Лабораторная работа 2. Решение задач классификации методами машинного обучения.

Сокуров Р.Е.

25.10.2024

Задание

Ход работы

1. Выполнение расчётов по статье.

1.1 Получение и разделение данных

Для начала, получили данные из датасета `iris` и разделили данные на обучающую и тестовую выборки в соотношении 80/20:

```
set.seed(111) # установка зерна случайной генерации
# Получение данных
data("iris") # импорт датасета
str(iris) # вывод структуры датасета
```

```
## 'data.frame':    150 obs. of  5 variables:
##  $ Sepal.Length: num  5.1 4.9 4.7 4.6 5 5.4 4.6 5 4.4 4.9 ...
##  $ Sepal.Width : num  3.5 3 3.2 3.1 3.6 3.9 3.4 3.4 2.9 3.1 ...
##  $ Petal.Length: num  1.4 1.4 1.3 1.5 1.4 1.7 1.4 1.5 1.4 1.5 ...
##  $ Petal.Width : num  0.2 0.2 0.2 0.2 0.2 0.4 0.3 0.2 0.2 0.1 ...
##  $ Species     : Factor w/ 3 levels "setosa","versicolor",..: 1 1 1 1 1 1 1 1 1 1 ...
```

```
summary(iris) # статистика всех переменных данных
```

```
##   Sepal.Length    Sepal.Width     Petal.Length    Petal.Width
##  Min.   :4.300   Min.   :2.000   Min.   :1.000   Min.   :0.100
##  1st Qu.:5.100   1st Qu.:2.800   1st Qu.:1.600   1st Qu.:0.300
##  Median :5.800   Median :3.000   Median :4.350   Median :1.300
##  Mean   :5.843   Mean   :3.057   Mean   :3.758   Mean   :1.199
##  3rd Qu.:6.400   3rd Qu.:3.300   3rd Qu.:5.100   3rd Qu.:1.800
##  Max.   :7.900   Max.   :4.400   Max.   :6.900   Max.   :2.500
##        Species
##  setosa    :50
##  versicolor:50
##  virginica :50
##
##
##
```

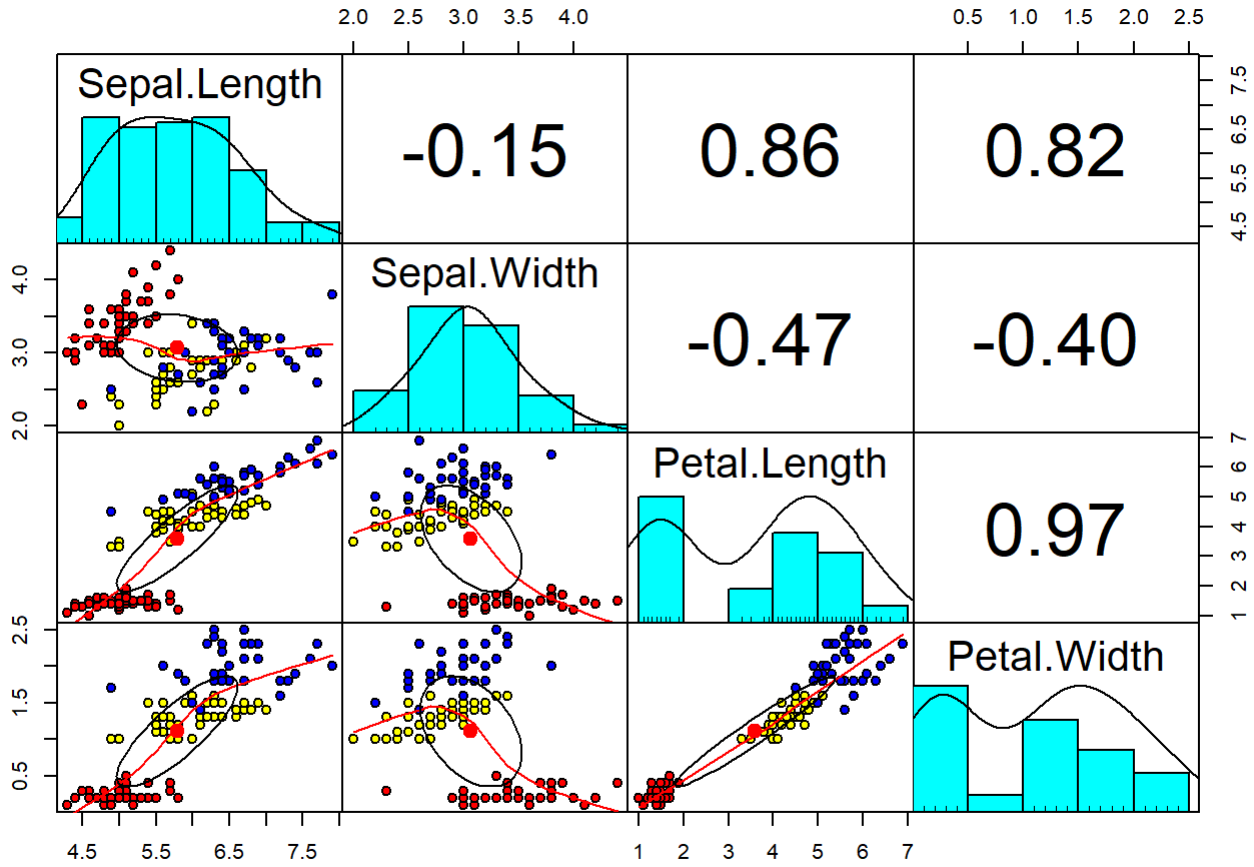
```
ind <- sample(2, nrow(iris), # делим датасет на две части по строкам
              replace = TRUE,
              prob = c(0.8, 0.2)) # делим части в отношении 80/20%
training <- iris[ind==1,] # 80% датасета в обучение
testing <- iris[ind==2,] # 20% в тест
```

1.2 Анализ датасета

Затем выполнили построение корелляционных матриц и матриц парных диаграмм (scatter plots):

```
library(psych) # подключаем библиотеку, которая содержит
               # функции для анализа данных

pairs.panels(training[,-5], # строим scatter plots для всех
                            # переменных в датасете
                            # кроме пятого столбца (species)
             gap = 0,       # промежуток между диаграммами равен 0
             bg = c("red", "yellow", "blue")[training$Species],
             # задаём разные цвета точек для каждого вида цветка
             # setosa - красный, versicolor - жёлтый, virginica - синий
             pch=21) # стиль маркера: круглый, с заливкой фона
```



Здесь на основной диагонали располагаются переменные датасета (и их распределение), выше от главной диагонали — корреляции переменных между собой, а ниже — диаграммы рассеивания.

Длина и ширина лепестка обладают высокой корреляцией. Это означает, что чем длиньше лепесток, тем он шире. Помимо этого, высоким значением корреляции обладают длины чашелистиков и лепестков, а также длина чашелистика и ширина лепестка.

Красные линии на диаграммах рассеивания — это кривые тренда (LOESS). Они отражают как переменные связаны друг с другом, а также характер этой зависимости. Например, эти кривые могут быть не прямыми, что и отражает нелинейные зависимости между признаками.

Доверительные эллипсы чёрного цвета (эллипсы разброса/ковариации) охватывают область, где находится большая часть данных для каждого вида цветков. ТаК, если эллипс вытянут вдоль диагонали, это указывает на сильную положительную или отрицательную корреляцию между переменным, а если эллипс более округлый, это означает, что корреляция между переменными слабая или отсутствует. Также, чем больше эллипс, тем больше вариация данных для данных признаков.

Как видно, в данном датасете несколько независимых переменных обладают мультиколлинеарностью. Это может привести к разным негативным последствиям, например:

1. Когда независимые переменные обладают высокой коллинеарностью, модель может неправильно оценить влияние каждого признака на результат;
2. Коэффициенты становятся менее значимыми в статистическом смысле, даже если на самом деле по отдельности эти переменные невероятно важны;
3. Модель может переобучитья и т.п.

Существует много разных способов борьбы с мультиколлинеарностью, например удалить одну из независимых переменных, использовать методы регуляризации и т.п.

В ходе данной работы был использован метод главных компонент (PCA - Principal Component Analysis) для преобразования корелированных переменных в новый набор компонент.

1.3 Метод главных компонент (PCA - Principal Component Analysis)

Поскольку метод главных компонент базируется на независимых переменных, был убран 5й столбец из датасета:

```
pc <- prcomp(training[,-5], # берём все независимые признаки
             center = TRUE, # центрируем данные
             scale. = TRUE) # масштабируем признаки
attributes(pc) # получаем атрибуты
```

```
## $names
## [1] "sdev"     "rotation" "center"   "scale"    "x"
##
## $class
## [1] "prcomp"
```

Далее был выполнен анализ результатов метода:

```
print(pc)
```

```
## Standard deviations (1, .., p=4):
## [1] 1.7173318 0.9403519 0.3843232 0.1371332
##
## Rotation (n x k) = (4 x 4):
##                     PC1         PC2        PC3        PC4
## Sepal.Length  0.5147163 -0.39817685  0.7242679  0.2279438
## Sepal.Width  -0.2926048 -0.91328503 -0.2557463 -0.1220110
## Petal.Length  0.5772530 -0.02932037 -0.1755427 -0.7969342
## Petal.Width   0.5623421 -0.08065952 -0.6158040  0.5459403
```

Здесь стандартное отклонение показывает, какое значение дисперсии данных объясняется каждой компонентой. PC1 имеет значение 1.72 — объсняет наибольшую долю дисперсии, PC4 0.14 объясняет наименьшую долю.

Затем отображается матрица нагрузок (rotation), которая показывает, как исходные признаки трансформируются в главные компоненты. Каждое значение здесь является коэффициентом, характеризующим вклад исходного признака в компоненту.

Далее был сформирован отчёт о важности каждой компоненты:

```
summary(pc)
```

```
## Importance of components:
##                           PC1    PC2     PC3    PC4
## Standard deviation     1.7173 0.9404 0.38432 0.1371
## Proportion of Variance 0.7373 0.2211 0.03693 0.0047
## Cumulative Proportion  0.7373 0.9584 0.99530 1.0000
```

Про стандартное отклонение уже было написано, а вот Proportion of Variance (Доля дисперсии) говорит о том, что 73.7% общей дисперсии объясняется PC1, ещё 22.1% — PC2. Таким образом лишь двумя компонентами объясняется уже 95,8% разброса данных.

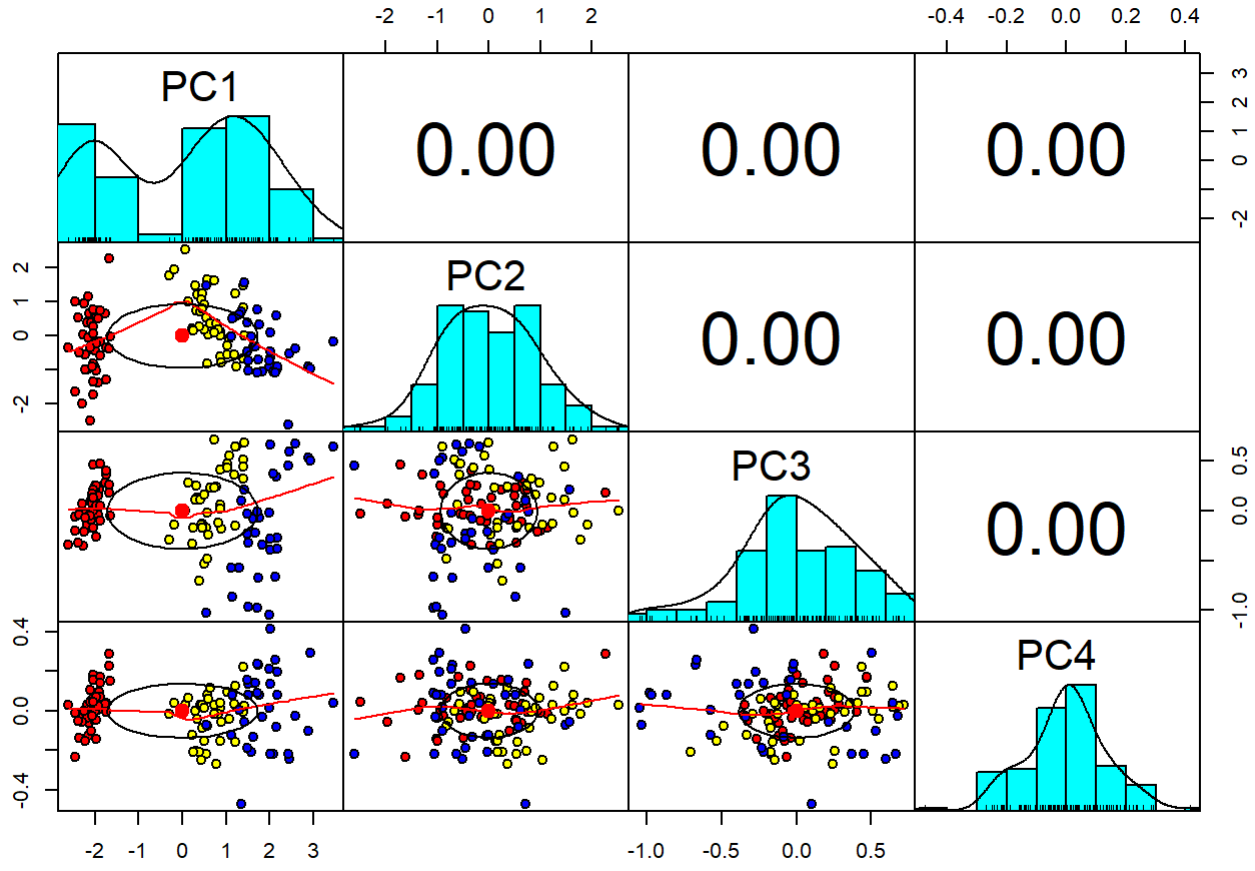
Cumulative Proportion (Накопленная доля) показывает, что все четыре компоненты кумулятивно (вместе) объясняют 100% разброса данных.

Далее было рассмотрено, исчезла ли проблема мультиколлинеарности или нет.

1.4 Влияние метода главных компонент на мультиколлинеарность

Вновь были построены корелляционные матрицы и матрицы парных диаграмм (scatter plots):

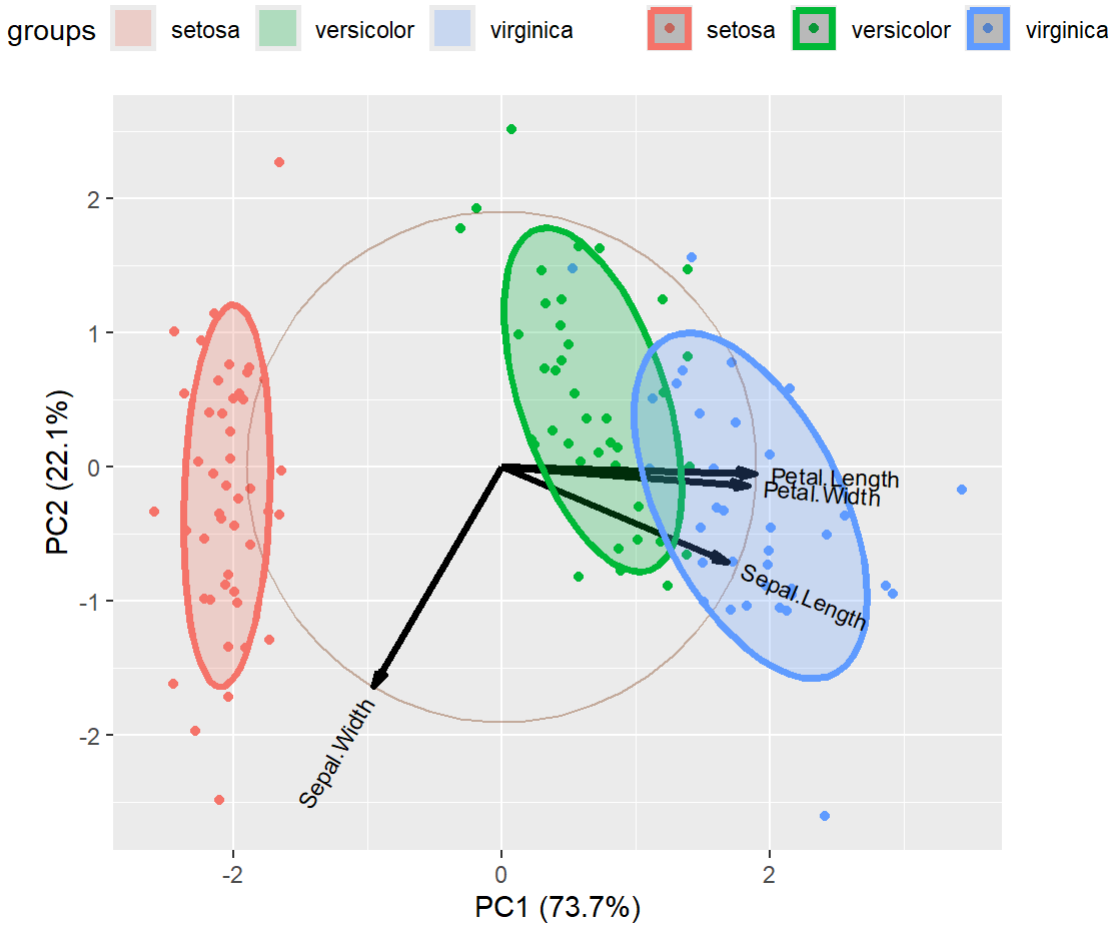
```
pairs.panels(pc$x,
             gap=0,
             bg = c("red", "yellow", "blue")[training$Species],
             pch=21)
```



Как видно из данного графика, проблема мультиколлинеарности полностью ушла.

1.5 Построение Bi-Plot

```
g <- ggbiplot(pc, # создаём график, передаём туда PC
              obs.scale = 1, # размер наблюдений равен 1
              var.scale = 1, # размер переменных равен 1
              groups = training$Species, # точки на графике будут раскрашены в зависимости от
группы (вида цветка)
              ellipse = TRUE, # доверительные эллипсы
              circle = TRUE, # добавляет окружность радиусом 1 для точек
              ellipse.prob = 0.68) # уровень вероятности для эллипсов
g <- g + scale_color_discrete(name = '') # настройка цветовой шкалы без легенды
g <- g + theme(legend.direction = 'horizontal',
               legend.position = 'top') # корректировка отображения легенды
print(g)
```



Здесь по оси X расположили PC1, которая объясняет 73,7% дисперсии, а по Y — PC2, которая объсняет 22,1%. Эллипсы отображают области вероятного нахождения точек (68% всех точек определенного класса). Стрелки - переменные исходного набора данных, длина стрелки показывает важность этой переменной для компоненты, а угол стрелки относительно двух осей отображает вклад переменной в каждую из компонент (можно проекции построить). Видно, что кластер setosa чётко отделён от остальных, в то время как versicolor и virginica являются схожими, при использовании данных параметров.

1.6 Предсказание с помощью многономиальной

логистической регрессии

```
trg <- predict(pc, training) # преобразование обучающих данных с помощью PCA
trg <- data.frame(trg, training[5]) # формируем дата-фрейм с ответами (5й столбец)
tst <- predict(pc, testing) # преобразование также тестовой выборки
tst <- data.frame(tst, testing[5]) # формируем дата-фрейм с ответами (5й столбец)
```

Для предсказаний была использована модель многономиальной логистической регрессии, поскольку зависимая переменная имеет более двух категорий (три вида цветка). В качестве обучающих данных были использованы две первые компоненты, потому что именно они объясняют 95% разброса данных.

```
library(nnet)
trg$Species <- relevel(trg$Species, ref = "setosa")
mymodel <- multinom(Species~PC1+PC2, data = trg)
```

```
## # weights:  12 (6 variable)
## initial  value 131.833475
## iter  10 value 20.607042
## iter  20 value 18.331120
## iter  30 value 18.204474
## iter  40 value 18.199783
## iter  50 value 18.199009
## iter  60 value 18.198506
## final  value 18.198269
## converged
```

```
summary(mymodel)
```

```
## Call:
## multinom(formula = Species ~ PC1 + PC2, data = trg)
##
## Coefficients:
##            (Intercept)      PC1      PC2
## versicolor   7.2345029 14.05161 3.167254
## virginica   -0.5757544 20.12094 3.625377
##
## Std. Errors:
##            (Intercept)      PC1      PC2
## versicolor    187.5986 106.3766 127.8815
## virginica     187.6093 106.3872 127.8829
##
## Residual Deviance: 36.39654
## AIC: 48.39654
```

1.7 Матрица ошибок (confusion matrix) и матрица неправильной классификации (misclassification error)

На обучающей выборке:

```
p <- predict(mymodel, trg)
tab <- table(p, trg$Species)
tab
```

```
##
## p            setosa versicolor virginica
##   setosa         45          0         0
##   versicolor      0         35         3
##   virginica       0          5        32
```

```
1 - sum(diag(tab))/sum(tab)
```

```
## [1] 0.06666667
```

Ошибка неправильной классификации 6.7%.

На тестовой выборке:

```
p1 <- predict(mymodel, tst)
tab1 <- table(p1, tst$Species)
tab1
```

```
##
## p1           setosa versicolor virginica
##   setosa          5          0         0
##   versicolor      0          9         3
##   virginica       0          1        12
```

```
1 - sum(diag(tab1))/sum(tab1)
```

```
## [1] 0.1333333
```

Ошибка неправильной классификации 13%.

2. Выполнение расчётов по выбранной теме.

2.1 Получение и разделение данных

Для самостоятельной части лабораторной работы был выбран датасет Human Activity Recognition Using Smartphones https://archive.ics.uci.edu/dataset/240/human+activity+recognition+using+smartphones (https://archive.ics.uci.edu/dataset/240/human+activity+recognition+using+smartphones). Данный датасет содержит информацию с различных датчиков смартфона (561 разных параметров), а задача состоит в следующем: необходимо корректно определить тип активности, которой занимался человек по информации с этих датчиков. Доступны 6 разных типов активности:

1. Ходьба;
2. Подъём по лестнице;
3. Спуск по лестнице;
4. Сидение;
5. Стояние;

6. Лежание.

Данные были предворительно предобработаны с помощью языка Python, а именно были собраны вместе названия столбцов, сами данные, ответы к ним (тип активности) и сохранены в файл data.csv. Импортируем его и разделим выборку в процентном соотношении 80/20:

```
data <- read.csv("data.csv")
ind <- sample(2, nrow(data), # делим датасет на две части по строкам
              replace = TRUE,
              prob = c(0.8, 0.2)) # делим части в отношении 80/20%
training <- data[ind==1,] # 80% датасета в обучение
testing <- data[ind==2,] # 20% в тест
```

2.2 Анализ датасета

Поскольку количество данных слишком велико (561 признак), то было принято решение вместо отрисовки графика создать корелляционную матрицу и сохранить её в файл excel. Он доступен в репозитории рядом с данным отчётом и называется `correlation_matrix.xlsx` (https://clck.ru/3EBWcL (https://clck.ru/3EBWcL)).

```
library(corrplot)
```

```
## corrplot 0.95 loaded
```

```
cor_matrix <- cor(data)  # Формируем корреляционную матрицу

cor_df <- as.data.frame(cor_matrix) # сохраняем её как дата-фрейм

# Округляем значения для удобства
cor_df <- round(cor_df, 2)

# Добавляем столбец с названиями переменных
cor_df <- cbind(Variable = rownames(cor_df), cor_df)

library(writexl)
# Сохраняем корреляционную матрицу в файл Excel
write_xlsx(cor_df, "correlation_matrix.xlsx")
```

В этом датасете множество переменных обладают высокой коллинеарностью, а также его размерность слишком велика (561 признаков). Для борьбы с обоими проблемами был использован метод главных компонент.

2.3 Метод главных компонент

Поскольку данный метод базируется на независимых переменных, был убран 562й столбец из датасета и сформирован отчёт о важности каждой компоненты (вывод ограничен до 17 строк):

```
pc <- prcomp(training[,-562], # берём все независимые признаки
             center = TRUE, # центрируем данные
             scale. = TRUE) # масштабируем признаки
summary(pc)
```

```
## Importance of components:
##                             PC1     PC2     PC3     PC4     PC5     PC6     PC7
## Standard deviation     16.8842 6.06946 3.94387 3.75022 3.25755 3.11270 2.79005
## Proportion of Variance  0.5082 0.06567 0.02773 0.02507 0.01892 0.01727 0.01388
## Cumulative Proportion   0.5082 0.57383 0.60155 0.62662 0.64554 0.66281 0.67668
##                             PC8     PC9    PC10    PC11    PC12    PC13    PC14
## Standard deviation     2.59948 2.37095 2.32875 2.21399 2.12562 2.04679 1.90222
## Proportion of Variance 0.01205 0.01002 0.00967 0.00874 0.00805 0.00747 0.00645
## Cumulative Proportion  0.68873 0.69875 0.70841 0.71715 0.72521 0.73267 0.73912
##                            PC15    PC16    PC17    PC18    PC19    PC20    PC21
## Standard deviation     1.88698 1.84596 1.82771 1.80612 1.78398 1.71233 1.67879
## Proportion of Variance 0.00635 0.00607 0.00595 0.00581 0.00567 0.00523 0.00502
## Cumulative Proportion  0.74547 0.75155 0.75750 0.76331 0.76899 0.77421 0.77924
##                            PC22    PC23    PC24    PC25    PC26    PC27    PC28
## Standard deviation     1.64935 1.64571 1.61328 1.58427 1.54383 1.53318 1.49907
## Proportion of Variance 0.00485 0.00483 0.00464 0.00447 0.00425 0.00419 0.00401
## Cumulative Proportion  0.78409 0.78891 0.79355 0.79803 0.80228 0.80647 0.81047
```

Всего главных компонент больше 560, но уже после 28й компоненты они начинают описывать менее 1.5% разброса данных. Поэтому для дальнейшего анализа использовалось 28 главных компонент.

2.4 Влияние метода главных компонент на мультиколлинеарность

Для проверки влияния метода на мультиколлинеарность, была составлена матрица корреляции для 28 компонент:

```
pc_data <- pc$x[, 1:28]  # Используем только первые 28 главных компонент

# Вычисляем корреляцию для главных компонент
cor_matrix <- cor(pc_data)

# Преобразуем в дата-фрейм и округляем значения
cor_df <- as.data.frame(round(cor_matrix, 2))

# Добавляем столбец с названиями переменных
cor_df <- cbind(Variable = rownames(cor_df), cor_df)

# Сохраняем корреляционную матрицу в файл Excel
write_xlsx(cor_df, "correlation_matrix_PC.xlsx")
```

Файл `correlation_matrix_PC.xlsx` доступен в репозитории рядом с данным отчётом (https://clck.ru/3EBWcL (https://clck.ru/3EBWcL)). Как видно из этого файла, теперь данные полностью независимы друг от друга.

2.5 Предсказание с помощью многономиальной

логистической регрессии

```
# Преобразуем обучающие данные с помощью PCA
trg <- predict(pc, training)

# Формируем дата-фрейм с ответами (5й столбец)
trg <- data.frame(trg[, 1:28], Activity = training[, 562]) # Используем только первые 28 PC

# Преобразование тестовой выборки
tst <- predict(pc, testing)
tst <- data.frame(tst[, 1:28], Activity = testing[, 562]) # Используем только первые 28 PC
```

Для предсказаний была использована модель многономиальной логистической регрессии, поскольку зависимая переменная имеет более двух категорий (6 видов активностей).

```
# Преобразуем переменную Activity в фактор
trg$Activity <- factor(trg$Activity)

# Устанавливаем референтное значение для целевой переменной
trg$Activity <- relevel(trg$Activity, ref = "1")

# Обучаем модель с использованием 28 главных компонент
mymodel <- multinom(Activity ~ ., data = trg) # Используем . для включения всех 75 PC
```

```
## # weights:  180 (145 variable)
## initial  value 10566.005590
## iter  10 value 2471.748809
## iter  20 value 2046.531064
## iter  30 value 1913.789361
## iter  40 value 1495.932453
## iter  50 value 1397.043101
## iter  60 value 1305.411145
## iter  70 value 1241.530726
## iter  80 value 1158.032320
## iter  90 value 1101.436836
## iter 100 value 1067.795454
## final  value 1067.795454
## stopped after 100 iterations
```

```
summary(mymodel)
```

```
## Call:
## multinom(formula = Activity ~ ., data = trg)
##
## Coefficients:
##   (Intercept)         PC1          PC2       PC3        PC4        PC5
## 2   1.3001658 -0.07676327 -0.540470830 0.9080627 -1.2096947 -0.1502676
## 3   1.6887758 -0.02864529  0.108480101 0.9396475 -1.6130782  0.1224742
## 4   1.9391743 -0.43741044 -0.085291041 1.0893810 -0.6004321  0.3250639
## 5   1.3532220 -0.25281693 -0.065700788 0.4515091 -0.3220754  1.2369001
## 6  -0.6130641 -0.19186849 -0.009192966 1.5931449 -2.1001180 -1.5652263
##            PC6       PC7       PC8       PC9        PC10        PC11
## 2 -0.617523174 0.4150115 0.1352794 0.4590591  0.50533943 -0.22655747
## 3  0.002003931 0.5157829 1.2473496 0.5685181  0.43893724  0.36981565
## 4  0.203514087 0.4819605 0.2525663 0.5462715 -0.26506461 -0.39621030
## 5  0.354292511 0.6809297 0.2322408 0.2436992  0.09778605  0.30314897
## 6  0.186803381 0.1563744 0.5007880 0.4119963 -0.15245022 -0.07048086
##          PC12       PC13        PC14        PC15        PC16        PC17
## 2 -0.28102144 -0.3172902  0.34104430 -0.56963218  0.31898098 -0.06898852
## 3  0.30375914 -0.3442236 -0.06677495 -0.28742207  0.26109285  0.34642458
## 4 -0.15288981 -0.4200848 -0.35316287  0.03009359  0.02196201  0.15571530
## 5 -0.01515624 -0.2670336  0.07707372  0.06729494 -0.03150335  0.41664391
## 6  0.12629725  0.3043689  0.47746296 -0.77639193  0.36166017 -0.06885556
##          PC18       PC19      PC20          PC21       PC22         PC23
## 2 -0.12269370  0.1882738 0.2811009  0.0009670393 -0.4715532  0.507320457
## 3  0.09717990 -0.1855089 0.0984283  0.0574645166  0.2161117  0.353766977
## 4  0.12956898  0.2475004 0.3607104 -0.0192694044 -0.1108453 -0.008790292
## 5 -0.28521410 -0.3167336 0.3664951 -0.0563231644 -0.2542485  0.397652198
## 6 -0.09049415 -0.0833214 0.4686978 -0.0943649353 -0.2199688  1.076122617
##         PC24        PC25       PC26       PC27         PC28
## 2 0.05315114 -0.58202926  0.1772297 -0.2012822 -0.007962881
## 3 0.16321270 -0.03112979  0.4847109  0.1541501  0.249087960
## 4 0.16278670  0.01778386  0.6840011 -0.2258149  0.077900249
## 5 0.08526852 -0.13083921  0.1789987  0.1486336 -0.183282624
## 6 0.19054564 -0.88155270 -0.2775452 -0.7505222 -0.271213208
##
## Std. Errors:
##   (Intercept)        PC1        PC2       PC3       PC4       PC5        PC6
## 2   0.7627428 0.03846055 0.05950342 0.1205390 0.1412483 0.1686358 0.08211424
## 3   0.6747944 0.03257494 0.05186926 0.1267528 0.1515391 0.1766428 0.07738058
## 4   0.7636262 0.04609850 0.06082745 0.1464282 0.1777684 0.1748604 0.10878297
## 5   0.6571391 0.03374293 0.05469408 0.1325636 0.1616883 0.1759679 0.11893435
## 6   0.8587360 0.03635486 0.06368226 0.1551733 0.2024622 0.2523693 0.11585853
##          PC7        PC8       PC9       PC10       PC11       PC12       PC13
## 2 0.09397944 0.09164763 0.1027630 0.09639804 0.06843557 0.07749632 0.07299259
## 3 0.09904811 0.11446742 0.1066550 0.10292679 0.07289450 0.08895443 0.08159848
## 4 0.12283151 0.14116467 0.1427831 0.14163788 0.13903156 0.12135882 0.13483458
## 5 0.12274787 0.12718635 0.1359644 0.14127339 0.13981886 0.12198861 0.13918720
## 6 0.13458303 0.15053276 0.1276404 0.14125497 0.13853590 0.13962839 0.14296362
##         PC14       PC15       PC16       PC17       PC18       PC19       PC20
## 2 0.08827426 0.09830727 0.08025382 0.08322012 0.09538874 0.08030377 0.08362073
## 3 0.09657756 0.08965179 0.09699431 0.08588964 0.10870331 0.08901199 0.09371759
## 4 0.12338288 0.14003668 0.13138841 0.13449697 0.13462927 0.13829955 0.13921198
## 5 0.12569107 0.13945011 0.13531805 0.13164777 0.13377145 0.14143883 0.13617376
## 6 0.15165968 0.16603070 0.15377331 0.15921792 0.15001155 0.14989836 0.16023492
##         PC21       PC22       PC23       PC24       PC25       PC26       PC27
```

```
## 2 0.08471228 0.1021972 0.1020323 0.09101921 0.1218243 0.1087665 0.09914151
## 3 0.09731751 0.1063003 0.1082610 0.09326457 0.1313259 0.1178082 0.11057636
## 4 0.13416342 0.1431097 0.1473721 0.17060616 0.1665122 0.1632715 0.13664519
## 5 0.13480497 0.1465059 0.1427418 0.16912971 0.1705127 0.1649637 0.14108491
## 6 0.15891214 0.1589670 0.1695466 0.18253507 0.1901845 0.1945459 0.16434176
##        PC28
## 2 0.1119907
## 3 0.1280676
## 4 0.1569192
## 5 0.1621404
## 6 0.1805704
##
## Residual Deviance: 2135.591
## AIC: 2425.591
```

2.6 Матрица ошибок (confusion matrix) и матрица неправильной классификации (misclassification error)

На обучающей выборке:

```
p <- predict(mymodel, trg)
tab <- table(p, trg$Activity)
tab
```

```
##
## p      1    2    3    4    5    6
##   1  970   21   15    0    0    0
##   2   14  822   27    0    1    0
##   3   12   18  753    0    0    0
##   4    0    0    0  831  130    6
##   5    1    0    0  184  978    0
##   6    0    1    0   10    0 1103
```

```
1 - sum(diag(tab))/sum(tab)
```

```
## [1] 0.07461421
```

Ошибка неправильной классификации 7.5%.

На тестовой выборке:

```
p1 <- predict(mymodel, tst)
tab1 <- table(p1, tst$Activity)
tab1
```

```
##
## p1    1   2   3   4   5   6
##   1 221   3   6   0   0   0
##   2   7 204   9   1   0   0
##   3   1   4 176   0   0   0
##   4   0   0   0 209  35   7
##   5   0   0   0  49 230   0
##   6   0   0   0   2   0 291
```

```
1 - sum(diag(tab1))/sum(tab1)
```

```
## [1] 0.08522337
```

Ошибка неправильной классификации 8.5%.

Вывод

В ходе данной лабораторной работы был исследован и использован метод главных компонент для исправления мультиколлинеарности независимых переменных, а также, во второй части работы, для уменьшения размерности датасета. Далее на основе получившихся независимых компонент была составлена модель многономиальной логистической регрессии для классификации данных на несколько классов. В самостоятельной части работы ошибка неправильной классификации на тестовом наборе составила 8.5%, что считается хорошим результатом.