БЕЛОРУССКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ
Факультет радиофизики и компьютерных технологий
Кафедра радиофизики и цифровых медиа технологий

Лабораторная работа по курсу
Статистическая радиофизика

Введение в машинное обучение. Решение задачи классификации

Минск 2023 г.

Цель работы: Изучить основы машинного обучения, а также основные алгоритмы классификации данных.

Общие сведения:

Машинное обучение (англ. machine learning, ML) – класс методов искусственного интеллекта, характерной чертой которых является не прямое решение задачи, а обучение в процессе применения решений множества сходных задач. Для построения таких методов используются средства математической статистики, численных методов, методов оптимизации, теории вероятностей, теории графов, различные техники работы с данными в цифровой форме.

Различают два типа обучения:

1. Обучение по прецедентам, или индуктивное обучение, основано на выявлении эмпирических закономерностей в данных.

2. Дедуктивное обучение предполагает формализацию знаний экспертов и их перенос в компьютер в виде базы знаний.

Дедуктивное обучение принято относить к области экспертных систем, поэтому термины машинное обучение и обучение по прецедентам можно считать синонимами.

Многие методы индуктивного обучения разрабатывались как альтернатива классическим статистическим подходам. Многие методы тесно связаны с извлечением информации (англ. information extraction), интеллектуальным анализом данных (data mining).

Раздел машинного обучения, с одной стороны, образовался в результате разделения науки о нейросетях на методы обучения сетей и виды топологий их архитектуры, с другой стороны – вобрал в себя методы математической статистики. Указанные ниже способы машинного обучения исходят из случая использования нейросетей, хотя существуют и другие методы, использующие понятие обучающей выборки — например, дискриминантный анализ, оперирующий обобщённой дисперсией и ковариацией наблюдаемой статистики, или байесовские классификаторы. Базовые виды нейросетей, такие как перцептрон и многослойный перцептрон (а также их модификации), могут обучаться как с учителем, так и без учителя, с подкреплением и самоорганизацией. Но некоторые нейросети и большинство статистических методов можно отнести только к одному из способов обучения. Поэтому, если нужно классифицировать методы машинного обучения в зависимости от способа обучения, будет некорректным относить нейросети к определенному виду, правильнее было бы типизировать алгоритмы обучения нейронных сетей.

- Обучение с учителем – для каждого прецедента задаётся пара «ситуация, требуемое решение»:
  - Искусственная нейронная сеть
  - Глубокое обучение
  - Метод коррекции ошибки
  - Метод обратного распространения ошибки
  - Метод опорных векторов
- Обучение без учителя – для каждого прецедента задаётся только «ситуация», требуется сгруппировать объекты в кластеры, используя данные о попарном сходстве объектов, и/или понизить размерность данных:
  - Альфа-система подкрепления
  - Гамма-система подкрепления
  - Метод ближайших соседей
- Обучение с подкреплением – для каждого прецедента имеется пара «ситуация, принятое решение»:

    - Генетический алгоритм.
- Активное обучение – отличается тем, что обучаемый алгоритм имеет возможность самостоятельно назначать следующую исследуемую ситуацию, на которой станет известен верный ответ
- Обучение с частичным привлечением учителя (англ. semi-supervised learning) – для части прецедентов задается пара «ситуация, требуемое решение», а для части – только «ситуация»
- Трансдуктивное обучение – обучение с частичным привлечением учителя, когда прогноз предполагается делать только для прецедентов из тестовой выборки
- Многозадачное обучение (англ. multi-task learning) – одновременное обучение группе взаимосвязанных задач, для каждой из которых задаются свои пары «ситуация, требуемое решение»
- Многовариантное обучение (англ. multiple-instance learning) – обучение, когда прецеденты могут быть объединены в группы, в каждой из которых для всех прецедентов имеется «ситуация», но только для одного из них (причем, неизвестно какого) имеется пара «ситуация, требуемое решение»
- Бустинг (англ. boosting – улучшение) – это процедура последовательного построения композиции алгоритмов машинного обучения, когда каждый следующий алгоритм стремится компенсировать недостатки композиции всех предыдущих алгоритмов.
- Байесовская сеть

Классические задачи, решаемые с помощью машинного обучения

- Классификация выполняется с помощью обучения с учителем на этапе собственно обучения.
- Кластеризация выполняется с помощью обучения без учителя
- Регрессия выполняется с помощью обучения с учителем на этапе тестирования, является частным случаем задач прогнозирования
- Понижение размерности данных и их визуализация выполняется с помощью обучения без учителя
- Восстановление плотности распределения вероятности по набору данных
- Одно классовая классификация и выявление новизны
- Построение ранговых зависимостей

Типы входных данных при обучении

- Признаковое описание объектов – наиболее распространённый случай
- Описание взаимоотношений между объектами, чаще всего отношения попарного сходства, выражаемые при помощи матрицы расстояний, ядер либо графа данных
- Временной ряд или сигнал
- Изображение или видеоряд

Типы функционалов качества

- При обучении с учителем – функционал качества может определяться как средняя ошибка ответов. Предполагается, что искомый алгоритм должен его минимизировать. Для предотвращения переобучения в минимизируемый функционал качества часто в явном или неявном виде добавляют регуляризатор.
- При обучении без учителя – функционалы качества могут определяться по-разному, например, как отношение средних межкластерных и внутрикластерных расстояний.
- При обучении с подкреплением – функционалы качества определяются физической средой, показывающей качество приспособления агента.

Целью машинного обучения является частичная или полная автоматизация решения сложных профессиональных задач в самых разных областях человеческой деятельности.

Машинное обучение имеет широкий спектр приложений:

- Распознавание речи
- Распознавание жестов
- Распознавание рукописного ввода
- Распознавание образов
- Техническая диагностика
- Медицинская диагностика
- Прогнозирование временных рядов
- Биоинформатика
- Обнаружение мошенничества
- Обнаружение спама
- Категоризация документов
- Биржевой технический анализ
- Финансовый надзор
- Кредитный скоринг
- Прогнозирование ухода клиентов
- Обучение ранжированию в информационном поиске

Сфера применений машинного обучения постоянно расширяется. Повсеместная информатизация приводит к накоплению огромных объёмов данных в науке, производстве, бизнесе, транспорте, здравоохранении. Возникающие при этом задачи прогнозирования, управления и принятия решений часто сводятся к обучению по прецедентам. Раньше, когда таких данных не было, эти задачи либо вообще не ставились, либо решались совершенно другими методами.

Задача классификации – задача, в которой имеется множество объектов (ситуаций), разделённых некоторым образом на классы. Задано конечное множество объектов, для которых известно, к каким классам они относятся. Это множество называется выборкой. Классовая принадлежность остальных объектов неизвестна. Требуется построить алгоритм, способный классифицировать произвольный объект из исходного множества.

Классифицировать объект – значит, указать номер (или наименование) класса, к которому относится данный объект.

Классификация объекта – номер или наименование класса, выдаваемый алгоритмом классификации в результате его применения к данному конкретному объекту.

В математической статистике задачи классификации называются также задачами дискриминантного анализа. В машинном обучении задача классификации решается, в

частности, с помощью методов искусственных нейронных сетей при постановке эксперимента в виде обучения с учителем.

Существуют также другие способы постановки эксперимента – обучение без учителя, но они используются для решения другой задачи – кластеризации или таксономии. В этих задачах разделение объектов обучающей выборки на классы не задаётся, и требуется классифицировать объекты только на основе их сходства друг с другом. В некоторых прикладных областях, и даже в самой математической статистике, из-за близости задач часто не различают задачи кластеризации от задач классификации.

Некоторые алгоритмы для решения задач классификации комбинируют обучение с учителем с обучением без учителя, например, одна из версий нейронных сетей Кохонена – сети векторного квантования, обучаемые с учителем.

Пусть X – множество описаний объектов, Y – конечное множество номеров (имён, меток) классов.

Существует неизвестная целевая зависимость – отображение $y^*\colon X \to Y$, значения которой известны только на объектах конечной обучающей выборки $X^m = \{(x_1,y_1),\dots,(x_m,y_m)\}$. Требуется построить алгоритм $a\colon X \to Y$, способный классифицировать произвольный объект $x \in X$.

Вероятностная постановка задачи

Более общей считается вероятностная постановка задачи. Предполагается, что множество пар «объект, класс» $X \times Y$ является вероятностным пространством с неизвестной вероятностной мерой $\mathbf{P}$.

Имеется $X^m = \{(x_1,y_1),\dots,(x_m,y_m)\}$ конечная обучающая выборка наблюдений, сгенерированная согласно вероятностной мере $\mathbf{P}$. Требуется построить алгоритм $a\colon X \to Y$, способный классифицировать произвольный объект $x \in X$.

Признаковое пространство

Признаком называется отображение $f\colon X \to D_f$, где D_f — множество допустимых значений признака. Если заданы признаки $f_1,\dots,f_n$, то вектор $x = (f_1(x),\dots,f_n(x))$ называется признаковым описанием объекта $x \in X$.

Признаковые описания допустимо отождествлять с самими объектами. При этом множество $X = D_{f_1} \times \dots \times D_{f_n}$ называют признаковым пространством.

В зависимости от множества D_f признаки делятся на следующие типы:

- бинарный признак: $D_f = \{0,1\}$;
- номинальный признак: D_f – конечное множество;
- порядковый признак: D_f – конечное упорядоченное множество;
- количественный признак: D_f – множество действительных чисел.

Часто встречаются прикладные задачи с разнотипными признаками, для их решения подходят далеко не все методы.

Пример решения задачи классификации: предсказание сорта ириса на основе размеров лепестков и чашелистиков

- Первоначально необходимо загрузить файл с данными в R-Studio

```
iris <- read.csv("Iris.csv")
```

- Просмотрим структуру данных с помощью функций head() или View()

```
head(iris)
```

```
##   Sepal.Length Sepal.Width Petal.Length Petal.Width Species
## 1          5.1         3.5          1.4         0.2  setosa
## 2          4.9         3.0          1.4         0.2  setosa
## 3          4.7         3.2          1.3         0.2  setosa
## 4          4.6         3.1          1.5         0.2  setosa
## 5          5.0         3.6          1.4         0.2  setosa
## 6          5.4         3.9          1.7         0.4  setosa
```

- Загрузим библиотеку для цветной визуализации данных

```
library(RColorBrewer)
```

- Создадим цветную палитру

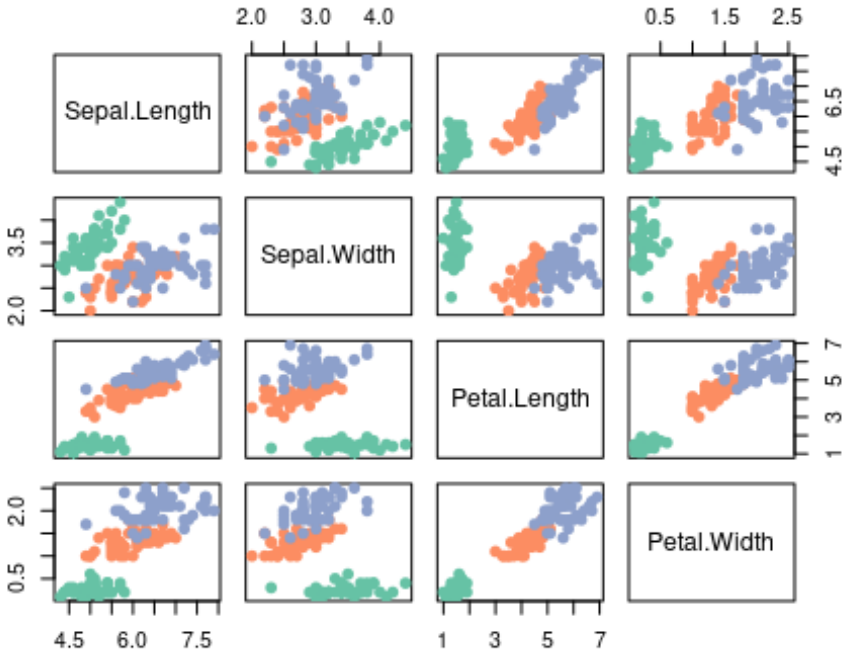
```
palette <- brewer.pal(3, "Set2")
```

- Преобразуем категориальный признак Species в фактор

```
iris$Species <- factor(iris$Species)
```

- Применим цветную палитру и визуализируем матрицу рассеивания

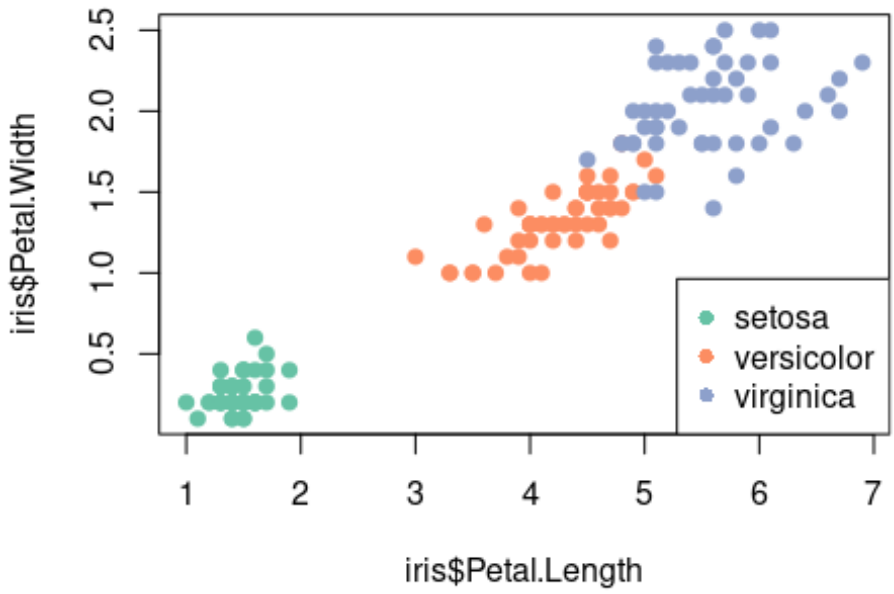
```
plot(
  x = iris[1:4],
  col = palette[as.numeric(iris$Species)],
  pch = 19)
```



- Построим диаграмму рассеивания зависимости длины лепестка от ширины

```
plot(
  x = iris$Petal.Length,
  y = iris$Petal.Width,
  col = palette[as.numeric(iris$Species)],
  pch = 19)
legend(
  x = "bottomright",
  legend = levels(iris$Species),
  col = palette,
```

```
  pch = 16
  )
```



- Сформируем обучающую и тестовую выборки. Для этого установим значение для воспроизведения одинаковой случайной последовательности каждый раз и выберем 100 случайных строк из 150 строк.

```
set.seed(42)
indexes <- sample(
  x = 1:150,
  size = 100)

train <- iris[indexes, ]
test <- iris[-indexes, ]
```

- Загрузим пакет для классификации. Обучим модель методом k-ближайших соседей. Спрогнозируем результаты модели и обобщим полученные результаты

```
library(caret)

## Loading required package: ggplot2

## Loading required package: lattice

knnModel <- knn3(
  formula = Species ~ .,
  data = train,
  k = 3)

knnPredictions <- predict(
  object = knnModel,
  newdata = test,
  type = "class")

table(
  x = knnPredictions,
  y = test$Species)
```

```
##             y
## x            setosa versicolor virginica
##   setosa         13          0         0
##   versicolor      0         17         2
##   virginica       0          0        18
```

- Создадим матрицу ошибок и выведем результаты

```
knnMatrix <- confusionMatrix(
  data = knnPredictions,
  reference = test$Species)
print(knnMatrix)
```

```
## Confusion Matrix and Statistics
##
##             Reference
## Prediction   setosa versicolor virginica
##   setosa         13          0         0
##   versicolor      0         17         2
##   virginica       0          0        18
##
## Overall Statistics
##
##                Accuracy : 0.96
##                  95% CI : (0.8629, 0.9951)
##     No Information Rate : 0.4
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.9393
##
##  Mcnemar's Test P-Value : NA
##
## Statistics by Class:
##
##                      Class: setosa Class: versicolor Class: virginica
## Sensitivity                   1.00            1.0000           0.9000
## Specificity                   1.00            0.9394           1.0000
## Pos Pred Value                1.00            0.8947           1.0000
## Neg Pred Value                1.00            1.0000           0.9375
## Prevalence                    0.26            0.3400           0.4000
## Detection Rate                0.26            0.3400           0.3600
## Detection Prevalence          0.26            0.3800           0.3600
## Balanced Accuracy             1.00            0.9697           0.9500
```

- Решим данную задачу с помощью деревьев решений. Для этого загрузим пакет для классификации с помощью деревьев решений, обучим модель и визуализируем результаты модели.

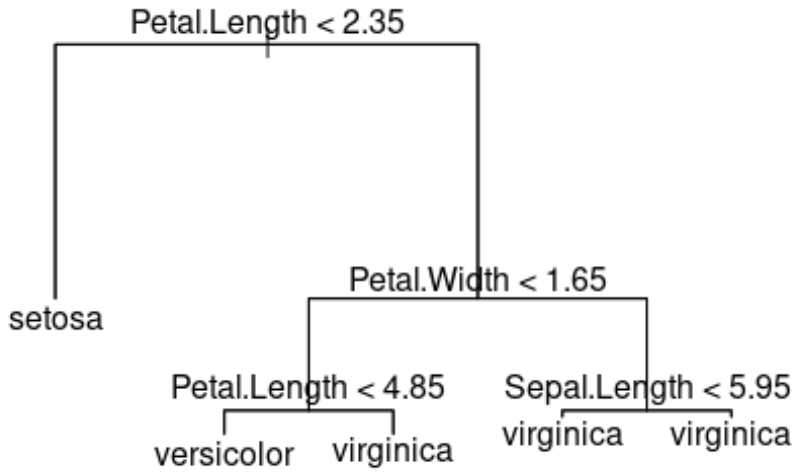
```
library(tree)

treeModel <- tree(
  formula = Species ~ .,
  data = train)
summary(treeModel)
```

```
##
## Classification tree:
```

```
## tree(formula = Species ~ ., data = train)
## Variables actually used in tree construction:
## [1] "Petal.Length" "Petal.Width"  "Sepal.Length"
## Number of terminal nodes:  5
## Residual mean deviance:  0.1445 = 13.72 / 95
## Misclassification error rate: 0.04 = 4 / 100
```

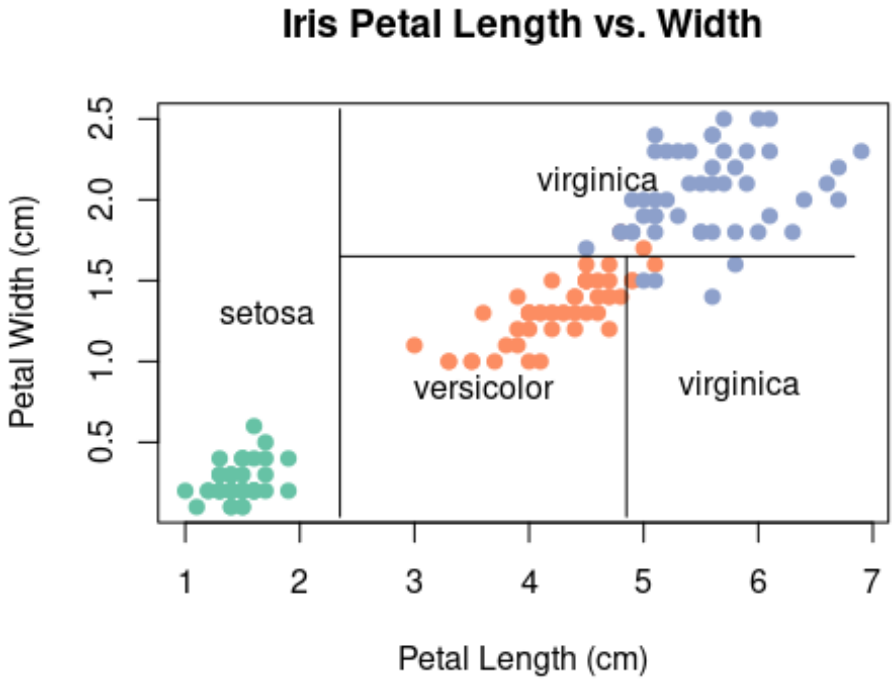
- Отрисуем дерево решений

```
plot(treeModel)
text(treeModel)
```



- Создадим диаграмму рассеивания и отрисуем границы участков дерева решений

```
treeModel1 = snip.tree(treeModel, nodes = c(12, 7))

## Warning in node.match(nodes, node, tree$frame$var == "<leaf>"): supplied n
odes
## 12 are leaves

plot(
  x = iris$Petal.Length,
  y = iris$Petal.Width,
  pch = 19,
  col = palette[as.numeric(iris$Species)],
  main = "Iris Petal Length vs. Width",
  xlab = "Petal Length (cm)",
  ylab = "Petal Width (cm)")
partition.tree(
  tree = treeModel1,
  label = "Species",
  add = TRUE)
```



- Сделаем предсказание полученной модели и рассмотрим полученные результаты

```
treePredictions <- predict(
  object = treeModel,
  newdata = test,
  type = "class")
treeMatrix <- confusionMatrix(
  data = treePredictions,
  reference = test$Species)
print(treeMatrix)
```

```
## Confusion Matrix and Statistics
##
##             Reference
## Prediction   setosa versicolor virginica
##   setosa         13          0         0
##   versicolor      0         16         1
##   virginica       0          1        19
##
## Overall Statistics
##
##                Accuracy : 0.96
##                  95% CI : (0.8629, 0.9951)
##     No Information Rate : 0.4
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.9391
##
##  Mcnemar's Test P-Value : NA
##
## Statistics by Class:
##
##                      Class: setosa Class: versicolor Class: virginica
## Sensitivity                   1.00            0.9412           0.9500
## Specificity                   1.00            0.9697           0.9667
## Pos Pred Value                1.00            0.9412           0.9500
```

```
## Neg Pred Value                1.00            0.9697            0.9667
## Prevalence                    0.26            0.3400            0.4000
## Detection Rate                0.26            0.3200            0.3800
## Detection Prevalence          0.26            0.3400            0.4000
## Balanced Accuracy             1.00            0.9554            0.9583
```

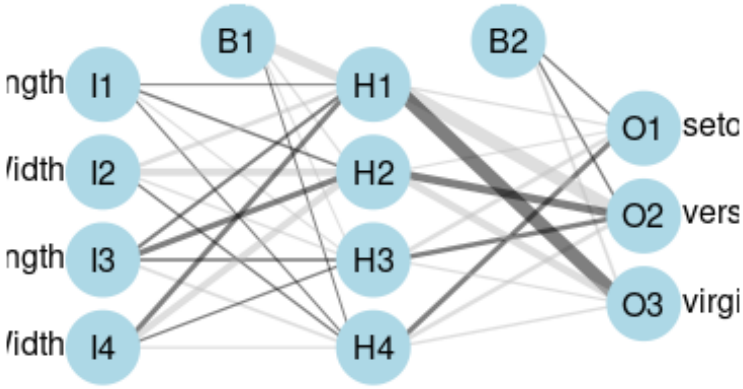
- Решим задачу классификации с помощью нейронных сетей. Для этого загрузим пакет для классификации с помощью нейронных сетей, обучим модель и визуализируем результаты модели

```
library(nnet)
neuralModel <- nnet(
  formula = Species ~ .,
  data = train,
  size = 4,
  decay = 0.0001,
  maxit = 500)

## # weights:  35
## initial  value 129.551829
## iter  10 value 49.985599
## iter  20 value 43.669724
## iter  30 value 43.665313
## iter  40 value 43.663217
## iter  50 value 42.513258
## iter  60 value 6.139024
## iter  70 value 2.127723
## iter  80 value 0.834528
## iter  90 value 0.609394
## iter 100 value 0.538797
## iter 110 value 0.504667
## iter 120 value 0.485511
## iter 130 value 0.438837
## iter 140 value 0.390736
## iter 150 value 0.372643
## iter 160 value 0.347384
## iter 170 value 0.331890
## iter 180 value 0.312769
## iter 190 value 0.306527
## iter 200 value 0.304712
## iter 210 value 0.301781
## iter 220 value 0.298428
## iter 230 value 0.297033
## iter 240 value 0.295715
## iter 250 value 0.293539
## iter 260 value 0.292254
## iter 270 value 0.290862
## iter 280 value 0.290755
## iter 290 value 0.290686
## iter 300 value 0.290656
## iter 310 value 0.290632
## iter 320 value 0.290626
## iter 330 value 0.290605
## final  value 0.290599
## converged

summary(neuralModel)
```

```
## a 4-4-3 network with 35 weights
## options were - softmax modelling  decay=1e-04
##  b->h1 i1->h1 i2->h1 i3->h1 i4->h1
## -16.15   0.25  -5.10   3.64   6.51
##  b->h2 i1->h2 i2->h2 i3->h2 i4->h2
##  -1.07   1.95 -11.34   7.92 -10.38
##  b->h3 i1->h3 i2->h3 i3->h3 i4->h3
##  -0.28  -0.66  -1.32   2.59   1.31
##  b->h4 i1->h4 i2->h4 i3->h4 i4->h4
##   0.27   0.63   1.36  -2.49  -1.29
##  b->o1 h1->o1 h2->o1 h3->o1 h4->o1
##   1.25  -1.23  -0.71  -5.06   6.41
##  b->o2 h1->o2 h2->o2 h3->o2 h4->o2
##   1.39 -23.12  12.47   6.08  -4.75
##  b->o3 h1->o3 h2->o3 h3->o3 h4->o3
##  -2.63  24.36 -11.77  -1.03  -1.66
```

- С помощью пакета NeuralNetTools визуализируем сеть

```
library(NeuralNetTools)
plotnet(neuralModel, alpha=0.5)
```



- Сделаем предсказание полученной модели и рассмотрим полученные результаты

```
neuralPredictions <- predict(
  object = neuralModel,
  newdata = test[, 1:4],
  type = "class")
neuralMatrix <- confusionMatrix(
  data = as.factor(neuralPredictions),
  reference = test$Species)
print(neuralMatrix)
```

```
## Confusion Matrix and Statistics
##
##             Reference
## Prediction   setosa versicolor virginica
##   setosa         13          0         0
##   versicolor      0         17         1
##   virginica       0          0        19
##
## Overall Statistics
##
##                Accuracy : 0.98
```

```
##                  95% CI : (0.8935, 0.9995)
##     No Information Rate : 0.4
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.9696
##
##  Mcnemar's Test P-Value : NA
##
## Statistics by Class:
##
##                      Class: setosa Class: versicolor Class: virginica
## Sensitivity                   1.00            1.0000           0.9500
## Specificity                   1.00            0.9697           1.0000
## Pos Pred Value                1.00            0.9444           1.0000
## Neg Pred Value                1.00            1.0000           0.9677
## Prevalence                    0.26            0.3400           0.4000
## Detection Rate                0.26            0.3400           0.3800
## Detection Prevalence          0.26            0.3600           0.3800
## Balanced Accuracy             1.00            0.9848           0.9750
```

- Сравним точность 3 классификаторов

```
print(knnMatrix$overall[1])

## Accuracy
##     0.96

print(treeMatrix$overall[1])

## Accuracy
##     0.96

print(neuralMatrix$overall[1])

## Accuracy
##     0.98
```

- Из полученных результатов можно сделать вывод о том, что наилучшим классификатором по точности является классификатор, основанный на использовании нейронной сети.

Задание: предсказать уровень страхового полюса

Создать новый проект: File → New Project → New Directory → New Project → Вводим имя новой директории R_Lab2 и указываем путь к папке, где будут храниться лабораторные работы. → Create project.

При выполнении данных команд у вас должен создаться новый проект. Далее создадим новый R Script, где непосредственно будут выполняться лабораторные работы. File → New File → R Script или комбинацией клавиш Ctrl + Shift + N. В появившемся окне будет вводиться код. Каждая команда с новой строки. Для выполнения команды следует нажать на кнопку Run (Ctrl + Enter).

Результаты выполнения будут отражаться в поле **Console**, **Enviroment** или **Plots**.

1. Загрузить файл с данными **Risk.csv**;
2. Просмотреть структуру данных с помощью функций head() или View(). **Результат с описанием структуры данных включить в отчёт**;

3. Загрузить библиотеку RColorBrew и создать цветовую палитру (**палитра Set2 с 3 цветами**);
4. Преобразовать категориальные признаки в фактор;
5. Визуализировать данные в виде цветной матрицы рассеивания, применив созданную палитру. **Полученную матрицу и ее описание включить в отчет**;
6. Создать диаграмму рассеивания возраста от BMI, используя созданную палитру. **Диаграмму с описанием включить в отчёт**;
7. Для функции set.seed() установить аргумент, равный 42. **Объяснить для чего используется данная функция в работе**;
8. Создать индексы для обучающей и тестовой выборок с помощью функции createDataPartition(), с параметрами p=0.8 и list=FALSE;
9. Используя созданные индексы создать обучающую и тестовую выборки;
10. Определить количество строк в обучающей и тестовой выборках с помощью функции nrow();
11. Загрузить пакет caret;
12. Обучить модель методом k-ближайших соседей, используя при этом формулу **Risk~Age+BMI+Gender+State.Rate**. **Объяснить выбранное количество соседей?**
13. Спрогнозировать результаты модели на основе тестовой выборки.
14. Создать матрицу ошибок и вывести результаты. **Какова точность модели?**
15. Загрузить библиотеку для классификации с помощью деревьев решений;
16. Обучить модель, используя следующую формулу **Risk ~ Age + BMI + Gender + State.Rate**;
17. Визуализировать данную модель;
18. Построить дерево модели;
19. Спрогнозировать результаты модели на основе тестовой выборки;
20. Создать матрицу ошибок;
21. Вывести результаты**. Какова точность модели?**
22. Загрузить пакет для классификации с помощью нейронных сетей;
23. Обучить модель, используя формулу **Risk ~ Age + BMI + Gender + State.Rate** и следующие параметры: **size = 10, decay = 0.00001, maxit = 500**;
24. Визуализировать модель;
25. Загрузить библиотеку для NeuralNetTools визуализации нейронной сети;
26. Построить модель нейронной сети;
27. Спрогнозировать результаты модели на основе тестовой выборки;
28. Создать матрицу ошибок;
29. Вывести результаты модели. **Какова точность модели?**
30. Сравнить результаты рассмотренных классификаторов и выбрать лучший.

Содержание отчёта

Файл отчёта должен содержать полный код программы с описанием производимых действий и ответами на вопросы.