14 stress 0.1295741   
## ... Procrustes: rmse 0.004592265 max resid 0.06890981   
## Run 15 stress 0.14909   
## Run 16 stress 0.1442248   
## Run 17 stress 0.1393085   
## Run 18 stress 0.1288349   
## ... New best solution  
## ... Procrustes: rmse 0.004464595 max resid 0.06889786   
## Run 19 stress 0.1404115   
## Run 20 stress 0.1282988   
## ... New best solution  
## ... Procrustes: rmse 0.002717603 max resid 0.05901673   
## \*\*\* Best solution was not repeated -- monoMDS stopping criteria:  
## 13: no. of iterations >= maxit  
## 7: scale factor of the gradient < sfgrmin

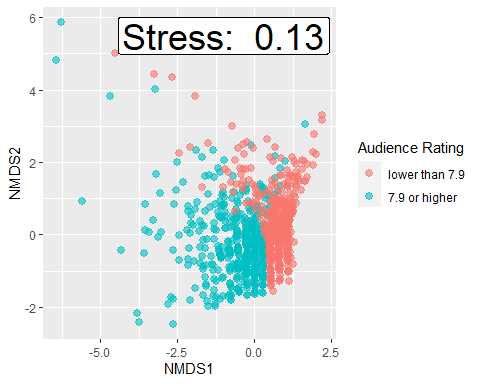
nmds\_result$stress # показатель говорит о том, насколько хорошо двумерное пространство подходит для отображения наших данных, то есть с какой точностью сработал алгоритм для наших данных (stress < 0.2 означает, что решение на два измерения хорошо подходит)

## [1] 0.1282988

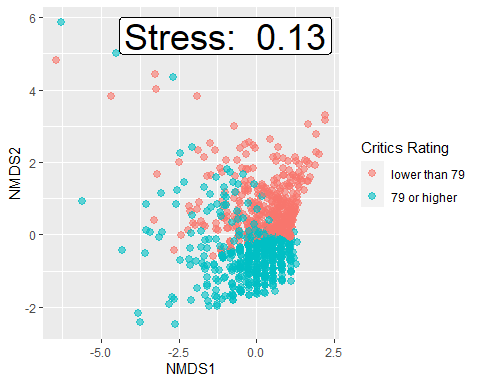
data\_scores <- as.data.frame(scores(nmds\_result))  
data\_scores <- cbind(data\_scores, df[, 6:10])

*Визуализируем наши данные при помощи координат, которые мы получили.*

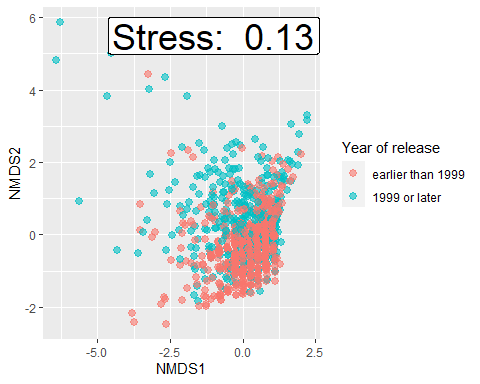
library(ggplot2)  
ggplot() +  
geom\_point(data = data\_scores, aes(x = NMDS1, y = NMDS2, color = Rating), size = 2.5, alpha = 0.6) +  
annotate(geom = 'label', x = -1, y = 5.5, size = 9.5, label = paste('Stress: ', round(nmds\_result$stress, digits = 2))) +  
scale\_color\_discrete(name = "Audience Rating", labels = c('lower than 7.9', '7.9 or higher'))



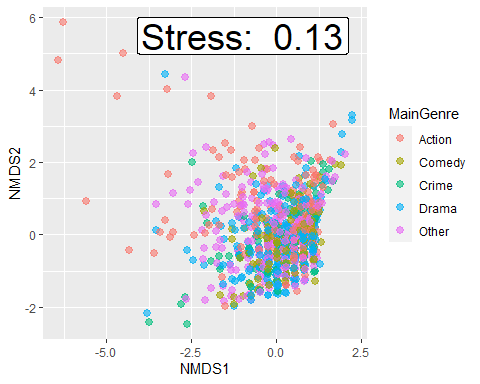
ggplot() +  
geom\_point(data = data\_scores, aes(x = NMDS1, y = NMDS2, color = Critics), size = 2.5, alpha = 0.6) +  
annotate(geom = 'label', x = -1, y = 5.5, size = 9.5, label = paste('Stress: ', round(nmds\_result$stress, digits = 2))) +  
scale\_color\_discrete(name = "Critics Rating", labels = c('lower than 79', '79 or higher'))

 *Как мы и предполагали, фильмы разделяются по рейтингу: в нашем случае горизонтальное измерение может быть описано при помощи информации об оценках аудитории, а вертикальное измерение через информацию об оценках критиков.*

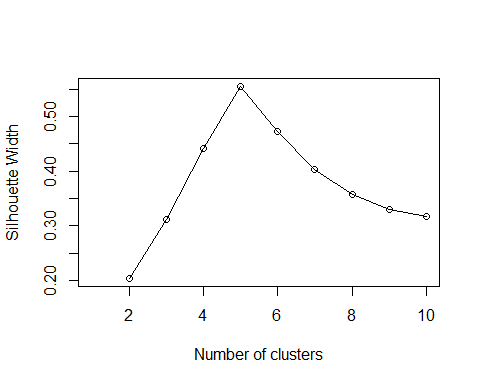
ggplot() +  
geom\_point(data = data\_scores, aes(x = NMDS1, y = NMDS2, color = Year), size = 2.5, alpha = 0.6) +  
annotate(geom = 'label', x = -1, y = 5.5, size = 9.5, label = paste('Stress: ', round(nmds\_result$stress, digits = 2))) +  
scale\_color\_discrete(name = "Year of release", labels = c('earlier than 1999', '1999 or later'))



ggplot() +  
 geom\_point(data = data\_scores, aes(x = NMDS1, y = NMDS2, color = MainGenre), size = 2.5, alpha = 0.6) +  
 annotate(geom = 'label', x = -1, y = 5.5, size = 9.5, label = paste('Stress: ', round(nmds\_result$stress, digits = 2)))

 *Как мы видим, в случае жанров или года релиза, визуализация не дает нам однозначно предположить, фильмы какого жанра (или какого года релиза) будут выше оценены критиками или аудиторией.* *Перейдем к задаче кластеризации. Для кластеризации мы будем использовать следующие переменные: длительность, год релиза, жанр, выручку, рейтинг аудитории и рейтинг критиков. Так как у нас и числовые, и категориальные переменные, используем алгоритм кластеризации PAM (gower distance).*

library(factoextra)  
library(cluster)  
library(fpc)  
df\_clust <- df[, 1:5]  
df\_clust <- as.data.frame(scale(df\_clust))   
df\_clust <- cbind(df\_clust, df[, 10])  
gower\_dist <- daisy(df\_clust, metric = "gower")  
sil\_width <- c(NA)  
for(i in 2:10){  
 pam\_fit <- pam(gower\_dist,  
 diss = TRUE,  
 k = i)  
 sil\_width[i] <- pam\_fit$silinfo$avg.width  
}  
plot(1:10, sil\_width,  
 xlab = "Number of clusters",  
 ylab = "Silhouette Width")  
lines(1:10, sil\_width) # оптимальное решение - 5 кластеров



set.seed(123)  
pam\_fit <- pam(gower\_dist, diss = TRUE, k = 5)  
pam\_fit$clusinfo

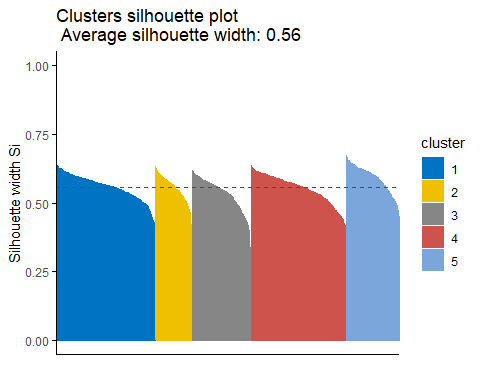
## size max\_diss av\_diss diameter separation  
## [1,] 289 0.2591401 0.09297776 0.4129711 0.1672705  
## [2,] 107 0.2810479 0.09431522 0.3966501 0.1726268  
## [3,] 172 0.2770521 0.10050997 0.4277344 0.1672705  
## [4,] 277 0.2128021 0.09423359 0.3749956 0.1684983  
## [5,] 155 0.2199266 0.08563907 0.3419781 0.1684983

dd <- cbind(df\_clust, cluster = pam\_fit$clustering)  
dd$cluster <- as.factor(dd$cluster)  
sapply(dd, class)

## Released\_Year IMDB\_Rating Meta\_score Gross Time   
## "numeric" "numeric" "numeric" "numeric" "numeric"   
## MainGenre cluster   
## "factor" "factor"

fviz\_silhouette(pam\_fit, palette = "jco", ggtheme = theme\_classic()) # проверяем насколько хорошо 5 кластеров описывают наши данные

## cluster size ave.sil.width  
## 1 1 289 0.56  
## 2 2 107 0.55  
## 3 3 172 0.53  
## 4 4 277 0.55  
## 5 5 155 0.59



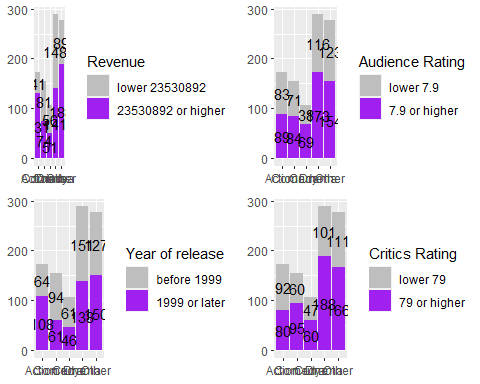
km\_stats <- cluster.stats(gower\_dist, pam\_fit$clustering)  
km\_stats$dunn

## [1] 0.3910617

dd <- cbind(dd, df[, 6:9])

*Сделаем несколько изображений, чтобы более подробно описать полученные кластеры с точки зрения жанров фильмов, года релиза, выручки и рейтингов. Как мы видим, жанр стал основной переменной, разделивший кластеры.*

library("gridExtra")  
p1 <- ggplot(dd, aes(MainGenre, fill = Revenue)) +  
geom\_bar() +  
scale\_fill\_manual(values = c("grey", "purple"), labels = c('lower 23530892', '23530892 or higher')) +   
theme(axis.title.x = element\_blank(),  
axis.title.y = element\_blank()) +   
geom\_text(aes(label= ..count..), stat = "count", position = position\_stack(vjust = 0.5))  
p2 <- ggplot(dd, aes(MainGenre, fill = Rating)) +  
geom\_bar() +  
scale\_fill\_manual(values = c("grey", "purple"), name = 'Audience Rating', labels = c('lower 7.9', '7.9 or higher')) +   
theme(axis.title.x = element\_blank(),  
 axis.title.y = element\_blank()) +   
geom\_text(aes(label= ..count..), stat = "count", position = position\_stack(vjust = 0.5))  
p3 <- ggplot(dd, aes(MainGenre, fill = Year)) +  
geom\_bar() +  
scale\_fill\_manual(values = c("grey", "purple"), name = 'Year of release', labels = c('before 1999', '1999 or later')) +   
 theme(axis.title.x = element\_blank(),  
 axis.title.y = element\_blank()) +   
geom\_text(aes(label= ..count..), stat = "count", position = position\_stack(vjust = 0.5))  
p4 <- ggplot(dd, aes(MainGenre, fill = Critics)) +  
geom\_bar() +  
scale\_fill\_manual(values = c("grey", "purple"), name = 'Critics Rating', labels = c('lower 79', '79 or higher')) +   
theme(axis.title.x = element\_blank(),  
 axis.title.y = element\_blank()) +   
geom\_text(aes(label= ..count..), stat = "count", position = position\_stack(vjust = 0.5))  
grid.arrange(p1, p2, p3, p4)

 *Далее мы будем решать задачу предсказания высокой выручки фильмов. Здесь для нас высокая выручка - равная или больше медианного значения 23530892. Для начала построим модель логистической регрессии.*

df\_logit <- df[, c(1:3, 5:6, 10)]  
levels(df\_logit$MainGenre) <- c('Other', 'Comedy', 'Action', 'Crime', 'Drama')  
model\_reg <- glm(Revenue ~ ., data = df\_logit, family = binomial)  
pR2 = 1 - model\_reg$deviance / model\_reg$null.deviance  
pR2

## [1] 0.04596996

library(DescTools)  
PseudoR2(model\_reg, c('McFadden', 'Nagel'))

## McFadden Nagelkerke   
## 0.04596996 0.08144842

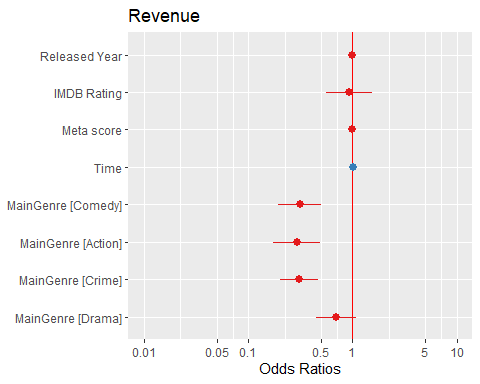
coef <- as.data.frame(exp(coef(model\_reg))/(1+exp(coef(model\_reg))))  
coef

## exp(coef(model\_reg))/(1 + exp(coef(model\_reg)))  
## (Intercept) 0.9673190  
## Released\_Year 0.4997598  
## IMDB\_Rating 0.4827740  
## Meta\_score 0.4986066  
## Time 0.5012788  
## MainGenreComedy 0.2387508  
## MainGenreAction 0.2257425  
## MainGenreCrime 0.2366327  
## MainGenreDrama 0.4119085

library(performance)  
check\_collinearity(model\_reg)

## # Check for Multicollinearity  
##   
## Low Correlation  
##   
## Term VIF VIF 95% CI Increased SE Tolerance Tolerance 95% CI  
## Released\_Year 1.16 [1.09, 1.26] 1.08 0.86 [0.79, 0.91]  
## IMDB\_Rating 1.17 [1.10, 1.27] 1.08 0.86 [0.79, 0.91]  
## Meta\_score 1.18 [1.11, 1.28] 1.09 0.85 [0.78, 0.90]  
## Time 1.14 [1.08, 1.24] 1.07 0.88 [0.80, 0.93]  
## MainGenre 1.08 [1.03, 1.20] 1.04 0.92 [0.84, 0.97]

library(sjPlot)  
plot\_model(model\_reg, vline.color = 'red')

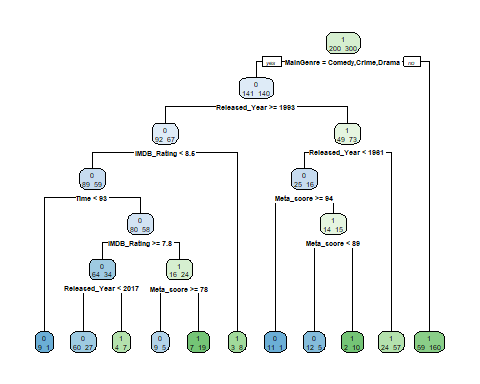


summary(model\_reg) # Значимыми для предсказания высокой выручки является переменная длительность фильма и переменная жанр фильма: фильмы с большей длительностью и жанром Other (так как мы перекодировали переменную жанр, в этом случае значит, что жанр должен быть не драма, комедия, экшен или криминал) с наибольшей вероятностью соберут большую кассу.

##   
## Call:  
## glm(formula = Revenue ~ ., family = binomial, data = df\_logit)  
##   
## Coefficients:  
## Estimate Std. Error z value Pr(>|z|)   
## (Intercept) 3.3877354 6.6519362 0.509 0.6106   
## Released\_Year -0.0009608 0.0030369 -0.316 0.7517   
## IMDB\_Rating -0.0689314 0.2611217 -0.264 0.7918   
## Meta\_score -0.0055735 0.0063677 -0.875 0.3814   
## Time 0.0051154 0.0025839 1.980 0.0477 \*   
## MainGenreComedy -1.1595403 0.2458541 -4.716 2.40e-06 \*\*\*  
## MainGenreAction -1.2325093 0.2655397 -4.642 3.46e-06 \*\*\*  
## MainGenreCrime -1.1712299 0.2171724 -5.393 6.93e-08 \*\*\*  
## MainGenreDrama -0.3560813 0.2231276 -1.596 0.1105   
## ---  
## Signif. codes: 0 '\*\*\*' 0.001 '\*\*' 0.01 '\*' 0.05 '.' 0.1 ' ' 1  
##   
## (Dispersion parameter for binomial family taken to be 1)  
##   
## Null deviance: 1357.3 on 999 degrees of freedom  
## Residual deviance: 1294.9 on 991 degrees of freedom  
## AIC: 1312.9  
##   
## Number of Fisher Scoring iterations: 4

*Далее для решения задачи предсказания большой выручки попробуем алгоритм дерева решений.*

library(rpart)  
df\_tree <- df[, c(1:3, 5:6, 10)]  
set.seed(456)  
train.index <- sample(c(1:dim(df\_tree)[1]), dim(df\_tree)[1]\*0.5) # разделяем выборку на обучающую и тестовую  
train.df <- df\_tree[train.index, ]  
valid.df <- df\_tree[-train.index, ]  
model <- rpart(Revenue ~ ., data = train.df, method = 'class', control = rpart.control(minsplit = 7, minbucket = 10, maxdepth = 10))  
library(rpart.plot)  
library(RColorBrewer)  
library(rattle)  
rpart.plot(model, type=2, extra = 1)

 *Как и в случае модели логистической регрессии, здесь мы видим, что жанр является важной переменной для предсказания высокой выручки фильма.* *Следующая развилка - это год релиза фильма: если релиз был в 1993 или позже, тогда для нас важен рейтинг зрителей; если ранее 1993, тогда следующая развилка - это год релиза ранее 1961 и затем рейтинг критиков. Интересное наблюдение, что если фильм недавнего производства (1993 и позже), тогда для предсказания высокой выручки важен рейтинг аудитории, но для более поздних фильмов (до 1961) важен рейтинг критиков. Однако мы должны с осторожностью интерпретировать эти результаты, потому что данная модель дает верное предсказание только в 61% случаев.*

predicted\_measure1 <- predict(model, train.df[ , -5], type = 'class')  
table\_mat1 <- table(train.df$Revenue, predicted\_measure1)  
table\_mat1 # смотрим какое число наблюдений было верно распознано на обучающей выборке

## predicted\_measure1  
## 0 1  
## 0 101 99  
## 1 39 261

predicted\_measure2 <- predict(model, valid.df[ , -5], type = 'class')  
table\_mat2 <- table(valid.df$Revenue, predicted\_measure2)  
table\_mat2 # смотрим какое число наблюдений было верно распознано на тестовой выборке

## predicted\_measure2  
## 0 1  
## 0 92 123  
## 1 72 213

accuracy\_Test <- sum(diag(table\_mat2)) / sum(table\_mat2)  
print(paste('Accuracy for test', accuracy\_Test))

## [1] "Accuracy for test 0.61"

*Чтобы улучшить модель, мы используем алгоритм Random Forest.*

library(randomForest)  
set.seed(3217)  
ntree.1 <- 45 # пробуем решения для разного числа деревьев, останавливаемся на том решении, где число ошибок распознавания не меняется  
nodesize.1 <-1  
keep.forest.1 <- TRUE  
modelRandomForest <- randomForest(Revenue ~., data = train.df, ntree=ntree.1, mtry=floor(sqrt(ncol(train.df))), replace=FALSE, nodesize = nodesize.1, importance=TRUE, localImp=FALSE, proximity=FALSE, norm.votes=TRUE, do.trace=ntree.1/10, keep.forest=keep.forest.1, corr.bias=FALSE, keep.inbag=FALSE)

## ntree OOB 1 2  
## 4: 43.48% 46.30% 41.67%  
## 8: 40.17% 46.07% 36.30%  
## 12: 42.28% 51.26% 36.33%  
## 16: 41.80% 48.00% 37.67%  
## 20: 39.60% 49.50% 33.00%  
## 24: 39.60% 49.00% 33.33%  
## 28: 40.60% 52.50% 32.67%  
## 32: 39.60% 53.00% 30.67%  
## 36: 39.40% 53.50% 30.00%  
## 40: 39.80% 53.50% 30.67%  
## 44: 39.40% 54.00% 29.67%

tree\_predicted1 <- predict(modelRandomForest, newdata = train.df[, -5])  
tree\_predicted2 <- predict(modelRandomForest, newdata = valid.df[, -5])  
tablerf1 <- table(train.df$Revenue, tree\_predicted1)  
tablerf2 <- table(valid.df$Revenue, tree\_predicted2) # смотрим улучшилась ли работа алгоритма по количеству верно распознанных наблюдений на тестовой выборке  
varImpPlot(modelRandomForest, sort=F) # можем увидеть какие переменные наиболее важны для классификатора

