МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РОССИЙСКОЙ ФЕДЕРАЦИИ
Федеральное государственное автономное образовательное учреждение
высшего образования
«СЕВЕРО-КАВКАЗСКИЙ ФЕДЕРАЛЬНЫЙ УНИВЕРСИТЕТ»

Методические указания по выполнению лабораторных работ

по дисциплине «Функциональное программирование»

для студентов направления подготовки
09.03.02 «Информационные системы и технологии»
Профиль «Прикладное программирование в интеллектуальных информационных системах»

Ставрополь, 2026

СОДЕРЖАНИЕ

ВВЕДЕНИЕ

1. Цели и задачи освоения дисциплины. Основная цель – дать студентам теоретическое представление о современных проблемамах в области систем искусственного интеллекта и машинного обучения, а также познакомить с путями их решения.

Пособие предназначено для студентов, обладающих теоретическими знаниями в области проектирования приложений и практическими навыками программирования (предпочтительно языки C, C++, C#, Python, R). Цель учебного пособия: сформировать у студентов целостный взгляд на современные тенденции в областях машинного обучения, искусственного интеллекта, анализа данныз; сформировать систему компетенций.

Учебное пособие (лабораторный практикум) по дисциплине «Функциональное программирование» для студентов направления 09.03.02 «Информационные системы и технологии». Пособие охватывает теоретические аспекты построения информационных систем на основе методов функционального программирования, а также предлагает студентам практические рекомендации по разработке систем на основе метеодов функционального программирования.

В пособии рассмотрены практические аспекты проектирования и разработки информационных систем для решения различных задач с использованием следующих подходов: линейных методов классификации, аппарата нейронных сетей, байесовских методов классификации, алгоритмов восстановления регрессии, алгоритмов логической классификации.

Многие задачи, возникающие в практических приложениях, не могут быть решены заранее известными методами или алгоритмами. Это происходит по той причине, что нам заранее не известны механизмы порождения исходных данных или же известная нам информация недостаточна для построения модели источника, генерирующего поступающие к нам данные. Как говорят, мы получаем данные из «черного ящика». В этих условиях ничего

не остается, как только изучать доступную нам последовательность исходных данных и пытаться строить предсказания совершенствуя нашу схему в процессе предсказания. Подход, при котором про- шлые данные или примеры используются для первоначального формирования и совершенствования схемы предсказания, называется методом машинного обучения (Machine Learning). Машинное обучение – чрезвычайно широкая и динамически развивающаяся область исследований, использующая огромное число теоретических и практических методов.

ЛАБОРАТОРНАЯ РАБОТА 1. ОСНОВЫ ПРОГРАММИРОВАНИЯ НА F#

Цели и задачи

Цель лабораторной работы: изучение программных средств для реализации функционального программирования.

Основные задачи:

– установка и настройка среды разработки Visual Studio;

– изучение принципов загрузки и очистки данных;

– получение навыков по предварительной обработке данных на языке F#;

– изучение основных библиотек F# для работы с данными.

Теоретическое обоснование

F# – это мультипарадигменный язык программирования, разработанный в Microsoft Research и предназначенный для исполнения на платформе Microsoft .NET. F# наследует от языков OCaml и Haskell и сочетает функциональную парадигму с императивными возможностями и объектной моделью .NET.

Базовый элемент синтаксиса F# – это выражение. Примеры выражений:

```
11
3 + 5 + 3
(1 + 2) * (3 + 4)
"Hello"
"Hello" + "␣World"
intToString 5
"Hello␣World" + 1
```

Результатом вычисления выражения является значение. Значение будем отделять от выражения символом =>:

```
11 => 11
3 + 5 + 3 => 11
(1 + 2) * (3 + 4) => 21
"Hello" => "Hello"
"Hello" + "␣World" => "Hello␣World"
intToString 5 => "5"
"Hello␣World" + 1 – ошибка!
```

Программа на F# состоит из деклараций. Рассмотрим декларацию переменных:

```
let x : int = 3 + 5 + 3
```

Переменной с именем x будет сопоставлено значение выражения 3 + 5 + 3 (типа int). В общем виде переменные объявляются конструкцией let (var) : (type) = (exp).

При вычислении последовательности деклараций let, вычисляется первая декларация, а затем её значение подставляется в следующие выражения, в места, где упоминается соответствующая переменная. Например:

```
let x : int = 2 + 3
let y : int = x + 1
let z : int = 5 + y
```

Шаг 1:

```
let x : int = 5
let y : int = x + 1
let z : int = 5 + y
```

Шаг 2:

```
let x : int = 5
let y : int = 6
let z : int = 5 + y
```

Шаг 3:

```
let x : int = 5
let y : int = 6
let z : int = 11
```

Как и в других языках, функция в языке F# имеет имя, может иметь параметры и принимать аргументы, а также функция имеет тело. В языке F# функции являются объектами первого порядка, т.е. могут возвращать функции и принимать функции в качестве аргумента.

Функции определяются с помощью ключевого слова let. Формат декларации в общем виде: let(func) (arg) = (body). Например, функцию f(x) = 3 ∗ x + 2 можно определить как:

```
let f (x : int) : int = (3*x) + 2
```

В случае, когда функции необходимо рекурсивно вызывать себя, в определение добавляется ключевое слово rec. Пример – функция вычисления факториала:

```
let rec fact n =
  if n = 0 then 1
  else n * fact (n - 1);;
```

Главная операция для функции – это применение – подстановка аргументов в тело функции:

```
    f 3
|-> (3 * 2) + 2
|-> 6 + 2
|-> 8
```

Есть несколько вариантов порядка применения:

• аппликативный – самый “естественный” порядок, при котором аргументы функции полностью вычисляются до её вызова. Такой порядок используется в большинстве языков программирования

• нормальный – порядок, при котором аргументы функции начинают вычисляться только тогда, когда в них возникает необходимость

• ленивый – вариант реализации нормального порядка, при котором не происходит дублирования вычислений

Для примера определим функцию double:

```
let double x = x + x
```

Пример аппликативного порядка:

```
    double (double 5)
|-> double (5 + 5)
|-> double 10
|-> 10 + 10
|-> 20
```

Пример нормального порядка:

```
    double (double 5)
|-> (double 5) + (double 5)
|-> 10 + 10
|-> 20
```

В F# есть поддержка ленивого порядка редукции с помощью ключевого слова lazy.

Оборудование и материалы

Для выполнения лабораторной работы рекомендуется использовать персональный компьютер со следующими программными средствами разработки (выбрать один или несколько програмных продуктов для практической реализации задач лабораторной работы): MS Visual Studio 2019 и выше.

Методика и порядок выполнения работы

Индивидуальное задание

1. Напишите программу, которая выяснит, какой порядок применения функций – аппликативный или нормальный – используется в F#? То же для любого императивного языка на ваш выбор. Подумайте, для каких задач какой порядок применения будет предпочтительнее/
2. Напишите итеративную версию алгоритма быстрого возведения в

степень по формуле

$$a^k = \begin{cases} a^{k/2} \cdot a^{k/2} & \text{если k} \vdots 2 \\ a \cdot a^{k-1} & \text{если k} \not\vdots 2 \end{cases}$$

3. Напишите функцию проверки числа на простоту. Реализуйте рекурсивную и итеративную версию. Хорошая скорость работы алгоритма – порядка O(√ n).
4. Нахождение площади фигуры методом Монте-Карло заключается во вписывании фигуры в прямоугольник и последовательной генерации большого числа случайно выбранных точек из этого прямоугольника. Итоговой площадью считается доля точек, попавших в фигуру относительно их общего количества, умноженная на площадь объемлющего прямоугольника:

$$S_{\Phi} = \frac{n_{\text{внутр}}}{n_{\text{всего}}} \cdot S_{\Pi}$$

Напишите программу, вычисляющую внутреннюю площадь эллипса. В качестве генератора случайных чисел придумайте какую-то функцию, которая производит нетривиальную трансформацию своего аргумента, так что результат функции трудно предсказуем и варьируется в значительных пределах. Для реализации программы вам придется хранить начальное значение (seed) своего генератора случайных чисел, принимать от него при каждом вызове, кроме сгенерированного псевдослучайного числа, еще и новый seed. Этот seed вы должны будете хранить от вызова к вызову.

Содержание отчета и его форма

Отчет по лабораторной работе должен содержать:

1. Номер и название лабораторной работы; задачи лабораторной работы.

2. Реализация каждого пункта подраздела «Индивидуальное задание» с приведением исходного кода программы, диаграмм и графиков для визуализации данных.

3. Ответы на контрольные вопросы.

4. Экранные формы (консольный вывод) и листинг программного кода с комментариями, показывающие порядок выполнения лабораторной работы, и результаты, полученные в ходе её выполнения.

Отчет о выполнении лабораторной работы подписывается студентом и сдается преподавателю.

Контрольные вопросы

1. Какие инструментальные средства используются для организации рабочего места специалиста Data Science?

2. Какие библиотеки Python используются для работы в области машинного обучения? Дайте краткую характеристику каждой библиотеке.

3. Почему при реализации систем машинного обучения широкое распространение получили библиотеки Python?

Список литературы

Для выполнения лабораторной работы, при подготовке к защите, а также для ответа на контрольные вопросы рекомендуется использовать следующие источники: [1-3, 9, 10].

ЛАБОРАТОРНАЯ РАБОТА 2. ФУНКЦИИ ВЫСШИХ ПОРЯДКОВ

Цели и задачи

Цель лабораторной работы: изучение программных средств для визуализации наборов данных.

Основные задачи:

– получение навыков создания одномерных данных;

–изучение основных типов регресии;

– получение навыков реализации модели линейной регрессии.

Теоретическое обоснование

Пример функции, вычисляющей квадратный корень методом Ньютона. Эта задача может быть разбита на подзадачи меньшего объёма:

1. Получить начальное приближение.
2. Итеративно повторять шаги 2 и 3.
3. Вычислить новое приближение
4. Проверить, является ли результат достаточно хорошим. Если да – вернуть ответ

В результате такой декомпозиции мы можем написать реализацию метода Ньютона, абстрагированную от конкретной задачи – от вычисления квадратного корня. Вынося, например, вычисление нового приближения в отдельную подзадачу, мы получим универсальную программу, с помощью которой можно вычислить приближение любой математической функции.

Пусть метод Ньютона принимаеть функцию вычисления нового приближения в качестве аргумента. Такая функция, манипулирующая другими функциями, называется функцией высшего порядка (higher-order function):

```
let newton first guess enough =
    let rec iter cur =
        if enough cur then cur
        else iter (guess cur)
    iter first
```

Теперь для вычисления квадратного корня достаточно реализовать несколько вспомогательных функций и передать их функции newton в качестве аргументов:

```
let isGood x y = abs (x - y*y) < 0.000001
let improve x y = ((x / y) + y) / 2.0
let sqrt x = newton 1. (improve x) (isGood x)
```

Вообще говоря, поддержка функций высшего порядка в языке означает способность языка манипулировать описаниями вычислений как полноценными значениями наравне с другими структурами данных. Для языков без такой способности существует ряд приёмов для имитации функций высшего порядка.

Функции высшего порядка иногда называют «операторами» или «функционалами» по аналогии с терминами функционального анализа.

Запишем функцию для суммирования значений i для i = a . . . b:

```
let rec sum a b =
    if a > b then 0
    else a + sum (a + 1) b
```

А теперь напишем функцию для суммирования квадратов:

```
let rec sumSq a b =
    if a > b then 0
    else a*a + sumSq (a + 1) b
```

Обе функции могут быть переписаны в виде итеративного процесса, но в этом примере это не важно. Код функций похож, как будто он написан с использованием общего шаблона. Присутствие такого шаблона является веским доводом в пользу того, что здесь скрыта полезная абстракция.

Действительно, в математической нотации эта абстракция называется “сигма-записью”:

$$sum = \sum_{i=a}^{b} i$$

$$sumSq = \sum_{i=a}^{b} i^2$$

Сила сигма-записи в том, что она выражает само понятие суммы вообще, а не просто некоторые конкретные суммы. Выразим эту идею в виде функции:

```
let rec sumOf op a next b =
    if a > b then 0
    else (op a) + sumOf op (next a) next b
```

Здесь функция next означает «приращение аргумента». Её можно использовать, например, при вычислении суммы ряда

$$\frac{\pi}{4} = \frac{1}{3 \cdot 5} + \frac{1}{7 \cdot 9} + \frac{1}{9 \cdot 11} + \dots$$

Выразим sum и sumSq через sumOf:

```
let id a = a
let inc a = a + 1
let square a = a*a
let sum a b = sumOf id a inc b
let sumSq a b = sumOf square a inc b
```

Такая запись на порядок понятней. Теперь sumOf выражает идею суммирования, а sum и sumSq – используют эту идею. Перепишем функцию sumOf в виде рекурсивного процесса:

```
let sumOf op a next b =
    let rec sumIter a s =
        if a > b then s
        else sumIter (next a) (s + (op a))
    sumIter a 0
```

Списком в F# называется упорядоченная последовательность значений одного типа. Например, список значений типа int можно определить как:

```
let listInts = [1; 2; 3; 100]
```

Пустой список обозначается как []. Для списков определена операция ::, позволяющая присоединить элемент к «голове» списка:

```
0 :: [3; 2; 1]
val it : int list = [0; 3; 2; 1]
```

Два списка можно “склеить” в новый список операцией @:

```
[true; true] @ [false; true; false]
val it : bool list = [true; true; false; true; false]
```

Список в F# неизменяем в том смысле, что в нём невозможно изменить элемент. Для работы со списками удобно использовать сопоставление с образцом:

```
match someList with
| head :: tail -> tail
| [] -> []
```

Такая операция сначала сравнит список someList с образцом head :: tail. Если это возможно, и список представим в виде конкатенации (склейки) головного элемента head и списка из остальных элементов (хвоста) tail, то выполнится первая ветка и результатом операции будет список-хвост tail. Если же нет, и список someList не содержит элементов (т.е. совпадает с []), результатом операции будет пустой список.

Напишем функцию суммирования списка в рекурсивном и итеративном варианте:

```
let rec sum list =
    match list with
    | head :: tail -> head + sum tail
    | [] -> 0

let sum list =
    let rec loop list acc =
        match list with
        | head :: tail -> loop tail (acc + head)
        | [] -> acc
    loop list 0
```

Аналогично суммированию чисел в интервале, здесь можно выделить общую идею – “свёртку” списка с помощью некоторой бинарной операции op и начального значения base.

Функции высшего порядка могут не только принимать другие функции в качестве аргумента, но и возвращать их в качестве результата.

Оборудование и материалы

Для выполнения лабораторной работы рекомендуется использовать персональный компьютер со следующими программными средствами разработки (выбрать один или несколько програмных продуктов для практической реализации задач лабораторной работы): MS Visual Studio 2015 и выше; среда разработки Java, интерпретатор Python (Anaconda) с библиотеками matplotlib, seaborn, numpy.

Методика и порядок выполнения работы

Индивидуальное задание

По согласованию с переподавателем выполните 3 задачи из списка ниже.

1. Неподвижной точкой функции f называется такое значение f, что f(x) = x. Её можно попробовать найти итеративной последовательностью приближений:

$$x, f(x), f(f(x)), f(f(f(x))), \ldots$$

Напишите процедуру поиска неподвижной точки. Найдите с её помощью значение золотого сечения φ.

2. Описанный выше метод поиска неподвижной точки может начать осциллировать и не сойтись, например, если искать sqrta как неподвижную точку функции $x \to \frac{a}{x}$. Тем не менее, можно использовать торможение усреднением: искать неподвижную точку не функции f(x), а функции $\frac{x+f(x)}{2}$. Реализуйте этот метод.
3. Напишите функцию fold, которая принимает список элементов, функциюбинарный оператор op и начальное значение base, а затем применяет op к двум аргументам – к base и к первому элементу списка x1. На следующем шаге fold применяет op к результату предыдущей операции и элементу x2 и так далее, в результате вычисляя формулу:

$$op\ (\ldots(op\ (op\ base\ x_1)\ x_2)\ldots)\ x_n$$

4. Напишите функцию dropWhile, которая удаляет самый длинный префикс (начальную часть) заданного списка, состоящий из элементов, удовлетворяющих некоторому предикату. В качестве аргументов она принимает предикат и исходный список, а возвращает новый список. Например,

```
dropWhile (fun x -> x < 5) [1..10]
val it : int list = [5; 6; 7; 8; 9; 10]

dropWhile (fun x -> x > 0) [1; 2; -3; 2; 1]
val it : int list = [-3; 2; 1]

dropWhile (fun x -> x % 2 = 0) [2; 4; 1; 2; 3]
val it : int list = [1; 2; 3]

dropWhile (fun x -> x = 1) [2; 2; 1; 1]
val it : int list = [2; 2; 1; 1]

dropWhile (fun x -> x <> 0) [1..5]
val it : int list = []
```

5. Реализовать списочные комбинаторы. Просто имя без комментариев означает одноименный стандартный комбинатор модуля List. Примечание: если можно, реализовать сразу итеративно.
6. Написать функцию, порождающую список перестановок заданного списка в каком-либо порядке.

```
permute [1..3];;
val it : int list list = [[1; 2; 3]; [1; 3; 2]; [2; 1;
    3]; [2; 3; 1]; [3;1; 2]; [3; 2; 1]]
```

7. Написать оператор суперпозиции, принимающий список функций и возвращающий единственную функцию, которая принимает аргумент x и последовательно “цепочкой” применяет к нему все функции из списка, передавая выходное значение (i − 1)-й функции на вход i-й.
8. Пусть с помощью АТД реализовано двоичное дерево поиска (все ключи левого поддерева меньше ключа в корне; все ключи правого поддерева больше ключа в корне). Напишите для него функцию memberOf, которая ищет заданный элемент в дереве и возвращает ссылку на вершину с ним либо какое-то уведомление об отсутствии (например, Nil-вершину).

9. Для бинарного дерева реализовать следующие функции:

   (a) Высота дерева.

   (b) Количество листьев.

   (c) Поиск элемента, удовлетворяющего предикату. В аргументах – три функции: предикат p, функция whenF ound, которой нужно передать вершину в случае успешного поиска, и функция whenNotF ound, которая вызывается в случае неуспешного.

   (d) Правый поворот дерева.

Содержание отчета и его форма

Отчет по лабораторной работе должен содержать:

1. Номер и название лабораторной работы; задачи лабораторной работы.

2. Реализация каждого пункта подраздела «Индивидуальное задание» с приведением исходного кода программы, диаграмм и графиков для визуализации данных.

3. Ответы на контрольные вопросы.

4. Экранные формы (консольный вывод) и листинг программного кода с комментариями, показывающие порядок выполнения лабораторной работы, и результаты, полученные в ходе её выполнения.

Отчет о выполнении лабораторной работы подписывается студентом и сдается преподавателю.

Контрольные вопросы

1. Какие инструментальные средства используются для организации рабочего места специалиста Data Science?

2. Какие библиотеки Python используются для работы в области машинного обучения? Дайте краткую характеристику каждой библиотеке.

3. Почему при реализации систем машинного обучения широкое распространение получили библиотеки Python?

Список литературы

Для выполнения лабораторной работы, при подготовке к защите, а также для ответа на контрольные вопросы рекомендуется использовать следующие источники: [1-4].

ЛАБОРАТОРНАЯ РАБОТА 3. СВЁРТКА

Цели и задачи

Цель лабораторной работы: изучение принципов построения информационных систем с использованием метрических методов классификации.

Основные задачи:

– изучение инструментария Python для реализации алгоритмов метрической классификации;

– изучение методов оптимизации параметров метрической классификации;

– освоение модификаций kNN-метода.

Теоретическое обоснование

Синтаксическое дерево – это промежуточное представление кода программы в компиляторах языков программирования. В процессе проверки синтаксического разбора программы парсер строит синтаксическое дерево, которое потом подвергается различным манипуляциям для оптимизации, проверки типов и т.д.

Суть свёртки определяется названием: она сворачивает сложную структуру данных в некоторое значение. Так можно решить много разных задач, в том числе и поставленные выше. Изобразим на доске пример сложной структуры в виде дерева, где каждый узел – это конструктор. Рядом изобразим дерево аналогичной структуры, только с операциями в узлах. Каждому конструктору соответствует отдельная операция.

Мы будем рассматривать упрощённую версию дерева из языка, определённого в FPF.

Тип такого дерева:

```
type expr =
     | Const of int
     | Var of string
     | Add of expr * expr
     | Mult of expr * expr
     | Observe of date * string
```

Задача. Написать функцию для расчёта выплаты, используя дерево.

Это – задача свёртки, по сути нам нужно свернуть дерево по неким правилам в одно значение. Теперь нужно определить операции, соответствующие разным конструкторам типа, которые будут решать задачу.

Можно решить данную задачу с помощью рекурсивно определённой функции. Но можно сделать это более элегантно, задав простые и нерекурсивные способы комбинирования значений, а всю рекурсию спрятать в оператор свёртки. Введём понятия “контекста свёртки”. В контексте будут храниться уже вычисленные значения предыдущих свёрток. Чтобы получить тип контекста применим преобразование к типу expr, чтобы получить новый тип данных.

1. Вводим новый параметр типа a

2. Заменяем рекурсивные вхождения expr в оригинальном типе на параметр a

3. Переименовываем конструкторы, добавляя Context.

Стало:

```
type 'a exprContext =
     | ConstContext of int
     | VarContext of string
     | AddContext of 'a * 'a
     | MultContext of 'a * 'a
     | ObserveContext of date * string
```

Наш контекст – типизированный, в нём могут храниться определённые значения типа a, а именно – промежуточные результаты. Теперь определим функцию на контексте, определяющую свёртку Она будет иметь тип ’a

exprContext -> a. Эта функции будет нерекурсивной и её достаточно для того, чтобы полностью определить свёртку. Для нашей задачи результатом свёртки будет int, т.к. результат вычисления выражения – число, а именно результат определяет тип a в контексте.

Таким образом, нам надо задать функцию вида ’int exprContext -> int.

```
// eval : 'int exprContext -> int
let eval expr =
    match expr with
    | ConstContext x          -> x
    | VarContext s            -> 42 // lookup in env
    | AddContext (x, y)       -> x + y
    | MultContext (x, y)      -> x * y
    | ObserveContext (d, a) -> lookup d a market
```

Здесь мы не делаем рекурсивных вызовов, а просто описываем как комбинировать элементы, считая что предыдущие вызовы у нас уже вычислены. Можно сказать, что эта функция знает как свернуть один уровень структуры данных.

Как теперь “запустить” эту функцию, чтобы она приняла дерево и выдала результат?

Для этого нам нужно преобразовать функцию f, которая определяет свёртку, в реальную свёртку. То есть у нас есть определение того, как свернуть один уровень дерева, и нам нужно свернуть всё дерево, обладая только этой функцией и знанием о типе дерева. Это делается с помощью функции такого рода:

```
// toFold : ('a exprContext -> 'a) -> expr -> 'a
let rec toFold ctx f =
    match ctx with
    | Const x         -> f (ConstContext x)
    | Var   s         -> f (VarContext s)
    | Add  (x, y)     -> f (AddContext (toFold f x, toFold f y))
    | Mult (x, y)     -> f (MultContext (toFold f x, toFold f y))
    | Observe (d, a) -> f (ObserveContext(d, a))
```

Такое обобщение свёрток на алгебраические структуры данных называется “катаморфизмом”.

Соответственно, финальная функция, которая будет решать нашу задачу выглядит так:

```
// e = 2 + 3 * 4
let e = Add(Const 2, Mult(Const 3, Const 4))
let solve e = cata eval e
let main = printfn "%A" (solve e)
// output: 14
```

Важно, что f – нерекурсивна, вся рекурсия спрятана в toFold.

Оборудование и материалы

Для выполнения лабораторной работы рекомендуется использовать персональный компьютер со следующими программными средствами разработки (выбрать один или несколько програмных продуктов для практической реализации задач лабораторной работы): MS Visual Studio 2013 и выше; среда разработки Java, интерпретатор Python (Anaconda).

Методика и порядок выполнения работы

Индивидуальное задание

1. Реализуйте катаморфизм на синтаксическом дереве, который возвращает множество переменных, использованных в выражении.
2. Разработайте алгебраический тип и свёртки для следующих задач.

   (a) У нас есть дерево документа. Элементами дерева могут быть:

   • параграф

   • картинка

   • ссылка

• текст

В каждом параграфе может быть произвольное количество картинок, подпараграфов, текста, ссылок и т.д. Задача состоит в том, чтобы извлечь все ссылки для того, чтобы вывести их в конце презентации. Напишите для этого катаморфизм.

(b) Напишите катаморфизм, который удалит все картинки из документа.

Содержание отчета и его форма

Отчет по лабораторной работе должен содержать:

1. Номер и название лабораторной работы; задачи лабораторной работы.

2. Реализация каждого пункта подраздела «Индивидуальное задание» с приведением исходного кода программы, диаграмм и графиков для визуализации данных.

3. Ответы на контрольные вопросы.

4. Экранные формы (консольный вывод) и листинг программного кода с комментариями, показывающие порядок выполнения лабораторной работы, и результаты, полученные в ходе её выполнения.

Отчет о выполнении лабораторной работы подписывается студентом и сдается преподавателю.

Контрольные вопросы

1. Допустим, тест на некое заболевание R дал положительный ответ, хотя на самом деле у испытуемого нет этого заболевания. Какую ошибку допустил тест?

2. Пусть в матрице ошибок TP = 5, TN = 90, FP = 10, FN = 5. Оцените метрики классификации для такой матрицы ошибок.

3. Перечислите возможные гиперпараметры модели логистической регрессии.

4. В чем заключается метод парзеновского окна?

5. Поясните принцип метода потенциальных функций.

6. Назовите, какие параметры оптимизируют в методах kNN?

Список литературы

Для выполнения лабораторной работы, при подготовке к защите, а также для ответа на контрольные вопросы рекомендуется использовать следующие источники: [1-5].

ЛАБОРАТОРНАЯ РАБОТА 4. СПИСОЧНЫЕ ГОМОМОРФИЗМЫ

Цели и задачи

Цель лабораторной работы: изучение принципов построения информационных систем с использованием логических методов классификации.

Основные задачи:

– освоение метода разложения матриц (метод главных компонент);

– освоение технологии внедрения алгоритмов на онове решающих деревьев в приложения;

– изучение параметров логической классификации;

– освоение модификаций логических методов классификации.

Теоретическое обоснование

Перед выполнением лабораторной работы необходимо ознакомиться с теорией, используя следующие источники: [1-5].

Оборудование и материалы

Для выполнения лабораторной работы рекомендуется использовать персональный компьютер со следующими программными средствами разработки (выбрать один или несколько програмных продуктов для практической реализации задач лабораторной работы): MS Visual Studio 2013 и выше; среда разработки Java, интерпретатор Python (Anaconda).

Методика и порядок выполнения работы

Задание

1. Реализуйте Rope с размером chunk = 1 (в листьях подвешены единичные элементы). Балансировка отсутствует. Rope должен быть параметризован тремя типовыми аргументами: типом данных, хранящихся в дереве, типом меры, и типом моноида, реализующего операцию над мерой. По-видимому, удобнейший способ это сделать в отсутствие классов типов – использовать аналогичную концепцию из мира ООП, интерфейсы. Заготовка приведена ниже.

```
type IMeasured<'V> =
    abstract Value : 'V

type IMonoid<'V> =
    abstract Zero : 'V
    abstract Plus : 'V -> 'V -> 'V

type Singleton<'T when 'T : (new : unit -> 'T)> private () =
    static let instance = new 'T()
    static member Instance = instance

type Tree<'T, 'M, 'V when 'M :> IMonoid<'V> and 'M : (new :
    unit -> 'M) and 'T :> IMeasured<'V>> =
    | Leaf of 'T
```

Здесь приведен интерфейс «измеримых» данных, интерфейс моноида, пример моноида, базовая реализация дерева (обратите внимание, как синтаксис F# позволяет наложить дополнительные ограничения на типовые параметры). Оператор :> значит «является подтипом». Отдельное внимание обратите на синглтон. Это известный паттерн в промышленном программировании: тип, у которого одновременно может существовать только один экземпляр. Это именно то, что нам нужно, чтобы пользоваться моноидами: моноидальная операция Plus принадлежит типу, а не его экземплярам, но для её вызовов экземпляры все-таки приходится создавать. Чтобы избавиться от этого неудобства, тип моноида делают синглтоном. Тогда, чтобы обращаться к операции, можно использовать следующий код:

```
let monoid = Singleton<SumMonoid>.Instance
let x = monoid.Plus l r
```

a) Для этого Rope реализуйте:

• Построение дерева по массиву (списку) данных, произвольным способом. Желательно организовать построение так, чтобы полученное дерево сразу имело высоту O(log N).

• Изменение элемента по индексу (с перевычислением мер по пути).

• Слияние двух веревок за O(1).

• Вычисление меры на подотрезке.

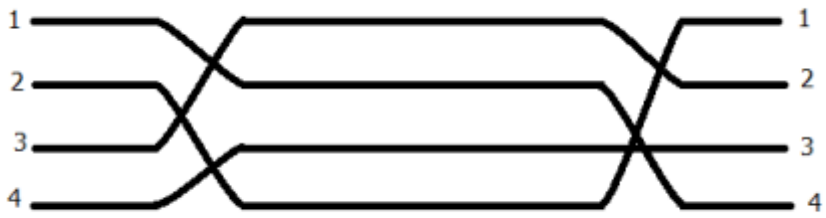
• Разрезание по предикату. Если какие-то из операций удастся реализовать через уже реализованные другие – отлично, это только лучше

(b) Реализуйте моноид статических характеристик выборки. Пользуясь классом System.Random, сгенерируйте 100 выборок размера 50, постройте из них дерево и выведите характеристики результирующей выборки. Проверьте результат, слив выборки вручную и подсчитав характеристики по определению.

(c) Имеется кабель из k проводов, который идет из точки A в точку B. Кабель выходит на поверхность в N местах, не считая начало и конец ($N \leq 105$). Между этими точками кабель залегает глубоко под землей, и доступа к нему нет.

У монтера-электрика дяди Васи выдался плохой день, и этим вечером он, основательно отдохнув, решил отомстить всему этому несправедливому миру. На протяжении вечера во время «основательного отдыха» периодически дядя Вася озлобляется и идет к кабелю – мстить за свой плохой день. Всего за день это произошло T раз ($T \leq 105$). Суть каждой мести заключается в том, что дядя Вася идет к какому-нибудь из N мест, откуда он может достать до кабеля, разрезает его и перепутывает в нем провода. Более строго, дядя Вася выбирает какое-то место $1 \leq i \leq N$ и разрезает кабель в этом месте. Теперь в левой руке (ближней к точке A) у него k проводов в ряд, и в правой руке

(ближней к точке B) у него тоже k проводов в ряд. Дядя Вася каким-то случайным образом присоединяет каждый из k проводов слева к какому-то из k проводов справа (взаимно однозначно), заматывает все изолентой и со злорадной улыбкой удаляется. Теперь информация, посланная из точки A по проводу №j ($1 \le j \le k$), придет в точку B не на выход №j, а на какой-то другой. Поскольку всю эту процедуру дядя Вася проделывал T раз в разных местах кабеля, все провода безнадежно перепутались. Наутро неблагодарные жильцы района нашли дядю Васю и убедительно попросили его рассказать им, какой же из выходов кабеля какому входу теперь соответствует. После продуктивного вечернего отдыха дядя Вася оказался не в силах решить такую тяжелую вычислительную задачу. Зато, к своему удивлению, он помнил в точности, какие локальные перепутывания в каких точках он производил. То есть, для каждого из T перепутываний в хронологическом порядке дядя Вася может рассказать, какие провода в левой руке он соединял с какими проводами в правой руке, а также где это все происходило. Естественно, что, поскольку перепутываний было много, то провод №1 в левой руке дяди Васи к концу вечера совершенно не обязательно соответствует проводу №1 в точке A. С правой рукой та же проблема. Вам дан список T воспоминаний дяди Васи. Каждое воспоминание содержит точку очередной диверсии n ($1 \le n \le N$) и список соответствий перепутывания в нумерации с точки зрения дяди Васи, глядящего в этот момент на разрезанный кабель (например, $1 \to 5$, $2 \to 3$, $3 \to 8$, . . .). Гарантируется, что воспоминания даны в хронологическом порядке. Пользуясь теорией моноидальных вычислений и запрограммированным ранее деревом, помогите жильцам района и вычислите итоговую таблицу соответствия проводов кабеля в точке 3 A проводам в точке B после всех махинаций дяди Васи. К примеру, пусть k = 4, N = 3, T = 2. Первое перепутывание происходит в точке n = 3 по схеме $1 \to 2$, $2 \to 4$, $3 \to 3$, $4 \to 1$, второе – в точке n = 1 по схеме $1 \to 2$, $2 \to 4$, $3 \to 1$, $4 \to 3$. Тогда в итоге провода будут соответствовать следующим образом: $1 \to 4$, $2 \to 1$, $3 \to 2$, $4 \to 3$ (см. рисунок).



Содержание отчета и его форма

Отчет по лабораторной работе должен содержать:

1. Номер и название лабораторной работы; задачи лабораторной работы.

2. Реализация каждого пункта подраздела «Индивидуальное задание» с приведением исходного кода программы, диаграмм и графиков для визуализации данных.

3. Ответы на контрольные вопросы.

4. Экранные формы (консольный вывод) и листинг программного кода с комментариями, показывающие порядок выполнения лабораторной работы, и результаты, полученные в ходе её выполнения.

Отчет о выполнении лабораторной работы подписывается студентом и сдается преподавателю.

Контрольные вопросы

1. Как связаны главные компоненты с исходными данными?

2. Сделайте грубую оценку сжатия данных, если исходная матрица имела размерность (4250, 7), а при восстановлении используются три главные компонены.

3. Сгенерируйте данные в виде эллипса с центром в точке (1.5, -2.5), радиусами (3, 2.5), углом 65 и количеством точек 1100. Оцените собственные

вектора, собственные значения, максимальные и минимальные значения в пространстве главных компонент.

4. Что такое многоклассовая информативность? Для чего она применяется?

5. Поясните назначение и алгоритм бинаризации количественных признаков.

6. Поясните порядок поиска закономерностей в форме конъюнкций.

Список литературы

Для выполнения лабораторной работы, при подготовке к защите, а также для ответа на контрольные вопросы рекомендуется использовать следующие источники: [1–5].

ЛАБОРАТОРНАЯ РАБОТА 5. КЛАСТЕРИЗАЦИЯ

Цели и задачи

Цель лабораторной работы: изучение принципов построения информационных систем с использованием линейных методов машинного обучения.

Основные задачи:

– освоение технологии внедрения алгоритмов линейной классификации в приложения;

– изучение основных приемов рабты с разреженными матрицами в ходе машинного обучения;

– освоение техники построения, обучения и оценки модели логистической регресии;

– освоение приемов работы с синтезированными признаками, масштабированием и настройкой гипперпараметров.

Теоретическое обоснование

Перед выполнением лабораторной работы необходимо ознакомиться с теорией построения регрессионных моделей машинного обучения, используя следующие источники: [1-5].

Оборудование и материалы

Для выполнения лабораторной работы рекомендуется использовать персональный компьютер со следующими программными средствами разработки (выбрать один или несколько програмных продуктов для практической реализации задач лабораторной работы): MS Visual Studio 2013 и выше; среда разработки Java, интерпретатор Python (Anaconda).

Методика и порядок выполнения работы

Учебная задача

Для простоты описания используем двумерный случай, однако все описанные метрики будут работать и в случаях пространств большей размерности.

На рисунке 5.1 представлены эквидистантные фигуры для различных метрик расстояний. Все точки, лежащие на прямых одного цвета, находятся на расстоянии 1 от центра координат с точки зрения соответствующей метрики расстояния. Запишем функцию для определения расстояния:

```
def distance(X1, X2, metric = 'euclidean', p = 2):
    if metric == 'euclidean':
        dist = np.sqrt(np.sum(np.square(X1 - X2).T,axis=0))
    if metric == 'cityblock':
        dist = np.sum(np.abs(X1 - X2).T,axis=0)
    if metric == 'Chebyshev':


            dist = np.max(np.abs(X1 - X2).T,axis=0)
        if metric == 'Minkowski':
            dist = np.power(np.sum(np.power(np.
abs(X1 - X2),p).T,axis=0),1/p)
        return dist
```

Функция работает как с векторами, так и с матрицами равной размерности.

Как правило, метрикой расстояния по умолчанию является евклидова метрика. Однако в разных задачах, особенно если анализируются категориальные переменные, могут хорошо проявлять себя и другие метрики. Выбор метрики расстояния тоже можно отнести к гиперпараметрам модели.

Кроме того, необходимо помнить о стандартизации или нормализации данных при использовании метрик расстояния. Иначе результаты вычисления

метрик могут быть некорректны, если используются данные, измеряющиеся в разных диапазонах.

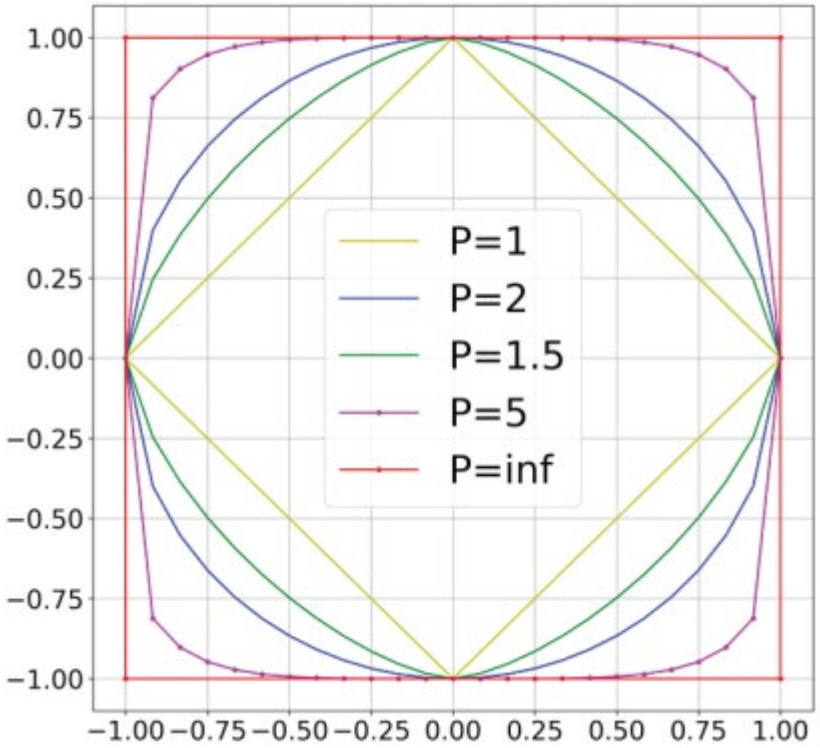


Рисунок 5.1. Эквидистантные фигуры для разных метрик расстояния

Одним из самых простых методов кластеризации является метод k-средних. Суть данного метода сводится к тому, чтобы найти заданное число кластеров (k) и их центры (так называемые центроиды) - такие, чтобы расстояние от центроидов до всех точек кластера было минимальны.

Прежде чем проводить кластеризацию, необходимо проинициализировать кластеры. Для этого выберем случайные индексы среди доступных в наборе данных:

```
    def init_centroids(X, n_clusters):
        centroid_idxs = np.random.randint(0, X.shape[0],
size = n_clusters)
        return X[centroid_idxs,:]
```

Посмотрим, как это работает для двух кластеров. Проведем первую кластеризацию. Для этого возьмем каждый центроид и посчитаем расстояние

от него до всех записей набора данных. Индексы значений для каждого кластера выберем как индексы минимальных расстояний до соответствующего центроида. Таким образом, нулевой кластер будет включать те точки набора данных, в которых расстояние до нулевого центроида меньше, чем до первого центроида.

```
    def predict(X, n_clusters, centroids, metric =
'euclidean', p = 2):
        distances = np.zeros((X.shape[0], n_clusters))
        for i,centr in enumerate(centroids):
            distances[:,i] = distance(centr,X, metric, p)
        cluster_label = np.argmin(distances,axis = 1)
        return cluster_label
```

Посмотрим, как распределились результаты кластеризации после случайной инициализации центроидов (рис. 5.2). Получилось достаточно интересно: оба центроида оказались относительно близко друг к другу. Тем не менее это нулевая итерация алгоритма.

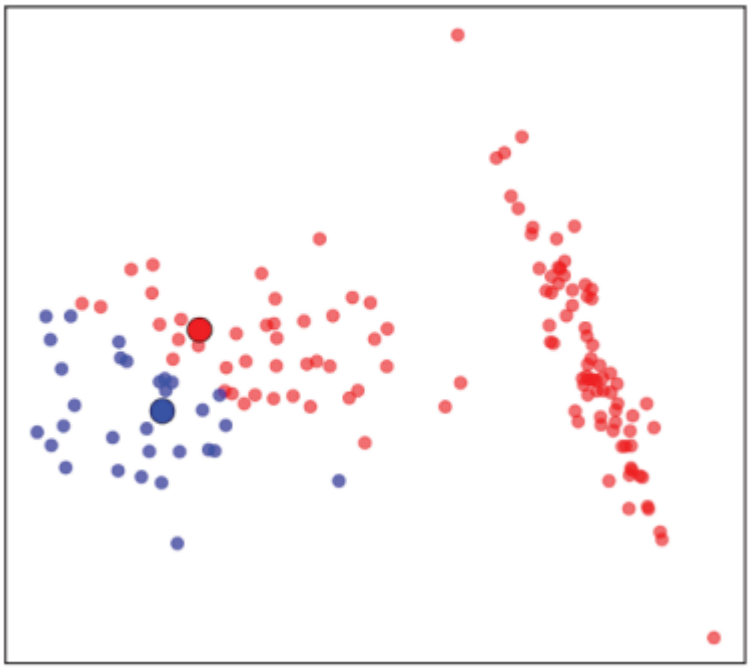


Рисунок 5.2. Результаты кластеризации после инициализации центроидов

Теперь выберем новые центроиды для каждого кластера как среднее значение по кластеру. В визуализации будут видны старые центроиды и новый

центроид с бóльшим радиусом (рис. 5.3). Видно, что центр для красного кластера значительно сместился.

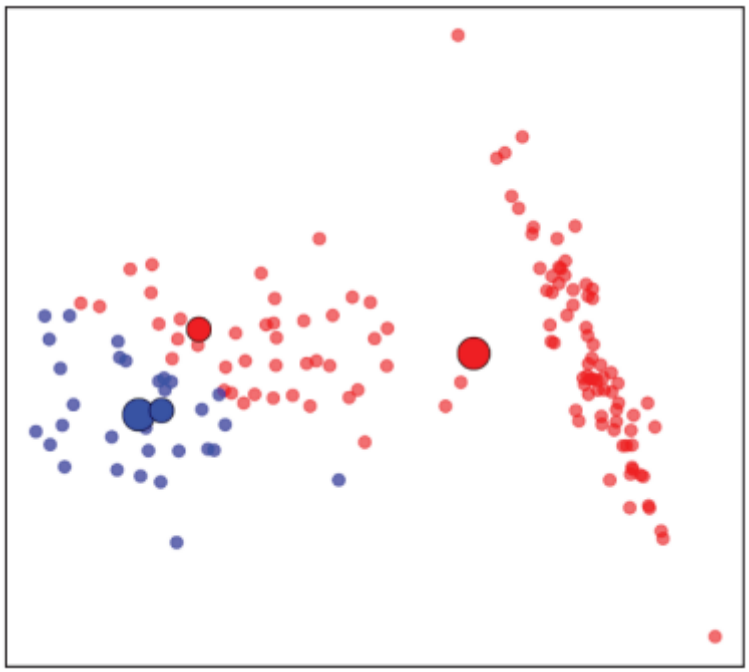


Рисунок 5.3. Результаты кластеризации после изменения центроидов

Рассчитаем относительное расстояние между старыми и новыми центроидами. Если расстояние между обновленными центроидами будет сравнительно небольшим, то есть центроиды перестанут менять позицию, то будем считать, что кластеризация закончена.

```
    def delta_centroids(centroids,old_centroids,
metric = 'euclidean', p = 2):
        return (distance(centroids,old_centroids, metric, p)/
distance(old_centroids, np.mean(old_centroids), metric, p)).mean()
```

Для автоматизации процесса реализуем процедуру итерационной кластеризации. Укажем порог относительного изменения центроидов – tol. В конце процедуры выведем результирующий номер итерации и расстояние между последним изменением центроидов:

```
    def fit(X, n_clusters, centroids, max_
iter=10, tol=0.01, metric = 'euclidean', p = 2):

        dcentr = np.inf

        for i in range(max_iter):

         old_centroids = np.copy(centroids)
        cluster_label=predict(X, n_clusters, centroids, metric, p)

            for k in range(n_clusters):
                c_idxs = np.flatnonzero(cluster_label==k)
                centroids[k] = X[c_idxs].mean(axis = 0)

          dcentr = delta_centroids(centroids,old_centroids, met-
ric, p)

            if dcentr<=tol:
                break

        print('Мы остановились на итерации:', i,', относитель-
ное изменение центроидов: ',dcentr)

        return cluster_label
```

Проверим и визуализируем результаты. Буквально через пару итераций получим следующее распределение (рис. 5.4).

Теперь объединим все наши наработки в один класс KMeans. Центроиды на последней итерации хранятся в атрибуте .centroids. Для того чтобы получить предсказания номера кластера, необходимо воспользоваться методом .fit_transform. Кроме того, мы добавили в класс KMeans атрибут .inertia — сумму квадратов расстояния точек кластеров до соответствующих центроидов.

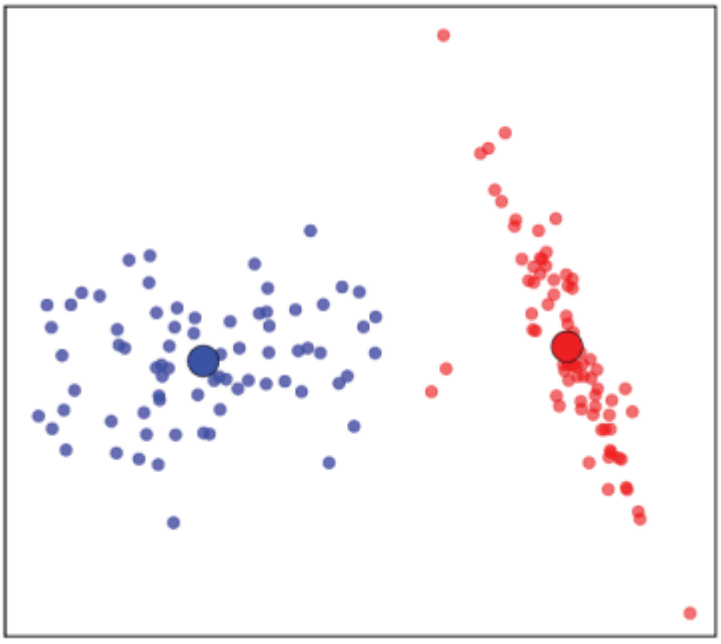


Рисунок 5.4 Результаты кластеризации после нескольких итераций

С помощью этого параметра можно попытаться ответить на вопрос, какое число кластеров оптимально для конкретных данных. Для этого можно использовать так называемый метод локтя (Elbow Method): проверяются различные количества кластеров и запоминаются конечные значения инерции для каждого числа кластеров, затем визуализируется зависимость (число кластеров, инерция). Как правило, получаются зависимости, подобные рис. 5.5.

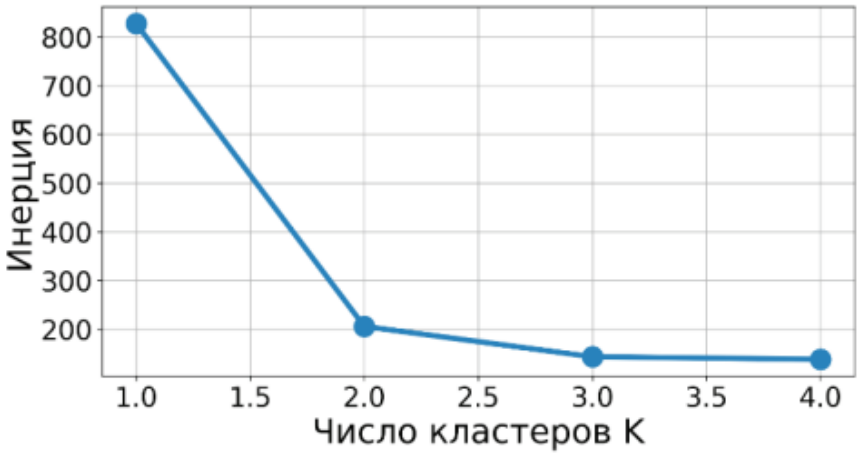


Рисунок 5.5 Визуализация метода локтя

Затем ищется такое число кластеров, которое напоминает перегиб согнутого локтя (в нашем случае это число k = 2). Большее количество кластеров неуместно: мы начинаем искусственно дробить большие кластеры на малые, нарушая целостную структуру данных (рис. 5.6). Метод k-средних можно применять не только к двумерным данным, но и к данным большей размерности. При этом рекомендуется пользоваться методами уменьшения размерности для возможности визуализации результата кластеризации.

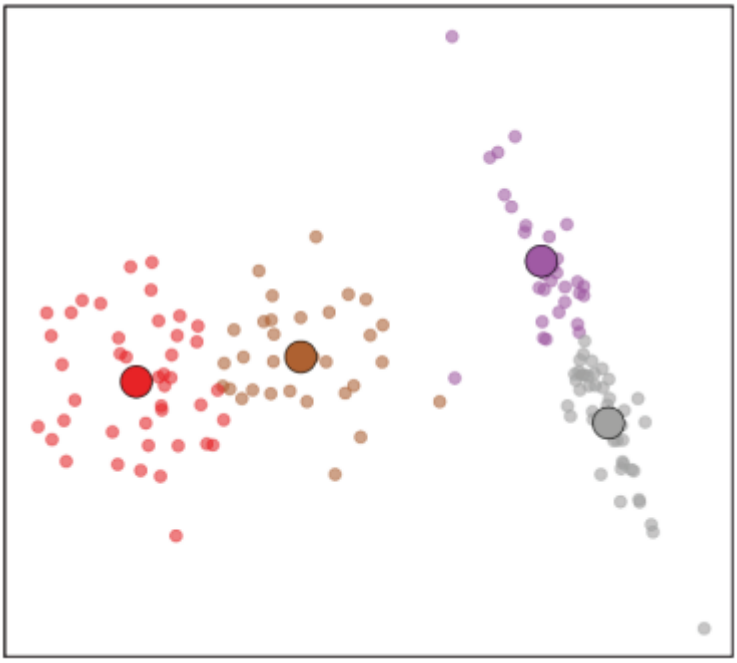


Рисунок 5.6. Кластеризация k-средних со слишком большим числом кластеров

В общем случае кластеризация применяется в контексте того, что истинные метки кластеров нам неизвестны. Именно поэтому эту задачу машинного обучения относят к задачам обучения без учителя. Однако иногда возникают тренировочные задачи, в которых метки известны или помимо числовых данных есть набор категориальных признаков, и требуется выяснить, связаны ли найденные кластеры с каким-то категориальным признаком

Индивидуальное задание

1. Студент самостоятельно выбирает набор данных и согласует свой выбор с преподавателем.
2. Сгенерируйте линейно разделимые данные с другими параметрами и проверьте, как работает алгоритм кластеризации k-средних на этих данных. Оцените оптимальное число кластеров по методу локтя.
3. Сгенерируйте данные, распределенные как знак инь-ян или концентрические круги, и проверьте, как работает алгоритм кластеризации k-средних на этих данных. Оцените оптимальное число кластеров по методу локтя.
4. Загрузите данные MNIST. Уменьшите размерность данных с использованием метода главных компонент. Примените кластеризацию k-средних. Оцените оптимальное число кластеров по методу локтя и связь кластеров с цифрами на изображениях.
5. Выполните кластеризацию для набора данных:

   • выполните кластеризацию для числовых признаков: используйте все числовые признаки, выполнив визуализацию в разных двумерных проекциях;

   • оцените оптимальное число кластеров по методу локтя;

   • оцените связь кластеров с категориальными признаками;

   • сравните результаты модели при использовании данных с применением метода главных компонент.
6. Сравните работу реализованных алгоритмов с функциями библиотеки scikit-learn – кластеризацией k-средних sklearn. cluster.KMeans.

Содержание отчета и его форма

Отчет по лабораторной работе должен содержать:

1. Номер и название лабораторной работы; задачи лабораторной работы.

2. Реализация каждого пункта подраздела «Индивидуальное задание» с приведением исходного кода программы, диаграмм и графиков для визуализации данных.

3. Ответы на контрольные вопросы.

4. Экранные формы (консольный вывод) и листинг программного кода с комментариями, показывающие порядок выполнения лабораторной работы, и результаты, полученные в ходе её выполнения.

Отчет о выполнении лабораторной работы подписывается студентом и сдается преподавателю.

Контрольные вопросы

1. . Оцените евклидово расстояние между векторами x1 {2, 5, 3 ,7} и x2 {2, 7, 1, 5}.

2. Оцените расстояние Чебышева между векторами x1 {0, 10, 4, 9} и x2 {3, 7, 0, 2}.

3. Есть три центроида c1 {1, 0, 0}, c2 {0, 1, 1}, c3{1, 0, 1} и точка x с координатами {2, 0, 2}. К какому кластеру следует отнести эту точку при использовании евклидовой метрики расстояния?

4. Как называется метод определения оптимального числа k (кластеров) с использованием анализа инерции?

Список литературы

Для выполнения лабораторной работы, при подготовке к защите, а также для ответа на контрольные вопросы рекомендуется использовать следующие источники: [1–5].

ЗАКЛЮЧЕНИЕ

Учебное пособие (лабораторный практикум) по дисциплине «Функциональное программирование» для студентов направления 09.03.02 «Информационные системы и технологии». Пособие охватывает теоретические аспекты построения информационных систем на основе методов искусственного интеллекта, а также предлагает студентам практические рекомендации по разработке систем на основе мтеодов искусственного интеллекта. Основное внимание уделяется теории обучения машин (машинное обучение, machine learning).

В пособии рассмотрены практические аспекты проектирования и разработки информационных систем для решения различных задач с использованием следующих подходов: линейных методов классификации, аппарата нейронных сетей, байесовских методов классификации, алгоритмов восстановления регрессии, алгоритмов логической классификации.

Многие задачи, возникающие в практических приложениях, не могут быть решены заранее известными методами или алгоритмами. Это происходит по той причине, что нам заранее не известны механизмы порождения исходных данных или же известная нам информация недостаточна для построения модели источника, генерирующего поступающие к нам данные. Как говорят, мы получаем данные из «черного ящика». В этих условиях ничего не остается, как только изучать доступную нам последовательность исходных данных и пытаться строить предсказания совершенствуя нашу схему в процессе предсказания. Подход, при котором про- шлые данные или примеры используются для первоначального формирования и совершенствования схемы предсказания, называется методом машинного обучения (Machine Learning). Машинное обучение – чрезвычайно широкая и динамически развивающаяся область исследований, использующая огромное число теоретических и практических методов.

СПИСОК ЛИТЕРАТУРЫ

Список основной литературы

1. Смит К. Программирование на F# – СПб.: Символ-Плюс, 2011. – 448 с.

2. Сузи, Р.А. Язык программирования Python Электронный ресурс : учебное пособие / Р.А. Сузи. - Язык программирования Python,2020-07-28. - Москва : Интернет-Университет Информационных Технологий (ИНТУИТ), 2016. - 350 c. - Книга находится в базовой версии ЭБС IPRbooks. - ISBN 5-9556-0058-2

3. Плахта Михаил. Грокаем функциональное программирование. – СПб: Питер, 2022. –512 с.

Список дополнительной литературы

4. Сошников Д. Функциональное программирование на F#. – М.: ДМК Пресс, 2016. – 385 с.

5. Седжвик, Р. Алгоритмы на C++ / Р. Седжвик. - 2-е изд., испр. - Москва : Национальный Открытый Университет «ИНТУИТ», 2016. - 1773 с., экземпляров неограниченно

6. https://archive.ics.uci.edu/ml/index.html – Репозиторий наборов данных для машинного обучения (Центр машинного обучения и интеллектуальных систем).

7. https://www.kaggle.com – Портал и система проведения соревнований по проблемам анализа данных.

8. https://www.mockaroo.com – Сайт для генерации наборов данных.

МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РФ

ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ АВТОНОМНОЕ ОБРАЗОВАТЕЛЬНОЕ
УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ
«СЕВЕРО-КАВКАЗСКИЙ ФЕДЕРАЛЬНЫЙ УНИВЕРСИТЕТ»

Методические указания

по организации и проведению самостоятельной работы
по дисциплине
«Функциональное программирование»
для студентов направления подготовки
09.03.02 «Информационные системы и технологии»
направленность (профиль)
«Прикладное программирование в интеллектуальных информационных системах»

Ставрополь, 2026

1. Общие положения

Самостоятельная работа – планируемая учебная, учебно-исследовательская, научно-исследовательская работа студентов, выполняемая во внеаудиторное (аудиторное) время по заданию и при методическом руководстве преподавателя, но без его непосредственного участия (при частичном непосредственном участии преподавателя, оставляющем ведущую роль за работой студентов).

Самостоятельная работа студентов (СРС) в ВУЗе является важным видом учебной и научной деятельности студента. Самостоятельная работа студентов играет значительную роль в рейтинговой технологии обучения.

К основным видам самостоятельной работы студентов относятся:

− формирование и усвоение содержания конспекта лекций на базе рекомендованной лектором учебной литературы, включая информационные образовательные ресурсы (электронные учебники, электронные библиотеки и др.);

− написание докладов;

− подготовка к семинарам, практическим и лабораторным работам, их оформление;

− составление аннотированного списка статей из соответствующих журналов по отраслям знаний (педагогических, психологических, методических и др.);

− выполнение учебно-исследовательских работ, проектная деятельность;

− подготовка практических разработок и рекомендаций по решению проблемной ситуации;

− выполнение домашних заданий в виде решения отдельных задач, проведения типовых расчетов, расчетно-компьютерных и индивидуальных работ по отдельным разделам содержания дисциплин и т.д.;

− компьютерный текущий самоконтроль и контроль успеваемости на базе электронных обучающих и аттестующих тестов;

− выполнение курсовых работ (проектов) в рамках дисциплин;

− выполнение выпускной квалификационной работы и др.

Методика организации самостоятельной работы студентов зависит от структуры, характера и особенностей изучаемой дисциплины, объема часов на ее изучение, вида заданий для самостоятельной работы студентов, индивидуальных качеств студентов и условий учебной деятельности.

Процесс организации самостоятельной работы студентов включает в себя следующие этапы:

- подготовительный (определение целей, составление программы, подготовка методического обеспечения, подготовка оборудования);
- основной (реализация программы, использование приемов поиска информации, усвоения, переработки, применения, передачи знаний, фиксирование результатов, самоорганизация процесса работы);
- заключительный (оценка значимости и анализ результатов, их систематизация, оценка эффективности программы и приемов работы, выводы о направлениях оптимизации труда).

Самостоятельная работа по дисциплине «Функциональное программирование» направлена на формирование следующих **компетенций**:

Код, формулировка компетенции	Код, формулировка индикатора
ПК-1 Способен разрабатывать программное обеспечение интеллектуальных информационных систем	ПК-1$_{ИД-1.1}$ – Демонстрирует знания в области применения алгоритмов и их адаптации к практическим задачам

	ПК-1$_{ИД-1.2}$ – Разрабатывает приложения на языке высокого уровня в том числе с использованием интеллектуальных технологий
	ПК-1$_{ИД-1.3}$ – Применяет современные автоматизированные технологии тестирования и отладки программного обеспечения
	ПК-1$_{ИД-1.4}$ – Применяет современные технологии непрерывного развертывания и доставки программного обеспечения
	ПК-1$_{ИД-1.5}$ – Разрабатывает приложения для различных целевых платформ (мобильные, серверные, встраиваемые, оконные)
ПК-5 Способен осуществлять автоматизированное проектирование информационных систем, в том числе интеллектуальных	ПК-5$_{ИД-5.1}$ – Проектирует новые и выполняет реинжиниринг существующих информационных систем в рамках решения профессиональных задач основываясь на данных предметной области и перспективах применения интеллектуальных моделей
	ПК-5$_{ИД-5.2}$ – Применяет автоматизированные инструментальные средства проектирования интеллектуальных информационных систем
	ПК-5$_{ИД-5.3}$ – Применяет технологии моделирования интеллектуальных информационных систем

2. Цель и задачи самостоятельной работы

Ведущая цель организации и осуществления СРС совпадает с целью обучения студента – формирование набора общенаучных, профессиональных и специальных компетенций будущего бакалавра по соответствующему направлению подготовки

При организации СРС важным и необходимым условием становятся формирование умения самостоятельной работы для приобретения знаний, навыков и возможности организации учебной и научной деятельности. Целью самостоятельной работы студентов является овладение фундаментальными знаниями, профессиональными умениями и навыками деятельности по профилю, опытом творческой, исследовательской деятельности. Самостоятельная работа студентов способствует развитию самостоятельности, ответственности и организованности, творческого подхода к решению проблем учебного и профессионального уровня.

Задачами СРС являются:

- систематизация и закрепление полученных теоретических знаний и практических умений студентов;
- углубление и расширение теоретических знаний;
- формирование умений использовать нормативную, правовую, справочную документацию и специальную литературу;
- развитие познавательных способностей и активности студентов: творческой инициативы, самостоятельности, ответственности и организованности;
- формирование самостоятельности мышления, способностей к саморазвитию, самосовершенствованию и самореализации;

- развитие исследовательских умений;
- использование материала, собранного и полученного в ходе самостоятельных занятий на семинарах, на практических и лабораторных занятиях, при написании курсовых и выпускной квалификационной работ, для эффективной подготовки к итоговым зачетам и экзаменам.

3. Технологическая карта самостоятельной работы студента

Коды реализуемых компетенций	Вид деятельности студентов	Средства и технологии оценки	Объем часов, в том числе (акад.)		
			СРС	Контактная работа с преподавателем	Всего
4 семестр					
ПК-1, ПК-5	Самостоятельное изучение литературы и источников	Собеседование	10	2	12
ПК-1, ПК-5	Подготовка к лабораторным занятиям	Защита ЛР	48	16	64
Итого за 4 семестр			58	18	76
Итого			58	18	76

4. Порядок выполнения самостоятельной работы студентом

4.1. Методические рекомендации по работе с учебной литературой

При работе с книгой необходимо подобрать литературу, научиться правильно ее читать, вести записи. Для подбора литературы в библиотеке используются алфавитный и систематический каталоги.

Важно помнить, что рациональные навыки работы с книгой - это всегда большая экономия времени и сил.

Правильный подбор учебников рекомендуется преподавателем, читающим лекционный курс. Необходимая литература может быть также указана в методических разработках по данному курсу.

Изучая материал по учебнику, следует переходить к следующему вопросу только после правильного уяснения предыдущего, описывая на бумаге все выкладки и вычисления (в том числе те, которые в учебнике опущены или на лекции даны для самостоятельного вывода).

При изучении любой дисциплины большую и важную роль играет самостоятельная индивидуальная работа.

Особое внимание следует обратить на определение основных понятий курса. Студент должен подробно разбирать примеры, которые поясняют такие определения, и уметь строить аналогичные примеры самостоятельно. Нужно добиваться точного представления о том, что изучаешь. Полезно составлять опорные конспекты. При изучении материала по учебнику полезно в тетради (на специально отведенных полях) дополнять

конспект лекций. Там же следует отмечать вопросы, выделенные студентом для консультации с преподавателем.

Выводы, полученные в результате изучения, рекомендуется в конспекте выделять, чтобы они при перечитывании записей лучше запоминались.

Опыт показывает, что многим студентам помогает составление листа опорных сигналов, содержащего важнейшие и наиболее часто употребляемые формулы и понятия. Такой лист помогает запомнить формулы, основные положения лекции, а также может служить постоянным справочником для студента.

Чтение научного текста является частью познавательной деятельности. Ее цель – извлечение из текста необходимой информации. От того на сколько осознанна читающим собственная внутренняя установка при обращении к печатному слову (найти нужные сведения, усвоить информацию полностью или частично, критически проанализировать материал и т.п.) во многом зависит эффективность осуществляемого действия.

Выделяют ***четыре основные установки в чтении научного текста***:

информационно-поисковый (задача – найти, выделить искомую информацию)

усваивающая (усилия читателя направлены на то, чтобы как можно полнее осознать и запомнить как сами сведения излагаемые автором, так и всю логику его рассуждений)

аналитико-критическая (читатель стремится критически осмыслить материал, проанализировав его, определив свое отношение к нему)

творческая (создает у читателя готовность в том или ином виде – как отправной пункт для своих рассуждений, как образ для действия по аналогии и т.п. – использовать суждения автора, ход его мыслей, результат наблюдения, разработанную методику, дополнить их, подвергнуть новой проверке).

Основные виды систематизированной записи прочитанного:

Аннотирование – предельно краткое связное описание просмотренной или прочитанной книги (статьи), ее содержания, источников, характера и назначения;

Планирование – краткая логическая организация текста, раскрывающая содержание и структуру изучаемого материала;

Тезирование – лаконичное воспроизведение основных утверждений автора без привлечения фактического материала;

Цитирование – дословное выписывание из текста выдержек, извлечений, наиболее существенно отражающих ту или иную мысль автора;

Конспектирование – краткое и последовательное изложение содержания прочитанного.

Конспект – сложный способ изложения содержания книги или статьи в логической последовательности. Конспект аккумулирует в себе предыдущие виды записи, позволяет всесторонне охватить содержание книги, статьи. Поэтому умение составлять план, тезисы, делать выписки и другие записи определяет и технологию составления конспекта.

Методические рекомендации по составлению конспекта:

1. Внимательно прочитайте текст. Уточните в справочной литературе непонятные слова. При записи не забудьте вынести справочные данные на поля конспекта;
2. Выделите главное, составьте план;
3. Кратко сформулируйте основные положения текста, отметьте аргументацию автора;
4. Законспектируйте материал, четко следуя пунктам плана. При конспектировании старайтесь выразить мысль своими словами. Записи следует вести четко, ясно.
5. Грамотно записывайте цитаты. Цитируя, учитывайте лаконичность, значимость мысли.

В тексте конспекта желательно приводить не только тезисные положения, но и их доказательства. При оформлении конспекта необходимо стремиться к емкости каждого предложения. Мысли автора книги следует излагать кратко, заботясь о стиле и

выразительности написанного. Число дополнительных элементов конспекта должно быть логически обоснованным, записи должны распределяться в определенной последовательности, отвечающей логической структуре произведения. Для уточнения и дополнения необходимо оставлять поля.

Овладение навыками конспектирования требует от студента целеустремленности, повседневной самостоятельной работы.

4.2. Методические рекомендации по подготовке к практическим и лабораторным занятиям

Для того чтобы практические и лабораторные занятия приносили максимальную пользу, необходимо помнить, что упражнение и решение задач проводятся по вычитанному на лекциях материалу и связаны, как правило, с детальным разбором отдельных вопросов лекционного курса. Следует подчеркнуть, что только после усвоения лекционного материала с определенной точки зрения (а именно с той, с которой он излагается на лекциях) он будет закрепляться на практических занятиях как в результате обсуждения и анализа лекционного материала, так и с помощью решения проблемных ситуаций, задач. При этих условиях студент не только хорошо усвоит материал, но и научится применять его на практике, а также получит дополнительный стимул (и это очень важно) для активной проработки лекции.

При самостоятельном решении задач нужно обосновывать каждый этап решения, исходя из теоретических положений курса. Если студент видит несколько путей решения проблемы (задачи), то нужно сравнить их и выбрать самый рациональный. Полезно до начала вычислений составить краткий план решения проблемы (задачи). Решение проблемных задач или примеров следует излагать подробно, вычисления располагать в строгом порядке, отделяя вспомогательные вычисления от основных. Решения при необходимости нужно сопровождать комментариями, схемами, чертежами и рисунками.

Следует помнить, что решение каждой учебной задачи должно доводиться до окончательного логического ответа, которого требует условие, и по возможности с выводом. Полученный ответ следует проверить способами, вытекающими из существа данной задачи. Полезно также (если возможно) решать несколькими способами и сравнить полученные результаты. Решение задач данного типа нужно продолжать до приобретения твердых навыков в их решении.

4.3. Методические рекомендации по самопроверке знаний

После изучения определенной темы по записям в конспекте и учебнику, а также решения достаточного количества соответствующих задач на практических занятиях и самостоятельно студенту рекомендуется, провести самопроверку усвоенных знаний, ответив на контрольные вопросы по изученной теме.

В случае необходимости нужно еще раз внимательно разобраться в материале.

Иногда недостаточность усвоения того или иного вопроса выясняется только при изучении дальнейшего материала. В этом случае надо вернуться назад и повторить плохо усвоенный материал. Важный критерий усвоения теоретического материала - умение решать задачи или пройти тестирование по пройденному материалу. Однако следует помнить, что правильное решение задачи может получиться в результате применения механически заученных формул без понимания сущности теоретических положений.

4.4. Методические рекомендации по написанию научных текстов (докладов, докладовов, эссе, научных статей и т.д.)

Перед тем, как приступить к написанию научного текста, важно разобраться, какова истинная цель вашего научного текста - это поможет вам разумно распределить свои силы и время.

Во-первых, сначала нужно определиться с идеей научного текста, а для этого необходимо научиться либо относиться к разным явлениям и фактам несколько критически (своя идея – как иная точка зрения), либо научиться увлекаться какими-то известными идеями, которые нуждаются в доработке (идея – как оптимистическая позиция и направленность на дальнейшее совершенствование уже известного). Во-вторых, научиться организовывать свое время, ведь, как известно, свободное (от всяких глупостей) время – важнейшее условие настоящего творчества, для него наконец-то появляется время. Иногда именно на организацию такого времени уходит немалая часть сил и талантов.

Писать следует ясно и понятно, стараясь основные положения формулировать четко и недвусмысленно (чтобы и самому понятно было), а также стремясь структурировать свой текст. Каждый раз надо представлять, что ваш текст будет кто-то читать и ему захочется сориентироваться в нем, быстро находить ответы на интересующие вопросы (заодно представьте себя на месте такого человека). Понятно, что работа, написанная «сплошным текстом» (без заголовков, без выделения крупным шрифтом наиболее важным мест и т, п.), у культурного читателя должна вызывать брезгливость и даже жалость к автору (исключения составляют некоторые древние тексты, когда и жанр был иной и к текстам относились иначе, да и самих текстов было гораздо меньше – не то, что в эпоху «информационного взрыва» и соответствующего «информационного мусора»).

Объем текста и различные оформительские требования во многом зависят от принятых в конкретном учебном заведении порядков.

Доклад - это самостоятельное исследование студентом определенной проблемы, комплекса взаимосвязанных вопросов.

Доклад не должна составляться из фрагментов статей, монографий, пособий. Кроме простого изложения фактов и цитат, в доклад е должно проявляться авторское видение проблемы и ее решения.

Рассмотрим основные этапы подготовки

а студентом.

Выполнение доклада начинается с выбора темы.

Затем студент приходит на первую консультацию к руководителю, которая предусматривает:

- обсуждение цели и задач работы, основных моментов избранной темы;
- консультирование по вопросам подбора литературы;
- составление предварительного плана.

Следующим этапом является работа с литературой. Необходимая литература подбирается студентом самостоятельно.

После подбора литературы целесообразно сделать рабочий вариант плана работы. В нем нужно выделить основные вопросы темы и параграфы, раскрывающие их содержание.

Составленный список литературы и предварительный вариант плана уточняются, согласуются на очередной консультации с руководителем.

Затем начинается следующий этап работы - изучение литературы. Только внимательно читая и конспектируя литературу, можно разобраться в основных вопросах темы и подготовиться к самостоятельному (авторскому) изложению содержания доклада. Конспектируя первоисточники, необходимо отразить основную идею автора и его позицию по исследуемому вопросу, выявить проблемы и наметить задачи для дальнейшего изучения данных проблем.

Систематизация и анализ изученной литературы по проблеме исследования позволяют студенту написать работу.

Рабочий вариант текста доклада предоставляется руководителю на проверку. На основе рабочего варианта текста руководитель вместе со студентом обсуждает возможности доработки текста, его оформление. После доработки доклад сдается на кафедру для его оценивания руководителем.

Требования к написанию доклада

Написание 1 доклада является обязательным условием выполнения плана СРС по любой дисциплине профессионального цикла.

Тема доклада может быть выбрана студентом из предложенных в рабочей программе или фонде оценочных средств дисциплины, либо определена самостоятельно, исходя из интересов студента (в рамках изучаемой дисциплины). Выбранную тему необходимо согласоваться с преподавателем.

Доклад должен быть написан научным языком.

Объем доклада должен составлять 20-25 стр.

Структура доклада:

- Введение (не более 3-4 страниц). Во введении необходимо обосновать выбор темы, ее актуальность, очертить область исследования, объект исследования, основные цели и задачи исследования.
- Основная часть состоит из 2-3 разделов. В них раскрывается суть исследуемой проблемы, проводится обзор мировой литературы и источников Интернет по предмету исследования, в котором дается характеристика степени разработанности проблемы и авторская аналитическая оценка основных теоретических подходов к ее решению. Изложение материала не должно ограничиваться лишь описательным подходом к раскрытию выбранной темы. Оно также должно содержать собственное видение рассматриваемой проблемы и изложение собственной точки зрения на возможные пути ее решения.
- Заключение (1-2 страницы). В заключении кратко излагаются достигнутые при изучении проблемы цели, перспективы развития исследуемого вопроса
- Список использованной литературы (не меньше 10 источников), в алфавитном порядке, оформленный в соответствии с принятыми правилами. В список использованной литературы рекомендуется включать работы отечественных и зарубежных авторов, в том числе статьи, опубликованные в научных журналах в течение последних 3-х лет и ссылки на ресурсы сети Интернет.
- Приложение (при необходимости).

Требования к оформлению:

- текст с одной стороны листа;
- шрифт Times New Roman;
- кегль шрифта 14;
- межстрочное расстояние 1,5;
- поля: сверху 2,5 см, снизу – 2,5 см, слева - 3 см, справа 1,5 см;
- доклад должен быть представлен в сброшюрованном виде.

Порядок защиты доклада:

Защита доклада проводится на практических занятиях, после окончания работы студента над ним и исправления всех недочетов, выявленных преподавателем в ходе консультаций. На защиту доклада отводится 5-7 минут времени, в ходе которого студент должен показать свободное владение материалом по заявленной теме. При защите доклада приветствуется использование мультимедиа-презентации.

Оценка доклада

Доклад оценивается по следующим критериям:

• соблюдение требований к его оформлению;

• необходимость и достаточность для раскрытия темы приведенной в тексте доклада информации;

• умение студента свободно излагать основные идеи, отраженные в докладе;

• способность студента понять суть задаваемых преподавателем и сокурсниками вопросов и сформулировать точные ответы на них.

Критерии оценки:

Оценка «отлично» выставляется студенту, если в докладе студент исчерпывающе, последовательно, четко и логически стройно излагает материал; свободно справляется с

задачами, вопросами и другими видами применения знаний; использует для написания доклада современные научные материалы; анализирует полученную информацию; проявляет самостоятельность при написании доклада.

Оценка «хорошо» выставляется студенту, если качество выполнения доклада достаточно высокое. Студент твердо знает материал, грамотно и по существу излагает его, не допуская существенных неточностей в ответе на вопросы по теме доклада.

Оценка «удовлетворительно» выставляется студенту, если материал доклада излагается частично, но пробелы не носят существенного характера, студент допускает неточности и ошибки при защите доклада, дает недостаточно правильные формулировки, наблюдаются нарушения логической последовательности в изложении материала.

Оценка «неудовлетворительно» выставляется студенту, если он не подготовил доклад или допустил существенные ошибки. Студент неуверенно излагает материал доклада, не отвечает на вопросы преподавателя.

4.5. Методические рекомендации по выполнению исследовательских проектов

Исследовательская проектная работа – это групповая работа, для выполнения которой необходим выбор и приложение научной методики к поставленной задаче, получение собственного теоретического или экспериментального материала, на основании которого необходимо провести анализ и сделать выводы об исследуемом явлении. Выполнение проекта – это всегда коллективная, творческая практическая работа, предназначенная для получения определенного продукта или научно-технического результата. Такая работа подразумевает четкое, однозначное формирование поставленной задачи, определение сроков выполнения намеченного, определение требований к разрабатываемому объекту.

Выполнение 1 группового проекта является обязательным условием выполнения самостоятельной работы по любой дисциплине профессионального цикла. Тема проектного задания может быть выбрана студентом из предложенных в рабочей программе или фонде оценочных средств дисциплины, либо определена самостоятельно, исходя из интересов студента (в рамках изучаемой дисциплины). Выбранную тему необходимо согласоваться с преподавателем.

Требования по выполнению и оформлению проекта

При выполнении проекта приветствуется работа в группе (2-3 человека). Проект – это исследовательская работа, в ходе которой студенты должны продемонстрировать владение навыками научного исследования, умения проводить анализ, обобщать информацию, делать выводы, предлагать свои решения проблемы, рассматриваемой в проекте.

При подготовке материалов проекта студенты должны продемонстрировать владение современными методами компьютерной обработки данных.

Критерии оценки работы участника проекта.

Для каждого из участников проекта оцениваются:

• профессиональные теоретические знания в соответствующей области;

• умение работать со справочной и научной литературой, осуществлять поиск необходимой информации в Интернет;

• умение работать с техническими средствами;

• умение пользоваться соответствующими выполняемому проекту информационными технологиями;

• умение готовить материалы проекта для презентации: составлять и редактировать тексты, формировать презентацию проекта;

• умение работать в команде;

• умение публично представлять результаты собственной деятельности;

• коммуникабельность, инициативность, творческие способности.

Критерии выставления оценки участникам проекта

Оценка	Профессиональные компетенции	Компетенции, связанные с использованием соответствующих выполняемому проекту технических средств и информационных технологий	Иные универсальные компетенции (коммуникабельность, инициативность, умение работать в «команде», управленческие навыки и т.д.)	Отчетность
«Отлично»	Работа выполнена на высоком профессиональном уровне. Представленный материал в основном фактически верен, допускаются негрубые фактические неточности. Студент свободно отвечает на вопросы, связанные с проектом.	Технические средства и информационные технологии освоены и использованы для реализации проекта полностью	Студент проявил инициативу, творческий подход, способность к выполнению сложных заданий, навыки работы в коллективе, организационные способности.	Проект представлен полностью и в срок.
«Хорошо»	Работа выполнена на достаточно высоком профессиональном уровне. Допущено до 4–5 фактических ошибок. Студент отвечает на вопросы, связанные с проектом, но недостаточно полно.	Обнаруживаются некоторые ошибки в использовании соответствующих технических средств и информационных технологий	Студент достаточно полно, но без инициативы и творческих находок выполнил возложенные на него задачи.	Проект представлен достаточно полно и в срок, но с некоторыми недоработками.
«Удовлетворительно»	Уровень недостаточно высок. Допущено до 8 фактических ошибок. Студент может ответить лишь на некоторые из заданных вопросов, связанных с проектом.	Обнаруживает недостаточное владение навыками работы с техническими средствами и соответствующими информационными технологиями	Студент выполнил большую часть возложенной на него работы.	Проект сдан со значительным опозданием (более недели) и не полностью
«Неудовлетворительно»	Работа не выполнена или выполнена на низком уровне. Допущено более 8 фактических ошибок. Ответы на связанные с	Навыков работы с техническими средствами нет, информационные технологии не освоены	Студент практически не работал, не выполнил свои задачи или выполнил лишь	Проект не сдан.

Оценка	Профессиональные компетенции	Компетенции, связанные с использованием соответствующих выполняемому проекту технических средств и информационных технологий	Иные универсальные компетенции (коммуникабельность, инициативность, умение работать в «команде», управленческие навыки и т.д.)	Отчетность
	проектом вопросы обнаруживают непонимание предмета и отсутствие ориентации в материале проекта.		отдельные не существенные поручения в групповом проекте.	

Студенты должны: защитить проект в режиме презентации, предъявить файлы выполненного проекта, уметь рассказать о технологиях, использованных ими при выполнении проекта, дать оценку работы каждого члена группы (*если проект групповой*).

4.6. Методические рекомендации по подготовке к экзаменам и зачетам

Изучение многих общепрофессиональных и специальных дисциплин завершается экзаменом. Подготовка к экзамену способствует закреплению, углублению и обобщению знаний, получаемых, в процессе обучения, а также применению их к решению практических задач. Готовясь к экзамену, студент ликвидирует имеющиеся пробелы в знаниях, углубляет, систематизирует и упорядочивает свои знания. На экзамене студент демонстрирует то, что он приобрел в процессе обучения по конкретной учебной дисциплине.

Экзаменационная сессия - это серия экзаменов, установленных учебным планом. Между экзаменами интервал 3-4 дня. Не следует думать, что 3-4 дня достаточно для успешной подготовки к экзаменам.

В эти 3-4 дня нужно систематизировать уже имеющиеся знания. На консультации перед экзаменом студентов познакомят с основными требованиями, ответят на возникшие у них вопросы. Поэтому посещение консультаций обязательно.

Требования к организации подготовки к экзаменам те же, что и при занятиях в течение семестра, но соблюдаться они должны более строго. Во-первых, очень важно соблюдение режима дня; сон не менее 8 часов в сутки, занятия заканчиваются не позднее, чем за 2-3 часа до сна. Оптимальное время занятий - утренние и дневные часы. В перерывах между занятиями рекомендуются прогулки на свежем воздухе, неутомительные занятия спортом. Во-вторых, наличие хороших собственных конспектов лекций. Даже в том случае, если была пропущена какая-либо лекция, необходимо во время ее восстановить (переписать ее на кафедре), обдумать, снять возникшие вопросы для того, чтобы запоминание материала было осознанным. В-третьих, при подготовке к экзаменам у студента должен быть хороший учебник или конспект литературы, прочитанной по указанию преподавателя в течение семестра. Здесь можно эффективно использовать листы опорных сигналов.

Вначале следует просмотреть весь материал по сдаваемой дисциплине, отметить для себя трудные вопросы. Обязательно в них разобраться. В заключение еще раз целесообразно повторить основные положения, используя при этом листы опорных сигналов.

Систематическая подготовка к занятиям в течение семестра позволит использовать время экзаменационной сессии для систематизации знаний.

Контроль самостоятельной работы студентов

Контроль самостоятельной работы проводится преподавателем в аудитории.

Предусмотрены следующие виды контроля: собеседование, оценка доклада, оценка презентации, оценка участия в круглом столе, оценка выполнения проекта.

Подробные критерии оценивания компетенций приведены в Фонде оценочных средств для проведения текущей и промежуточной аттестации.

Список литературы для выполнения СРС

Перечень основной литературы:

1. Мэннинг Дж., Батфилд-Эддисон П. Unity для разработчика. Мобильные мультиплатформенные игры/ Мэннинг Дж., Батфилд-Эддисон П. –М.:Спб.: Питер, 2018. 304 с.

2. Торн А. Искусство создания сценариев в Unity [Электронный ресурс]. М. : ДМК Пресс, 2016. - 360 с. Режим доступа: http://www.studentlibrary.ru/book/ISBN9785970603819.html

Перечень дополнительной литературы:

1. Торн А. Основы анимации в Unity [Электронный ресурс]. М. : ДМК Пресс, 2016. - 176 с. Режим доступа: http://www.studentlibrary.ru/book/ISBN9785970603772.html

2. Шейдеры и эффекты в Unity. Книга рецептов [Электронный ресурс] / Кенни Ламмерс - М.: ДМК Пресс, 2014. -274 с. Режим доступа: http://www.studentlibrary.ru/book/ISBN9785940747376.html

Методическая литература:

1. Методические рекомендации для самостоятельной работы студентов по дисциплине «Функциональное программирование»
2. Методические указания к лабораторным работам по дисциплине «Функциональное программирование»

Интернет-ресурсы:

1. http://el.ncfu.ru/ – система управления обучением ФГАОУ ВО СКФУ. Дистанционная поддержка дисциплины «Функциональное программирование».