

Факторный и дискриминантный анализ.

Вейбер Е.Н. 23.М08-мм

02-05-2024

Введение

Это отчет о проведении факторного и дискриминантного анализа. Вариант 18.

Построение матрицы факторных нагрузок и двумерной диаграммы первых двух факторов

```
# Загрузка необходимых библиотек
```

```
library(nlstools)
```

```
library(ggplot2)
```

```
library(tidyverse)
```

```
library(readr)
```

```
library(corrplot)
```

```
library(tidyr)
```

```
library(GGally)
```

```
library(plotly)
```

```
library(reshape2)
```

```
# Загрузка данных
```

```
data <- read_delim("~/Downloads/addicts.csv", delim = ";")
```

```
head(data)
```

```
## # A tibble: 6 × 27
```


```
##   prcod intpla sex age educat curwor asi1_med asi2_emp asi3_alc asi4  
_dr
```

```
##   <dbl> <dbl> <dbl> <dbl> <dbl> <dbl> <chr> <chr> <chr> <chr>  
>
```

```
## 1      4      1      0     18      1      1 0,19      0,7      0,12      0,3
```

```
## 2      2      2      1     30      4      1 0,44      0,2      0,01      0,3
```

```
## 3      2      1      0     23      2      0 0,50      1,0      0,30      0,3
```

```
## 4      4      1      0      20      2      1 0,00      0,8      0,05      0,3
## 5      3      2      0      20      2      0 0,00      0,8      0,78      0,2
## 6      1      1      0      24      2      0 0,52      0,5      0,10      0,3
## #  17 more variables: asi5_leg <chr>, asi6_soc <chr>, asi7_psy <chr>,
## #   asid3_dyr <chr>, tlfba2 <chr>, tlfbh2 <chr>, st <chr>, ha <chr>, se
## #   <chr>,
## #   cravin <chr>, rabdru <dbl>, rubsex <dbl>, gaf <dbl>, bdi <chr>,
## #   sstat1 <dbl>, end <chr>, endpo <chr>
```

```
# Подготовка данных
```

```
my_data <- data[, c("rubsex", "tlfba2", "asi3_alc", "sstat1", "tlfbh2")]
```

```
# Преобразование из <chr> с запятой в <dbl>
```

```
# Преобразование всех колонок в data
```

```
for (i in names(my_data)) {
```

```
  if (is.character(my_data[[i]])) {
```

```
    my_data[[i]] <- gsub(" ", "", my_data[[i]]) # Удаляем пробелы
```

```
    my_data[[i]] <- gsub(",", ".", my_data[[i]]) # Заменяем запятые на точки
```

```
    my_data[[i]] <- as.numeric(my_data[[i]]) # Преобразование в числовой тип
```

```
  }
```

```
}
```

```
library(base)
```

```
# Пропускаем NA значения
```

```
my_data <- na.omit(my_data)
```

```
# Выполнение PCA
```

```
pca_result <- prcomp(my_data, scale. = TRUE) # Масштабирование данных для стандартизации
```

```
# Получение факторных нагрузок
```

```
loadings <- pca_result$rotation
```

```
# Создание DataFrame для факторных нагрузок
```

```
loadings_df <- as.data.frame(loadings)
```

```
names(loadings_df) <- paste0("PC", 1:ncol(loadings_df))
```

```
# Печать результатов
```

```
print(loadings_df)
```

##		PC1	PC2	PC3	PC4	PC5
##	rubsex	0.04404728	-0.71338811	0.009373247	-0.69837028	0.03644632
##	tlfba2	0.69521278	0.09774063	0.022558729	-0.01855853	0.71152838
##	asi3_alc	0.67727888	0.17964607	-0.043907292	-0.17737327	-0.68966041
##	sstat1	0.12121025	-0.38605279	-0.826926465	0.38938594	-0.02902629
##	tlfbh2	0.20332342	-0.54791851	0.560060726	0.57345606	-0.12619439

```
# Просмотр долей объясненной дисперсии
```

```
print(summary(pca_result))
```

```
## Importance of components:
```

##		PC1	PC2	PC3	PC4	PC5
##	Standard deviation	1.1356	1.0697	0.9981	0.9215	0.8491
##	Proportion of Variance	0.2579	0.2288	0.1992	0.1698	0.1442
##	Cumulative Proportion	0.2579	0.4868	0.6860	0.8558	1.0000

```
# Извлечение результатов PCA для первых двух главных компонент
```

```
scores <- as.data.frame(pca_result$x[, 1:2])
```

```
# Создание двумерной диаграммы первых двух компонент
```

```
ggplot(scores, aes(x = PC1, y = PC2)) +
```

```
  geom_point(aes(color = my_data$tlfbh2), alpha = 0.7) + # Цвет точек по з  
ависимой переменной, если нужно
```

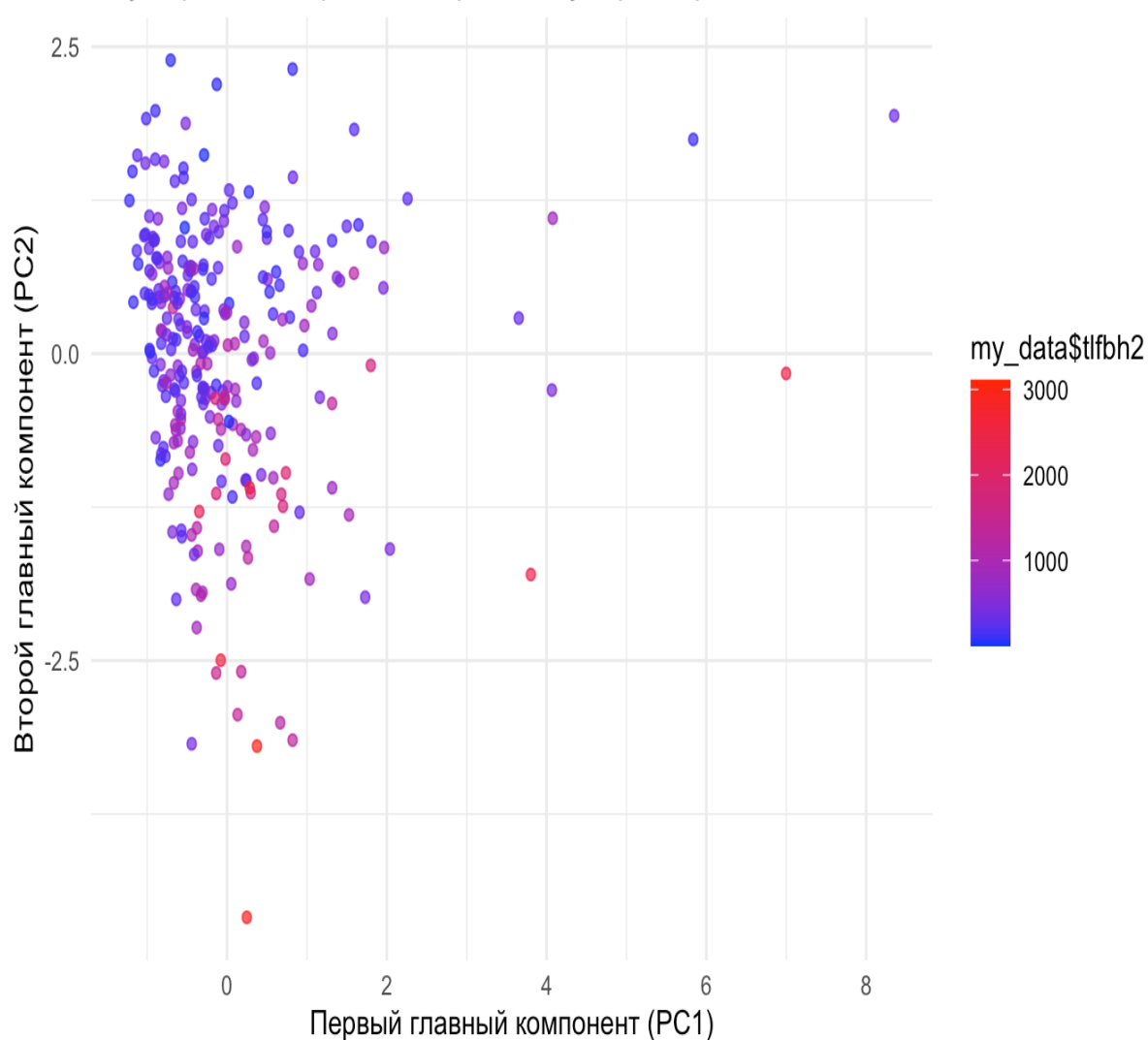
```
  labs(x = "Первый главный компонент (PC1)", y = "Второй главный компонент  
(PC2)",
```

```
        title = "Двумерная диаграмма первых двух факторов") +
```

```
  theme_minimal() +
```

```
  scale_color_gradient(low = "blue", high = "red") # Изменение градиента ц  
вета, если это необходимо
```

Двумерная диаграмма первых двух факторов



Факторные нагрузки:

- **PC1:**
 - `rubsex`: 0.044, маленькое значение, показывает слабую связь переменной с первым компонентом.
 - `tlfba2`: 0.695, значительное положительное влияние на первый компонент.
 - `asi3_alc`: 0.677, также значительное положительное влияние на первый компонент.
 - `sstat1`: 0.121, небольшое положительное влияние.
 - `tlfbh2`: 0.203, умеренное положительное влияние.
- **PC2:**
 - `rubsex`: -0.713, значительное отрицательное влияние.
 - `tlfba2`: 0.098, очень небольшое положительное влияние.
 - `asi3_alc`: 0.180, небольшое положительное влияние.
 - `sstat1`: -0.386, умеренное отрицательное влияние.
 - `tlfbh2`: -0.548, значительное отрицательное влияние.

Интерпретация:

- **PC1** может быть интерпретирован как компонент, который в основном связан с переменными `tlfba2` и `asi3_alc`, что может указывать на общую тенденцию в данных, связанную с этими двумя переменными.
- **PC2** выявляет взаимосвязь между `rubsex`, `sstat1`, и `tlfbh2`, где `rubsex` и `tlfbh2` имеют сильные отрицательные нагрузки, что может предполагать различное воздействие или отрицательную корреляцию этих переменных относительно других переменных.

Вклад в общую дисперсию:

- **PC1** (Первый главный компонент):
 - Стандартное отклонение: 1.1356.
 - Доля объяснённой дисперсии: 25.79% (0.2579).
- **PC2** (Второй главный компонент):
 - Стандартное отклонение: 1.0697.
 - Доля объяснённой дисперсии: 22.88% (0.2288).

Кумулятивный вклад:

- Суммарный вклад первых двух компонентов в дисперсию данных составляет **48.68%** (0.4868), что показывает, что почти половина общей изменчивости данных может быть описана первыми двумя главными компонентами.


Это означает, что первые два компонента эффективно захватывают значительную часть информации, содержащейся в исходных данных. Такой уровень объяснённой дисперсии считается достаточно высоким в контексте PCA, особенно если учитывать, что вся модель с пятью компонентами объясняет 100% дисперсии.

Построение дискриминантной функции, оценка граничного значения в случае эвристической и байесовской процедуры

```
# Подгружаем необходимые библиотеки
library(MASS)
library(caret) # Для разделения данных и оценки точности

# Загрузка данных
data <- read_delim("~/Downloads/addicts.csv", delim = ";")
head(data)

## # A tibble: 6 × 27
```

```
## prcod intpla sex age educat curwor asi1_med asi2_emp asi3_alc asi4_dr
## <dbl> <dbl> <dbl> <dbl> <dbl> <dbl> <chr> <chr> <chr> <chr>
>
## 1 4 1 0 18 1 1 0,19 0,7 0,12 0,3
## 2 2 2 1 30 4 1 0,44 0,2 0,01 0,3
## 3 2 1 0 23 2 0 0,50 1,0 0,30 0,3
## 4 4 1 0 20 2 1 0,00 0,8 0,05 0,3
## 5 3 2 0 20 2 0 0,00 0,8 0,78 0,2
## 6 1 1 0 24 2 0 0,52 0,5 0,10 0,3
## #  17 more variables: asi5_leg <chr>, asi6_soc <chr>, asi7_psy <chr>,
## # asid3_dyr <chr>, tlfba2 <chr>, tlfbh2 <chr>, st <chr>, ha <chr>, se
## # cravin <chr>, rabdru <dbl>, rubsex <dbl>, gaf <dbl>, bdi <chr>,
## # sstat1 <dbl>, end <chr>, endpo <chr>
```

```
# Подготовка данных
```

```
my_data <- data[, c("rubsex", "tlfba2", "asi3_alc", "sstat1", "tlfbh2", "end")]
```

```
head(my_data)
```

```
## # A tibble: 6 × 6
## rubsex tlfba2 asi3_alc sstat1 tlfbh2 end
## <dbl> <chr> <chr> <dbl> <chr> <chr>
## 1 4 8,9 0,12 60 231,7 0
## 2 5 0,3 0,01 50 727,8 0
## 3 1 7,9 0,30 55 1 189,0 0
## 4 4 1,3 0,05 58 671,0 0
## 5 4 72,0 0,78 58 50,0 0
## 6 2 6,0 0,10 68 1 844,0 0
```

```
# Преобразование из <chr> с запятой в <dbl>
```

```
# Преобразование всех колонок в data
```

```
for (i in names(my_data)) {
  if (is.character(my_data[[i]])) {
    my_data[[i]] <- gsub(" ", "", my_data[[i]]) # Удаляем пробелы
    my_data[[i]] <- gsub(",", ".", my_data[[i]]) # Заменяем запятые на точки
    my_data[[i]] <- as.numeric(my_data[[i]]) # Преобразование в числовой тип
  }
}
```

```

# Пропускаем NA значения
my_data <- na.omit(my_data)

target <- my_data$end
head(target)
## [1] 0 0 0 0 0 0

# Разделение данных на тренировочные и тестовые
set.seed(81) # Для воспроизводимости разделения
split <- createDataPartition(my_data$end, p = 0.7, list = FALSE)
train_data <- my_data[split, ]
test_data <- my_data[-split, ]

# Обучение дискриминантной модели
lda_model <- lda(end ~ ., data = train_data)

# Предсказание на тестовой выборке
predictions <- predict(lda_model, newdata = test_data)
y_pred <- predictions$class

# Точность
accuracy <- sum(test_data$end == y_pred) / nrow(test_data)
print(sprintf('Точность дискриминантной модели: %.3f', accuracy))
## [1] "Точность дискриминантной модели: 0.699"

# Создание confusion matrix
conf_matrix <- confusionMatrix(as.factor(y_pred), as.factor(test_data$end))

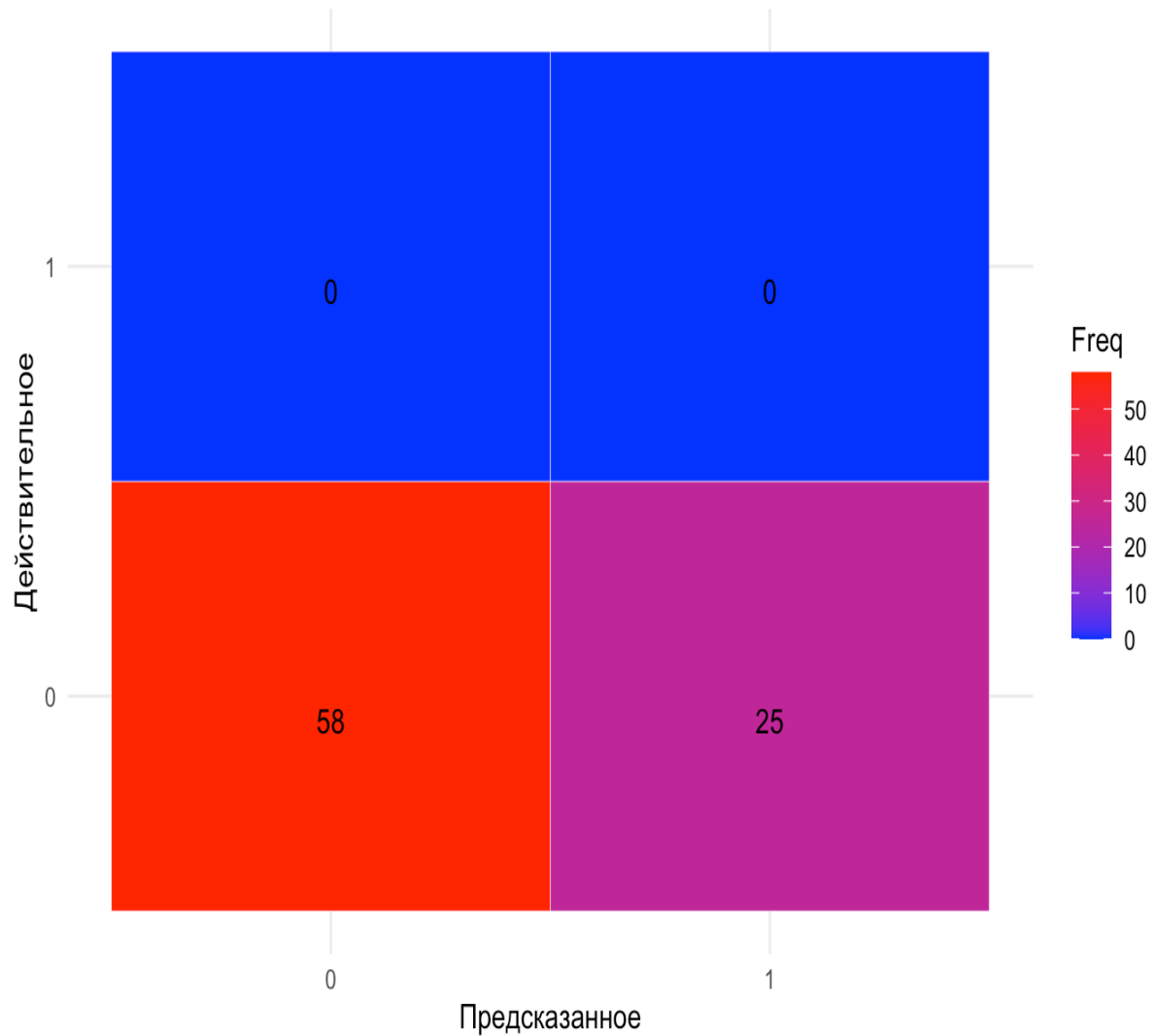
# Получение таблицы сопряженности
conf_table <- as.table(conf_matrix$table)

# Построение тепловой карты матрицы классификации
ggplot(data = as.data.frame(conf_table), aes(x = Reference, y = Prediction,
fill = Freq)) +
  geom_tile(color = "white") +
  geom_text(aes(label = Freq), vjust = 1.5, color = "black") +
  scale_fill_gradient(low = "blue", high = "red") +

```

```
labs(title = "Матрица классификации", x = "Предсказанное", y = "Действительное") +  
theme_minimal()
```

Матрица классификации



```
print(conf_matrix)  
  
## Confusion Matrix and Statistics  
##  
##           Reference  
## Prediction  0   1  
##           0 58 25  
##           1  0  0  
##  
##           Accuracy : 0.6988  
##           95% CI : (0.5882, 0.7947)
```



```
##      No Information Rate : 0.6988
##      P-Value [Acc > NIR] : 0.5538
##
##              Kappa : 0
##
##  McNemar's Test P-Value : 1.587e-06
##
##              Sensitivity : 1.0000
##              Specificity : 0.0000
##              Pos Pred Value : 0.6988
##              Neg Pred Value :    NaN
##              Prevalence : 0.6988
##              Detection Rate : 0.6988
##      Detection Prevalence : 1.0000
##      Balanced Accuracy : 0.5000
##
##      'Positive' Class : 0
##
```

```
#####
```

```
# Вычисление дискриминантных значений
```

```
lda_values <- predict(lda_model, my_data)$x
```

```
# Расчет эвристического граничного значения
```

```
heuristic_threshold <- mean(lda_values[,1]) # Предполагаем, что речь идет о первой дискриминантной функции
```

```
# Расчет Байесовского граничного значения
```

```
priors <- lda_model$prior
```

```
means <- colMeans(lda_values)
```

```
scaling_coef <- lda_model$scaling[, 1] # Предполагаем, что речь идет о первой дискриминантной функции
```

```
intercept <- -sum(means * scaling_coef)
```

```
bayesian_threshold <- (log(priors[2] / priors[1]) - intercept) / scaling_coef
```

```
print(paste("Эвристическое граничное значение:", heuristic_threshold))
```

```
## [1] "Эвристическое граничное значение: 0.0253528574442031"
```

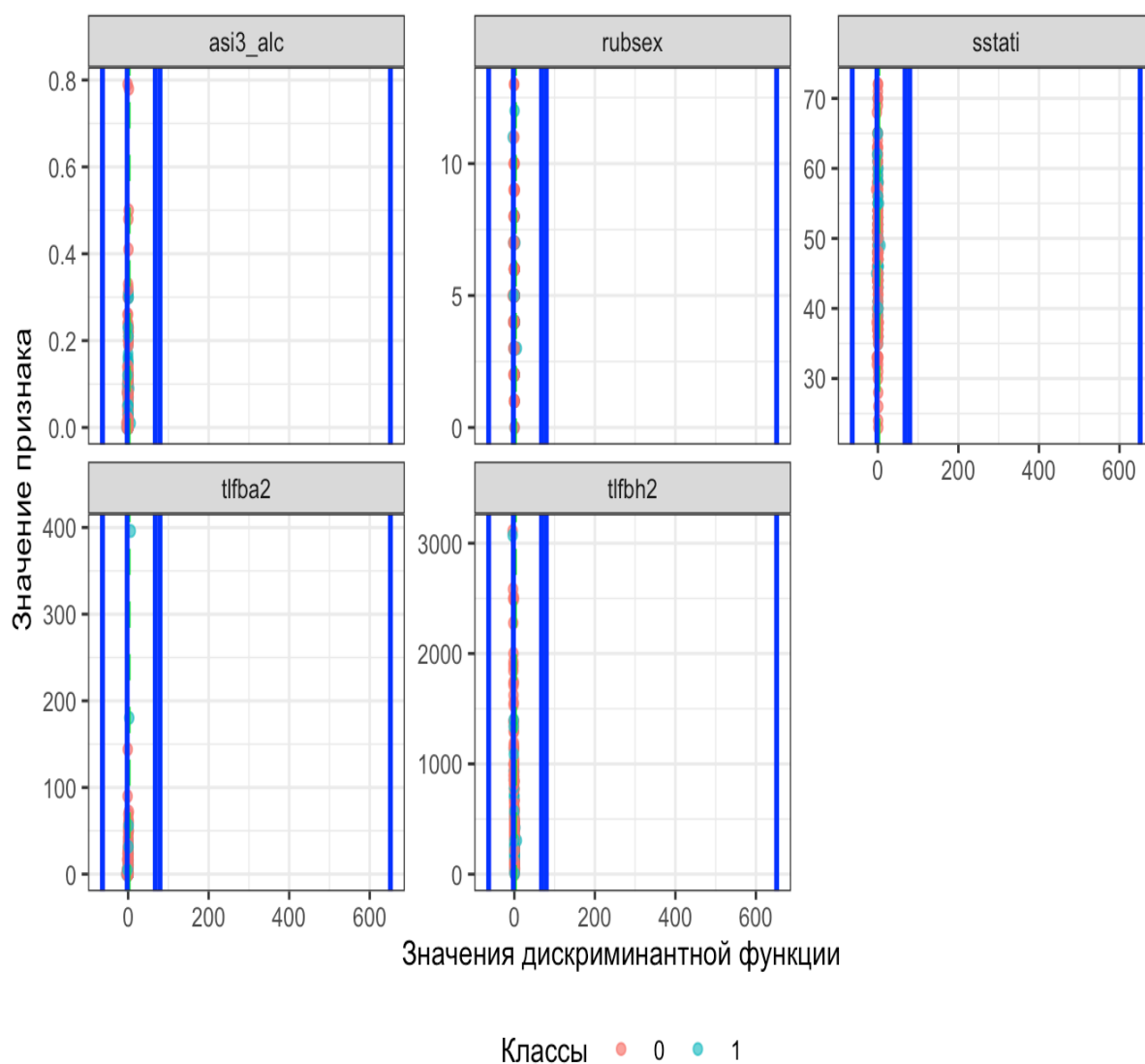
```
print(paste("Байесовское граничное значение:", bayesian_threshold))
```

```
## [1] "Байесовское граничное значение: 79.2265420512626"
## [2] "Байесовское граничное значение: -63.749649880233"
## [3] "Байесовское граничное значение: -2.38379702931434"
## [4] "Байесовское граничное значение: 67.4365966905426"
## [5] "Байесовское граничное значение: 651.423075998572"

my_data$lda_values <- lda_values
features <- c("rubsex", "tlfba2", "asi3_alc", "sstati", "tlfbh2")
my_data_long <- my_data %>%
  pivot_longer(cols = features, names_to = "feature", values_to = "value")

# Создание ggplot графика с фасетами для каждой переменной из 'features'
p <- ggplot(my_data_long, aes(x = lda_values, y = value, color = factor(end
))) +
  geom_point(alpha = 0.7) +
  facet_wrap(~ feature, scales = "free_y") +
  geom_vline(xintercept = heuristic_threshold, color = "green", linetype =
"dashed", size = 1) +
  geom_vline(xintercept = bayesian_threshold, color = "blue", linetype = "s
olid", size = 1) +
  labs(color = "Классы") +
  theme_bw() +
  theme(legend.position = "bottom") +
  labs(x = "Значения дискриминантной функции", y = "Значение признака")

# Вывод графика
print(p)
```



Основные характеристики модели:

1. Confusion Matrix (Матрица ошибок):

- True Negative (TN, Истинно отрицательные): 58
- False Positive (FP, Ложно положительные): 25
- False Negative (FN, Ложно отрицательные): 0
- True Positive (TP, Истинно положительные): 0

Модель не смогла верно классифицировать ни один истинно положительный случай ($TP = 0$), что говорит о том, что она полностью игнорирует или не может определить второй класс.

2. Accuracy (Точность): 69.88%

- Точность модели сравнима с No Information Rate (NIR), что означает, что модель не лучше случайного угадывания класса, основанного на самом частом классе в данных.

3. Кappa (Каппа коэффициент): 0

- Значение Каппа, равное нулю, подтверждает, что согласие между предсказаниями и фактическими значениями не лучше случайного.
- 4. **McNemar's Test (Тест Макнемара):**
 - P-value ~ 0.00000159, что указывает на статистически значимое различие между ложно положительными и ложно отрицательными результатами. В данном случае, модель сильно склонна к ошибкам первого рода (ложные положительные).
- 5. **Sensitivity (Чувствительность): 100%**
 - Модель идеально определяет истинно отрицательные результаты (все нули корректно классифицированы), но при этом не определяет истинно положительные вовсе.
- 6. **Specificity (Специфичность): 0%**
 - Модель не определяет ни одного истинно положительного результата, что делает специфичность нулевой.
- 7. **Positive Predictive Value (PPV, Положительная предсказательная стоимость): 69.88%**
 - Совпадает с долей класса 0 в данных, поскольку модель всегда предсказывает класс 0.
- 8. **Negative Predictive Value (NPV, Отрицательная предсказательная стоимость): NaN**
 - Не определено, поскольку нет истинно положительных или ложно отрицательных предсказаний.
- 9. **Balanced Accuracy (Сбалансированная точность): 50%**
 - Среднее значение чувствительности и специфичности, показывает, что модель работает плохо в терминах равновесия между обнаружением классов.

Эвристическое граничное значение

Эвристическое граничное значение, равное приблизительно 0.025, представляет собой среднее значение дискриминантных оценок по всем наблюдениям в датасете.

Байесовские граничные значения

Каждое из этих значений представляет пороговое значение для различных дискриминантных функций. Различные пороги могут указывать на разные правила для разделения данных на классы в зависимости от условий, определенных этими функциями. Большое и малое значения, особенно такие как 651.42 и -63.75, могут указывать на то, что для некоторых классов очень мало наблюдений или что классы сильно отличаются по своим характеристикам, что приводит к экстремальным значениям в расчетах порогов.

Анализ графика

На графике изображены панели для разных переменных (`asi3_alc`, `rubsex`, `sstat1`, `tlfba2`, `tlfbh2`), каждая из которых показывает распределение значений переменной относительно значений дискриминантной функции. Разные цвета точек представляют разные классы, указанные в переменной `target`.

Наблюдения:

1. **Разделение классов:** Во всех панелях классы (обозначенные разными цветами) хорошо разделяются вдоль оси дискриминантной функции. Это свидетельствует о том, что дискриминантная функция эффективно разделяет классы по этим переменным.
2. **Граничные значения:** Зелёная пунктирная линия (эвристическое граничное значение) и синие сплошные линии (байесовские граничные значения) показывают пороги для классификации. Видно, что байесовские граничные значения могут варьироваться значительно между разными переменными.
3. **Сжатие оси X:** Ось X, представляющая значения дискриминантной функции, показывает, что большинство значений сосредоточено в узком диапазоне, что может указывать на высокую дискриминантную способность функции для этого набора данных.

Построение деревьев классификации

```
library(rpart)

# Модель с информационным выигрышем (Используем критерий "entropy")
tree_gain <- rpart(factor(train_data$end) ~ ., data = train_data, method =
"class", parms = list(split = "information"))

# Стандартная модель (Используем критерий "gini")
tree_standard <- rpart(factor(train_data$end) ~ ., data = train_data, metho
d = "class", parms = list(split = "gini"))

# Предсказания для модели на основе информационного выигрыша
y_pred_gain <- predict(tree_gain, test_data, type = "class")

# Предсказания для стандартной модели
y_pred_standard <- predict(tree_standard, test_data, type = "class")

# Расчет точности для модели с информационным выигрышем
accuracy_gain <- confusionMatrix(y_pred_gain, factor(test_data$end))$overal
l['Accuracy']

# Расчет точности для стандартной модели
accuracy_standard <- confusionMatrix(y_pred_standard, factor(test_data$end)
)$overall['Accuracy']

# Вывод результатов
cat(sprintf("Точность модели с информационным выигрышем: %.3f\n", accuracy_
gain))

## Точность модели с информационным выигрышем: 0.566
```

```

cat(sprintf("Точность стандартной модели: %.3f\n", accuracy_standard))
## Точность стандартной модели: 0.602

library(gridExtra)

plot_confusion_matrix <- function(y_true, y_pred, title) {
  # Преобразуем y_pred и y_true в факторы для работы confusionMatrix
  conf_matrix <- confusionMatrix(as.factor(y_pred), as.factor(y_true))
  print(conf_matrix)

  # Преобразуем таблицу матрицы ошибок в dataframe для ggplot
  conf_matrix_df <- as.data.frame(conf_matrix$table)
  names(conf_matrix_df) <- c("Reference", "Prediction", "Freq")

  # Построение матрицы ошибок с помощью ggplot
  ggplot(conf_matrix_df, aes(x = Reference, y = Prediction, fill = Freq)) +
    geom_tile(colour = "white") + # Здесь теперь корректно используется x
и y
    geom_text(aes(label = Freq), vjust = 1.5, color = "black") +
    scale_fill_gradient(low = "white", high = "steelblue") +
    labs(title = title, x = "Предсказанное", y = "Действительное") +
    theme_minimal() +
    theme(axis.text.x = element_text(angle = 45, hjust = 1),
          plot.title = element_text(hjust = 0.5))
}

# Создание графиков для каждой модели
p1 <- plot_confusion_matrix(test_data$end, y_pred_gain, "Модель дерево с ин
формационным выигрышем")

## Confusion Matrix and Statistics
##
##              Reference
## Prediction  0  1
##           0 45 23
##           1 13  2
##
##              Accuracy : 0.5663
##              95% CI : (0.4529, 0.6747)
##              No Information Rate : 0.6988

```

```
##      P-Value [Acc > NIR] : 0.9962
##
##      Kappa : -0.1626
##
##      McNemar's Test P-Value : 0.1336
##
##      Sensitivity : 0.7759
##      Specificity : 0.0800
##      Pos Pred Value : 0.6618
##      Neg Pred Value : 0.1333
##      Prevalence : 0.6988
##      Detection Rate : 0.5422
##      Detection Prevalence : 0.8193
##      Balanced Accuracy : 0.4279
##
##      'Positive' Class : 0
##
```

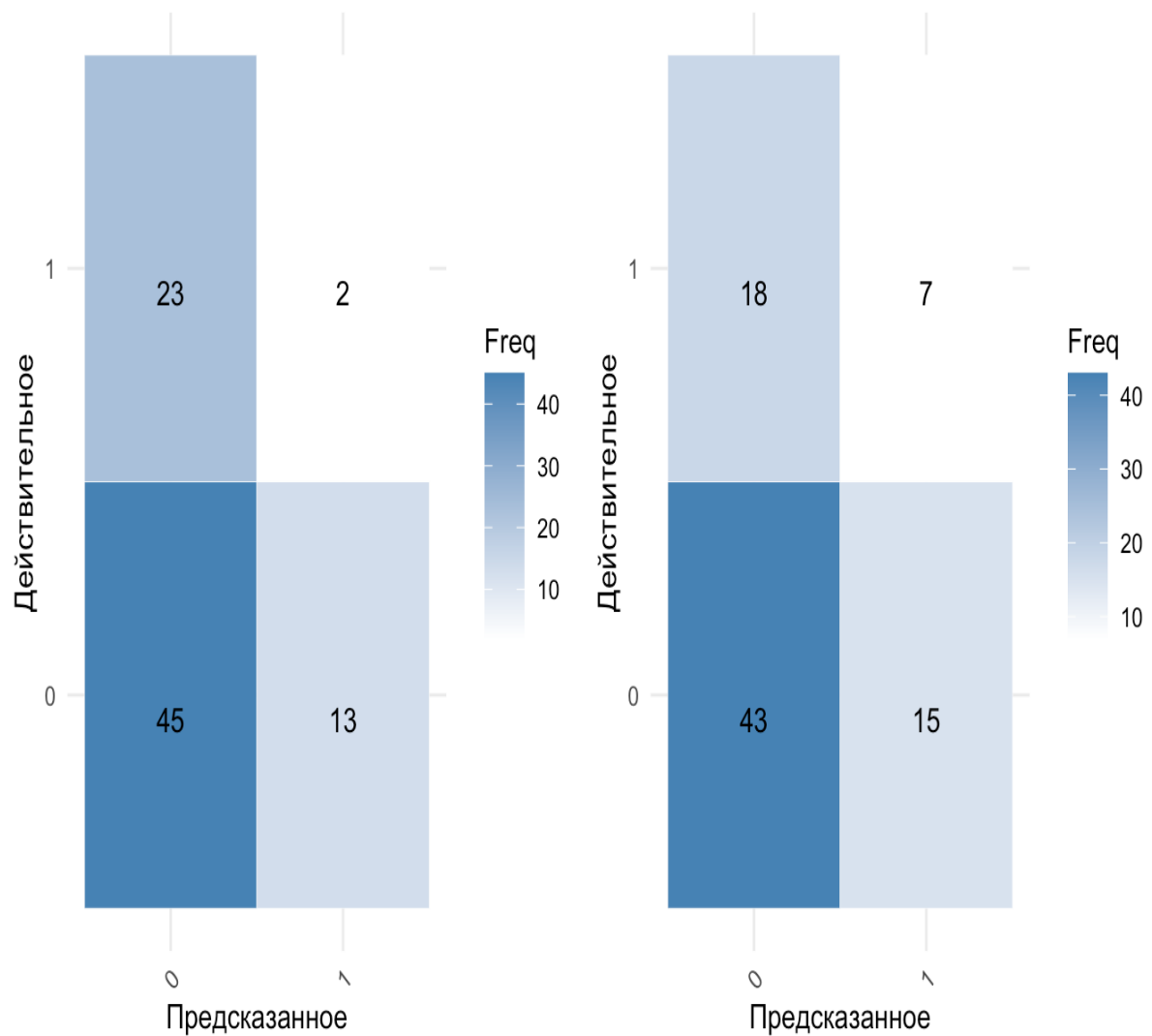
```
p2 <- plot_confusion_matrix(test_data$end, y_pred_standard, "Модель дерева
стандартное")
```

```
## Confusion Matrix and Statistics
##
##      Reference
## Prediction  0  1
##      0 43 18
##      1 15  7
##
##      Accuracy : 0.6024
##      95% CI : (0.489, 0.7083)
##      No Information Rate : 0.6988
##      P-Value [Acc > NIR] : 0.9769
##
##      Kappa : 0.0221
##
##      McNemar's Test P-Value : 0.7277
##
##      Sensitivity : 0.7414
##      Specificity : 0.2800
```

```
##          Pos Pred Value : 0.7049
##          Neg Pred Value : 0.3182
##          Prevalence     : 0.6988
##          Detection Rate  : 0.5181
##          Detection Prevalence : 0.7349
##          Balanced Accuracy : 0.5107
##
##          'Positive' Class : 0
##
```

```
# Объединение графиков в один ряд
grid.arrange(p1, p2, ncol = 2)
```

ь дерево с информационным выигрышем Модель дерево стандартное

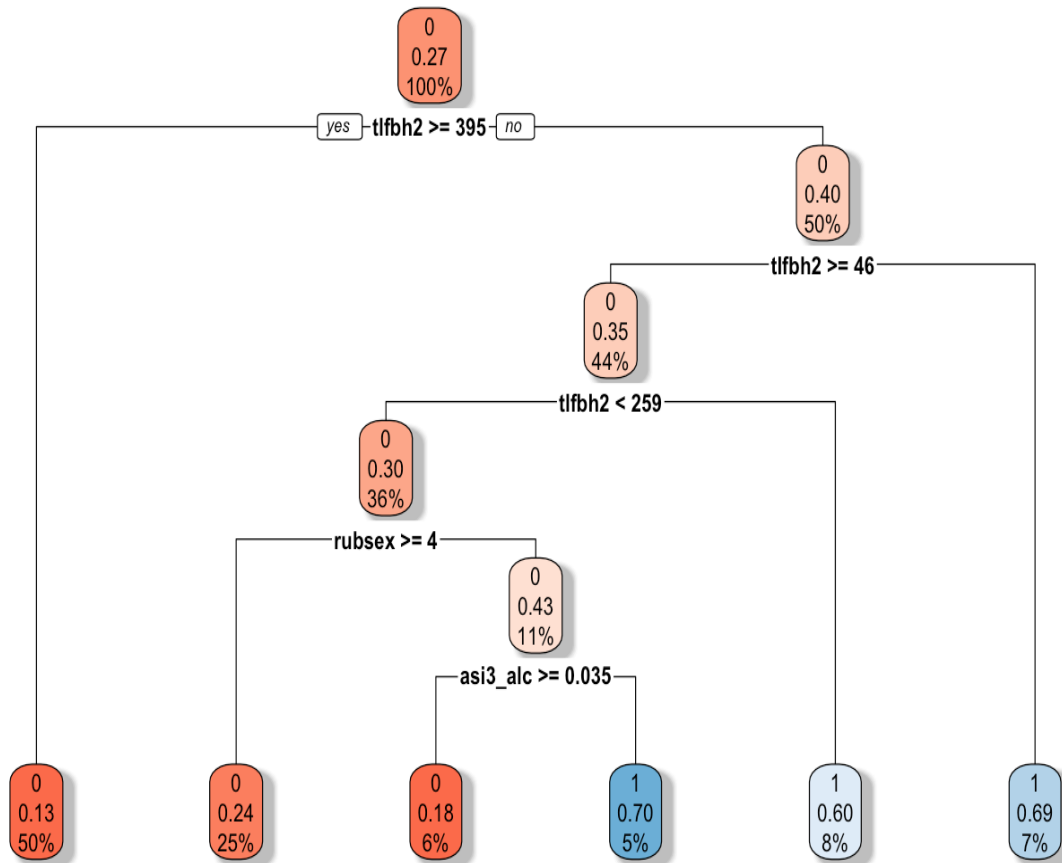


```
library(rpart.plot)
# Визуализация дерева
```



```
rpart.plot(tree_gain, main="Модель дерево с информационным выигрышем", box.palette="RdBu", shadow.col="gray", border.col="black")
```

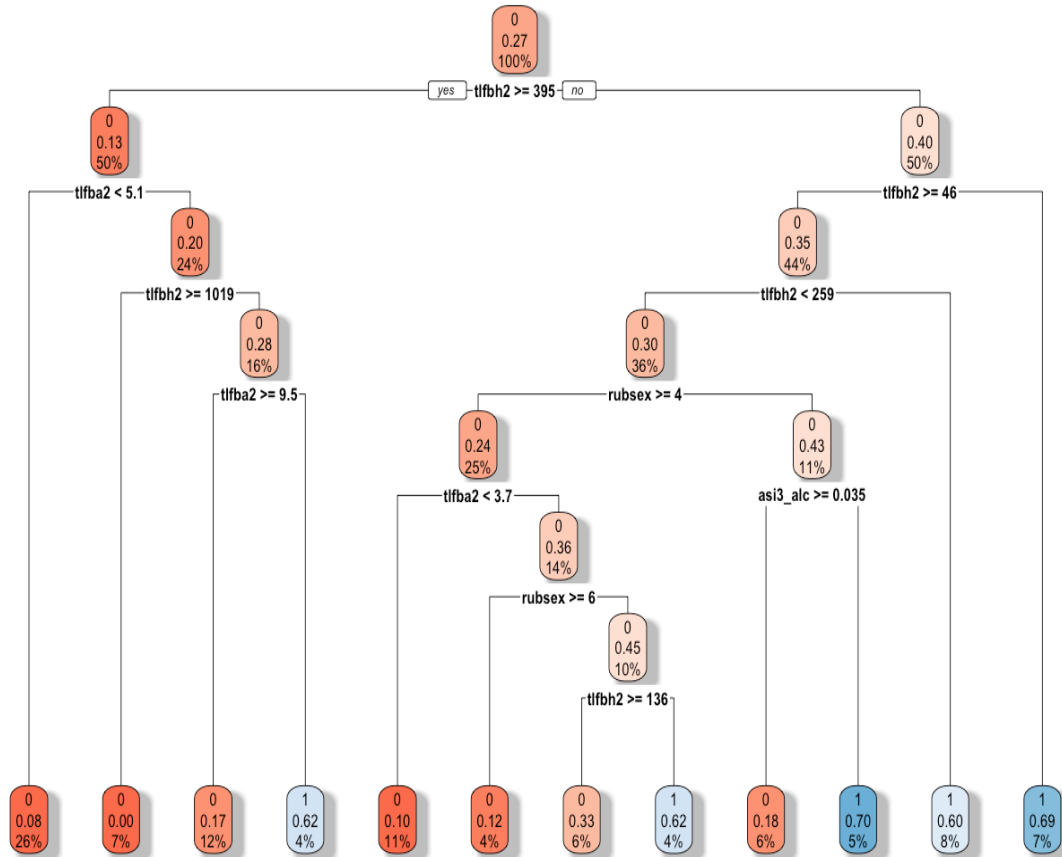
Модель дерево с информационным выигрышем



Визуализация дерева

```
rpart.plot(tree_standard, main="Модель дерево стандартное", box.palette="RdBu", shadow.col="gray", border.col="black")
```

Модель дерево стандартное



Из результатов видно, что точность стандартной модели дерева решений, использующей индекс Джини (0.602), выше, чем модели, использующей критерий информационного выигрыша (0.566).

Исходя из вывода матрицы ошибок и статистики, видно, что модель дерева решений с критерием информационного выигрыша показывает довольно низкую общую точность (Accuracy) 56.63% и негативное значение Каппа (Карра), что указывает на слабое согласие между предсказаниями и фактическими значениями.

Результаты для стандартной модели дерева решений, использующей критерий Джини, также показывают умеренную точность (Accuracy) 60.24%.