

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

МЕТОДЫ ИСКУССТВЕННОГО ИНТЕЛЛЕКТА

Методические указания к лабораторным работам

Направление подготовки:

09.04.02 «Информационные системы и технологии»

Квалификация выпускника: магистр

Ставрополь, 2025

Н 63 Методы искусственного интеллекта: Методические указания к лабораторным работам для студентов направления 09.04.02 «Информационные системы и технологии» / Николаев Е.И. – Ставрополь: Изд-во СКФУ, 2025. – 163 с.

Учебное пособие (лабораторный практикум) по дисциплине «Методы искусственного интеллекта» для студентов направления 09.04.02 «Информационные системы и технологии». Пособие охватывает практические аспекты построения информационных систем на основе методов искусственного интеллекта с применением современных технологий разработки информационных систем и инструментария анализа данных. Основное внимание уделяется теории обучения машин (машинное обучение, machine learning). Пособие предназначено для студентов, обладающих теоретическими знаниями в области проектирования приложений и практическими навыками программирования (предпочтительно языки C++, C#, Python, Java, R). Цель учебного пособия: сформировать у студентов практические навыки разработки информационных систем на основе методов машинного обучения, искусственного интеллекта, анализа данных; сформировать систему компетенций.

УДК 004.41 (075.8)
ББК 22.18я73

Автор:

канд. техн. наук, доцент **Е.И. Николаев**

ОГЛАВЛЕНИЕ

ВВЕДЕНИЕ.....	5
ЛАБОРАТОРНАЯ РАБОТА 1. ИНСТРУМЕНТАЛЬНЫЕ СРЕДСТВА МАШИННОГО ОБУЧЕНИЯ	7
ЛАБОРАТОРНАЯ РАБОТА 2. ВИЗУАЛИЗАЦИЯ ДАННЫХ	17
ЛАБОРАТОРНАЯ РАБОТА 3. МЕТРИЧЕСКИЕ МЕТОДЫ КЛАССИФИКАЦИИ	32
ЛАБОРАТОРНАЯ РАБОТА 4. ЛОГИЧЕСКИЕ МЕТОДЫ КЛАССИФИКАЦИИ	42
ЛАБОРАТОРНАЯ РАБОТА 5. ЛОГИСТИЧЕСКАЯ РЕГРЕССИЯ	53
ЛАБОРАТОРНАЯ РАБОТА 6. ЛИНЕЙНАЯ РЕГРЕССИЯ	70
ЛАБОРАТОРНАЯ РАБОТА 7. РАЗРАБОТКА ЕДИНОГО ШАБЛОНА ПРЕДВАРИТЕЛЬНОЙ ОБРАБОТКИ ДАННЫХ.....	84
ЛАБОРАТОРНАЯ РАБОТА 8. ПОСТРОЕНИЕ ПАЙПЛАЙНА ОДНОМЕРНОЙ РЕГРЕССИИ.....	90
ЛАБОРАТОРНАЯ РАБОТА 9. ИСПОЛЬЗОВАНИЕ РАЗРАБОТАННОГО ПАЙПЛАЙНА ДЛЯ МНОГОМЕРНОЙ РЕГРЕССИИ	96
ЛАБОРАТОРНАЯ РАБОТА 10. ПОЛИНОМИАЛЬНАЯ РЕГРЕССИЯ	103
ЛАБОРАТОРНАЯ РАБОТА 11. СНИЖЕНИЕ РАЗМЕРНОСТИ.....	109
ЛАБОРАТОРНАЯ РАБОТА 12. АНАЛИЗ МНОГОМЕРНЫХ ДАННЫХ С ИСПОЛЬЗОВАНИЕМ МЕТОДОВ СНИЖЕНИЯ РАЗМЕРНОСТИ.....	116
ЛАБОРАТОРНАЯ РАБОТА 13. МНОГОСЛОЙНАЯ НЕЙРОННАЯ СЕТЬ.....	119
ЛАБОРАТОРНАЯ РАБОТА 14. РАЗРАБОТКА НЕЙРОНА НА PYTORCH	124

ЛАБОРАТОРНАЯ РАБОТА 15. МНОГОСЛОЙНЫЙ	
ПЕРСЕПТРОН НА PYTORCH.....	132
ЛАБОРАТОРНАЯ РАБОТА 16. КЛАССИФИКАЦИЯ	
ИЗОБРАЖЕНИЙ С ИСПОЛЬЗОВАНИЕМ MLP	141
ЛАБОРАТОРНАЯ РАБОТА 17. СВЕРТОЧНАЯ НЕЙРОННАЯ	
СЕТЬ	156
ЗАКЛЮЧЕНИЕ	161
СПИСОК ЛИТЕРАТУРЫ	162

ВВЕДЕНИЕ

Учебное пособие по дисциплине «Методы искусственного интеллекта» для студентов направления 09.04.02 «Информационные системы и технологии». Пособие охватывает теоретические аспекты построения информационных систем на основе методов искусственного интеллекта. Основное внимание уделяется теории обучения машин (машинное обучение, machine learning).

Основная задача науки и реальной жизни – получение правильных предсказаний о будущем поведении сложных систем на основании их прошлого поведения. 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 прошлые 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. Данное пособие ни в какой мере не претендует на какое-либо исчерпывающее изложение содержания данной области. Основная цель – дать студентам теоретическое представление о современных математических проблемах в области систем искусственного интеллекта, 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 данных; сформировать систему компетенций.

ЛАБОРАТОРНАЯ РАБОТА 1. ИНСТРУМЕНТАЛЬНЫЕ СРЕДСТВА МАШИННОГО ОБУЧЕНИЯ

Цели и задачи

Цель лабораторной работы: изучение программных средств для организации рабочего места специалиста по анализу данных и машинному обучению.

Основные задачи:

- установка и настройка среды разработки Python;
- изучение принципов загрузки и очистки данных;
- получение навыков по предварительной обработке данных на языке Python;
- изучение основных библиотек Python для работы с данными.

Теоретическое обоснование

Перед выполнением лабораторной работы необходимо ознакомиться с базовыми принципами языка Python, 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 программных продуктов для практической реализации задач лабораторной работы): MS Visual Studio 2013 и выше; среда разработки Java, интерпретатор Python (Anaconda).

Методика и порядок выполнения работы

Перед выполнением индивидуального задания рекомендуется выполнить все пункты учебной задачи.

Учебная задача

Необходимо организовать подготовку данных для построения модели (допустим модели классификации). В качестве данных выбран набор данных об ирисах Фишера. Это, пожалуй, самый известный набор данных, с которого многие начинают исследование алгоритмов машинного обучения.

Данный набор данных предназначен для построения модели классификации. Данные о 150 экземплярах ириса (рис. 1.1), по 50 экземпляров из трёх видов – Ирис щетинистый (*Iris setosa*), Ирис виргинский (*Iris virginica*) и Ирис разноцветный (*Iris versicolor*). Для каждого экземпляра измерялись четыре характеристики (в сантиметрах):

- 1) длина наружной доли околоцветника (sepal length);
- 2) ширина наружной доли околоцветника (sepal width);
- 3) длина внутренней доли околоцветника (petal length);
- 4) ширина внутренней доли околоцветника (petal width).

На основании этого набора данных требуется построить правило классификации, определяющее вид растения по данным измерений. Это задача многоклассовой классификации, так как имеется три класса – три вида ириса.



а)



б)



в)

Рисунок 1.1 – Внешний вид классифицируемых ирисов: а) *Iris setosa*; б) *Iris virginica*; в) *Iris versicolor*

1. Необходимо скачать набор данных из репозитория Center for Machine Learning and Intelligent Systems (необходим только один текстовый файл с данными измерений): <http://archive.ics.uci.edu/ml/datasets/Iris>.

Файл Iris.data при просмотре выглядит следующим образом (рис. 1.2):

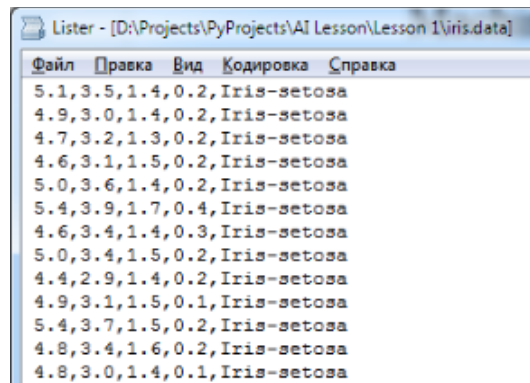
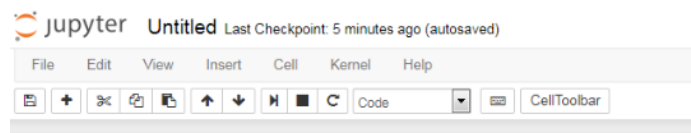


Рисунок 1.2 – Внешний вид данных файла Iris.data

2. Использовать текстовые редакторы для просмотра и анализа данных из определенных наборов – нерациональный вариант. Поэтому запустим Jupyter Notebook и начнем работать с загруженным набором с использованием среды Python. Используем метод `genfromtxt()` из пакета `scipy`.



Лабораторная работа 1

Исследование набора данных ирисы Фишера

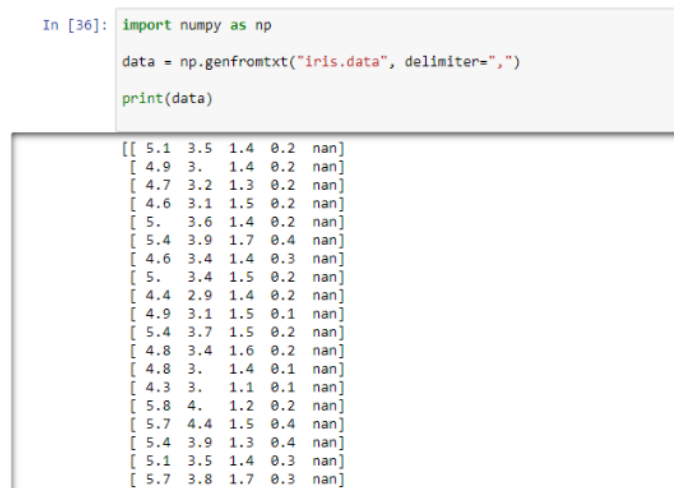


Рисунок 1.3 – Загрузка данных файла Iris.data

Метод `genfromtxt()` возвращает массив `numpy` (тип `numpy.ndarray`). Следует обратить внимание, что пятый столбец содержит неопределенные значения `numpy.NaN` (объясните – почему?).

3. Производить вывод всего источника данных – нерациональный путь. В реальных задачах данных может оказаться слишком много, поэтому

чаще всего используют подвыборку данных для поверхностного обзора исследуемой обучающей выборки (рис. 1.4).

```
In [37]: print ( "Data type : ", type(data) )
print ( "Data shape : ", data.shape )
print ( data[:10] )

Data type : <class 'numpy.ndarray'>
Data shape : (150, 5)
[[ 5.1  3.5  1.4  0.2 nan]
 [ 4.9  3.  1.4  0.2 nan]
 [ 4.7  3.2  1.3  0.2 nan]
 [ 4.6  3.1  1.5  0.2 nan]
 [ 5.  3.6  1.4  0.2 nan]
 [ 5.4  3.9  1.7  0.4 nan]
 [ 4.6  3.4  1.4  0.3 nan]
 [ 5.  3.4  1.5  0.2 nan]
 [ 4.4  2.9  1.4  0.2 nan]
 [ 4.9  3.1  1.5  0.1 nan]]
```

Рисунок 1.4 – Начальное исследование Iris.data

Из представленного фрагмента видно, что data – это двумерный массив размером 150x5, или можно сказать, что это одномерный массив, каждый элемент которого также одномерный массив размером 5 элементов.

4. В рамках данной задачи необходимо все-таки получить значения пятого столбца. Для этого желательно использовать другой подход (рис. 1.5):

```
In [69]: data1 = np.genfromtxt("iris.data", delimiter=";", dtype=None)
print(data1.shape)
print(type(data1))
print(type(data1[0]))
print(type(data1[0][4]))
print(data1[:10])

(150,)
<class 'numpy.ndarray'>
<class 'numpy.void'>
<class 'numpy.bytes_'>
[(5.1, 3.5, 1.4, 0.2, b'Iris-setosa') (4.9, 3.0, 1.4, 0.2, b'Iris-setosa')
 (4.7, 3.2, 1.3, 0.2, b'Iris-setosa') (4.6, 3.1, 1.5, 0.2, b'Iris-setosa')
 (5.0, 3.6, 1.4, 0.2, b'Iris-setosa') (5.4, 3.9, 1.7, 0.4, b'Iris-setosa')
 (4.6, 3.4, 1.4, 0.3, b'Iris-setosa') (5.0, 3.4, 1.5, 0.2, b'Iris-setosa')
 (4.4, 2.9, 1.4, 0.2, b'Iris-setosa') (4.9, 3.1, 1.5, 0.1, b'Iris-setosa')]
```

Рисунок 1.5 – Загрузка данных разного типа в массив

Сразу же желательно (на первоначальных этапах изучения Python) проводить анализ типов различных значений. На рис. 1.6 представлен еще один вариант загрузки данных в массив numpy.ndarray.

```

In [70]: dt = np.dtype("f8, f8, f8, f8, U30")
data2 = np.genfromtxt("iris.data", delimiter=",", dtype=dt)
print(data2.shape)
print(type(data2))
print(type(data2[0]))
print(type(data2[0][4]))
print(data2[:10])

(150,)
<class 'numpy.ndarray'>
<class 'numpy.void'>
<class 'numpy.str_'>
[(5.1, 3.5, 1.4, 0.2, 'Iris-setosa') (4.9, 3.0, 1.4, 0.2, 'Iris-setosa')
 (4.7, 3.2, 1.3, 0.2, 'Iris-setosa') (4.6, 3.1, 1.5, 0.2, 'Iris-setosa')
 (5.0, 3.6, 1.4, 0.2, 'Iris-setosa') (5.4, 3.9, 1.7, 0.4, 'Iris-setosa')
 (4.6, 3.4, 1.4, 0.3, 'Iris-setosa') (5.0, 3.4, 1.5, 0.2, 'Iris-setosa')
 (4.4, 2.9, 1.4, 0.2, 'Iris-setosa') (4.9, 3.1, 1.5, 0.1, 'Iris-setosa')]

```

Рисунок 1.6 – Загрузка данных в массив с типом, определяемым пользователем

Поясните различие в структурах данных, получаемых с использованием представленных листингов.

5. Было загружено 150 элементов данных, но даже при такой маленькой выборке невозможно что-либо сказать о наборе данных. Для получения дополнительной информации необходимо визуализировать загруженные данные. В нашем случае каждый элемент данных представлен вещественными признаками – это существенно упрощает визуализацию (рис. 1.7). Но сложность заключается в том, что приходится работать с элементами 4-мерного пространства, поэтому строится не графическое представление распределения, а отдельные проекции.

6. Уже из графического распределения на рис. 1.7 видно, что тип ирисов Setosa хорошо отделяется. На данном графике представлено отображение в плоскости признаков ('Sepal Width', 'Sepal Length') Но исследователь имеет возможность построить столько графиков, сколько необходимо для глубокого анализа данных. Изменим ячейку In[72] в соответствии с листингом, представленным на рис. 1.8.

```
In [72]: import matplotlib as mpl
import matplotlib.pyplot as plt

# Данные из отдельных столбцов
sepal_length = [] # Sepal Length
sepal_width = [] # Sepal Width
petal_length = [] # Petal Length
petal_width = [] # Petal Width

# Выполняем обход всей коллекции data2
for dot in data2:
    sepal_length.append(dot[0])
    sepal_width.append(dot[1])
    petal_length.append(dot[2])
    petal_width.append(dot[3])

# Строим графики по проекциям данных
# Учитываем, что каждые 50 типов ирисов идут последовательно
plt.figure(1)
setosa, = plt.plot(sepal_length[:50], sepal_width[:50], 'ro', label='Setosa')
versicolor, = plt.plot(sepal_length[50:100], sepal_width[50:100], 'g^', label='Versicolor')
virginica, = plt.plot(sepal_length[100:150], sepal_width[100:150], 'bs', label='Virginica')
plt.legend(bbox_to_anchor=(1.05, 1), loc=2, borderaxespad=0.)
plt.xlabel('Sepal Length')
plt.ylabel('Sepal Width')

plt.show()
```

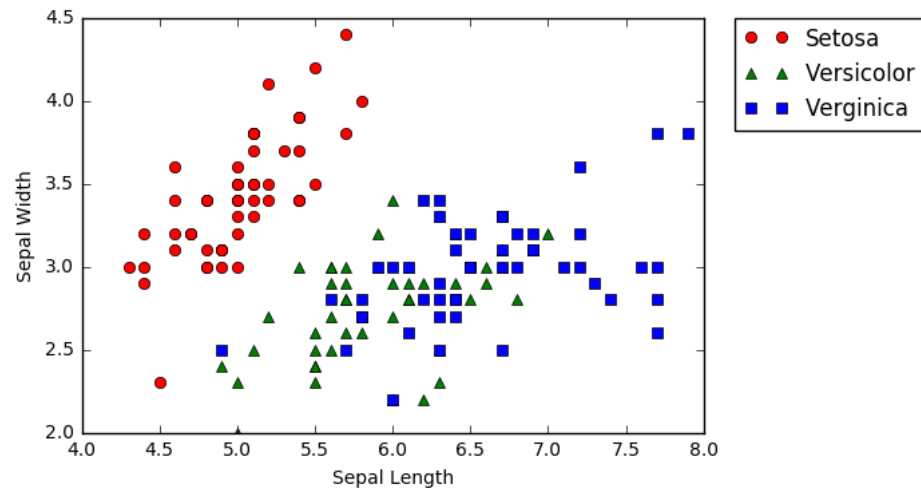


Рисунок 1.7 – Простейший анализ данных по графическому представлению


```

In [74]: import matplotlib as mpl
import matplotlib.pyplot as plt

# Данные из отдельных столбцов
sepal_length = [] # Sepal Length
sepal_width = [] # Sepal Width
petal_length = [] # Petal Length
petal_width = [] # Petal Width

# Выполняем обход всей коллекции data2
for dot in data2:
    sepal_length.append(dot[0])
    sepal_width.append(dot[1])
    petal_length.append(dot[2])
    petal_width.append(dot[3])

# Строим графики по проекциям данных
# Учитываем, что каждые 50 типов ирисов идут последовательно
plt.figure(1)
setosa, = plt.plot(sepal_length[:50], sepal_width[:50], 'ro', label='Setosa')
versicolor, = plt.plot(sepal_length[50:100], sepal_width[50:100], 'g^', label='Versicolor')
virginica, = plt.plot(sepal_length[100:150], sepal_width[100:150], 'bs', label='Virginica')
plt.legend(bbox_to_anchor=(1.05, 1), loc=2, borderaxespad=0.)
plt.xlabel('Sepal Length')
plt.ylabel('Sepal Width')

plt.figure(2)
setosa, = plt.plot(sepal_length[:50], petal_length[:50], 'ro', label='Setosa')
versicolor, = plt.plot(sepal_length[50:100], petal_length[50:100], 'g^', label='Versicolor')
virginica, = plt.plot(sepal_length[100:150], petal_length[100:150], 'bs', label='Virginica')
plt.legend(bbox_to_anchor=(1.05, 1), loc=2, borderaxespad=0.)
plt.xlabel('Sepal Length')
plt.ylabel('Petal Length')

plt.figure(3)
setosa, = plt.plot(sepal_length[:50], petal_width[:50], 'ro', label='Setosa')
versicolor, = plt.plot(sepal_length[50:100], petal_width[50:100], 'g^', label='Versicolor')
virginica, = plt.plot(sepal_length[100:150], petal_width[100:150], 'bs', label='Virginica')
plt.legend(bbox_to_anchor=(1.05, 1), loc=2, borderaxespad=0.)
plt.xlabel('Sepal Length')
plt.ylabel('Petal Width')

plt.show()

```

Рисунок 1.8 – Построение простейшего графика для отображения различных проекций данных

Вывод данного кода представлен на рис. 1.9.

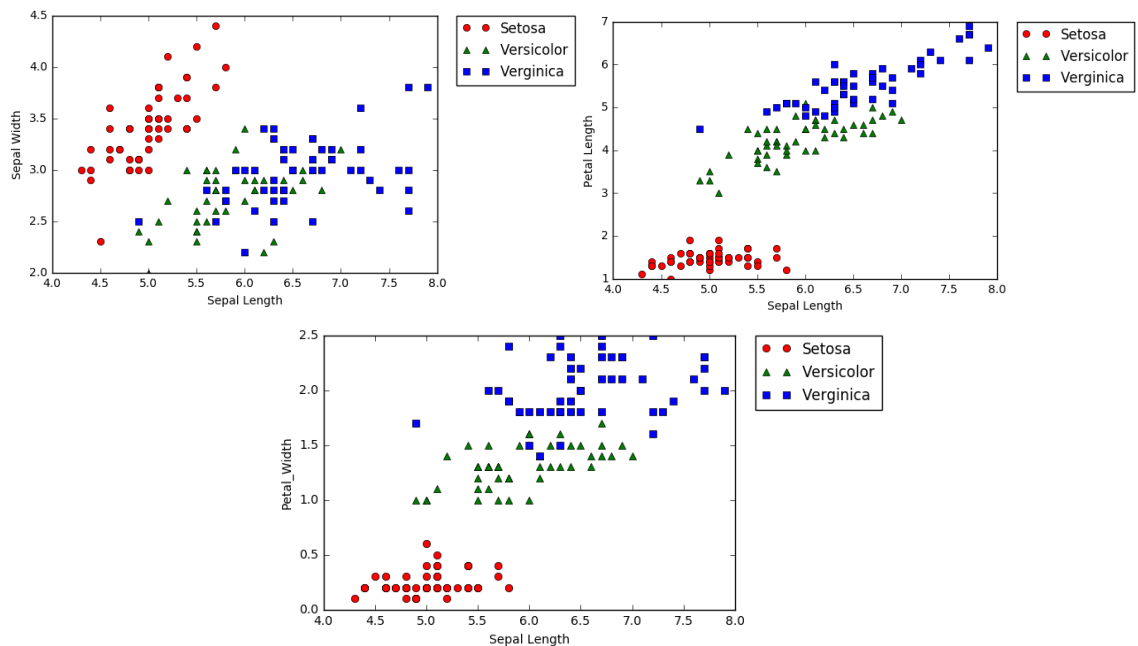


Рисунок 1.9 – Вывод графика для отображения различных проекций данных

Из графиков 1.9 уже хорошо видно, что множество *Setosa* хорошо отделимо, а множества *Versicolor* и *Verginica* представляют собой множества, разделение которых является непростой задачей.

Постройте другие проекции исходных данных. Сколько всего различных проекций можно построить для данного набора данных?

Важные замечания

1. Несмотря на кажущуюся простоту и «понятность» данных в результате визуализации, исследователь не должен делать поспешных выводов (например, было бы ошибочно делать вывод по рис. 1.9 о том, что ирисы *Setosa* те, у которых *petal width* менее 0,75). Следует помнить, что цель первичного исследования данных – получение представления о структуре и природе данных, а не построение модели предсказания, классификации и т.п.

1. В качестве среды разработки используйте языки программирования Python, Java или C#. По согласованию с преподавателем студент может самостоятельно выбрать язык программирования и среду разработки (при этом студенту необходимо критически обосновать свой выбор).

2. При выборе набора данных (*data set*) на ресурсах [9, 10] необходимо согласовать свой выбор с другими студентами группы и преподавателем, так как работа над одинаковыми наборами данных недопустима.

3. В рамках данного лабораторного курса рекомендуется использовать инструментарий Python (библиотеки, среду разработки) для решения поставленных задач.

Индивидуальное задание

1. Подберите набор данных на ресурсах [9, 10] и согласуйте свой выбор с преподавателем. Студент может предложить набор данных в соответствии с тематикой магистерского исследования.

2. Проведите первичный анализ данных. В результате анализа данных студент должен предоставить следующую информацию о наборе данных:

2.1. Описание набора данных, пояснения, позволяющие лучше понять природу данных. Назначение набора данных и возможные модели, которые можно построить на основе данного набора данных (практические задачи, решаемые с использованием данного обучающего набора данных). Описание каждого признака и его тип.

2.2. Форма набора данных: количество элементов набора, количество признаков, количество пропущенных значений, среднее значение отдельных признаков, максимальные и минимальные значения отдельных признаков и прочие показатели. Предположения, которые можно сделать, проведя первичный анализ.

2.3. Графические представления, позволяющие судить о неоднородности исследуемого набора данных. Построение графиков желательно произвести по нескольким проекциям.

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. ВИЗУАЛИЗАЦИЯ ДАННЫХ

Цели и задачи

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

Основные задачи:

- установка и настройка matplotlib, seaborn;
- изучение основных типов графиков библиотеки matplotlib;
- изучение основных типов графиков библиотеки seaborn;
- получение навыков анализа данных по визуальным представлениям данных.

Теоретическое обоснование

Перед выполнением лабораторной работы необходимо ознакомиться с базовыми принципами языка Python, используя следующие источники: [1-5]. Особое внимание необходимо уделить репозитарию [5] с исходными кодами.

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

Перед выполнением индивидуального задания рекомендуется выполнить все пункты учебной задачи.

Учебная задача

Выполним анализ набора данных «Предсказание ухода клиента». Данный набор данных используется в качестве учебного набора при изучении методов прогнозирования. Набор представляет собой данные об активности клиента телекоммуникационной компании (количество часов разговоров, видеозвонков, ночные и дневные разговоры и прочие). Набор данных подходит для обучения моделей логистической регрессии, моделей классификации (CNN, kNN, Logic tree). Набор данных можно получить в репозитории [5] или на портале Kaggle [4].

Рассмотрим основные признаки, представленный в наборе. Загрузим набор данных с использованием pandas и выведем признаки набора данных (рисунок 2.1).

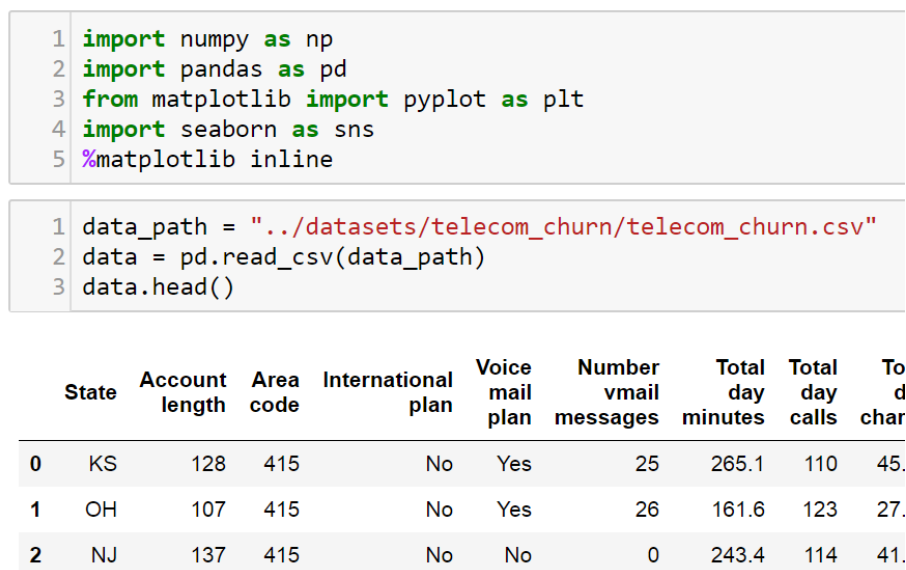


Рисунок 2.1 – Загрузка данных и получение и первичный анализ признаков

Набор данных telecom_churn.csv содержит большое количество признаков. Для детального изучения воспользуемся методом info() класса DataFrame (рисунок 2.2).

```

1 data.info()

<class 'pandas.core.frame.DataFrame'>
RangeIndex: 3333 entries, 0 to 3332
Data columns (total 20 columns):
State                3333 non-null object
Account length       3333 non-null int64
Area code            3333 non-null int64
International plan    3333 non-null object
Voice mail plan       3333 non-null object
Number vmail messages 3333 non-null int64
Total day minutes     3333 non-null float64
Total day calls       3333 non-null int64
Total day charge      3333 non-null float64
Total eve minutes     3333 non-null float64
Total eve calls       3333 non-null int64
Total eve charge      3333 non-null float64
Total night minutes   3333 non-null float64
Total night calls     3333 non-null int64
Total night charge    3333 non-null float64
Total intl minutes    3333 non-null float64
Total intl calls      3333 non-null int64
Total intl charge     3333 non-null float64
Customer service calls 3333 non-null int64
Churn                 3333 non-null bool
dtypes: bool(1), float64(8), int64(8), object(3)
memory usage: 498.1+ KB

```

Рисунок 2.2 – Информация о признаках набора данных

Графики, используемые при анализе данных, делят не по библиотекам, с использованием которых они строятся, а по типам признаков, для анализа которых предназначены графики.

6.1.1. Визуализация количественных признаков

Для представления распределения простого количественного признака подходит обычная гистограмма, содержащаяся во всех библиотеках (рисунок 2.3).

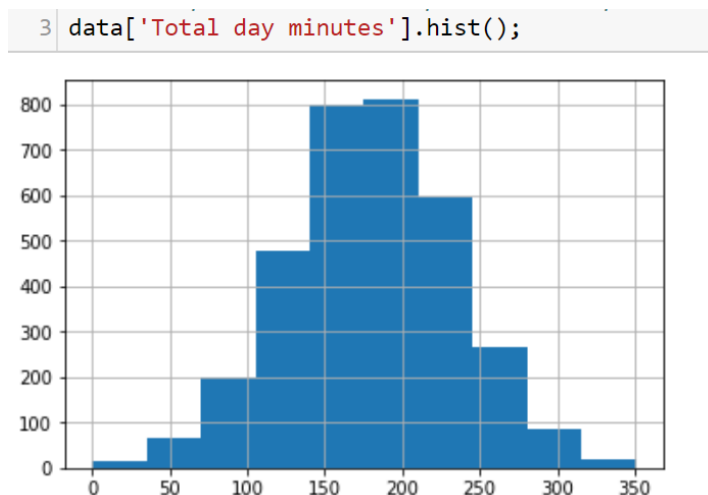
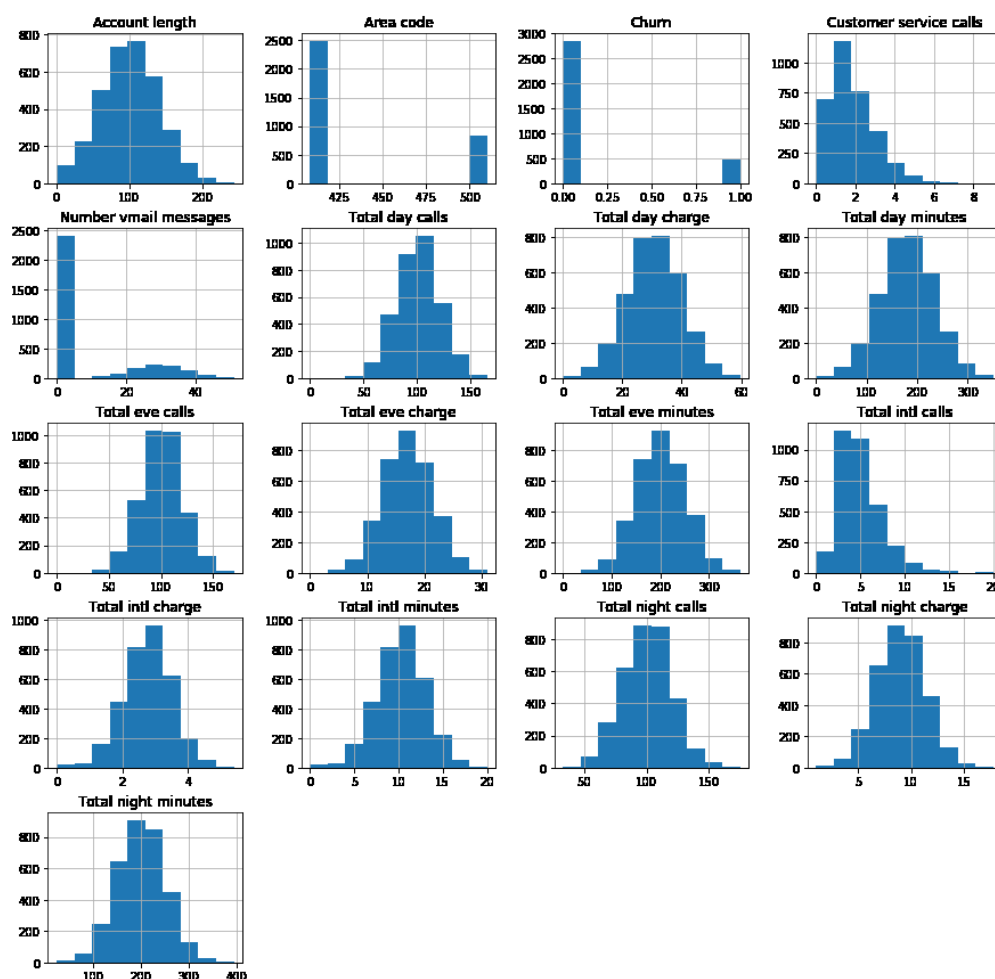


Рисунок 2.3 – Информая о признаках набора данных

Для построения гистограммы вызывается метод **hist()** класса `DataFrame`. На самом деле используется метод из библиотеки `matplotlib`. Метод `hist()` можно использовать для построения гистограмм по нескольким признакам (рисунок 2.4). При этом неколичественные признаки игнорируются.

Рисунок 2.4 – Применение метода `hist()` для визуализации распределения нескольких признаков

Аналогичный тип графика можно получить с использованием `matplotlib` (рисунок 2.5). Если необходимо построить график распределения, аналогичный представленному на рисунке 2.3, то нужно выполнить дополнительные расчеты (рисунок 2.6).


```
1 plt.bar(data.index, data['Total day minutes'])
2 plt.show()
```

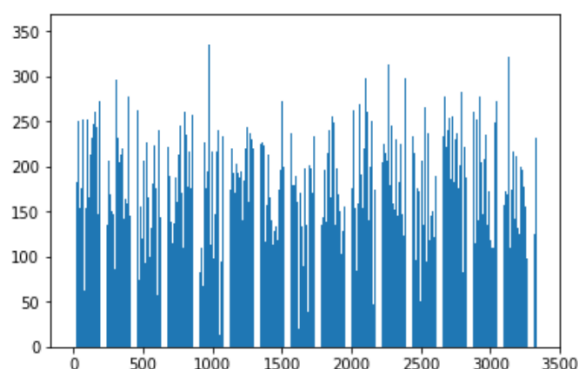


Рисунок 2.5 – Построение гистограммы с использованием matplotlib

```
1 hist = data['Total day minutes'].value_counts()
2 plt.bar(hist.index, hist);
```

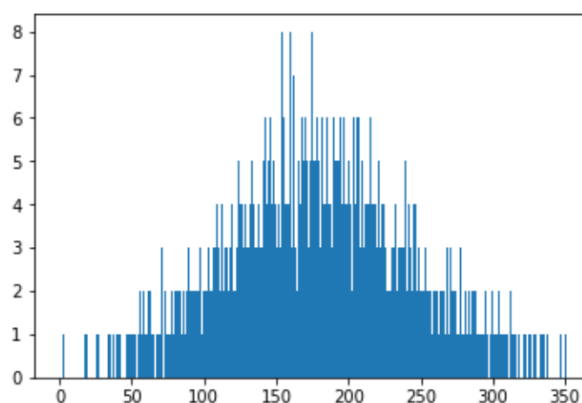


Рисунок 2.6 – Использование matplotlib для представления распределения значений признака

Один из эффективных типов графиков для анализа количественных признаков – это «ящики с усами» (boxplot). На рисунке 2.7 показан код и реализованный график. Для анализа нескольких признаков графики boxplot также эффективны. На рисунке 2.8 представлен код и результат построения графиков для анализа пяти штатов с максимальным объемом дневных звонков.

```
5 sns.boxplot(data['Total day minutes']);
```

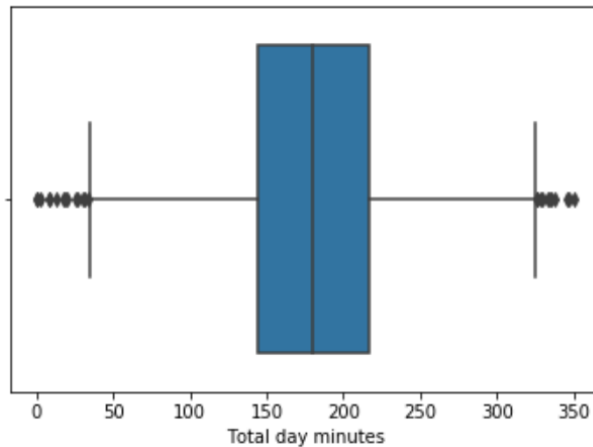


Рисунок 2.7 – График «ящик с усами» для отдельного признака

```
1 top_data = data[['State', 'Total day minutes']]
2 top_data = top_data.groupby('State').sum()
3 top_data = top_data.sort_values('Total day minutes', ascending=False)
4 top_data = top_data[:5].index.values
5 sns.boxplot(y='State',
6             x='Total day minutes',
7             data=data[data.State.isin(top_data)], palette='Set3');
```

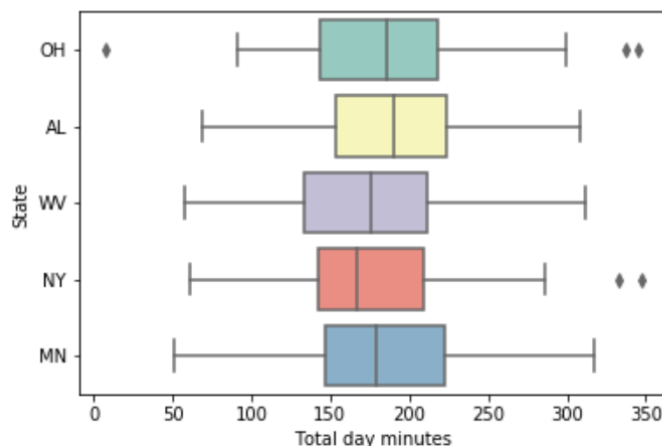


Рисунок 2.8 – Использование boxplot для анализа признака для пяти штатов

График boxplot состоит из коробки, усов и точек. Коробка показывает интерквартильный размах распределения, то есть соответственно 25% (первая квартиль, $Q1$) и 75% ($Q3$) перцентили. Черта внутри коробки обозначает медиану распределения (можно получить с использованием метода `median()` в `pandas` и `numpy`). Усы отображают весь разброс точек кроме выбросов, то есть минимальные и максимальные значения, которые попадают в промежуток $(Q1 - 1,5 \cdot IQR, Q3 + 1,5 \cdot IQR)$, где $IQR = Q3 - Q1$ – интерквартильный размах. Точками на графике обозначаются выбросы (outliers), то есть те

значения, которые не вписываются в промежуток значений, заданный усами графика (рисунок 2.9).

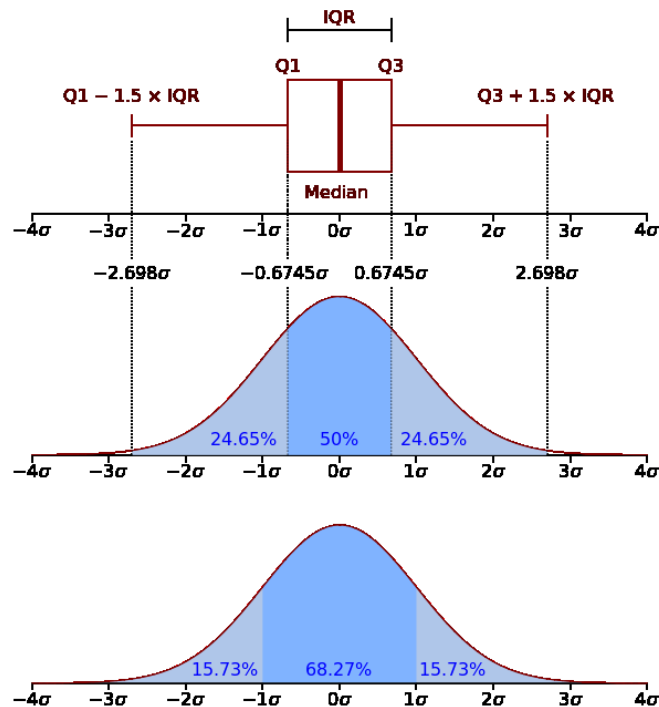


Рисунок 2.9 – Структура графика типа «ящик с усами»

6.1.2. Категориальные признаки

Типичным категориальным признаком в анализируемом наборе данных является «Штат» (State). Под категориальный признак подходит также «Отказ» (Churn) (хотя он является логическим). На рисунке 2.10 представлены графики типа `countplot()` из библиотеки `seaborn`, которые строят гистограммы, но не по сырым данным, а по рассчитанному количеству разных значений признака.

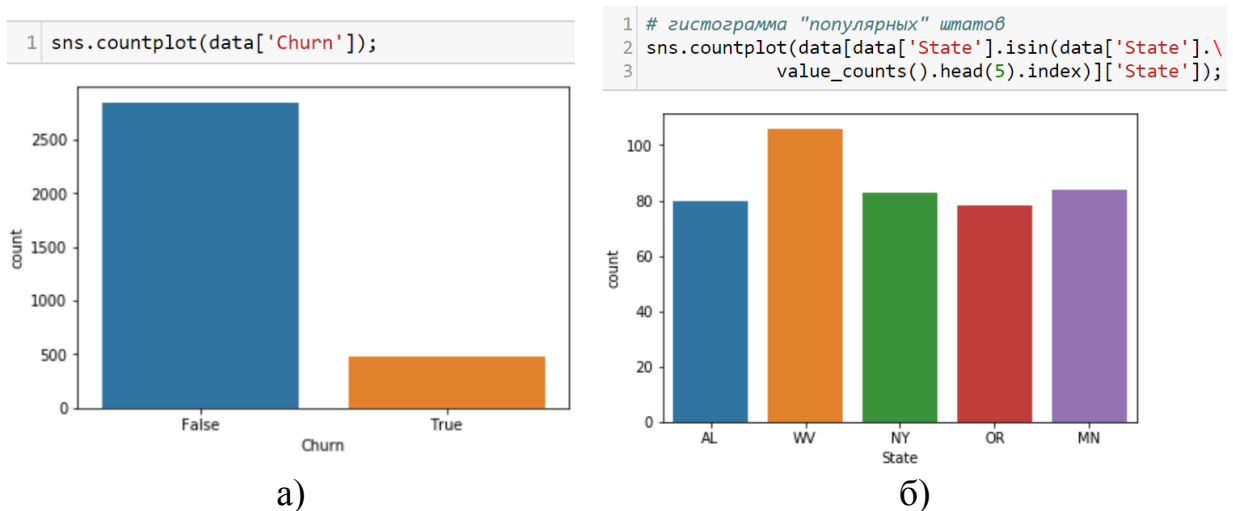


Рисунок 2.10 – График countplot: а) визуализация распределения признака Churn; б) визуализация пяти популярных штатов

6.1.3. Визуализация соотношения количественных признаков

Одним из вариантов визуализации соотношения количественных признаков является диаграмма по нескольким признакам (рисунки 2.4, 2.8). Рассмотрим пример демонстрирующий сравнение распределений показателей, связанных с финансовыми затратами клиентов. Упрощенно, можно сказать, что это все показатели, содержащие подстроку «charge» в имени показателя. На рисунке 2.11 представлен код для отбора требуемых показателей.

```
1 # Отбор числовых признаков, содержащих слово 'charge'
2 feats = [f for f in data.columns if 'charge' in f]
3 feats

['Total day charge',
 'Total eve charge',
 'Total night charge',
 'Total intl charge']
```

Рисунок 2.11 – Отбор показателей, связанных с затратами клиентов

После отбора интересующих показателей можно построить диаграммы для сравнения (рисунок 2.12).

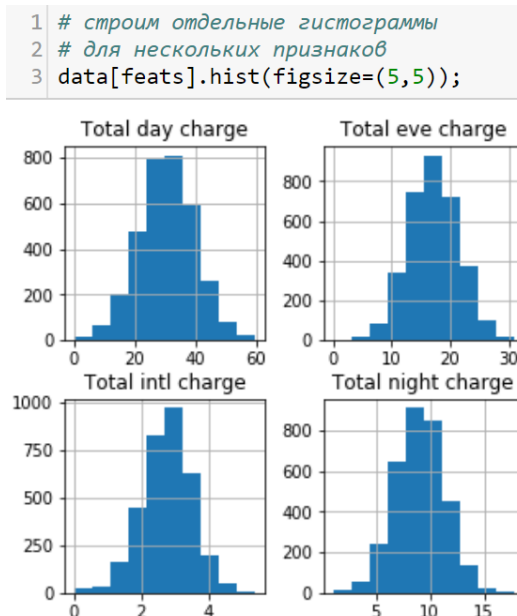


Рисунок 2.12 – Диаграммы для сравнения распределения числовых показателей

Часто используют попарное сравнение признаков для обеспечения широкого взгляда на набор данных (рисунок 2.13). На диагональных графиках рисунка 2.13 представлены гистограммы распределения отдельного признака, на внедиагональных позициях – попарные распределения.

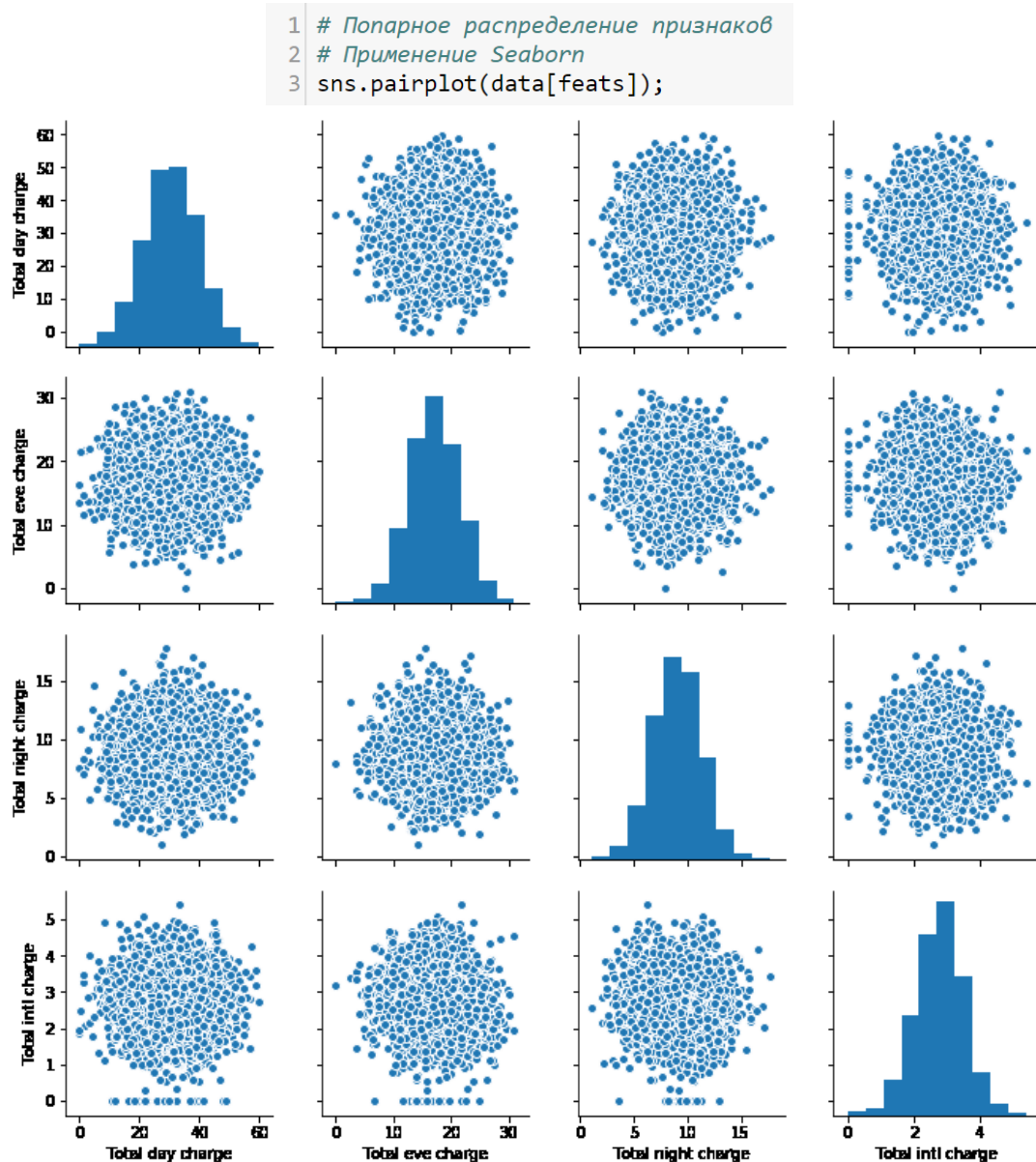


Рисунок 2.13 – Попарное распределение признаков

Можно реализовать юолее сложные графики. Например, если требуется добавить к существующим признакам, целевой признак Churn (количество отказов) и раскрасить разные типы элементов, то можно воспользоваться попарными распределениями, но с отображением подмножеств отказов (рисунок 2.14).

До сих пор использовались возможности библиотеки seaborn, а также методы pandas (которые производят визуализацию, обращаясь к библиотеке matplotlib). Библиотека matplotlib наиболее известная и широко применяемая при анализе данных в рамках стека технологий python.

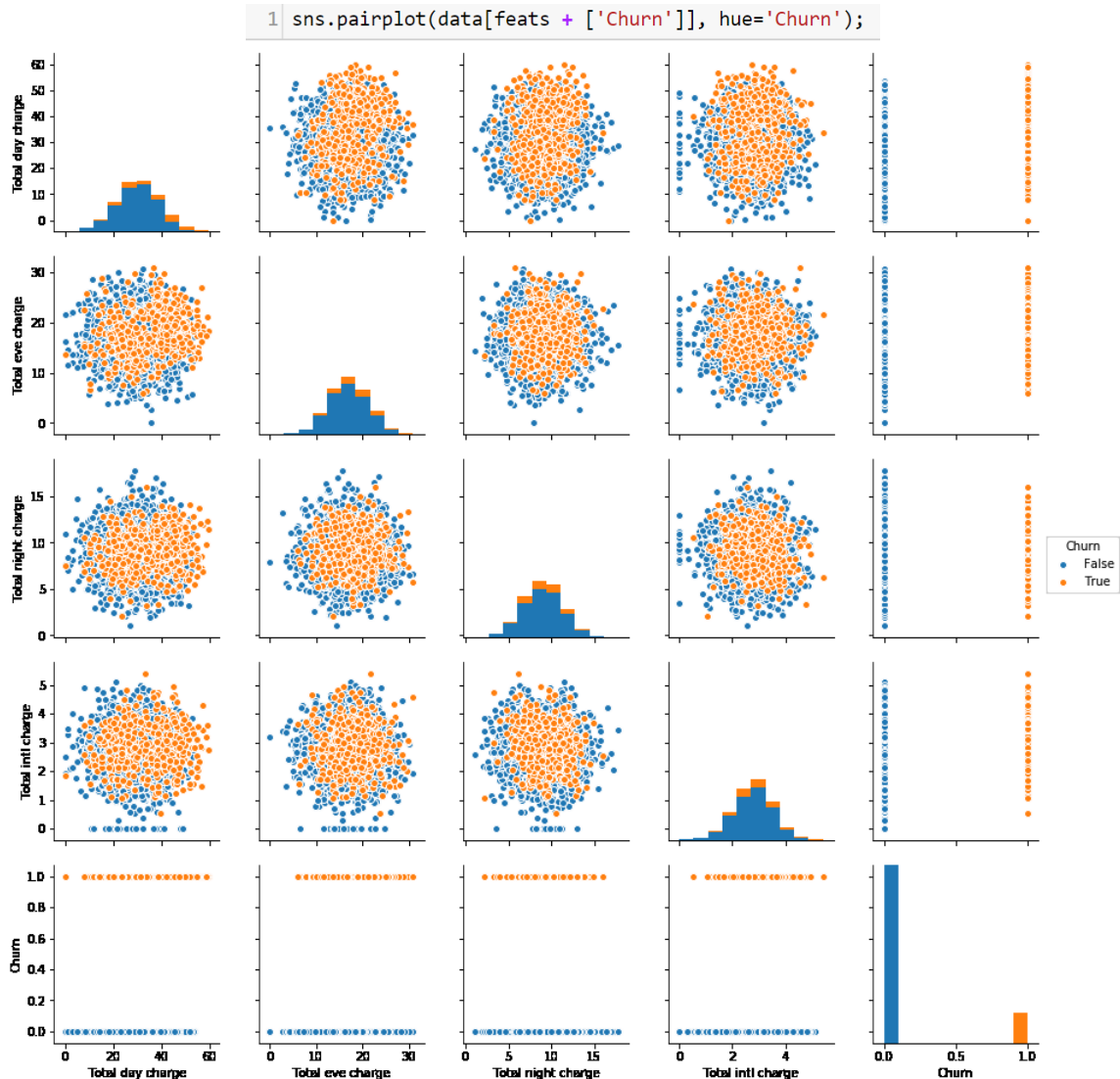


Рисунок 2.14 – Попарное распределение признаков с визуализацией отказов

На рисунке 2.15 показан пример использования графика scatter библиотеки matplotlib, предназначенного для вывода множества точек.


```

1 plt.scatter(data['Total day charge'],
2             data['Total intl charge'],
3             color='lightblue', edgecolors='blue')
4 plt.xlabel('Дневные начисление')
5 plt.ylabel('Международн. начисление')
6 plt.title('Распределение по 2 признакам');

```

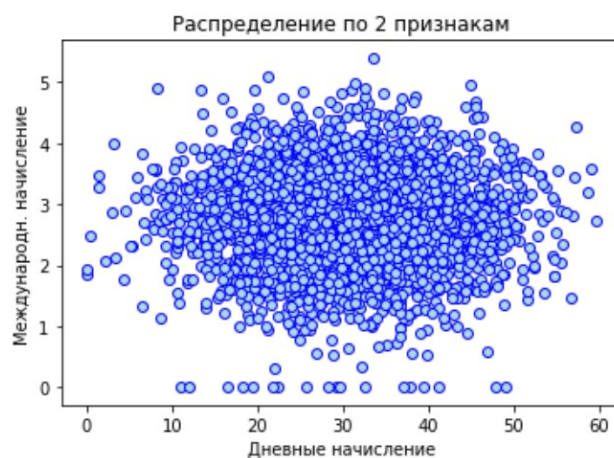


Рисунок 2.15 – График scatter библиотеки matplotlib

На рисунке 2.16 показан пример более тонкой настройки параметров графика.

```

1 # Раскрашивание данных
2 # Цвет в зависимости от ухода клиента
3 c = data['Churn'].map({False: 'lightblue', True: 'orange'})
4 edge_c = data['Churn'].map({False: 'blue', True: 'red'})
5 # Настройка графика
6 plt.scatter(data['Total day charge'], data['Total intl charge'],
7            color=c, edgecolors=edge_c
8            )
9 plt.xlabel('Дневные начисление')
10 plt.ylabel('Международн. начисление');

```

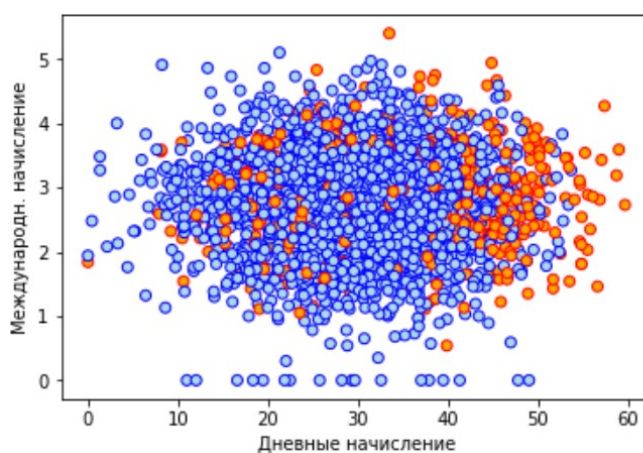


Рисунок 2.16 – Настройка графика: цвет точки зависит от целевого значения признака

График на рисунке 2.16 можно построить различными способами, например, можно добавлять множества точек отдельными подмножествами, указывая параметры визуализации для каждого подмножества (рисунок 2.17).

```

4 # Ушедшие клиенты
5 data_churn = data[data['Churn']]
6 # Оставшиеся клиенты
7 data_loyal = data[~data['Churn']]
8
9 plt.scatter(data_churn['Total day charge'],
10            data_churn['Total intl charge'],
11            color='orange',
12            edgecolors='red',
13            label='Ушли')
14
15 plt.scatter(data_loyal['Total day charge'],
16            data_loyal['Total intl charge'],
17            color='lightblue',
18            edgecolors='blue',
19            label='Остались')
20
21 plt.xlabel('Дневные начисление')
22 plt.ylabel('Международн. начисление')
23 plt.title('Распределение клиентов')
24 plt.legend();

```



а)

б)

Рисунок 2.17 – Построение отдельных подмножеств с легендой; а) исходный код; б) полученный график

В реальных задачах машинного обучения при первичном анализе данных необходимо выявить корреляции признаков обучающей выборки. В пакете Pandas имеется встроенный инструмент для этого – метод **corr()** класса **DataFrame**. На рисунке 2.18 показан фрагмент вывода этой функции.

```

1 # Применяется функция corr() из Pandas
2 data.corr()

```

	Account length	Area code	Number vmail messages	Total day minutes	Total day calls	Total day charge	Total eve minutes	Total
Account length	1.000000	-0.012463	-0.004628	0.006216	0.038470	0.006214	-0.006757	0.01
Area code	-0.012463	1.000000	-0.001994	-0.008264	-0.009646	-0.008264	0.003580	-0.01
Number vmail messages	-0.004628	-0.001994	1.000000	0.000778	-0.009548	0.000776	0.017562	-0.00
Total day minutes	0.006216	-0.008264	0.000778	1.000000	0.006750	1.000000	0.007043	0.01

Рисунок 2.18 – Определение коррелирующих признаков набора данных

Полученная матрица имеет размер 17×17 . Это незначительный размер (в реальных задачах машинного обучения размеры матриц корреляции имеют порядки $10^6 - 10^{10}$ и более), но даже для матрицы рассматриваемого набора

данных проанализировать корреляцию признаков вручную – трудоемкая задача. Например, можно использовать скрипты, для выделения больших коэффициентов корреляции. Но лучше использовать специальный тип графика – heatmap (рисунок 2.19).

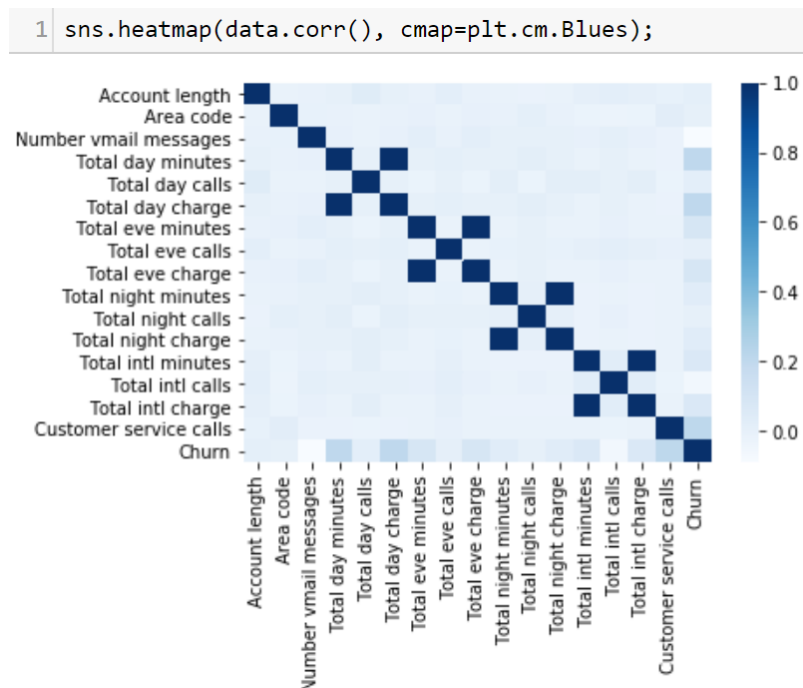


Рисунок 2.19 – Визуализация матрица корреляции с использованием графика типа heatmap

Коррелирующие признаки обычно удаляются и не рассматриваются в процессе обучения.

Важные замечания

1. Статья о типах графиков при первичном анализе данных:
<https://medium.com/open-machine-learning-course/open-machine-learning-course-topic-2-visual-data-analysis-in-python-846b989675cd>
2. В качестве среды разработки используйте языки программирования Python, Java или C#. По согласованию с преподавателем студент может самостоятельно выбрать язык программирования и среду разработки (при этом студенту необходимо критически обосновать свой выбор).

2. При выборе набора данных (data set) на ресурсах [3, 4] необходимо согласовать свой выбор с другими студентами группы и преподавателем, так как работа над одинаковыми наборами данных недопустима.

3. В рамках данного лабораторного курса рекомендуется использовать инструментарий Python (библиотеки, среду разработки) для решения поставленных задач.

Индивидуальное задание

1. Подберите набор данных на ресурсах [3, 4] и согласуйте свой выбор с преподавателем. Студент может предложить набор данных в соответствии с тематикой магистерского исследования.

2. Проведите первичный анализ данных. Особое внимание следует уделить графическому представлению распределений признаков, визуализации взаимосвязей, позволяющие судить о наборе данных. Построение графиков желательно произвести по нескольким проекциям. При анализе данных использовать как можно более разнообразные типы графиков.

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. МЕТРИЧЕСКИЕ МЕТОДЫ КЛАССИФИКАЦИИ

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

Перед выполнением лабораторной работы необходимо ознакомиться с теорией построения метрических классификаторов, используя следующие источники: [1-5]. Особое внимание необходимо уделить репозитарию [5] с исходными кодами.

Оборудование и материалы

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

Методика и порядок выполнения работы

Перед выполнением индивидуального задания рекомендуется выполнить все пункты учебной задачи.

Учебная задача

В рамках данной задачи рассматривается построение классификатора с использованием метода ближайших соседей. В качестве набора данных используются данные об ирисах Фишера.

В рамках данной лабораторной работы рекомендуется использование библиотеки `pandas`. `Pandas` – это библиотека Python, предоставляющая широкие возможности для анализа данных. Данные, с которыми работают специалисты Data Science, часто хранятся в табличном формате (`.csv`, `.tsv`, `.xlsx`, ...). С помощью библиотеки `Pandas` данные удобно загружать, обрабатывать и анализировать с помощью SQL-подобных запросов. `Pandas` предоставляет широкие возможности визуального анализа табличных данных в связке с библиотеками `Matplotlib` и `Seaborn`.

Основными структурами данных в `Pandas` являются классы `Series` и `DataFrame`. Первый из них представляет собой одномерный индексированный массив данных некоторого фиксированного типа. Второй – это двумерная структура данных, представляющая собой таблицу, каждый столбец которой содержит данные одного типа. Можно представлять её как словарь объектов типа `Series`. Структура `DataFrame` отлично подходит для представления реальных данных: строки соответствуют признаковым описаниям отдельных объектов, а столбцы соответствуют признакам.

1. На рис. 3.1 представлен код в Python Notebook для загрузки исходного набора данных.

```
In [8]: import pandas as pd
import numpy as np

data_source = 'iris.data'
d = pd.read_table(data_source, delimiter=',')
d.head()
```

Out[8]:

	5.1	3.5	1.4	0.2	Iris-setosa
0	4.9	3.0	1.4	0.2	Iris-setosa
1	4.7	3.2	1.3	0.2	Iris-setosa
2	4.6	3.1	1.5	0.2	Iris-setosa
3	5.0	3.6	1.4	0.2	Iris-setosa
4	5.4	3.9	1.7	0.4	Iris-setosa

Рисунок 3.1 – Использование `pandas` для загрузки данных

Следует обратить внимание, что первая строка набора данных интерпретировалась как шапка таблицы (название столбцов). Данную неточность необходимо исправить следующим образом (рис. 3.2). В таком случае столбцы получают порядковые номера в качестве названий столбцов.

```
In [11]: import pandas as pd
import numpy as np

data_source = 'iris.data'
d = pd.read_table(data_source, delimiter=',', header=None)
d.head()
```

Out[11]:

	0	1	2	3	4
0	5.1	3.5	1.4	0.2	Iris-setosa
1	4.9	3.0	1.4	0.2	Iris-setosa
2	4.7	3.2	1.3	0.2	Iris-setosa
3	4.6	3.1	1.5	0.2	Iris-setosa
4	5.0	3.6	1.4	0.2	Iris-setosa

Рисунок 3.2 – Добавление шапки DataFrame по умолчанию

Исследователь также может дать символьные имена столбцам при загрузке (рис. 3.3).

```
In [19]: import pandas as pd
import numpy as np

data_source = 'iris.data'
d = pd.read_table(data_source, delimiter=',',
                  header=None,
                  names=['sepal_length', 'sepal_width',
                        'petal_length', 'petal_width', 'answer'])
d.head()
```

Out[19]:

	sepal_length	sepal_width	petal_length	petal_width	answer
0	5.1	3.5	1.4	0.2	Iris-setosa
1	4.9	3.0	1.4	0.2	Iris-setosa
2	4.7	3.2	1.3	0.2	Iris-setosa
3	4.6	3.1	1.5	0.2	Iris-setosa
4	5.0	3.6	1.4	0.2	Iris-setosa

Рисунок 3.3 – Добавление шапки DataFrame с символьными именами столбцов

2. После загрузки данных можно визуализировать полученный набор данных. Для визуализации будем использовать библиотеку seaborn (рис. 3.4).

```
In [18]: import seaborn as sb
          %matplotlib inline
          sb.pairplot(d)
```

Рисунок 3.4 – Добавление шапки DataFrame по умолчанию

В результате будет выведен график, отображающий распределение объектов попарно по различным признакам (рис. 3.5).

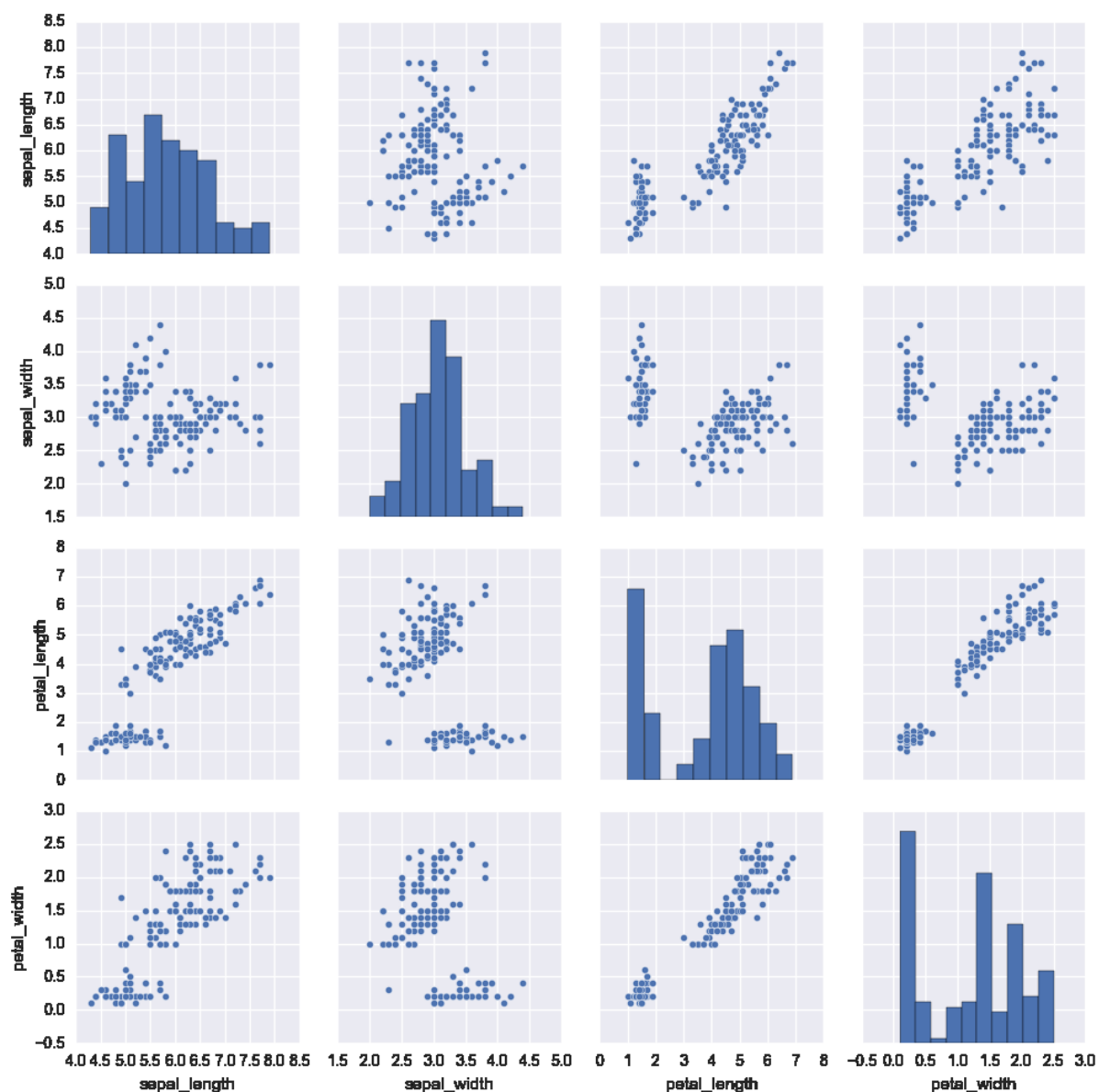


Рисунок 3.5 – Попарное признаковое распределение ирисов

На графике попарного распределения видно преимущество символического обозначения столбцов – график легче интерпретировать. Отдельные классы не отмечаются различными цветами, но видно, что на отдельных подграфиках множества точек разделены. Следует обратить внимание на подграфики, расположенные по диагонали. Подумайте, что они отображают?

3. Для придания отдельным классам своих цветов необходимо указать, по какому признаку разделяются точки (рис. 3.6).

```
In [23]: import seaborn as sb
          %matplotlib inline
          sb.pairplot(d, hue='answer')
```

Рисунок 3.6 – Построение графика с указанием признака отдельных классов

Результат представлен на рис. 3.7.

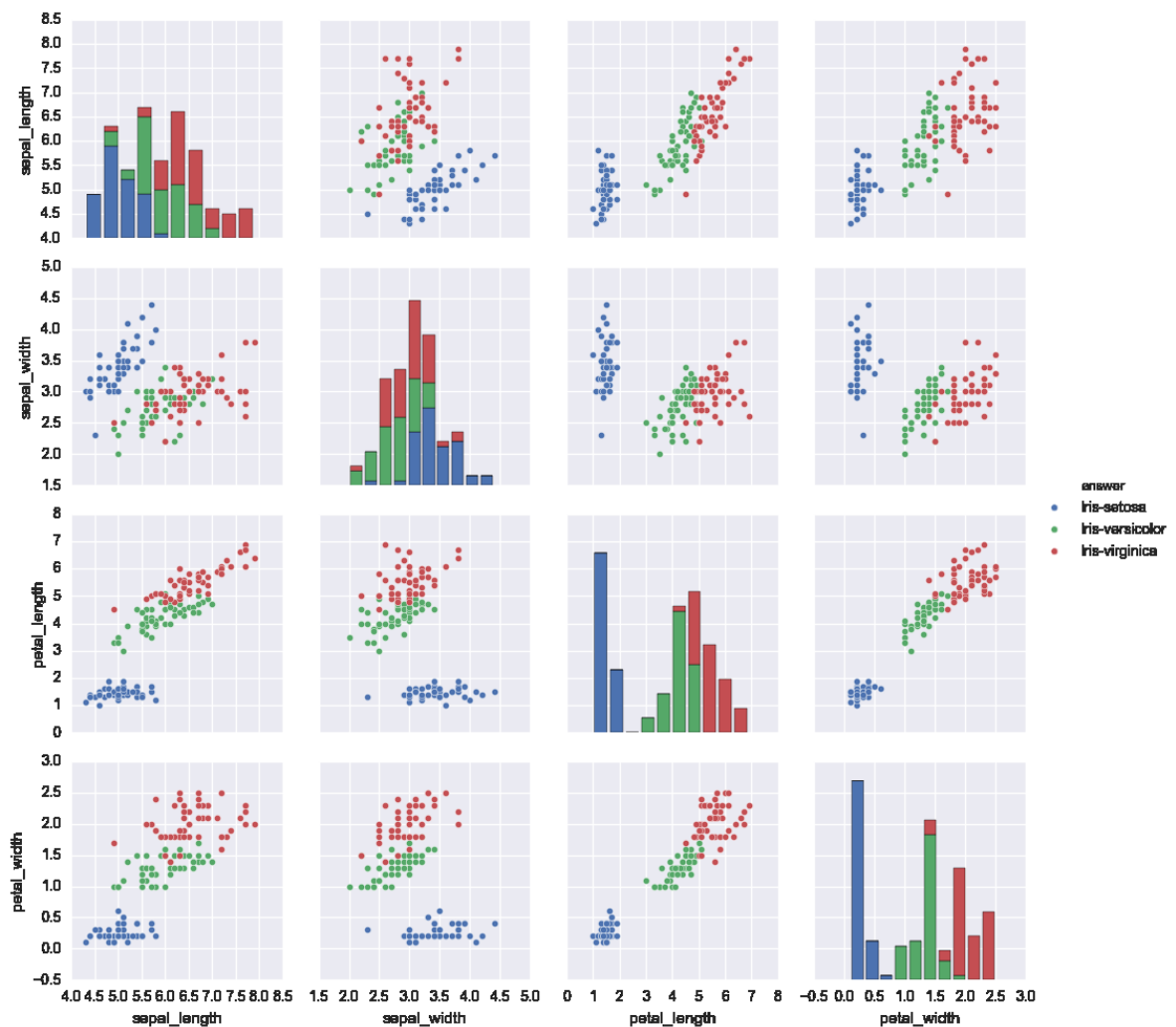


Рисунок 3.7 – Попарное признаковое распределение ирисов с разделением на классы

Можно изменить маркеры каждого класса. Для этого необходимо использовать код: `sb.pairplot(d, hue='answer', markers=["o", "s", "D"])`.

4. Перейдем к построению модели. Модель метрической классификации должна обеспечивать следующий алгоритм работы: пользователь вводит новое признаковое описание объекта (объект отсутствует в обучающей выборке), а алгоритм классификации относит новый объект к одному из классов ирисов.

5. Существует несколько вариаций метода ближайших соседей. Каждая модель предполагает наличие различных параметров для оптимизации. Воспользуемся библиотекой `scikit` для построения классификатора (рис. 3.8).

```
from sklearn.neighbors import KNeighborsClassifier
from sklearn.metrics import accuracy_score

X_train = d[['sepal_length', 'sepal_width', 'petal_length', 'petal_width']]
y_train = d['answer']

K = 3

# Создание и настройка классификатора
knn = KNeighborsClassifier(n_neighbors=K)
# построение модели классификатора (процедура обучения)
knn.fit(X_train, y_train)

# Использование классификатора
# Объявление признаков объекта
X_test = np.array([[1.2, 1.0, 2.8, 1.2]])
# Получение ответа для нового объекта
target = knn.predict(X_test)
print(target)
```

```
['Iris-versicolor']
```

Рисунок 3.8 – Основные этапы решения задачи классификации методом ближайших соседей с использованием библиотеки `scikit`

Для представленного объекта `X_test` попробуйте поменять значение признаков и проследите за изменением значения выходного класса.

6. Модель построена и выдает ответ для новых (отсутствующих в исходной выборке) объектов. Но, анализируя код на рис. 3.8, следует отметить следующие недостатки такого подхода:

- в качестве количества ближайших соседей выбрано значение $K=3$, выбор данного значения не обосновывается, но в данном методе именно данный параметр должен оптимизироваться;
- отсутствует какое-либо графическое представление модели, нет визуализации процесса принятия решения.

Исправим данные недостатки.

7. Займемся обоснованием выбора оптимального значения количества ближайших соседей. Для этого будем использовать простейшую оценку качества hold-out (рис. 3.9).

```
from sklearn.model_selection import train_test_split

X_train, X_holdout, y_train, y_holdout = \
    train_test_split(d[['sepal_length', 'sepal_width',
                        'petal_length', 'petal_width']],
                    d['answer'],
                    test_size=0.3,
                    random_state=17)

knn = KNeighborsClassifier(n_neighbors=3)
knn.fit(X_train, y_train)
knn_pred = knn.predict(X_holdout)
accur = accuracy_score(y_holdout, knn_pred)
print('accuracy: ', accur)

accuracy:  0.977777777778
```

Рисунок 3.9 – Оценка точности классификатора с использованием методики hold-out

В качестве эксперимента попробуйте поменять значение количества соседей и рассмотрите изменение точности классификатора.

8. Еще одна оценка качества – cross validation (CV) error. На рис. 3.10 показан алгоритм получения оценки точности классификации CV и процедура выбора оптимального значения количества соседей в алгоритме kNN на основе данной оценки.

```

from sklearn.model_selection import cross_val_score
import matplotlib.pyplot as plt

# Значения параметра K
k_list = list(range(1,50))
# Пустой список для хранения значений точности
cv_scores = []
# В цикле проходим все значения K
for k in k_list:
    knn = KNeighborsClassifier(n_neighbors=k)
    scores = cross_val_score(knn, d.ix[ : , 0:4 ], d['answer'], cv=10, scoring='accuracy')
    cv_scores.append(scores.mean())

# Вычисляем ошибку (misclassification error)
MSE = [1-x for x in cv_scores]

# Строим график
plt.plot(k_list, MSE)
plt.xlabel('Количество соседей (K)');
plt.ylabel('Ошибка классификации (MSE)')
plt.show()

# Ищем минимум
k_min = min(MSE)

# Пробуем найти прочие минимумы (если их несколько)
all_k_min = []
for i in range(len(MSE)):
    if MSE[i] <= k_min:
        all_k_min.append(k_list[i])

# печатаем все K, оптимальные для модели
print('Оптимальные значения K: ', all_k_min)

```

Рисунок 3.10 – Реализация процедуры выбора оптимального параметра на основе cross validation error

Вывод для данного кода представлен на рис. 3.11.

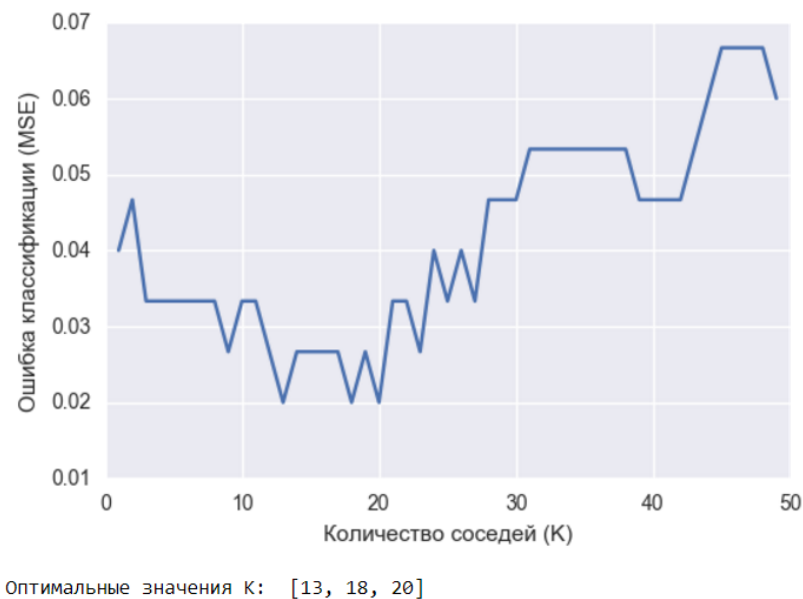


Рисунок 3.11 – Визуализация выбора оптимального параметра на основе cross validation error

Важные замечания

1. При выборе набора данных (data set) на ресурсах [3, 4] необходимо согласовать свой выбор с другими студентами группы и преподавателем с целью недопустимости выбора одинаковых вариантов.

2. В рамках данного лабораторного курса рекомендуется использовать инструментарий Python (библиотеки, среду разработки) для решения поставленных задач.

3. При выборе набора данных следует отдавать предпочтение тем наборам, которые имеют следующие характеристики: содержат не более 5 признаков на объект; все признаки – числовые; желательно отсутствие пропусков в данных.

Индивидуальное задание

1. Студент самостоятельно выбирает набор данных на ресурсах [3, 4] для построения классификатора с использованием метода ближайших соседей и согласует свой выбор с преподавателем.

2. Выполните построение модели классификации на основе метода ближайших соседей. В ходе решения задачи необходимо решить следующие подзадачи:

2.1 Построение классификатора с заданием K (количества ближайших соседей) пользователем;

2.2 Вычисление оценки hold-out для различных значений K , а также для различных долей обучающей и тестирующей подвыборок;

2.3 Вычисление оценки cross validation для различных значений K , а также для различных значений fold (количества подмножеств при кросс-валидации).

2.4 Вычислите оптимальные значения K . Обоснуйте свой выбор. Продемонстрируйте использование полученного классификатора.

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. Поясните особенности основных методов метрической классификации: метод ближайшего соседа, метод k ближайших соседей.
2. Поясните основные принципы и этапы реализации метода kNN .
3. Поясните принцип выбора количества соседних объектов, по которым определяется принадлежность целевого объекта к результирующему классу.
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. ЛОГИЧЕСКИЕ МЕТОДЫ КЛАССИФИКАЦИИ

Цели и задачи

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

Основные задачи:

- освоение технологии внедрения алгоритмов на основе решающих списков в приложения;
- освоение технологии внедрения алгоритмов на основе решающих деревьев в приложения;
- изучение параметров логической классификации;
- освоение модификаций логических методов классификации.

Теоретическое обоснование

Перед выполнением лабораторной работы необходимо ознакомиться с теорией построения логических классификаторов, 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 программных продуктов для практической реализации задач лабораторной работы): MS Visual Studio 2013 и выше; среда разработки Java, интерпретатор Python (Anaconda).

Указания по технике безопасности

Студенты должны следовать общепринятой технике безопасности для пользователей персональных компьютеров. Не следует самостоятельно

производить ремонт технических средств, установку и удаление программного обеспечения. В случае обнаружения неисправностей необходимо сообщить об этом администратору компьютерного класса (обслуживающему персоналу лаборатории).

Методика и порядок выполнения работы

Учебная задача

В рамках учебной задачи необходимо произвести построение классификатора на основе логического дерева. В качестве набора данных используется набор данных об ирисах Фишера.

1. Подключим библиотеки, которые потребуются для загрузки и первичного анализа данных (рис. 4.1).

```
import numpy as np
import pandas as pd

%matplotlib inline

import seaborn as sns
from matplotlib import pyplot as plt

data_source = 'iris.data'
d = pd.read_table(data_source, delimiter=',',
                  header=None,
                  names=['sepal_length', 'sepal_width',
                        'petal_length', 'petal_width', 'answer'])
dX = d.ix[ : , 0:4 ]
dy = d['answer']
print(dX.head())
print(dy.head())
```

	sepal_length	sepal_width	petal_length	petal_width
0	5.1	3.5	1.4	0.2
1	4.9	3.0	1.4	0.2
2	4.7	3.2	1.3	0.2
3	4.6	3.1	1.5	0.2
4	5.0	3.6	1.4	0.2
0	Iris-setosa			
1	Iris-setosa			

Рисунок 4.1 – Использование pandas для загрузки набора данных

2. Для построения дерева классификации воспользуемся специальным классом `sklearn.tree.DecisionTreeClassifier`. Оценим точность модели методом hold-out (рис. 4.2). Следует обратить внимание, что если в

методе ближайших соседей производилась оптимизация по одному параметру K – количеству ближайших соседей, то при создании модели `DecisionTreeClassifier` необходимо указать два параметра: максимальную глубину дерева (`max_depth`) и количество признаков разделения дерева (`max_features`).

```
from sklearn.tree import DecisionTreeClassifier
from sklearn.model_selection import train_test_split
from sklearn.metrics import accuracy_score

# Подмножества для hold-out
X_train, X_holdout, y_train, y_holdout = \
train_test_split(dX, dy, test_size=0.3, random_state=12)

# Обучение модели
tree = DecisionTreeClassifier(max_depth=5,
                             random_state=21,
                             max_features=2)

tree.fit(X_train, y_train)

# Получение оценки hold-out
tree_pred = tree.predict(X_holdout)
accur = accuracy_score(y_holdout, tree_pred)
print(accur)
```

0.977777777778

Рисунок 4.2 – Обучение модели классификации и оценка ее точности методом hold-out

3. Произведем оценку точности модели по методу cross validation (рис. 4.3), а также сделаем выводы об оптимальном значении параметра `max_depth`.


```

from sklearn.model_selection import cross_val_score

# Значения параметра max_depth
d_list = list(range(1,20))
# Пустой список для хранения значений точности
cv_scores = []
# В цикле проходим все значения K
for d in d_list:
    tree = DecisionTreeClassifier(max_depth=d,
                                  random_state=21,
                                  max_features=2)
    scores = cross_val_score(tree, dX, dy, cv=10, scoring='accuracy')
    cv_scores.append(scores.mean())

# Вычисляем ошибку (misclassification error)
MSE = [1-x for x in cv_scores]

# Строим график
plt.plot(d_list, MSE)
plt.xlabel('Макс. глубина дерева (max_depth)');
plt.ylabel('Ошибка классификации (MSE)')
plt.show()

# Ищем минимум
d_min = min(MSE)

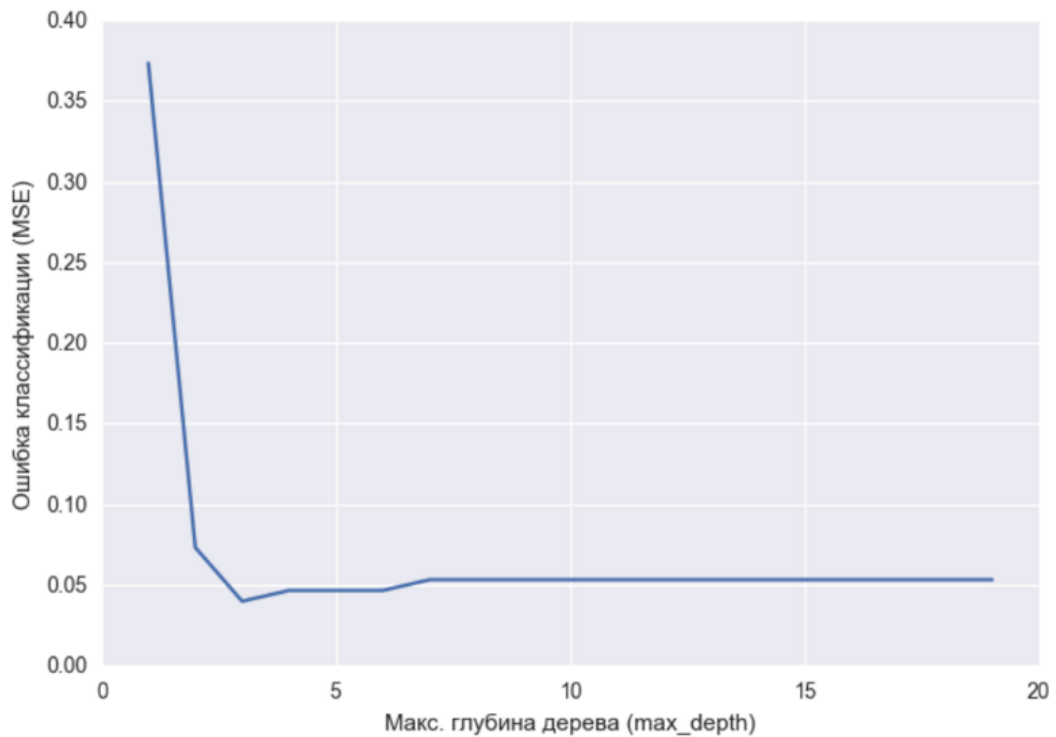
# Пробуем найти прочие минимумы (если их несколько)
all_d_min = []
for i in range(len(MSE)):
    if MSE[i] <= d_min:
        all_d_min.append(d_list[i])

# печатаем все K, оптимальные для модели
print('Оптимальные значения max_depth: ', all_d_min)

```

Рисунок 4.3 – Оценка точности модели методом cross validation и нахождение оптимального значения параметра max_depth

В результате работы данного кода будет получен вывод (рис. 4.4).



Оптимальные значения max_depth: [3]

Рисунок 4.4 – Вывод зависимости значения MSE от параметра max_depth

4. Оптимальное значение параметра max_depth модели получено, но в модели присутствует еще один параметр max_features, который был установлен в значение 2 (не изменялся и не оптимизировался). Для проведения cross validation по всем параметрам воспользуемся классом GridSearchCV пакета sklearn.model_selection (рис. 4.5).

```

from sklearn.model_selection import GridSearchCV, cross_val_score
from sklearn import tree

dtc = DecisionTreeClassifier(max_depth=10, random_state=21, max_features=2)

tree_params = { 'max_depth': range(1,20), 'max_features': range(1,4) }
tree_grid = GridSearchCV(dtc, tree_params, cv=10, verbose=True, n_jobs=-1)
tree_grid.fit(dX, dy)

print('\n')
print('Лучшее сочетание параметров: ', tree_grid.best_params_)
print('Лучшие баллы cross validation: ', tree_grid.best_score_)

# Генерируем графическое представление дерева
tree.export_graphviz(tree_grid.best_estimator_,
                     feature_names=dX.columns,
                     class_names=dy.unique(),
                     out_file='iris_tree.dot',
                     filled=True, rounded=True)

```

Fitting 10 folds for each of 57 candidates, totalling 570 fits

```

Лучшее сочетание параметров: {'max_features': 2, 'max_depth': 3}
Лучшие баллы cross validation: 0.96

```

```
[Parallel(n_jobs=-1)]: Done 570 out of 570 | elapsed: 3.9s finished
```

Рисунок 4.5 – Нахождение оптимальных параметров модели логической классификации

Поясните вывод данного фрагмента. Поясните значение таких величин как fold, candidate, fit. Какие значения принимают данные величины в данном коде и почему?

Следует обратить внимание, что в результате оценки оптимальных параметров, фактически, было построено оптимальное дерево классификации. Доступ к данному дереву производится через поле `best_estimator_` класса `GridSearchCV`. В коде (рис. 4.5) производится экспорт полученного дерева в формат `.dot`. Полученный формат можно преобразовать в `.png` через сервис сайта <http://webgraphviz.com>. Полученное дерево представлено на рис. 4.6.

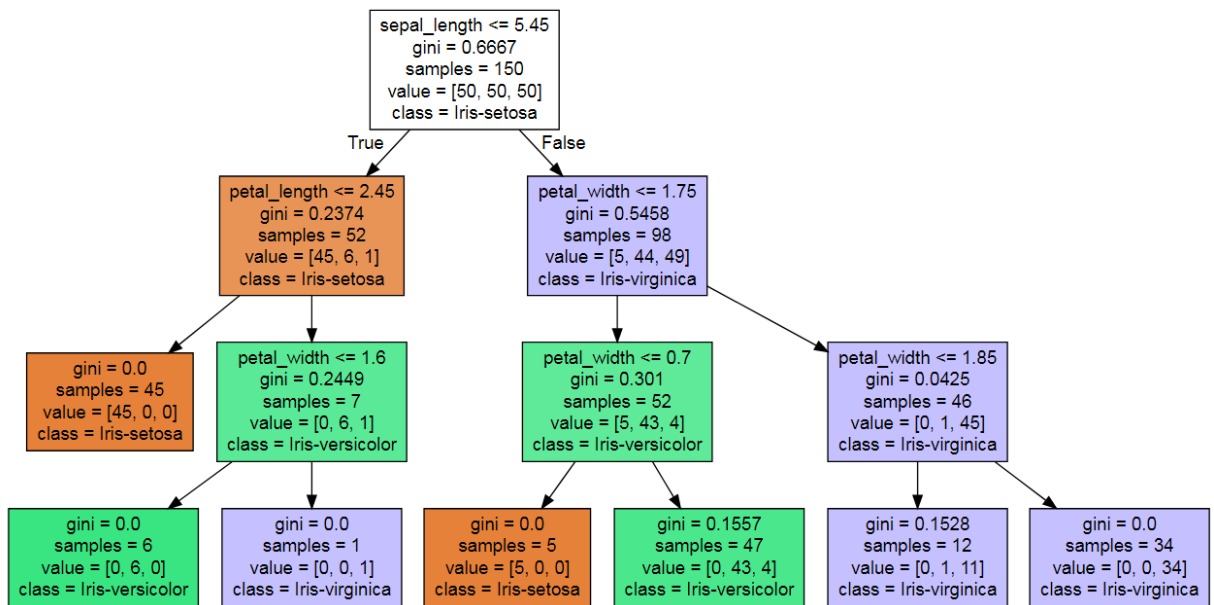


Рисунок 4.6 – Нахождение оптимальных параметров модели логической классификации

Поясните значения переменных в узлах полученного дерева: gini, samples, value.

5. Оптимальные параметры определены, можно обучить модель и использовать ее для классификации (рис. 4.7).

```

# Построим области решения для оптимального дерева
# max_features = 2, max_depth = 3

dtc = DecisionTreeClassifier(max_depth=3,
                             random_state=21,
                             max_features=2)

dtc.fit(dX, dy)
res = dtc.predict([[5.1, 3.5, 1.4, 0.2]])
print(res)

['Iris-setosa']
  
```

Рисунок 4.7 – Использование модели логической классификации

6. В заключении построим еще одну визуализацию процесса логической классификации – покажем решающие границы модели классификации (рис. 4.8).

```

plot_markers = ['r*', 'g^', 'bo']
answers = dy.unique()

# Создаем подграфики для каждой пары признаков
f, places = plt.subplots(4, 4, figsize=(16,16))

fmin = dX.min()-0.5
fmax = dX.max()+0.5
plot_step = 0.02

# Обходим все subplot
for i in range(0,4):
    for j in range(0,4):

        # Строим решающие границы
        if(i != j):
            xx, yy = np.meshgrid(np.arange(fmin[i], fmax[i], plot_step),
                                np.arange(fmin[j], fmax[j], plot_step))
            model = DecisionTreeClassifier(max_depth=3, random_state=21, max_features=2)
            model.fit(dX.ix[:, [i,j]], dy)
            p = model.predict(np.c_[xx.ravel(), yy.ravel()])
            p = p.reshape(xx.shape)
            p[p==answers[0]] = 0
            p[p==answers[1]] = 1
            p[p==answers[2]] = 2
            places[i,j].contourf(xx, yy, p, cmap='Pastel1')

        # Обход всех классов
        for id_answer in range(len(answers)):
            idx = np.where(dy == answers[id_answer])
            if i==j:
                places[i, j].hist(dX.iloc[idx].ix[:,i],
                                color=plot_markers[id_answer][0],
                                histtype = 'step')
            else:
                places[i, j].plot(dX.iloc[idx].ix[:,i], dX.iloc[idx].ix[:,j],
                                plot_markers[id_answer],
                                label=answers[id_answer], markersize=6)

        if j==0:
            places[i, j].set_ylabel(dX.columns[j])

        if i==3:
            places[i, j].set_xlabel(dX.columns[i])

```

Рисунок 4.8 – Построение решающих границ

Вывод данного кода представлен на рис. 4.9.

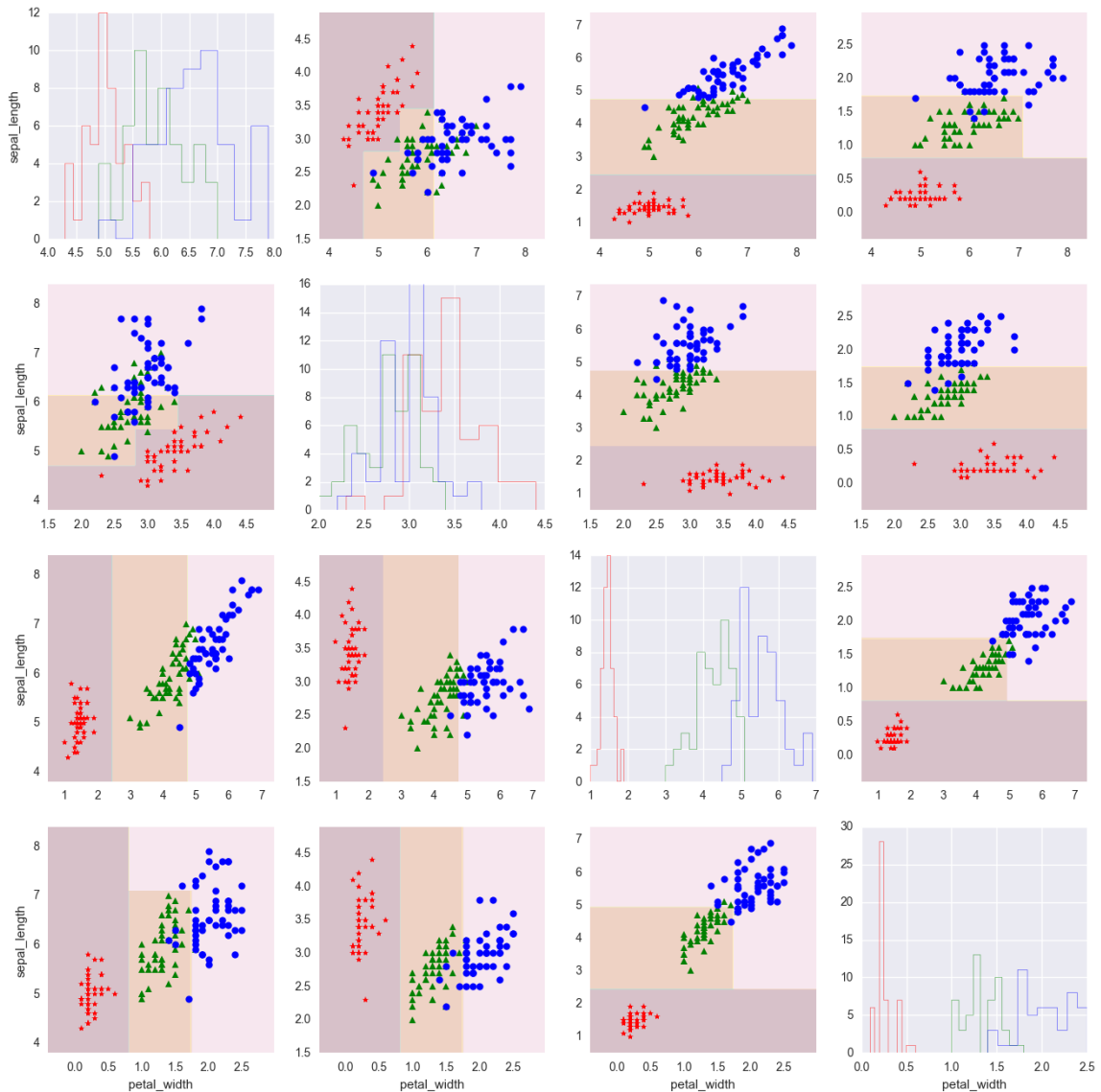


Рисунок 4.9 – Решающие границы логической модели классификации

7. Выполните индивидуальное задание.

Важные замечания

1. При выборе набора данных (data set) на ресурсах [3, 4] необходимо согласовать свой выбор с другими студентами группы и преподавателем с целью недопустимости выбора одинаковых вариантов.

2. В рамках данного лабораторного курса рекомендуется использовать инструментарий Python (библиотеки, среду разработки) для решения поставленных задач.

Индивидуальное задание

1. Студент самостоятельно выбирает набор данных на ресурсах [3, 4] для построения классификатора с использованием метода логической классификации и согласует свой выбор с преподавателем.

2. Выполните построение модели классификации на основе дерева классификации. В ходе решения задачи необходимо решить следующие подзадачи:

2.1 Построение логического классификатора с заданием `max_depth` (максимальной глубины) и `max_features` (максимального количества признаков) пользователем (установить любые); визуализация дерева решений для выбранных исследователем параметров (в формате .png)

2.2 Вычисление оценки `cross validation` (MSE) для различных значений `max_depth` (построить график зависимости);

2.3 Вычисление оценки `cross validation` (MSE) для различных значений `max_features` (построить график зависимости);

2.4 Вычислите оптимальные значения `max_depth` и `max_features`. Обоснуйте свой выбор. Продемонстрируйте использование полученного классификатора.

2.5 Выведите дерево в формате .png;

2.6 Выведите решающие границы полученной модели.

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. Поясните принцип построения дерева решений.
2. Укажите статистическое определение информативности.
3. Поясните энтропийное определение информативности.
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. ЛОГИСТИЧЕСКАЯ РЕГРЕССИЯ

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 работы с разреженными матрицами в ходе машинного обучения;
- освоение техники построения, обучения и оценки модели логистической регрессии;
- освоение приемов работы с синтезированными признаками, масштабированием и настройкой гиперпараметров.

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 программных продуктов для практической реализации задач лабораторной работы): MS Visual Studio 2013 и выше; среда разработки Java, интерпретатор Python (Anaconda).

Указания по технике безопасности

Студенты должны следовать общепринятой технике безопасности для пользователей персональных компьютеров. Не следует самостоятельно производить ремонт технических средств, установку и удаление программного обеспечения. В случае обнаружения неисправностей необходимо сообщить об этом администратору компьютерного класса (обслуживающему персоналу лаборатории).

Методика и порядок выполнения работы

Учебная задача

В рамках учебной задачи разберем проблему «Catch Me If You Can: Intruder Detection through Webpage Session (<https://inclass.kaggle.com/c/catch-me-if-you-can-intruder-detection-through-webpage-session-tracking>).

Решается задача идентификации взломщика по его поведению в сети Интернет. Например, взломщик почтового ящика будет вести себя не так, как владелец ящика: он может не удалять сообщения сразу по прочтении, как это делал хозяин, он будет по-другому ставить флажки сообщениям и даже по-своему двигать мышкой. Тогда такого злоумышленника можно идентифицировать и «выкинуть» из почтового ящика, предложив хозяину войти по SMS-коду. Похожие механизмы разрабатываются, например, в Google Analytics и описываются в научных статьях, найти можно многое по фразам «Traversal Pattern Mining» и «Sequential Pattern Mining».

В учебной задаче рассмотрим похожую задачу: алгоритм будет анализировать последовательность из нескольких веб-сайтов, посещенных подряд одним и тем же человеком, и определять, Элис это или взломщик (кто-то другой). В качестве метрики в этом соревновании используется ROC AUC.

В обучающей выборке `train_sessions.csv` представлены следующие признаки:

- site_i – это индексы посещенных сайтов (расшифровка дана в pickle-файле со словарем site_dic.pkl);
- time_j – время посещения сайтов site_j;
- target – целевой признак; факт того, что сессия принадлежит Элис (то есть что именно Элис ходила по всем этим сайтам).

Задача – сделать прогнозы для сессий в тестовой выборке (test_sessions.csv), определить, принадлежат ли они Элис.

Не обязательно ограничиваться только предложенной выборкой train_sessions.csv – в train.zip даны исходные данные о посещенных пользователями веб-страницах, по которым можно сформировать свою обучающую выборку.

Переходим к решению:

1. Подключем необходимые библиотеки:

```
from __future__ import division, print_function
%matplotlib inline
from matplotlib import pyplot as plt
import seaborn as sns

import pickle
import numpy as np
import pandas as pd
from scipy.sparse import csr_matrix
from scipy.sparse import hstack
from sklearn.preprocessing import StandardScaler
from sklearn.metrics import roc_auc_score
from sklearn.linear_model import LogisticRegression
```

Рисунок 5.1 – Код подключения необходимых библиотек в jupyter notebook

2. Проведем первичное обследование набора данных. Выполним загрузку данных (рис. 5.2).

```

train_df = pd.read_csv('Dataset/train_sessions.csv', index_col='session_id')
test_df = pd.read_csv('Dataset/test_sessions.csv', index_col='session_id')

times = ['time%s' % i for i in range(1, 11)]
train_df[times] = train_df[times].apply(pd.to_datetime)
test_df[times] = test_df[times].apply(pd.to_datetime)
train_df = train_df.sort_values(by='time1')

print(train_df.shape)
print(test_df.shape)
train_df.head()

```

```
(253561, 21)
```

```
(82797, 20)
```

	site1	time1	site2	time2	site3	time3	site4	time4	site5	time5
session_id										
21669	56	2013-01-12 08:05:57	55.0	2013-01-12 08:05:57	NaN	NaT	NaN	NaT	NaN	NaT

Рисунок 5.2 – Загрузка и первичная обработка данных

В каждой строке набора данных содержатся признаки сессии:

– site1 – индекс первого посещенного сайта в сессии;

– time1 – время посещения первого сайта в сессии;

...

– site10 – индекс 10-го посещенного сайта в сессии;

– time10 – время посещения 10-го сайта в сессии;

– target – целевая переменная, принимает значение 1 для сессий Элис и 0 для сессий других пользователей.

Сессии пользователей выделены таким образом, что они не могут быть длинее получаса или содержит более 10 сайтов. То есть сессия считается оконченной либо когда пользователь посетил 10 сайтов подряд, либо когда сессия заняла по времени более 30 минут.

В таблице встречаются пропущенные значения, это значит, что сессия состоит менее, чем из 10 сайтов.

3. Заменяем пропущенные значения нулем и приведем колонки к единому типу. Также загрузим словарь сайтов и посмотрим как он выглядит:

```

sites = ['site%s' % i for i in range(1, 11)]
train_df[sites] = train_df[sites].fillna(0).astype('int')
test_df[sites] = test_df[sites].fillna(0).astype('int')

with open(r"Dataset/site_dic.pkl", "rb") as input_file:
    site_dict = pickle.load(input_file)

sites_dict = pd.DataFrame(list(site_dict.keys()),
                           index=list(site_dict.values()),
                           columns=['site'])
print(u'всего сайтов:', sites_dict.shape[0])
sites_dict.head()

```

всего сайтов: 48371

	site
11774	i62.tinypic.com
27343	static.weezbe.com
18896	forum.sports.fr
25836	webservices.viamichelin.com
36271	www.clipartof.com

Рисунок 5.3 – Загрузка словаря сайтов формата .pkl

4. Покажем, какие сайты являются топовыми, наиболее популярными по посещаемости:

```
top_sites = pd.Series(train_df[sites].fillna(0).values.flatten()
                      ).value_counts().sort_values(ascending=False).head(5)
print(top_sites)
sites_dict.ix[top_sites.index]
```

```
21      123776
0       122730
23       87619
782      77055
22       58258
dtype: int64
```

	site
21	www.google.fr
0	NaN
23	www.google.com
782	annotathon.org
22	apis.google.com

Рисунок 5.4 – Определение наиболее посещаемых сайтов

5. Основную работу будем проводить с временными параметрами сессий. Для этого создадим отдельный фрейм данных для хранения начала сессии, лкнчания сессии и ее длительности (рис. 5.5).

```
# создадим отдельный датафрейм
time_df = pd.DataFrame(index=train_df.index)
time_df['target'] = train_df['target']

# найдем время начала и окончания сессии
time_df['min'] = train_df[times].min(axis=1)
time_df['min'] = pd.to_datetime(time_df['min'])
time_df['max'] = train_df[times].max(axis=1)
time_df['max'] = pd.to_datetime(time_df['max'])

# вычислим длительность сессии и переведем в секунды
time_df['seconds'] = \
    (time_df['max'].values - \
     time_df['min'].values) / np.timedelta64(1, 's')

time_df.head()
```

	target	min	max	seconds
session_id				
21669	0	2013-01-12 08:05:57	2013-01-12 08:05:57	0.0
54843	0	2013-01-12 08:37:23	2013-01-12 09:07:09	1786.0
77292	0	2013-01-12 08:50:13	2013-01-12 08:50:17	4.0
114021	0	2013-01-12 08:50:17	2013-01-12 08:50:20	3.0
146670	0	2013-01-12 08:50:20	2013-01-12 08:50:22	2.0

Рисунок 5.5 – Создание отдельного DataFrame для хранения временных параметров сессий

Чтобы перейти к обучению модели, необходимо подготовить данные. С самого начала выделим целевую переменную и удалим ее из обучающей выборки. Теперь и обучающая, и тестовая выборки будут иметь одинаковое количество колонок, поэтому объединим их в один общий датафрейм. Таким образом, все преобразования будут выполняться одновременно как для обучающей, так и для тестовой выборок. С одной стороны, это ведет к тому, что у обеих выборок будет одно пространство признаков (можно не волноваться, что забыли сделать преобразование какого-то признака для одной из выборок), но с другой возрастает время обработки. Для больших выборок может оказаться невозможным сделать преобразования одновременно для обеих выборок (а иногда преобразования придется разбивать на несколько этапов только для обучающей/тестовой выборки). Здесь же мы будем делать преобразования для объединенной таблицы

целиком, а перед обучением или прогнозированием просто возьмем нужную ее часть.

```
# наша целевая переменная
y_train = train_df['target']
# объединенная таблица исходных данных
full_df = pd.concat([train_df.drop('target', axis=1), test_df])
# индекс, по которому будем отделять обучающую выборку от тестовой
idx_split = train_df.shape[0]

print('y_train', y_train.shape)
print('train_df', train_df.shape)
print('full_df', full_df.shape)

y_train (253561,)
train_df (253561, 21)
full_df (336358, 20)
```

Рисунок 5.6 – Слияние обучающей и тестовой выборок

6. Для первой модели будем использовать только посещенные сайты в сессии (но не будем обращать внимание на временные признаки). За таким выбором данных для модели стоит следующая идея: у Элис есть свои излюбленные сайты, и чем чаще вы видим эти сайты в сессии, тем выше вероятность, что это сессия Элис, и наоборот. Подготовим данные, из всей таблицы выберем только признаки site1, site2, ... , site10. Пропущенные значения заменены нулем. Вот как выглядят первые строки таблицы:

```
full_sites = full_df[sites]
full_sites.head()
```

	site1	site2	site3	site4	site5	site6	site7	site8	site9	site10
session_id										
21669	56	55	0	0	0	0	0	0	0	0
54843	56	55	56	55	0	0	0	0	0	0
77292	946	946	951	946	946	945	948	784	949	946
114021	945	948	949	948	945	946	947	945	946	946
146670	947	950	948	947	950	952	946	951	946	947

Рисунок 5.7 – Обучающая выборка на основе посещаемых сайтов

Сессии представляют собой последовательность индексов сайтов, и данные в таком виде неудобны для линейных методов. В соответствии с нашей гипотезой (у Элис есть изблюбленные сайты) надо преобразовать эту таблицу таким образом, чтобы каждому возможному сайту соответствовал свой отдельный признак (колонка), а его значение равнялось бы количеству посещений этого сайта в сессии:

```
sites_flatten = full_sites.values.flatten()
full_sites_sparse = csr_matrix(([1] * sites_flatten.shape[0],
                                sites_flatten,
                                range(0, sites_flatten.shape[0] + 10,
                                      10)))[:, 1:]

print('sites_flatten', sites_flatten.shape)
print('full_sites_sparse', full_sites_sparse.shape)
print('Calculation Memory Size: ', full_sites_sparse.shape[0] * \
      full_sites_sparse.shape[0] * 4)

sites_flatten (3363580,)
full_sites_sparse (336358, 48371)
Calculation Memory Size: 452546816656
```

```
# Сколько места занимает разреженная матрица в памяти
print('{0} elements * {1} bytes = {2} bytes'.format(
    full_sites_sparse.count_nonzero(),
    4, full_sites_sparse.count_nonzero() * 4))
print('Real Memory Size = {0} bytes'.format(full_sites_sparse.data.nbytes))

1866898 elements * 4 bytes = 7467592 bytes
Real Memory Size = 7467592 bytes
```

Рисунок 5.8 – Построение разреженной матрицы

7. Разреженную матрицу будем использовать как набор данных для обучения. Напишем простую функцию для получения качества обучения (рис. 5.9). В исходном наборе данных 90% прецедентов будут использоваться для обучения, а 10% – для валидации. Затем

```
def get_auc_lr_valid(X, y, C=1.0, seed=17, ratio = 0.9):
    idx = int(round(X.shape[0] * ratio))
    lr = LogisticRegression(C=C,
                            random_state=seed,
                            n_jobs=-1).fit(X[:idx, :], y[:idx])
    y_pred = lr.predict_proba(X[idx:, :])[:, 1]
    score = roc_auc_score(y[idx:], y_pred)
    return score
```

```
%%time
X_train = full_sites_sparse[:idx_split, :]
print(get_auc_lr_valid(X_train, y_train))
```

```
0.919523803917
Wall time: 3.71 s
```

Рисунок 5.9 – Обучение на части выборки и вычисление качества модели

Теперь обучим модель на всей выборке и вычислим качество:

```
# функция для записи прогнозов в файл
def write_to_submission_file(predicted_labels, out_file,
                             target='target', index_label="session_id"):
    predicted_df = pd.DataFrame(predicted_labels,
                                index = np.arange(1, predicted_labels.shape[0] + 1),
                                columns=[target])
    predicted_df.to_csv(out_file, index_label=index_label)
```

```
%%time
lr = LogisticRegression(C=1.0, random_state=17).fit(X_train, y_train)
X_test = full_sites_sparse[idx_split:, :]
y_test = lr.predict_proba(X_test)[:, 1]
write_to_submission_file(y_test, 'baseline_sites.csv')
```

```
Wall time: 5.13 s
```

Рисунок 5.10 – Обучение модели линейной регрессии на полной выборке

8. Модель построена и обучена! Точность, оцениваемая по ROC AUC, имеет значение 0,9195. Но перед использованием необходимо провести анализ: можно ли повысить точность модели за счет добавления новых признаков.

9. Добавим в обучающую выборку новый признак – месяц, в котором происходила сессия (рис. 5.11).

```
# датафрейм для анализа новых признаков
full_new_feat = pd.DataFrame(index=full_df.index)
# добавим признак start_month
full_new_feat['start_month'] = full_df['time1'].apply(
    lambda ts: 100 * ts.year + ts.month)
full_new_feat.head()
```

	start_month
session_id	
21669	201301
54843	201301
77292	201301
114021	201301
146670	201301

Рисунок 5.11 – Создание нового фрейма для анализа нового синтезированного признака

Добавим синтезированный признак в обучающую выборку, проведем обучение и оценим точность полученной модели:

```
# добавляем новый признак в разреженную матрицу
print('full_new_feat.shape', full_new_feat.shape)
tmp = full_new_feat[['start_month']].as_matrix()
print('tmp.shape', tmp.shape)
print('full_sites_sparse.shape', full_sites_sparse.shape)
X_train = csr_matrix(hstack([full_sites_sparse[:,idx_split:], tmp[:,idx_split:]]))

# считаем метрику на валидационной выборке
print(get_auc_lr_valid(X_train, y_train))

full_new_feat.shape (336358, 1)
tmp.shape (336358, 1)
full_sites_sparse.shape (336358, 48371)
0.750835486018
```

Рисунок 5.12 – Обучение модели логистической регрессии с синтезированным признаком

Очевидно, что точность модели уменьшилась после добавления признака. Масштабируем новый признак и выполним процедуру обучения еще раз:

```
# добавим новый стандартизированный признак в разреженную матрицу
tmp = StandardScaler().fit_transform(full_new_feat[['start_month']])
X_train = csr_matrix(hstack([full_sites_sparse[:,idx_split:], tmp[:,idx_split,:]]))
# считаем метрику на валидационной выборке
print(get_auc_lr_valid(X_train, y_train))
```

```
0.919698615157
```

Рисунок 5.13 – Обучение модели логистической регрессии с синтезированным масштабированным признаком

Точность модели повысилась, то есть масштабирование признака имеет положительное влияние на модель.

10. Добавим еще один синтезированный признак: количество уникальных сайтов (n_unique_sites) в сессии (рис. 5.14).

```
# масштабируем новый признак
full_new_feat['n_unique_sites'] = (full_sites.as_matrix()>0).sum(1)
tmp = StandardScaler().fit_transform(full_new_feat[['start_month', 'n_unique_sites']])
X_train = csr_matrix(hstack([full_sites_sparse[:,idx_split:], tmp[:,idx_split,:]]))
print('X_train.shape', X_train.shape)
# обучение, ошибка
print(get_auc_lr_valid(X_train, y_train))
```

```
X_train.shape (253561, 48373)
```

```
0.916113928018
```

Рисунок 5.14 – Обучение модели с новым признаком n_unique_sites

11. Пойдем по пути добавления новых признаков. Добавим еще два признака: час начала сессии (start_hour), логический признак, отражающий утреннее начало сессии (morning).

```
full_new_feat['start_hour'] = time_df['min'].dt.hour
full_new_feat['morning'] = (full_new_feat['start_hour'] <= 11).astype(np.int32)
full_new_feat
```

	start_month	n_unique_sites	start_hour	morning
session_id				
21669	201301	2	8	1
54843	201301	4	8	1
77292	201301	10	8	1
114021	201301	10	8	1
146670	201301	10	8	1

Рисунок 5.15 – Формирование матрицы новых синтезированных признаков для обучения модели

12. Проведем обучение модели с использованием новых признаков:

```
%%time
# формируем обучающую выборку
tmp_scaled = StandardScaler().fit_transform(full_new_feat[['start_month', 'start_hour', 'morning']])
X_train = csr_matrix(hstack([full_sites_sparse[:,idx_split,:],
                             tmp_scaled[:,idx_split,:]]))

# зафиксируем качество с параметрами по умолчанию
score_C_1 = get_auc_lr_valid(X_train, y_train)
print(score_C_1)

0.959149798439
Wall time: 3.6 s
```

Рисунок 5.16 – Обучения модели с новыми признаками и масштабированием

13. При обучении модели мы используем гиперпараметр модели C – коэффициент регуляризации. Подберем коэффициент регуляризации, доставляющий максимальную точность модели.

```
%%time
# набор возможных значений C
Cs = np.logspace(-3, 1, 10)

scores = []

for C in Cs:
    sc = get_auc_lr_valid(X_train, y_train, C=C)
    scores.append(sc)
    print(C, ' -> ', sc)

#from tqdm import tqdm
#for C in tqdm(Cs):
#    scores.append(get_auc_lr_valid(X_train, y_train, C=C))

0.001 -> 0.820627741427
0.00278255940221 -> 0.895920059635
0.00774263682681 -> 0.938832143168
0.0215443469003 -> 0.956320664198
0.0599484250319 -> 0.960675849238
0.16681005372 -> 0.961202396393
0.464158883361 -> 0.960319282616
1.29154966501 -> 0.958667935331
3.5938136638 -> 0.955761207846
10.0 -> 0.951321485419
Wall time: 30.7 s

np.max(scores)

0.96120239639339355
```

Рисунок 5.17 – Получение точности модели при различных значениях гиперпараметра

По полученным значениям AUC-ROC построим график.

```
plt.plot(Cs, scores, 'ro-')
plt.xscale('log')
plt.xlabel('C')
plt.ylabel('AUC-ROC')
plt.title('Подбор коэффициента регуляризации')
# горизонтальная линия -- качество модели с коэффициентом по умолчанию
plt.axhline(y=score_C_1, linewidth=.5, color = 'b', linestyle='dashed')
plt.show()
```

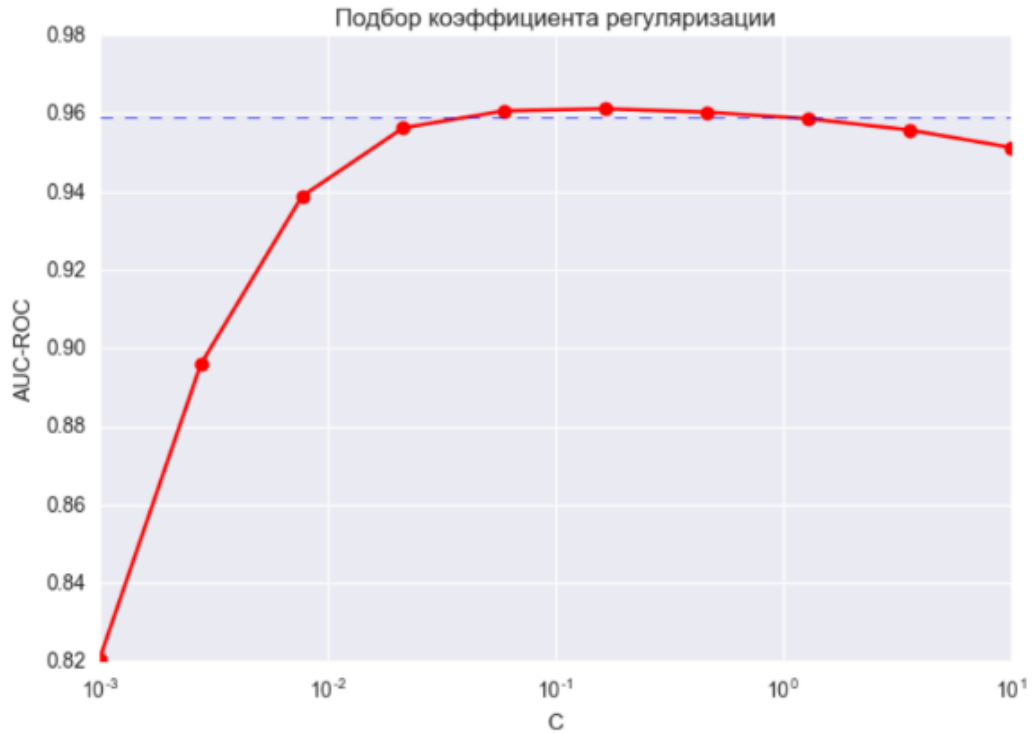


Рисунок 5.18 – График зависимости AUC-ROC от гиперпараметра

14. Обучим модель со значением гиперпараметра C равным 0,1668101.

```
# 0.16681005372 -> 0.961202396393
score_result = get_auc_lr_valid(X_train, y_train, C=0.1668101)
print(score_result)

0.961203453111
```

Рисунок 5.19 – Получение финальной модели и ее точности

15. В заключении обучим модель с найденным оптимальным значением коэффициента регуляризации.

```

# подготовим данные для обучения и теста
tmp_scaled = StandardScaler().fit_transform(
    full_new_feat[['start_month', 'start_hour', 'morning']])
X_train = csr_matrix(hstack([full_sites_sparse[:idx_split,:],
                             tmp_scaled[:idx_split,:]]))
X_test = csr_matrix(hstack([full_sites_sparse[idx_split:,:],
                             tmp_scaled[idx_split:,:]]))
# обучим модель на всей выборке с оптимальным коэффициентом регуляризации
lr = LogisticRegression(C=C, random_state=17).fit(X_train, y_train)
# сделаем прогноз для тестовой выборки
y_test = lr.predict_proba(X_test)[: , 1]
# запишем его в файл, готовый для сабмита
write_to_submission_file(y_test, 'baseline_res.csv')

```

Рисунок 5.20 – Обучение на всей выборке и запись прогноза в файл

16. Выполните индивидуальное задание.

Важные замечания

1. При выборе набора данных (data set) на ресурсах [3, 4] необходимо согласовать свой выбор с другими студентами группы и преподавателем с целью недопустимости выбора одинаковых вариантов.
2. В рамках данного лабораторного курса рекомендуется использовать инструментарий Python (библиотеки, среду разработки) для решения поставленных задач.

Индивидуальное задание

1. Студент самостоятельно выбирает набор данных на ресурсах [3, 4] для построения классификатора с использованием метода логистической регрессии и согласует свой выбор с преподавателем.
2. Выполните построение модели логистической регрессии. В ходе решения задачи необходимо решить следующие подзадачи:
 - 2.1 В рамках выполнения задания необходимо ввести 2 новых синтезированных признака, оценить изменение точности модели после добавления признаков (с применением масштабирования признаков и без).

2.2 После определения необходимого набора признаков необходимо сформировать набор значений гиперпараметра модели и выбрать оптимальное значение. Постройте график (AUC-ROC, гиперпараметр C).

2.3 После обучения модели необходимо сохранить на диск значения бейслайн.

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. За какие года представлены данные в обучающей и тестовой выборках? Ответ нужно аргументировать кодом на Python.
2. Исследуйте с использованием Python исходный набор данных и дайте ответ на следующий вопрос: какие сайты чаще всего посещает Элис? Для ответа на вопрос можно доработать код пункта 4 учебного задания.
3. На языке Python реализуйте проверку следующих утверждений:
 - Сессия Элис в среднем короче, чем сессия остальных пользователей.
 - Доля сессий Элис в выборке превышает 1%.
 - Диапазоны длительности сессий Элис и остальных пользователей примерно одинаковы.

– Доля сессий Элис длительностью 40 с и более составляет менее четверти.

4. В рамках подпункта 9 учебной задачи постройте график зависимости количества сессий Элис от синтезированного признака `start_month`. Проверьте следующие утверждения:

– с начала 2013 года по середину 2014 года количество ежемесячных сессий уменьшилось;

– в целом количество сессий Элис за месяц постоянно на протяжении всего периода;

– с начала 2013 года по середину 2014 года количество ежемесячных сессий возросло.

5. Какие методы классификации являются линейными? Укажите основные параметры линейной модели классификации. Что такое гиперпараметры линейной модели?

6. Поясните назначение и принципы реализации методов стохастического градиента.

7. Что такое «линейно разделимая выборка»?

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 6. ЛИНЕЙНАЯ РЕГРЕССИЯ

Цели и задачи

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

Основные задачи:

- освоение методологии работы с моделями линейной регрессии в задачах машинного обучения;
- освоение методик работы с линейными моделями в python;
- освоение методики применения методов регрессии;
- изучение основных параметров регрессионных моделей.

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 программных 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

В рамках учебной задачи разберем проблему «Прогноз популярности статьи на Хабре» (<https://habrahabr.ru>). Подробности (<https://inclass.kaggle.com/c/howpop-habrahabr-favs>).

Файл `howpop_test.csv` содержит тестовые объекты. Файл `howpop_train.csv` содержит обучающую выборку. Целевая переменная – `favs_lognorm`. Файлы `howpop_test.jsonlines` и `howpop_train.jsonlines` содержат полные описания статей в формате JSON. Целевая переменная – `favs_lognorm`.

Следует обратить внимание, что `howpop_train.jsonlines` – файл с размером 4 ГБ.

Переходим к решению:

17. Подключем необходимые библиотеки:

```
from __future__ import division, print_function
import numpy as np
import pandas as pd
import scipy
from matplotlib import pyplot as plt
import seaborn as sns

from sklearn.model_selection import train_test_split
from sklearn.linear_model import Ridge
from sklearn.metrics import mean_squared_error

%matplotlib inline
```

Рисунок 6.1 – Код подключения библиотек в jupyter notebook

18. Проведем первичное обследование набора данных. Выполним загрузку данных (рис. 6.2). Такое представление обучающей и тестовой выборок не дает осмысленного представления о структуре данных, вывод тяжело читается. Трансформируем вывод следующим образом (рис. 6.3): в методе `head()` будем выводить одну строку, а также транспонируем вывод.

```

train_df = pd.read_csv('dataset/howpop_train.csv')
test_df = pd.read_csv('dataset/howpop_test.csv')

print('////////////////////////////////')
print(train_df.head())
print('////////////////////////////////')
print(test_df.head())

```

```
////////////////////////////////
```

	url	domain	post_id	\
0	https://habrahabr.ru/post/18284/	habrahabr.ru	18284	
1	https://habrahabr.ru/post/18285/	habrahabr.ru	18285	
2	https://habrahabr.ru/post/18286/	habrahabr.ru	18286	
3	https://habrahabr.ru/post/18291/	habrahabr.ru	18291	
4	https://geektimes.ru/post/18294/	geektimes.ru	18294	

	published	author	flow	polling	content_len	\
0	2008-01-01 18:19:00	@Tapac	develop	False	4305	
1	2008-01-01 18:30:00	@DezmASter	design	False	7344	
2	2008-01-01 18:34:00	@DezmASter	design	False	8431	
3	2008-01-02 01:32:00	@Taoorus	design	False	5662	
4	2008-01-02 14:34:00	@dennydo	NaN	False	3706	

	title	comments	favs	views	\
0	Новогодний подарок блоггерам – WordPress 2.3.2	0	0	236	
1	Сумасшедшие яйца, или сервис для отслеживания ...	1	1	353	
2	Сумасшедшие яйца, или сервис для отслеживания ...	47	72	1200	
3	Сглаживание шрифтов, и субпиксельная отрисовка	102	36	5700	
4	Почему мне не нравится iPhone	230	6	1400	

Рисунок 6.2 – Получение шапки набора данных

```

train_df = pd.read_csv('dataset/howpop_train.csv')
test_df = pd.read_csv('dataset/howpop_test.csv')

print('////////////////////////////////')
print(train_df.head(1).T)
print('////////////////////////////////')
print(test_df.head(1).T)

```

Рисунок 6.3 – Получение читаемого представления данных в jupyter notebook

Вывод для данного кода представлен на рис. 6.4.

```

////////////////////////////////////
url                                     0
                                     https://habrahabr.ru/post/18284/
domain                               habrahabr.ru
post_id                             18284
published                           2008-01-01 18:19:00
author                              @Tapac
flow                                develop
polling                             False
content_len                          4305
title                               Новогодний подарок блоггерам – WordPress 2.3.2
comments                             0
favs                                 0
views                               236
votes_plus                           0
votes_minus                           0
views_lognorm                         -0.792687
favs_lognorm                         -1.34407
comments_lognorm                      -2.43687
////////////////////////////////////
                                     0
url                                   https://habrahabr.ru/post/314080/
domain                               habrahabr.ru
post_id                             314080
published                           2016-11-01 01:05:00
author                              @fsou11
flow                                develop
polling                             True
content_len                          20132
title                               Опыт использования MassTransit 3.0

```

Рисунок 6.4 – Столбцы набора данных

19. Определим размер выборок:

```

print('Тестовый набор: \t', test_df.shape)
print('Обучающая выборка: \t', train_df.shape)

```

```

Тестовый набор:          (3990, 9)
Обучающая выборка:      (134137, 17)

```

Рисунок 6.5 – Определение размера тестового и обучающего наборов

20. Получим общую информацию по фрейму:

```
print(train_df.info())

<class 'pandas.core.frame.DataFrame'>
RangeIndex: 134137 entries, 0 to 134136
Data columns (total 17 columns):
url                134137 non-null object
domain             134137 non-null object
post_id            134137 non-null int64
published           134137 non-null object
author             97657 non-null object
flow               97048 non-null object
polling            134137 non-null bool
content_len        134137 non-null int64
title              134137 non-null object
comments           134137 non-null int64
favs               134137 non-null int64
views              134137 non-null int64
votes_plus         133566 non-null float64
votes_minus        133566 non-null float64
views_lognorm      134137 non-null float64
favs_lognorm       134137 non-null float64
comments_lognorm   134137 non-null float64
dtypes: bool(1), float64(5), int64(5), object(6)
memory usage: 16.5+ MB
None
```

Рисунок 6.6 – Общая информация по обучающей выборке

21. Рассмотрим, каким образом упорядочены данные в `train_df` по временной оси (по `published`). Для этого используем код (рис. 6.7). Данные упорядочены по полю `published` (рис. 6.8).

```
# копируем столбец данных published
ser_data = train_df['published'].apply(lambda ts: pd.to_datetime(ts))
print('Ряд с датами столбца published')
print(ser_data.head())
print('Размер объекта Series: ', df1.shape)
ser_data.apply(lambda el: el.value).plot()
```

Рисунок 6.7 – Код для анализа отдельного поля `published`

```

Ряд с датами столбца published
0    2008-01-01 18:19:00
1    2008-01-01 18:30:00
2    2008-01-01 18:34:00
3    2008-01-02 01:32:00
4    2008-01-02 14:34:00
Name: published, dtype: datetime64[ns]
Размер объекта Series: (134137,)

<matplotlib.axes._subplots.AxesSubplot at 0x1bfc9c18>

```

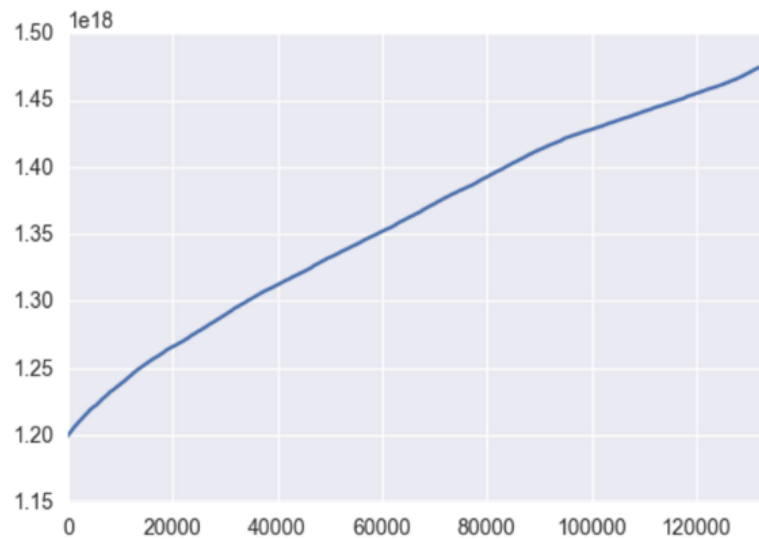


Рисунок 6.8 – График, построенный по ряду published

22. Продолжим анализировать набор данных и определим, существует ли корреляция между отдельными признаками. Для этого воспользуемся методом `pandas.DataFrame.corr()`. Вызвав реализацию функции по умолчанию `train_df.corr()`, получим коэффициенты корреляции.

```
corr = train_df.corr()
corr
```

	post_id	polling	content_len	comments	favs	views	votes_plus	votes_minus	views_lognorm	favs_lognorm	comments_lognorm
post_id	1.000000	0.081628	0.241384	-0.147763	0.089313	0.132385	-0.187885	-0.235594	-0.010376	-0.005845	-0.001605
polling	0.081628	1.000000	-0.003337	0.042605	0.006882	0.025660	-0.033505	0.027116	-0.003523	-0.036385	0.071417
content_len	0.241384	-0.003337	1.000000	-0.023544	0.308194	0.204101	0.068779	-0.078686	0.246063	0.356481	0.073132
comments	-0.147763	0.042605	-0.023544	1.000000	0.164166	0.290035	0.613961	0.457638	0.349568	0.278942	0.662740
favs	0.089313	0.006882	0.308194	0.164166	1.000000	0.634304	0.416241	0.062877	0.456097	0.587982	0.263239
views	0.132385	0.025660	0.204101	0.290035	0.634304	1.000000	0.396849	0.128654	0.585105	0.406782	0.326427
votes_plus	-0.187885	-0.033505	0.068779	0.613961	0.416241	0.396849	1.000000	0.464168	0.414232	0.449712	0.525081
votes_minus	-0.235594	0.027116	-0.078686	0.457638	0.062877	0.128654	0.464168	1.000000	0.146609	0.088226	0.355458
views_lognorm	-0.010376	-0.003523	0.246063	0.349568	0.456097	0.585105	0.414232	0.146609	1.000000	0.688811	0.546530
favs_lognorm	-0.005845	-0.036385	0.356481	0.278942	0.587982	0.406782	0.449712	0.088226	0.688811	1.000000	0.479476
comments_lognorm	-0.001605	0.071417	0.073132	0.662740	0.263239	0.326427	0.525081	0.355458	0.546530	0.479476	1.000000

Рисунок 6.8 – Матрица корреляции для признаков train_df

По данной матрице необходимо определить коррелирующие признаки (с коэффициентом корреляции больше 0.9). Искать такие значения по

представленной матрице – достаточно сложная задача, которая повлечет ошибки, поэтому представим матрицу корреляции в удобном для анализа виде:

`corr[corr>0.9].replace({np.nan : ''})`

	post_id	polling	content_len	comments	favs	views	votes_plus	votes_minus	views_lognorm	favs_lognorm	com
post_id	1										
polling		1									
content_len			1								
comments				1							
favs					1						
views						1					
votes_plus							1				
votes_minus								1			
views_lognorm									1		
favs_lognorm										1	
comments_lognorm											1

Рисунок 6.9 – Матрица корреляции для признаков train_df

Таким образом устанавливаем, что признаки набора данных независимы.

23. Проведем еще одно исследование: определим как распределены публикации по годам (рис. 6.10).

```
df = train_df.copy()
df['published'] = pd.to_datetime(df['published']).dt.year
ss = df['published'].value_counts()
print(ss.sort_index())
```

```
2008      7743
2009     10783
2010     13091
2011     15063
2012     15972
2013     15537
2014     16180
2015     23452
2016     16316
Name: published, dtype: int64
```

Рисунок 6.10 – Код для построения графика распределения публикаций по годам

Для более наглядного представления о распределении по годам построим график:

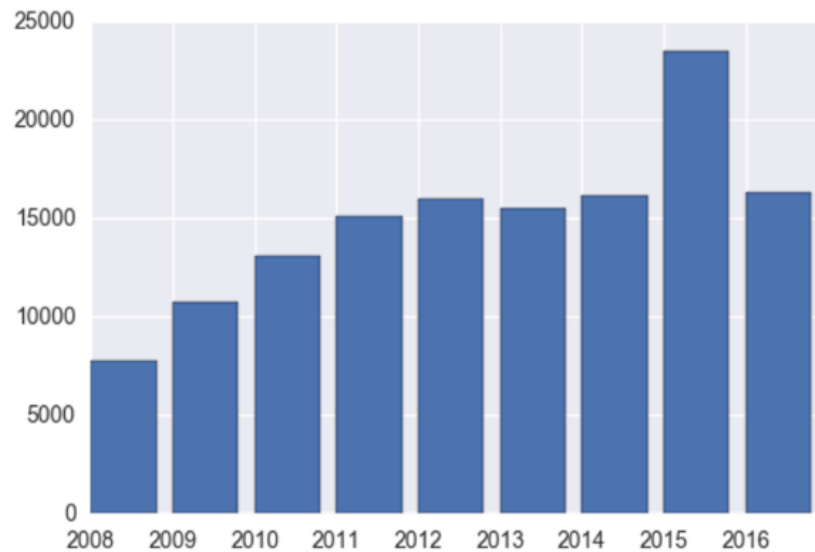


Рисунок 6.11 – График распределения публикаций по годам

24. Первичное исследование набора данных завершено. Перейдем непосредственно к построению модели обучения. Разделим исходный набор данных на тренировочную и тестовую подвыборки, а также отберем признаки, которые будут использоваться в процессе обучения.

```
features = ['author', 'flow', 'domain', 'title']
train_size = int(0.7 * train_df.shape[0])
print('Размер исходного набора: ', len(train_df), \
      '\nРазмер обучающей подвыборки: ', train_size)
#отделяем признаки от целевой переменной
X, y = train_df.ix[:, features], train_df['favs_lognorm']
X_test = test_df.ix[:, features]
X_train, X_valid = X.iloc[:train_size, :], X.iloc[train_size:,:]
y_train, y_valid = y.iloc[:train_size], y.iloc[train_size:]
```

```
Размер исходного набора: 134137
Размер обучающей подвыборки: 93895
```

Рисунок 6.12 – Код для подготовки подвыборок

25. Для анализа контента (содержимого файлов howpor_*.jsonlines) используем TfidfVectorizer из пакета sklearn. TF-IDF (от англ. TF – term frequency, IDF – inverse document frequency) – статистическая мера, используемая для оценки важности слова в контексте документа, являющегося частью коллекции документов или корпуса. Вес некоторого слова

пропорционален количеству употребления этого слова в документе, и обратно пропорционален частоте употребления слова в других документах коллекции. TfidfVectorizer преобразует тексты в матрицу TF-IDF признаков.

Основные параметры TfidfVectorizer в sklearn:

- min_df – при построении словаря слова, которые встречаются реже, чем указанное значение, игнорируются;
- max_df – при построении словаря слова, которые встречаются чаще, чем указанное значение, игнорируются;
- analyzer – определяет, строятся ли признаки по словам или по символам (буквам);
- ngram_range – определяет, формируются ли признаки только из отдельных слов или из нескольких слов (в случае с analyzer='char' задает количество символов). Например, если указать analyzer='word' и ngram_range=(1,3), то признаки будут формироваться из отдельных слов, из пар слов и из троек слов;
- stop_words – слова, которые игнорируются при построении матрицы.

Создадим объект TfidfVectorizer и обучим его на данных:

```
from sklearn.feature_extraction.text import TfidfVectorizer

vectorizer_title = TfidfVectorizer(min_df=3, max_df=0.3, ngram_range=(1,3))

vX_train_title = vectorizer_title.fit(X_train['title'])
print('vX_train_title.vocabulary_: ', len(vX_train_title.vocabulary_))
vX_valid_title = vectorizer_title.fit(X_valid['title'])
print('vX_train_title.vocabulary_: ', len(vX_train_title.vocabulary_))
vX_test_title = vectorizer_title.fit(X_test['title'])
print('vX_train_title.vocabulary_: ', len(vX_train_title.vocabulary_))

X_train_title = vectorizer_title.fit_transform(X_train['title'])
print('X_train_title.shape: ', X_train_title.shape)
X_valid_title = vectorizer_title.transform(X_valid['title'])
print('X_valid_title.shape: ', X_valid_title.shape)
X_test_title = vectorizer_title.transform(X_test['title'])
print('X_test_title.shape: ', X_test_title.shape)

vX_train_title.vocabulary_: 50624
vX_train_title.vocabulary_: 28528
vX_train_title.vocabulary_: 2768
X_train_title.shape: (93895, 50624)
X_valid_title.shape: (40242, 50624)
X_test_title.shape: (3990, 50624)
```

Рисунок 6.13 – Создание TfidfVectorizer

Получили три словаря и три матрицы TF-IDF. Для доступа к словарям можно обратиться к полю `vocabulary_` объекта `TfidfVectorizer`. К этому времени объект должен быть обучен – должен быть вызван метод `fit_transform()`.

```
X_test_title.vocabulary_
{'код': 1284,
 'почему': 1973,
 'дыры': 1005,
 'поможет': 1952,
 'евгений': 1006,
 'user group': 521,
 'точностью': 2530,
 'космические': 1338,
 'рынке': 2249,
 'под android': 1905,
 'луну': 1399,
 'каждой': 1216,
```

Рисунок 6.14 – Формат словаря `TfidfVectorizer`

Данные в словаре хранятся в формате `{'термин': индекс признака,...}`.

26. Построим TF-IDF-матрицы для этих же признаков, но с использованием параметра `analyzer='char'` (рис. 6.15).

```
vectorizer_title_ch = TfidfVectorizer(analyzer='char')

vX_train_title_ch = vectorizer_title_ch.fit(X_train['title'])
print('vX_train_title_ch.vocabulary_: ', len(vX_train_title_ch.vocabulary_))
vX_valid_title_ch = vectorizer_title_ch.fit(X_valid['title'])
print('vX_valid_title_ch.vocabulary_: ', len(vX_valid_title_ch.vocabulary_))
vX_test_title_ch = vectorizer_title_ch.fit(X_test['title'])
print('vX_test_title_ch.vocabulary_: ', len(vX_test_title_ch.vocabulary_))

X_train_title_ch = vectorizer_title_ch.fit_transform(X_train['title'])
print('X_train_title_ch.shape: ', X_train_title_ch.shape)
X_valid_title_ch = vectorizer_title_ch.transform(X_valid['title'])
print('X_valid_title_ch.shape: ', X_valid_title_ch.shape)
X_test_title_ch = vectorizer_title_ch.transform(X_test['title'])
print('X_test_title_ch.shape: ', X_test_title_ch.shape)

vX_train_title_ch.vocabulary_: 218
vX_valid_title_ch.vocabulary_: 165
vX_test_title_ch.vocabulary_: 136
X_train_title_ch.shape: (93895, 218)
X_valid_title_ch.shape: (40242, 218)
X_test_title_ch.shape: (3990, 218)
```

Рисунок 6.15 – Построение TfidfVectorizer с параметром analyzer='char'

27. Для обработки остальных признаков будем использовать DictVectorizer из sklearn. Признаки ['author', 'flow', 'domain'] имеют категориальную природу, для них TfidfVectorizer неприменим.

```
vectorizer_feats = DictVectorizer()
tmp_dict_train = X_train[feats].fillna('-').T.to_dict().values()
tmp_dict_valid = X_valid[feats].fillna('-').T.to_dict().values()
tmp_dict_test = X_test[feats].fillna('-').T.to_dict().values()
```

```
X_train_feats = vectorizer_feats.fit_transform(tmp_dict_train)
X_valid_feats = vectorizer_feats.transform(tmp_dict_valid)
X_test_feats = vectorizer_feats.transform(tmp_dict_test)
print(X_train_feats.shape)
print(X_valid_feats.shape)
print(X_test_feats.shape)
```

```
(93895, 17869)
(40242, 17869)
(3990, 17869)
```

Рисунок 6.16 – Построение DictVectorizer для категориальных признаков

28. Выполним объединение полученных матриц:

```
# объединение матриц, построенных на предыдущих этапах
X_train_new = scipy.sparse.hstack([X_train_title, \
                                   X_train_feats, \
                                   X_train_title_ch])
X_valid_new = scipy.sparse.hstack([X_valid_title, \
                                   X_valid_feats, \
                                   X_valid_title_ch])
X_test_new =  scipy.sparse.hstack([X_test_title, \
                                   X_test_feats, \
                                   X_test_title_ch])

print(X_train_new.shape)
print(X_valid_new.shape)
print(X_test_new.shape)
```

```
(93895, 68711)
(40242, 68711)
(3990, 68711)
```

Рисунок 6.17 – Объединение IDF-матриц

Следует обратить внимание, что операция объединения допустима над полученными матрицами (поясните почему).

29. Для обучения выбрана линейная модель регрессии с L2-регуляризацией.

```
%%time
model_1 = Ridge(alpha=.1, random_state=1)
model_1.fit(X_train_new, y_train)
```

Wall time: 10 s

```
train_preds1 = model_1.predict(X_train_new)
valid_preds1 = model_1.predict(X_valid_new)

print('Ошибка на трейне: ', mean_squared_error(y_train, train_preds1))
print('Ошибка на тесте: ', mean_squared_error(y_valid, valid_preds1))
```

Ошибка на трейне: 0.184107031582

Ошибка на тесте: 0.996233534497

Рисунок 6.18 – Обучение модели с параметром регуляризации $\alpha=0.1$

30. Выполним обучение еще одной модели и проверим ошибки:

```
%%time
model_2 = Ridge(alpha=1.0, random_state=1)
model_2.fit(X_train_new, y_train)
```

Wall time: 4.12 s

```
train_preds2 = model_2.predict(X_train_new)
valid_preds2 = model_2.predict(X_valid_new)

print('Ошибка на трейне: ', mean_squared_error(y_train, train_preds2))
print('Ошибка на тесте: ', mean_squared_error(y_valid, valid_preds2))
```

Ошибка на трейне: 0.28624407471

Ошибка на тесте: 0.80695156251

Рисунок 6.19 – Обучение модели с параметром регуляризации $\alpha=1$

Важные замечания

1. При выборе набора данных (data set) на ресурсах [3, 4] необходимо согласовать свой выбор с другими студентами группы и преподавателем с целью недопустимости выбора одинаковых вариантов.

2. В рамках данного лабораторного курса рекомендуется использовать инструментарий Python (библиотеки, среду разработки) для решения поставленных задач.

Индивидуальное задание

1. Студент самостоятельно выбирает набор данных на ресурсах [3, 4] для построения классификатора с использованием метода логической классификации и согласует свой выбор с преподавателем.

2. Выполните построение модели классификации на основе дерева классификации. В ходе решения задачи необходимо решить следующие подзадачи:

2.1 Построение логического классификатора с заданием `max_depth` (максимальной глубины) и `max_features` (максимального количества признаков) пользователем (установить любые); визуализация дерева решений для выбранных исследователем параметров (в формате .png)

2.2 Вычисление оценки `cross validation` (MSE) для различных значений `max_depth` (построить график зависимости);

2.3 Вычисление оценки `cross validation` (MSE) для различных значений `max_features` (построить график зависимости);

2.4 Вычислите оптимальные значения `max_depth` и `max_features`. Обоснуйте свой выбор. Продемонстрируйте использование полученного классификатора.

2.5 Выведите дерево в формате .png;

2.6 Выведите решающие границы полученной модели.

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 методы классификации являются линейными?
2. Укажите основные параметры линейной модели классификации.
3. Поясните назначение и принципы реализации методов стохастического градиента.
4. В чем заключается главная идея метода опорных векторов?
5. Что такое «линейно разделимая выборка»?
6. Поясните назначение ядер и спрямляющих пространств в алгоритмах линейной классификации.

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 7. РАЗРАБОТКА ЕДИНОГО ШАБЛОНА ПРЕДВАРИТЕЛЬНОЙ ОБРАБОТКИ ДАННЫХ

Цели и задачи

Цель лабораторной работы: изучение теоретических принципов и инструментальных средств для построения пайплайна для предварительной обработки данных.

Основные задачи:

- предварительная обработка данных;
- изучение библиотек для предварительной обработки данных;
- масштабирование признаков;
- представление категориальных данных;
- построение пайплайна для предварительной обработки данных.

Теоретическое обоснование

Для решения задач машинного обучения часто приходится повторять различные блоки кода, которые являются единообразными для разных задач, принадлежащих одному классу (регрессия, классификация, кластеризация и т.д.). Данное обстоятельство приводит к повторяющемуся шаблонному коду. Такой код называется boilerplate-код или просто boilerplate. С другой стороны, единообразная последовательность действий, которую выполняет разработчик при решении задач машинного обучения часто называется пайплайном (machine learning pipeline).

Рассмотрим простейший пайплан для решения задачи регрессии. Для решения задачи регрессии необходимо реализовать (в общем случае) следующие стадии:

1. Загрузка набора данных.
2. Заполнение пропусков данных в соответствии с выбранной стратегией.
3. Масштабирование признаков.

4. Обработка категориальных признаков.
5. Разделение на тестовую и тренировочную выборку.
6. Обучение модели.
7. Интерпретация и визуализация результатов.

Заполнение пропусков в данных

```
missingvalues = SimpleImputer(missing_values = np.nan, strategy = 'mean',
verbose = 0)

missingvalues = missingvalues.fit(X[:, 1:3])
X[:, 1:3]=missingvalues.transform(X[:, 1:3])
```

Масштабирование признаков (standartisation и normalization)

```
from sklearn.preprocessing import StandardScaler
```

Разделение на тестовую и обучающую выборки:

```
from sklearn.model_selection import train_test_split
from sklearn.cross_validation import train_test_split
```

Перед выполнением лабораторной работы необходимо ознакомиться с базовыми принципами языка Python, 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 программных продуктов для практической реализации задач лабораторной работы): MS Visual Studio 2013 и выше; среда разработки Java, интерпретатор Python (Anaconda).

Методика и порядок выполнения работы

Для тестирования универсального пайплайна будет использоваться модель линейной регрессии (LinearRegression из библиотеки sklearn).

Учебная задача

Условие. Построить пайплайн, реализующий первичную обработку данных.

Решение. Для решения задачи необходимо написать скрипт на языке Python (рисунок 7.1).

1. Подключение библиотек

```
import numpy as np
import matplotlib.pyplot as plt
import pandas as pd
```

2. Загрузка данных и разделение на матрицу признаков и зависимую переменную

```
dataset = pd.read_csv('Data.csv')
dataset.head()
```

	Country	Age	Salary	Purchased
0	France	44.0	72000.0	No
1	Spain	27.0	48000.0	Yes
2	Germany	30.0	54000.0	No
3	Spain	38.0	61000.0	No
4	Germany	40.0	NaN	Yes

```
X = dataset.iloc[:, :-1].values
y = dataset.iloc[:, 3].values
print ("Матрица признаков"); print(X)
print ("Зависимая переменная"); print(y)
```

Матрица признаков

```
[['France' 44.0 72000.0]
 ['Spain' 27.0 48000.0]
 ['Germany' 30.0 54000.0]
 ['Spain' 38.0 61000.0]
 ['Germany' 40.0 nan]
 ['France' 35.0 58000.0]
 ['Spain' nan 52000.0]
 ['France' 48.0 79000.0]
 ['Germany' 50.0 83000.0]
 ['France' 37.0 67000.0]]
```

Зависимая переменная

```
['No' 'Yes' 'No' 'No' 'Yes' 'Yes' 'No' 'Yes' 'No' 'Yes']
```

3. Обработка пропущенных значений

```
from sklearn.preprocessing import Imputer
imputer = Imputer(missing_values = 'NaN', strategy = 'mean', axis = 0)
imputer = imputer.fit(X[:, 1:3])
X[:, 1:3] = imputer.transform(X[:, 1:3])
print(X)
```

```
[['France' 44.0 72000.0]
 ['Spain' 27.0 48000.0]
 ['Germany' 30.0 54000.0]
 ['Spain' 38.0 61000.0]
 ['Germany' 40.0 63777.77777777778]
 ['France' 35.0 58000.0]
 ['Spain' 38.77777777777778 52000.0]
 ['France' 48.0 79000.0]
 ['Germany' 50.0 83000.0]
 ['France' 37.0 67000.0]]
```

4. Обработка категориальных данных

4.1 Замена категории кодом (LabelEncoder)

```
from sklearn.preprocessing import LabelEncoder
labelencoder_y = LabelEncoder()
print("Зависимая переменная до обработки")
print(y)
y = labelencoder_y.fit_transform(y)
print("Зависимая переменная после обработки")
print(y)
```

Зависимая переменная до обработки
 ['No' 'Yes' 'No' 'No' 'Yes' 'Yes' 'No' 'Yes' 'No' 'Yes']
 Зависимая переменная после обработки
 [0 1 0 0 1 1 0 1 0 1]

4.2 Применение OneHotEncoder

```
from sklearn.preprocessing import OneHotEncoder
labelencoder_X = LabelEncoder()
X[:, 0] = labelencoder_X.fit_transform(X[:, 0])
onehotencoder = OneHotEncoder(categorical_features = [0])
X = onehotencoder.fit_transform(X).toarray()
print("Перекодировка категориального признака")
print(X)
```

Перекодировка категориального признака

[	1.00000000e+00	0.00000000e+00	0.00000000e+00	4.40000000e+01
	7.20000000e+04]			
[	0.00000000e+00	0.00000000e+00	1.00000000e+00	2.70000000e+01
	4.80000000e+04]			
[	0.00000000e+00	1.00000000e+00	0.00000000e+00	3.00000000e+01
	5.40000000e+04]			
[	0.00000000e+00	0.00000000e+00	1.00000000e+00	3.80000000e+01
	6.10000000e+04]			

Рисунок 7.1 – Код Python, отражающий общий пайплайн для предварительной обработки данных

Индивидуальное задание

1. Подберите набор данных на ресурсах [5-7] и согласуйте свой выбор с преподавателем. Студент может предложить набор данных в соответствии с тематикой магистерского исследования.

2. Реализуйте первичную обработку данных загруженного набора. Выполните полный спектр операций для загруженного набора данных: загрузка, визуализация, обработка пропущенных значений, обработка категориальных данных и разделение выборки на тестовую и тренировочную.

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. Какая библиотека python предназначена для управления наборами данных: numpy, pandas, sklearn, opencv, matplotlib?
2. Какая стратегия является нежелательной при обработке пропусков в данных?
 - а) замена пропущенных значений в столбце медианным значением по данному столбцу;
 - б) удаление строк, содержащих пропуски в данных;
 - в) замена пропущенных значений в столбце средним арифметическим значением по данному столбцу;
 - г) замена пропущенных значений в столбце наиболее часто встречающимся значением по данному столбцу;

3. Обоснуйте ответ на следующую проблему предварительной обработки данных: имеется независимая категориальная переменная u , которая представляет собой категориальный признак, опеределенный на домене {C#, Java, Python, R}. Нужно ли применять к данному целевому признаку OneHotEncoder?

4. Поясните принцип разбиения набора данных на обучающую и тестовую выборку. Какое соотношение «тестовая:обучающая» наиболее оптимально: 20:80, 50:50, 25:75, 5:95, 40:30?

5. Какой код лучше использовать при загрузке данных из csv-файла?

а) `dataset = read_csv("data.csv")`

б) `dataset = import("data.csv")`

в) `dataset = read.csv("data.csv")`

г) `dataset = import.csv("data.csv")`

д) `dataset = read_xls("data.csv")`

Список литературы

Для выполнения лабораторной работы, при подготовке к защите, а также для ответа на контрольные вопросы рекомендуется использовать следующие источники: [1, 2, 5-7].

ЛАБОРАТОРНАЯ РАБОТА 8. ПОСТРОЕНИЕ ПАЙПЛАЙНА ОДНОМЕРНОЙ РЕГРЕССИИ

Цели и задачи

Цель лабораторной работы: разработка единого пайплайна для решения задачи регрессии.

Основные задачи:

- реализовать конвейер для выполнения всех стадий обработки данных при решении задачи одномерной регрессии;
- получение теоретических представлений о задаче регрессии;
- получение навыков использования пайплайна при решении задачи машинного обучения;
- получение навыков рефакторинга кода в задачах машинного обучения.

Теоретическое обоснование

Для решения задачи одномерной регрессии необходимо использовать универсальный пайплайн предварительной обработки данных. К имеющемуся шаблонному коду необходимо добавить код для обучения модели, интерпретации и визуализации результатов.

Линейная регрессия – метод восстановления зависимости между двумя переменными. Пусть задана модель регрессии – параметрическое семейство функций $g(x, \alpha)$, где $\alpha \in \mathbb{R}^p$ – вектор параметров модели. Определим функционал качества аппроксимации целевой зависимости на выборке X^ℓ как сумму квадратов ошибок:

$$Q(\alpha, X^\ell) = \sum_{i=1}^{\ell} (g(x_i, \alpha) - y_i)^2. \quad (8.1)$$

Обучение по методу наименьших квадратов (МНК) состоит в том, чтобы найти вектор параметров α^* , при котором достигается минимум среднего квадрата ошибки на заданной обучающей выборке X^ℓ :

$$\alpha^* = \arg \min_{\alpha \in \mathbb{R}^p} Q(\alpha, X^\ell). \quad (8.2)$$

Стандартный способ решения этой оптимизационной задачи – воспользоваться необходимым условием минимума. Если функция $g(x, \alpha)$ достаточное число раз дифференцируема по α , то в точке минимума выполняется система p уравнений относительно p неизвестных:

$$\frac{\partial Q}{\partial \alpha}(\alpha, X^\ell) = 2 \sum_{i=1}^{\ell} (g(x_i, \alpha) - y_i) \frac{\partial g}{\partial \alpha}(x_i, \alpha) = 0. \quad (8.3)$$

С использованием библиотек машинного обучения формулы (8.1) – (8.2) можно реализовать автоматически, но следует понимать, что конкретно реализует каждый метод.

Оборудование и материалы

Для выполнения лабораторной работы рекомендуется использовать персональный компьютер со следующими программными средствами разработки (выбрать один или несколько программных 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

Перед выполнением индивидуального задания рекомендуется выполнить все пункты учебной задачи.

Учебная задача

Условие. Построить пайплайн, реализующий решение задачи линейной одномерной регрессии.

Решение. Для решения задачи необходимо написать скрипт на языке Python (рисунок 8.1).

1. Подключение библиотек

```
import numpy as np
import matplotlib.pyplot as plt
import pandas as pd
```

2. Загрузка данных и разделение на матрицу признаков и зависимую переменную

```
dataset = pd.read_csv('./Salary_Data.csv')
dataset.head()
```

	YearsExperience	Salary
0	1.1	39343.0
1	1.3	46205.0
2	1.5	37731.0
3	2.0	43525.0
4	2.2	39891.0

```
x = dataset.iloc[:, :-1].values
y = dataset.iloc[:, 1].values
print ("Матрица признаков"); print(x[:5])
print ("Зависимая переменная"); print(y[:5])
```

Матрица признаков

```
[[ 1.1]
 [ 1.3]
 [ 1.5]
 [ 2. ]
 [ 2.2]]
```

Зависимая переменная

```
[ 39343.  46205.  37731.  43525.  39891.]
```


Lesson 7. Построение пайплайна регрессии Last Checkpoint: 5 minutes ago (autosaved)

View Insert Cell Kernel Widgets Help

Trusted








Code

5. Разделение выборки на тестовую и тренировочную

```
# from sklearn.cross_validation import train_test_split
from sklearn.model_selection import train_test_split
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size = 1/4, random_state = 0)
```

6. Обучение линейной модели регрессии

```
from sklearn.linear_model import LinearRegression
regressor = LinearRegression()
regressor.fit(X_train, y_train)
```

LinearRegression(copy_X=True, fit_intercept=True, n_jobs=1, normalize=False)

7. Предсказание, обработка и визуализация результатов

```
y_pred = regressor.predict(X_test)
print(y_pred)
```

```
[ 41056.25705466 123597.70938378  65443.50433372  63567.56223533
 116093.94099022 108590.17259667 117031.91203942  64505.53328452]
```

```
plt.scatter(X_train, y_train, color = 'red')
plt.plot(X_train, regressor.predict(X_train), color = 'blue')
plt.title('Salary vs Experience (Training set)')
plt.xlabel('Years of Experience')
plt.ylabel('Salary')
plt.show()
```



```
plt.scatter(X_test, y_test, color = 'red')
plt.plot(X_train, regressor.predict(X_train), color = 'blue')
plt.title('Salary vs Experience (Test set)')
plt.xlabel('Years of Experience')
plt.ylabel('Salary')
plt.show()
```



Рисунок 8.1 – Код Python, отражающий общий пайплайн для задачи одномерной регрессии

Индивидуальное задание

1. Подберите набор данных на ресурсах [5-7] и согласуйте свой выбор с преподавателем. Студент может предложить набор данных в соответствии с тематикой магистерского исследования.
2. Постройте модель регрессии.

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. Почему при реализации линейной модели регрессии нет необходимости выполнять масштабирование признаков?
2. Почему при реализации модели линейной регрессии в качестве функции потерь используется квадратичное отклонение, а не модуль отклонения?
3. Что именно реализовано в методе `fit(X, y)` класса `LinearRegression`?
4. Что такое *p*-значение? Как *p*-значение используется при оптимизации моделей регрессии?
5. Поясните назначение метода `predict` класса `LinearRegression`.
6. Поясните назначение метода `plot` и `scatter` класса `pyplot`.
7. По какой подвыборке необходимо оценивать точность модели машинного обучения: тестовой или тренировочной?

Список литературы

Для выполнения лабораторной работы, при подготовке к защите, а также для ответа на контрольные вопросы рекомендуется использовать следующие источники: [1, 2, 5-7].

ЛАБОРАТОРНАЯ РАБОТА 9. ИСПОЛЬЗОВАНИЕ РАЗРАБОТАННОГО ПАЙПЛАЙНА ДЛЯ МНОГОМЕРНОЙ РЕГРЕССИИ

Цели и задачи

Цель лабораторной работы: научиться применять разработанный пайплайн для тиражирования кода с целью решения широкого круга задач машинного обучения.

Основные задачи:

- получение навыков рефакторинга кода в проектах машинного обучения;
- получение навыков определения ключевых признаков в задачах машинного обучения;
- получение навыков реализации ключевых стратегий оптимизации моделей регрессии.

Теоретическое обоснование

При решении задач многомерной регрессии исследователю необходимо решить ряд подзадач:

1. Определить коррелированность признаков.
2. Определить, какие признаки существенны при построении модели регрессии.

Проблема определения значимых признаков связана с проблемой снижения размерности.

Важное значение при многомерной регрессии приобретает обработка категориальных признаков. Часто необходимо заменить категориальный признак на набор фиктивных переменных.

К проблеме выбора значимых переменных существует несколько стратегий (фактически это методы построения модели многомерной регрессии):

1. All-in. В данном подходе производится включение всех признаков в модель.
2. Backward Elimination. В подходе предполагается обучение модели с учетом всех признаков и удаление признаков по одному на основе их значимости до достижения ситуации, когда останутся только значимые признаки.
3. Forward Selection. Подход предполагает начальное тестирование модели с одним признаком (тестируется каждый признак). Затем добавляются по одному наиболее значимые признаки.
4. Bidirectional Elimination. Подход совмещает стратегии 2 и 3.
5. Score Comparison.

Оборудование и материалы

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

Методика и порядок выполнения работы

Перед выполнением индивидуального задания рекомендуется выполнить все пункты учебной задачи.

Учебная задача

Задание. На основе разработанного пайплайна для линейной одномерной регрессии разработать многомерную модель регрессии.

Решение. Для разработки модели необходимо реализовать следующий код:

1.1 Подключение библиотек

```
1 import numpy as np
2 import matplotlib.pyplot as plt
3 import pandas as pd
```

executed in 11.9s, finished 21:21:46 2019-10-11

1.2 Загрузка данных и разделение на матрицу признаков и зависимую переменную

```
1 dataset = pd.read_csv('50_Startups.csv')
2 dataset.head()
```

executed in 14ms, finished 21:49:36 2019-10-11

	R&D Spend	Administration	Marketing Spend	State	Profit
0	165349.20	136897.80	471784.10	New York	192261.83
1	162597.70	151377.59	443898.53	California	191792.06
2	153441.51	101145.55	407934.54	Florida	191050.39
3	144372.41	118671.85	383199.62	New York	182901.99
4	142107.34	91391.77	366168.42	Florida	166187.94

```
1 X = dataset.iloc[:, :-1].values
2 y = dataset.iloc[:, 4].values
3 print ("Матрица признаков"); print(X[:5])
4 print ("Зависимая переменная"); print(y[:5])
```

executed in 6ms, finished 21:50:21 2019-10-11

Матрица признаков

```
[[165349.2 136897.8 471784.1 'New York']
 [162597.7 151377.59 443898.53 'California']
 [153441.51 101145.55 407934.54 'Florida']
 [144372.41 118671.85 383199.62 'New York']
 [142107.34 91391.77 366168.42 'Florida']]
```

1.3 Обработка пропущенных значений

```
1 # from sklearn.preprocessing import Imputer
2 # imputer = Imputer(missing_values = 'NaN', strategy = 'mean', axis = 0)
3 # imputer = imputer.fit(X[:, 1:3])
4 # X[:, 1:3] = imputer.transform(X[:, 1:3])
5 # print(X)
```

1.4 Обработка категориальных данных

1.4.1 Замена категории кодом (LabelEncoder)

```
1 # from sklearn.preprocessing import LabelEncoder
2 # labelencoder_y = LabelEncoder()
3 # print("Зависимая переменная до обработки")
4 # print(y)
5 # y = labelencoder_y.fit_transform(y)
6 # print("Зависимая переменная после обработки")
7 # print(y)
```

1.4.2 Применение OneHotEncoder

```
1 from sklearn.preprocessing import LabelEncoder, OneHotEncoder
2 labelencoder = LabelEncoder()
3 X[:, 3] = labelencoder.fit_transform(X[:, 3])
4 onehotencoder = OneHotEncoder(categorical_features = [3])
5 X = onehotencoder.fit_transform(X).toarray()
6 print("Перекодировка категориального признака")
7 print(X[:,4,:])
```

executed in 8ms, finished 21:50:24 2019-10-11

Перекодировка категориального признака

```
[[0.000000e+00 0.000000e+00 1.000000e+00 1.653492e+05 1.368978e+05
  4.717841e+05]
 [1.000000e+00 0.000000e+00 0.000000e+00 1.625977e+05 1.513775e+05
  4.438985e+05]
 [0.000000e+00 1.000000e+00 0.000000e+00 1.534415e+05 1.011455e+05
  4.079345e+05]
 [0.000000e+00 0.000000e+00 1.000000e+00 1.443724e+05 1.186718e+05
  3.831996e+05]]
```

1.5 Для предотвращения мультиколлинеарности необходимо избавиться от одной из фиктивных переменных, добавленных в результате обработки категориальных признаков

```
1 X = X[:, 1:]
2 print(X[:, 1:])
```

executed in 4ms, finished 21:50:50 2019-10-11

```
[[0.0000000e+00 1.0000000e+00 1.6534920e+05 1.3689780e+05 4.7178410e+05]
 [0.0000000e+00 0.0000000e+00 1.6259770e+05 1.5137759e+05 4.4389853e+05]
 [1.0000000e+00 0.0000000e+00 1.5344151e+05 1.0114555e+05 4.0793454e+05]
 [0.0000000e+00 1.0000000e+00 1.4437241e+05 1.1867185e+05 3.8319962e+05]]
```

1.6 Разделение выборки на тестовую и тренировочную

```
1 # from sklearn.cross_validation import train_test_split
2 from sklearn.model_selection import train_test_split
3 X_train, X_test, y_train, y_test = train_test_split(X, y, test_size = 0.2, random_state = 0)
```

executed in 3ms, finished 21:51:04 2019-10-11

1.7 Обучение линейной модели регрессии

```
1 from sklearn.linear_model import LinearRegression
2 regressor = LinearRegression()
3 regressor.fit(X_train, y_train)
```

executed in 5ms, finished 21:51:10 2019-10-11

1.8 Обработка результатов, тюнинг модели

1.8.1 Предсказание

```
1 y_pred = regressor.predict(X_test)
2 print(y_pred)
```

executed in 3ms, finished 21:51:13 2019-10-11

```
[103015.20159795 132582.27760817 132447.73845176 71976.09851257
 178537.48221058 116161.24230165 67851.69209675 98791.73374686
 113969.43533013 167921.06569553]
```

Рисунок 9.1 – Код Python для построения модели многомерной регрессии

На данный момент произведено обучение модели на всем наборе признаков. Для оптимизации модели реализуем стратегию Back Elimination (рисунок 9.2).

1.8.2 Оптимизация модели

```
1 import statsmodels.formula.api as sm
2 X = np.append(arr = np.ones((50, 1)).astype(int), values = X, axis = 1)
3 X_opt = X[:, [0, 1, 2, 3, 4, 5]]
4 regressor_OLS = sm.OLS(endog = y, exog = X_opt).fit()
5 regressor_OLS.summary()
```

executed in 22ms, finished 21:51:17 2019-10-11

	coef	std err	t	P> t	[0.025	0.975]
const	5.013e+04	6884.820	7.281	0.000	3.62e+04	6.4e+04
x1	198.7888	3371.007	0.059	0.953	-6595.030	6992.607
x2	-41.8870	3256.039	-0.013	0.990	-6604.003	6520.229
x3	0.8060	0.046	17.369	0.000	0.712	0.900
x4	-0.0270	0.052	-0.517	0.608	-0.132	0.078
x5	0.0270	0.017	1.574	0.123	-0.008	0.062

```
X_opt = X[:, [0, 1, 3, 4, 5]]
regressor_OLS = sm.OLS(endog = y, exog = X_opt).fit()
regressor_OLS.summary()
```

executed in 20ms, finished 21:52:56 2019-10-11

	coef	std err	t	P> t	[0.025	0.975]
const	5.011e+04	6647.870	7.537	0.000	3.67e+04	6.35e+04
x1	220.1585	2900.536	0.076	0.940	-5621.821	6062.138
x2	0.8060	0.046	17.606	0.000	0.714	0.898
x3	-0.0270	0.052	-0.523	0.604	-0.131	0.077
x4	0.0270	0.017	1.592	0.118	-0.007	0.061


```

X_opt = X[:, [0, 3, 4, 5]]
regressor_OLS = sm.OLS(endog = y, exog = X_opt).fit()
regressor_OLS.summary()

```

executed in 19ms, finished 21:53:11 2019-10-11

	coef	std err	t	P> t	[0.025	0.975]
const	5.012e+04	6572.353	7.626	0.000	3.69e+04	6.34e+04
x1	0.8057	0.045	17.846	0.000	0.715	0.897
x2	-0.0268	0.051	-0.526	0.602	-0.130	0.076
x3	0.0272	0.016	1.655	0.105	-0.006	0.060


```

X_opt = X[:, [0, 3, 5]]
regressor_OLS = sm.OLS(endog = y, exog = X_opt).fit()
regressor_OLS.summary()

```

executed in 20ms, finished 21:53:30 2019-10-11

	coef	std err	t	P> t	[0.025	0.975]
const	4.698e+04	2689.933	17.464	0.000	4.16e+04	5.24e+04
x1	0.7966	0.041	19.266	0.000	0.713	0.880
x2	0.0299	0.016	1.927	0.060	-0.001	0.061


```

X_opt = X[:, [0, 3]]
regressor_OLS = sm.OLS(endog = y, exog = X_opt).fit()
regressor_OLS.summary()

```

executed in 18ms, finished 21:53:49 2019-10-11

Dep. Variable:	y	R-squared:	0.947
Model:	OLS	Adj. R-squared:	0.945
Method:	Least Squares	F-statistic:	849.8
Date:	Fri, 11 Oct 2019	Prob (F-statistic):	3.50e-32
Time:	21:53:49	Log-Likelihood:	-527.44
No. Observations:	50	AIC:	1059.
Df Residuals:	48	BIC:	1063.
Df Model:	1		
Covariance Type:	nonrobust		

	coef	std err	t	P> t	[0.025	0.975]
const	4.903e+04	2537.897	19.320	0.000	4.39e+04	5.41e+04
x1	0.8543	0.029	29.151	0.000	0.795	0.913

Omnibus:	13.727	Durbin-Watson:	1.116
Prob(Omnibus):	0.001	Jarque-Bera (JB):	18.536
Skew:	-0.911	Prob(JB):	9.44e-05
Kurtosis:	5.361	Cond. No.	1.65e+05

Рисунок 9.2 – Оптимизация модели многомерной регрессии

Выполните индивидуальное задание.

Индивидуальное задание

1. Подберите набор данных на ресурсах [5-7] и согласуйте свой выбор с преподавателем. Студент может предложить набор данных в соответствии с тематикой магистерского исследования.
2. Постройте модель многомерной регрессии с использованием стратегии backward elimination.

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. Почему при реализации многомерной линейной регрессии необходимо добавить фиктивный признак с единственным значением 1.0?
2. то такое фиктивная переменная? Поясните причину удаления одной фиктивной переменной, возникающей при перекодировке категориального признака.
3. На основе какого критерия можно выбирать удаляемый признак в алгоритме back elimination.
4. В чем заключается алгоритм all-in regression?

5. В чем заключается алгоритм forward selection regression?
6. В чем заключается алгоритм Bidirectional Elimination?
7. Стратегия Backward Elimination предполагает удаление признаков на основе анализа р-критерия. Как реализовать удаление признаков в автоматическом режиме?

Список литературы

Для выполнения лабораторной работы, при подготовке к защите, а также для ответа на контрольные вопросы рекомендуется использовать следующие источники: [1, 2, 5-7].

ЛАБОРАТОРНАЯ РАБОТА 10. ПОЛИНОМИАЛЬНАЯ РЕГРЕССИЯ

Цели и задачи

Цель лабораторной работы: научиться применять разработанный пайплайн для тиражирования кода с целью решения задачи полиномиальной регрессии.

Основные задачи:

- получение навыков рефакторинга кода в проектах машинного обучения;
- изучение поведения модели полиномиальной регрессии при изменении степени полинома;
- освоение модификаций kNN-метода.

Теоретическое обоснование

Линейная и параболическая модели являются частными случаями более сложной модели – полиномиальной. Построить модель регрессии – это значит найти параметры той функции, которая будет в ней фигурировать. Для линейной регрессии – два параметра: коэффициент и свободный член.

Полиномиальная регрессия может применяться в математической статистике при моделировании трендовых составляющих временных рядов. Временной ряд – это, по сути, ряд чисел, которые зависят от времени. Например, средние значения температуры воздуха по дням за прошедший год, или доход предприятия по месяцам. Порядок моделируемого полинома оценивается специальными методами, например, критерием серий. Цель построения модели полиномиальной регрессии в области временных рядов всё та же – прогнозирование.

Указания по технике безопасности

Студенты должны следовать общепринятой технике безопасности для пользователей персональных компьютеров. Не следует самостоятельно производить ремонт технических средств, установку и удаление программного обеспечения. В случае обнаружения неисправностей необходимо сообщить об этом администратору компьютерного класса (обслуживающему персоналу лаборатории).

Методика и порядок выполнения работы

Перед выполнением индивидуального задания рекомендуется выполнить все пункты учебной задачи.

Учебная задача

Задание. На основе разработанного пайплайна для линейной одномерной регрессии разработать полиномиальную модель регрессии.

Решение. Для разработки модели необходимо реализовать следующий код:

1.1 Подключение библиотек

```
import numpy as np
import matplotlib.pyplot as plt
import pandas as pd
```

executed in 510ms, finished 22:35:43 2019-10-11

1.2 Загрузка данных и разделение на матрицу признаков и зависимую переменную

```
dataset = pd.read_csv('Position_Salaries.csv')
dataset.head()
```

executed in 23ms, finished 22:35:54 2019-10-11

	Position	Level	Salary
0	Business Analyst	1	45000
1	Junior Consultant	2	50000
2	Senior Consultant	3	60000
3	Manager	4	80000
4	Country Manager	5	110000

1.6 Обучение модели

1.6.1 Обучение линейной модели

```
from sklearn.linear_model import LinearRegression
lin_reg = LinearRegression()
lin_reg.fit(X, y)
```

executed in 884ms, finished 22:38:01 2019-10-11

```
LinearRegression(copy_X=True, fit_intercept=True, n_jobs=None,
                 normalize=False)
```

1.6.2 Обучение полиномиальной модели

```
from sklearn.preprocessing import PolynomialFeatures
poly_reg = PolynomialFeatures(degree = 10)
X_poly = poly_reg.fit_transform(X)
poly_reg.fit(X_poly, y)
lin_reg_2 = LinearRegression()
lin_reg_2.fit(X_poly, y)
```

executed in 45ms, finished 22:59:17 2019-10-11

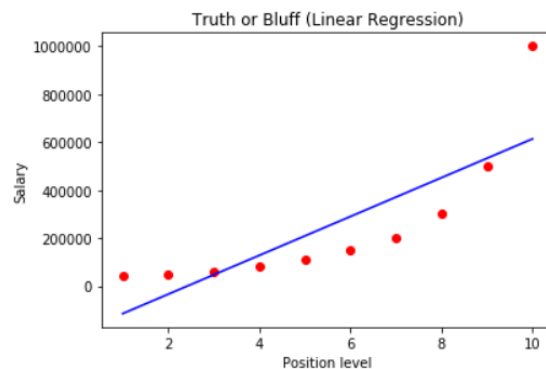
```
LinearRegression(copy_X=True, fit_intercept=True, n_jobs=None,
                 normalize=False)
```

1.7 Предсказание, обработка и визуализация результатов

```
y_pred_lin = lin_reg.predict([[6.5]])
y_pred_poly = lin_reg_2.predict(poly_reg.fit_transform([[6.5]]))
print(y_pred_lin, y_pred_poly)
plt.scatter(X, y, color = 'red')
plt.plot(X, lin_reg.predict(X), color = 'blue')
plt.title('Truth or Bluff (Linear Regression)')
plt.xlabel('Position level')
plt.ylabel('Salary')
plt.show()
```

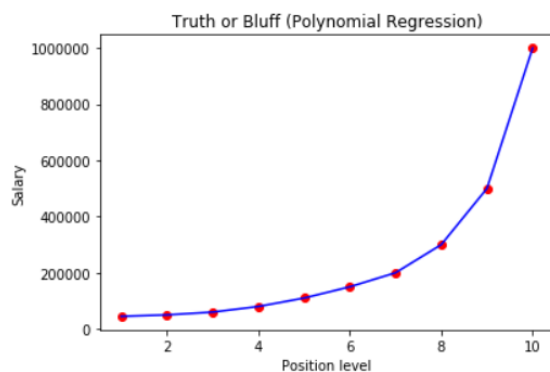
executed in 137ms, finished 22:53:36 2019-10-11

[330378.78787879] [176235.71632004]



```
plt.scatter(X, y, color = 'red')
plt.plot(X, lin_reg_2.predict(poly_reg.fit_transform(X)), color = 'blue')
plt.title('Truth or Bluff (Polynomial Regression)')
plt.xlabel('Position level')
plt.ylabel('Salary')
plt.show()
```

executed in 134ms, finished 22:59:21 2019-10-11



```
X_grid = np.arange(min(X), max(X), 0.1)
X_grid = X_grid.reshape((len(X_grid), 1))
plt.scatter(X, y, color = 'red')
plt.plot(X_grid, lin_reg_2.predict(poly_reg.fit_transform(X_grid)), color = 'blue')
plt.title('Truth or Bluff (Polynomial Regression)')
plt.xlabel('Position level')
plt.ylabel('Salary')
plt.show()
```

executed in 135ms, finished 22:59:24 2019-10-11

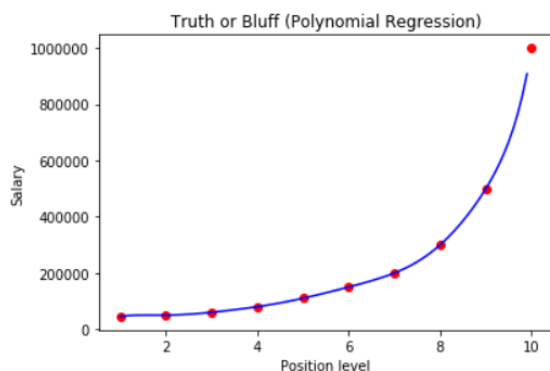
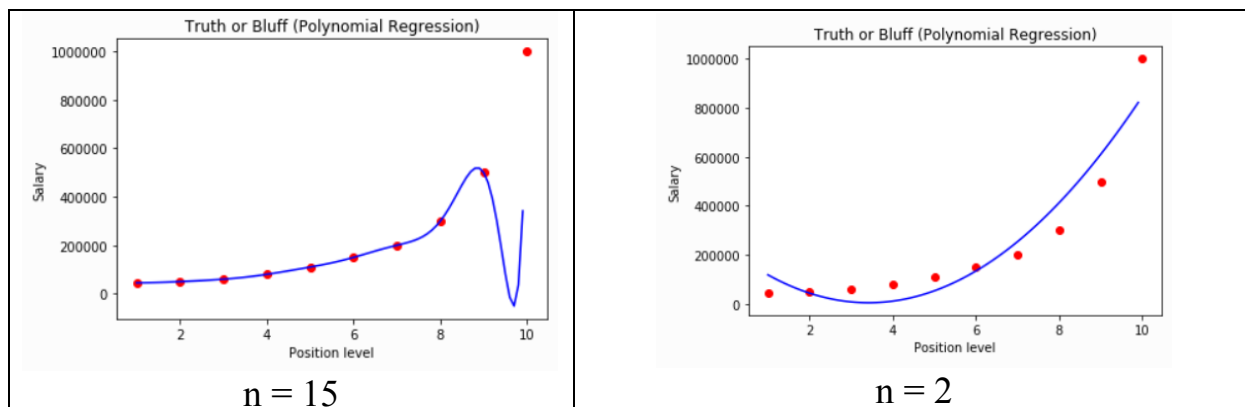


Рисунок 10.1 – Код Python для построения модели полиномиальной регрессии

Рассмотрим как изменяется модель при изменении степени аппроксимирующего полинома (рисунок 10.2).



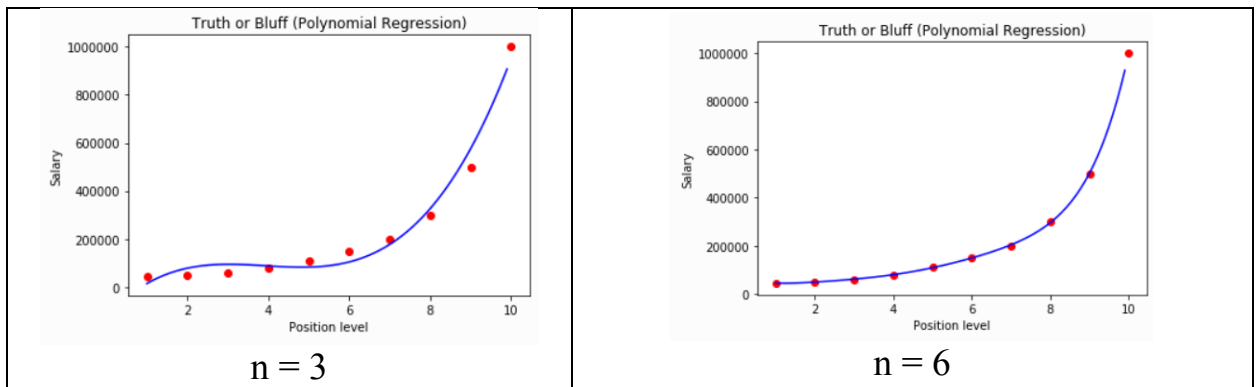


Рисунок 10.2 – Анализ модели полиномиальной регрессии

Очевидно, что при реализации полиномиальной регрессии нет необходимости в немотивированном увеличении степени аппроксимации. Выполните индивидуальное задание.

Индивидуальное задание

1. Подберите набор данных на ресурсах [5-7] и согласуйте свой выбор с преподавателем. Студент может предложить набор данных в соответствии с тематикой магистерского исследования.
2. Постройте модель полиномиальной регрессии с использованием. Проанализируйте кривые аппроксимации при различных степенях полинома.

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. Почему при реализации многомерной линейной регрессии необходимо добавить фиктивный признак с единственным значением 1.0?
2. Что такое фиктивная переменная? Поясните причину удаления одной фиктивной переменной, возникающей при перекодировке категориального признака.
3. С использованием какого класса создается модель полиномиальной регрессии?
4. Поясните принцип преобразования признаков при построении полиномиальной регрессии.
5. Возможно ли применение технологий масштабирования признаков при реализации полиномиальной регрессии?

Список литературы

Для выполнения лабораторной работы, при подготовке к защите, а также для ответа на контрольные вопросы рекомендуется использовать следующие источники: [1, 2, 5-7].

ЛАБОРАТОРНАЯ РАБОТА 11. СНИЖЕНИЕ РАЗМЕРНОСТИ

1. Цели и задачи

Цель лабораторной работы: научиться использовать обучаемые алгоритмы снижения размерности.

Основные задачи:

- получение навыков рефакторинга кода в проектах машинного обучения;
- получение навыков по использованию обучаемых алгоритмов снижения размерности;
- получение навыков по аргументированному выбору и анализу алгоритмов снижения размерности.

2. Теоретическое обоснование

2.1 Необходимость снижения размерности

Считается, что модель стоит обучать на как можно большем объеме разнообразных данных. Но иногда, если информации становится слишком много, она тормозит или останавливает процесс обучения. Чтобы модель работала нормально, нужно уменьшить количество признаков, выбрав самые эффективные. Для этого применяют методы снижения размерности.

Уменьшение размерности данных – это подход упрощения сложных наборов данных для облегчения их обработки. По мере того как данные растут и становятся более сложными, извлекать информацию становится все труднее, а визуализация становится более накладной. Методы уменьшения размерности данных решают эту проблему, предоставляя меньшее количество измерений (столбцов) при сохранении наиболее важной информации. Мы можем потерять некоторые детали, но получить более простое представление данных, которое легче обрабатывать и сравнивать.

Уменьшение размерности широко используется в области машинного обучения и анализа данных. Его цель состоит в том, чтобы упростить обработку данных за счет уменьшения количества объектов в наборе при сохранении ключевой информации. Когда мы сталкиваемся с данными большой размерности, ее уменьшение может помочь нам снизить вычислительную сложность, повысить производительность и результативность модели.

2.2 Термин «проклятие размерности»

Обучая модели, нужно предоставить им на входе переменные (например, цвет и стоимость товара). Их количество называют размерностью.

Термин «проклятие размерности» в 1961 году ввел американский математик Ричард Беллман. Он описал трудности при обработке датасетов с большим количеством параметров: возрастающую сложность вычислений, необходимость большого объема памяти для хранения датасетов, рост шума и проблему переобучения искусственных нейронных сетей.

Беллман объяснил рост размерности системы на примере единичного интервала $[0,1]$. 100 точек способны заполнить этот интервал при частоте от 0,01. Но если мы создадим модель 10-мерного куба – нам потребуется уже 10^{20} точек, то есть в 10^{18} раз больше. Беллман видел решение этой проблемы в снижении размерности пространства и переносе данных на пространства с меньшей размерностью.

При кластеризации большое количество переменных приводит к тому, что точки данных выглядят равноудаленными друг от друга. Это еще одно негативное следствие проклятия размерности.

Снижение размерности – это метод подготовки данных перед обучением модели. Оно может быть выполнено после очистки и масштабирования данных. Модель, обученная на данных с широким набором признаков, подвергается риску переобучения. Это приводит к снижению точности.

Борьба с переобучением – основная цель уменьшения размерности. Чем меньше предположений делает модель, тем она проще.

2.3 Методы снижения размерности

Для снижения размерности используют **методы выбора и проектирования переменных**. Выбор переменных – это самый простой способ уменьшения размерности. Проектирование – создание новых переменных на основе существующих путем их преобразования.

Например, нам нужна модель, которая прогнозирует средний чек. У нас есть массив данных, который подробно описывает каждого покупателя. Столбец с цветом глаз не помог бы предсказать чек. В отличие от информации о доходе покупателей.

Метод порога отклонения (Variance Threshold) – простой способ выбора переменных. Он отбрасывает все признаки, в которых дисперсия не превышает заданного порогового значения. Порог выбирается с учетом данных. Маленькая дисперсия часто встречается у признаков, в которых значение многих строк совпадает (например, признак «миллиардер», скорее всего, будет бесполезен).

Метод одномерного выбора признаков (Univariate Feature Selection) применяет статистические тесты. Он называется одномерным, потому что анализирует переменные по очереди, сравнивая их с целевым показателем. Признаки, которые слабо коррелируют с ним, отбрасываются.

Самые распространенные методы проектирования переменных используют линейные преобразования. Например, **PCA (Principal Component Analysis)**. Его применяют для уменьшения размерности континуальных данных (температура, вес, уровень гемоглобина в крови). Он позволяет снизить потерю информации при уменьшении размерности. PCA применяется, если некоторые признаки коррелируют друг с другом. Он создает из них независимые линейные комбинации. Часто дисперсии получившихся признаков сильно отличаются. В таком случае признаки с

низкой дисперсией можно отбросить, потому что они менее информативны. Этот метод решает две задачи: создает более информативные признаки и уменьшает шум в данных.

Факторный анализ (Factor analysis), как и PCA, создает новые признаки, комбинируя существующие. Его отличие в том, что ФА предполагает наличие реальных скрытых факторов, которые определяют наблюдаемые. Кроме того, он объясняет часть дисперсии наблюдаемых параметров ошибками измерения, а часть – реальной дисперсией скрытых факторов. Данные должны быть количественными, измеряться по интегральной шкале или шкале отношений. Категориальные данные (религия, место рождения, пол) не подходят для этого метода.

Например, когда человек выбирает мебель, его решение основано на ряде факторов: его возрасте, доходе, площади квартиры. Это скрытые факторы (их не знают владельцы магазина), и задача анализа — выявить их. Используйте этот метод, если считаете, что такие факторы существуют.

LDA (Linear Discriminant Analysis) используется для задач классификации. Этот метод часто применяют для уменьшения размерности при предварительной обработке данных. Цель – спроецировать набор данных на пространство меньшей размерности с хорошей разделимостью кластеров. Метод делает предположения о нормально распределенных классах и равных ковариациях классов. Его можно применять даже в случаях, когда классов больше двух.

Методы нелинейного преобразования используются, когда данные не лежат в линейном подпространстве. Если линейное подпространство трехмерного пространства можно представить как плоский лист бумаги, то примером нелинейного многообразия будет свернутый лист.

К нелинейным методам относится, например, **многомерное масштабирование (MDS)**, **изометрическое отображение объектов (Isomap)**, **локально-линейное вложение (LLE)**, **собственное отображение**

Гессе, t-SNE (стохастическое вложение соседей с t-распределением), UMAP (Uniform Manifold Approximation and Projection) и другие.

Популярный метод уменьшения размерности, который дает впечатляющие результаты, – автокодировщик. Это тип искусственной нейросети, который обучается копировать свои входные данные в выходные, имея уменьшенный, по сравнению с входным и выходным, промежуточный слой. Автокодировщик сжимает входные данные в значения весов промежуточного слоя, а затем реконструирует выходные данные из этих значений.

3. Методика и порядок выполнения работы

3.1 Учебная задача

Задание. На основе предоставленного набора данных wineQualityReds.csv выполните снижение размерности. Исходный набор данных содержит сведения о различных марках вин. В наборе присутствуют признаки, представленные в таблице 9.1.

Таблица 9.1 – Признаки набора данных wineQualityReds.csv

Признак набора данных	Описание
fixed.acidity	фиксированная кислотность (г/дм ³)
volatile.acidity	летучая кислотность (г /дм ³)
citric.acid	лимонная кислота (г /дм ³)
residual.sugar	остаточный сахар (г /дм ³)
chlorides	хлориды (г/дм ³)
free.sulfur.dioxide	свободный диоксид серы (мг/дм ³)
total.sulfur.dioxide	общий диоксид серы (мг/дм ³)
density	плотность (г/см ³)
pH	pH
sulphates	сульфаты (г/дм ³)
alcohol	спирт (% по объему) Выходная переменная (на основе сенсорных данных)
quality	качество (оценка от 0 до 10)

Решение.

Различные методы понижения размерности представлены в листинге:

<https://github.com/enikolaev/MMO/blob/main/Lesson%2010.%20Dimensionality%20Reduction/LabWork10.ipynb>

3.2 Индивидуальное задание

1. Подберите набор данных на ресурсах [5-7] и согласуйте свой выбор с преподавателем. Студент может предложить синтезированный набор данных.
2. Реализуйте процедуры понижения размерности с использованием согласованного набора данных.
3. Продемонстрируйте отличия в использовании многомерного набора данных при обучении модели и набора данных пониженной размерности. Например на задаче классификации.

4. 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.

5. Контрольные вопросы

1. Что такое проклятие размерности?
2. Перечислите основные алгоритмы понижения размерности.
3. Опишите алгоритм и основную идею метода PCA.
4. Опишите концепцию метода t-SNE.

6. Список литературы

Для выполнения лабораторной работы, при подготовке к защите, а также для ответа на контрольные вопросы рекомендуется использовать следующие источники: [1, 2, 5-7].

ЛАБОРАТОРНАЯ РАБОТА 12. АНАЛИЗ МНОГОМЕРНЫХ ДАННЫХ С ИСПОЛЬЗОВАНИЕМ МЕТОДОВ СНИЖЕНИЯ РАЗМЕРНОСТИ

1. Цели и задачи

Цель лабораторной работы: научиться применять обучаемые алгоритмы снижения размерности для визуализации наборов данных с большой размерности.

Основные задачи:

- получение навыков рефакторинга кода в проектах машинного обучения;
- получение навыков по использованию обучаемых алгоритмов снижения размерности;
- получение навыков по аргументированному выбору и анализу алгоритмов снижения размерности.

2. Теоретическое обоснование

Необходимо повторить теоретический материал работы 11.

3. Методика и порядок выполнения работы

3.1 Учебная задача

Задание. На основе предоставленного набора данных Fashion-MNIST выполните снижение размерности. Исходный набор данных содержит изображения размером 28x28. Для понимания структуры данных, понимания возможности их использования для решения задач кластеризации, классификации, необходимо выполнить визуализацию данных изображений с отображением расстояния между отдельными экземплярами.

Решение.

Применение метода PCA представлено по ссылке:

<https://github.com/enikolaev/MMO/blob/main/Lesson%2011.%20Dimensionality%20Reduction%20for%20Complex%20data%20visualization/LabWork11.ipynb>

3.2 Индивидуальное задание

1. Реализуйте по аналогии с учебной задачей визуализацию набора данных Fashion-MNIST с использованием методов снижения размерности: ICA, TruncatedSVD, MDS, t-SNE, UMAP.

2. Подберите набор данных на ресурсах [5-7] и согласуйте свой выбор с преподавателем. Необходимо, чтобы набор данных содержал изображения. Не рекомендуется выбирать MNIST или Fashion-MNIST.

3. Реализовать визуализацию выбранного набора данных с использованием нескольких методов снижения размерности: ICA, TruncatedSVD, MDS, t-SNE, UMAP.

4. Выполните сравнение используемых методов снижения размерности. Проанализируйте, какие метрики качества можно использовать, чтобы сравнить рассмотренные методы. Каким образом можно сделать вывод, что один метод лучше другого?

4. 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.

5. Контрольные вопросы

1. Что такое проклятие размерности?
2. Перечислите основные алгоритмы понижения размерности.
3. Опишите алгоритм и основную идею метода PCA.
4. Опишите концепцию метода t-SNE.
5. Существует ли предел снижения размерности?

6. Список литературы

Для выполнения лабораторной работы, при подготовке к защите, а также для ответа на контрольные вопросы рекомендуется использовать следующие источники: [1, 2, 5-7].

ЛАБОРАТОРНАЯ РАБОТА 13. МНОГОСЛОЙНАЯ НЕЙРОННАЯ СЕТЬ

1. Цели и задачи

Цель лабораторной работы: научиться применять простейшую модель искусственной нейронной сети для решения задачи классификации.

Основные задачи:

- научиться использовать возможности стандартной библиотеки для построения искусственной нейронной сети;
- исследование процесса обучения искусственной нейронной сети;
- получение навыков работы с параметрами и гиперпараметрами искусственной нейронной сети;
- решение задачи классификации данных с использованием искусственной нейронной сети.

2. Теоретическое обоснование

Многослойный персептрон (Multi-Layered Perseptron, MLP) – это тип искусственной нейронной сети, состоящей из нескольких слоев взаимосвязанных искусственных нейронов (рисунок 13.1).

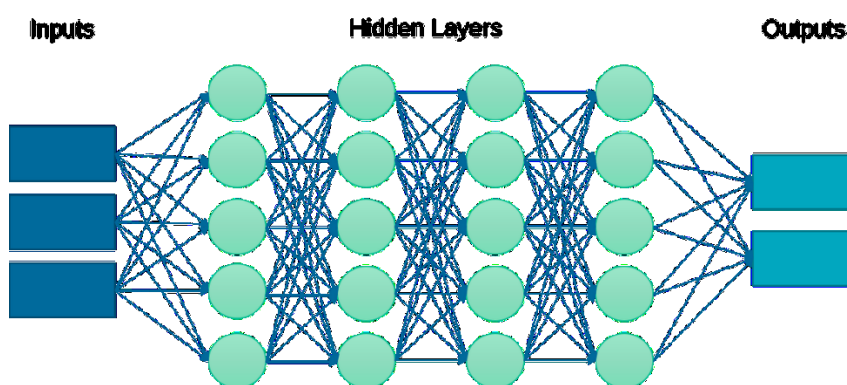


Рисунок 13.1 – Схема многослойного персептрона

Это нейронная сеть с прямой связью, что означает, что информация течет только в одном направлении от входа к выходу, без петель. Входной слой MLP состоит из нейронов, которые представляют входные данные, а

выходной слой состоит из нейронов, которые производят окончательный прогноз. Между входным и выходным слоями есть один или несколько скрытых слоев, выполняющих промежуточные вычисления над входными данными. Каждый нейрон в MLP получает входные данные от предыдущего слоя, применяет нелинейную функцию активации к взвешенной сумме этих входных данных, и создает один вывод, который передается на следующий уровень (рисунок 13.2).

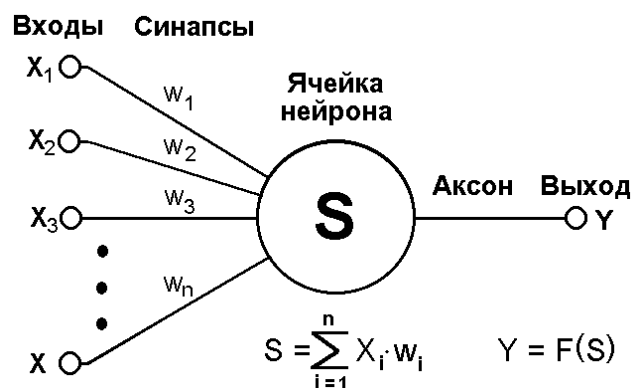


Рисунок 13.2 – Схема нейрона искусственной нейронной сети

На рисунке 13.3 показана схема биологической нервной клетки.

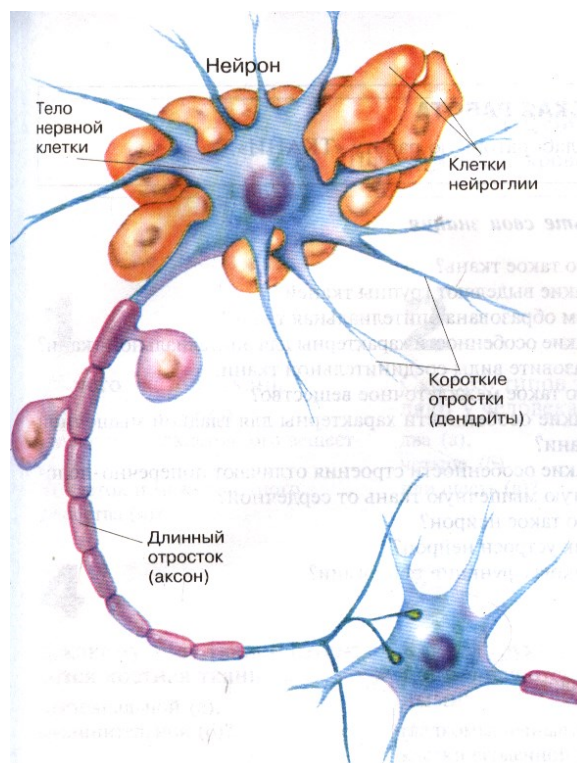


Рисунок 13.3 – Схема биологической нервной клетки

MLP обычно используются для контролируемых задач обучения, таких как классификация и регрессия, где цель состоит в том, чтобы предсказать целевую переменную на основе входных признаков. Их также можно использовать для неконтролируемых задач обучения, таких как уменьшение размерности, где цель состоит в том, чтобы изучить низкоразмерное представление данных.

3. Методика и порядок выполнения работы

3.1 Учебная задача

Задание. Для синтетического набора данных реализовать решение задачи классификации с использованием нейросетевой модели библиотеки scikit-learn.

Решение.

Применение MLPClassifier библиотеки scikit-learn представлено по ссылке:

<https://github.com/enikolaev/MMO/blob/main/12.%20MLP/LabWork12.ipynb>

3.2 Индивидуальное задание

1. Реализуйте обучение моделей для синтетических наборов данных различной природы (make_circles, make_moons и прочие). Выполните визуализацию границ решения.

2. Реализуйте несколько моделей искусственных нейронных сетей с различными гиперпараметрами и архитектурами (Изменяйте количество слоев, количество нейронов в скрытых слоях). Сравните области решений обученных моделей.

3. Реализуйте обучение одной модели нейронной сети с использованием различных алгоритмов оптимизации. Сравните получившиеся области решения.

4. 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.

5. Контрольные вопросы

1. Опишите архитектуру многослойного персептрона.
2. Опишите структуру искусственного нейрона. Каким образом элементы данной структуры определяются в классе `MLPClassifier`?
3. С использованием какого параметра можно определить метод оптимизации в `MLPClassifier` и какие оптимизаторы можно использовать?
4. С использованием какого параметра можно определить функцию активации в `MLPClassifier` и какие функции можно использовать?
5. Опишите параметры и их назначение в методе `make_classification`, создающего синтетический набор данных.
6. Опишите параметры и их назначение при создании классификатора с использованием `MLPClassifier`.

6. Список литературы

Для выполнения лабораторной работы, при подготовке к защите, а также для ответа на контрольные вопросы рекомендуется использовать следующие источники: [1, 2, 5-7].

ЛАБОРАТОРНАЯ РАБОТА 14. РАЗРАБОТКА НЕЙРОНА НА PYTORCH

Цели и задачи

Цель лабораторной работы: изучение возможностей библиотеки PyTorch.

Основные задачи:

- разработка нейрона с использованием PyTorch;
- получение навыков использования различных функций потерь;
- исследование работы простейшего нейрона на различных наборах данных.

Теоретическое обоснование

Определим важнейшие аспекты, которые играют принципиальную роль в построении любой нейронной сети (все их исследователь задает явно):

- непосредственно, сама архитектура нейросети (сюда входят типы функций активации у каждого нейрона);
- начальная инициализация весов каждого слоя;
- метод оптимизации нейросети (сюда ещё входит метод изменения `learning_rate`);
- размер батчей (`batch_size`);
- количество итераций обучения (`num_epochs`);
- функция потерь (`loss`);
- тип регуляризации нейросети (для каждого слоя можно свой).

Кроме вышеуказанных аспектов, важное значение приобретают факторы, связанные с самой задачей и данными:

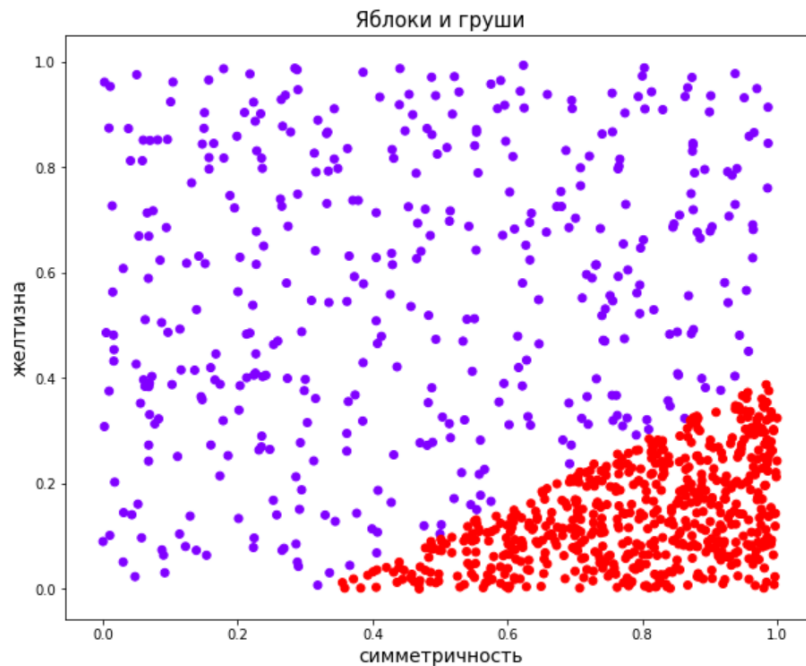
- качество выборки (непротиворечивость, чистота, корректность постановки задачи);
- размер выборки.

Пример построения классификатора с использованием одного нейрона
(в задаче используется набор данных «Яблоки и груши»):

```
import pandas as pd
from matplotlib import pyplot as plt
```

```
data = pd.read_csv("../data/apples_pears.csv")
```

```
plt.figure(figsize=(10, 8))
plt.scatter(data.iloc[:, 0], data.iloc[:, 1], c=data['target'], cmap='rainbow')
plt.title('Яблоки и груши', fontsize=15)
plt.xlabel('симметричность', fontsize=14)
plt.ylabel('желтизна', fontsize=14)
plt.show();
```



```
X = data.iloc[:, :2].values # матрица объекты-признаки
y = data['target'].values.reshape((-1, 1)) # классы (столбец из нулей и единиц)
```

В модуле `torch.nn` лежат все необходимые вещи для конструирования нейронок, а в модуле `torch.optim` лежат все необходимые вещи для выбора метода оптимизации нейросети:

```
import torch
from torch.nn import Linear, Sigmoid
```

Есть два пути объявления нейросетей в PyTorch:

- функциональный (Functional);
- последовательный (Sequential).

Рассмотрим второй путь (он чуть более user-friendly), к первому потом ещё вернёмся, и построим таким способом один нейрон (точно такой же, который мы реализовывали раньше с помощью Numpy):

```
num_features = X.shape[1]

neuron = torch.nn.Sequential(
    Linear(num_features, out_features=1),
    Sigmoid()
)

neuron

Sequential(
  (0): Linear(in_features=2, out_features=1, bias=True)
  (1): Sigmoid()
)
```

Пока что мы просто создали объект класса Sequential, который состоит из одного линейного слоя размерности (num_features, 1) и последующего применения сигмоиды. Но уже сейчас его можно применить к объекту (тензору), просто веса в начале инициализируются случайно и при forward_pass мы получим какой-то ответ пока что необученного нейрона:

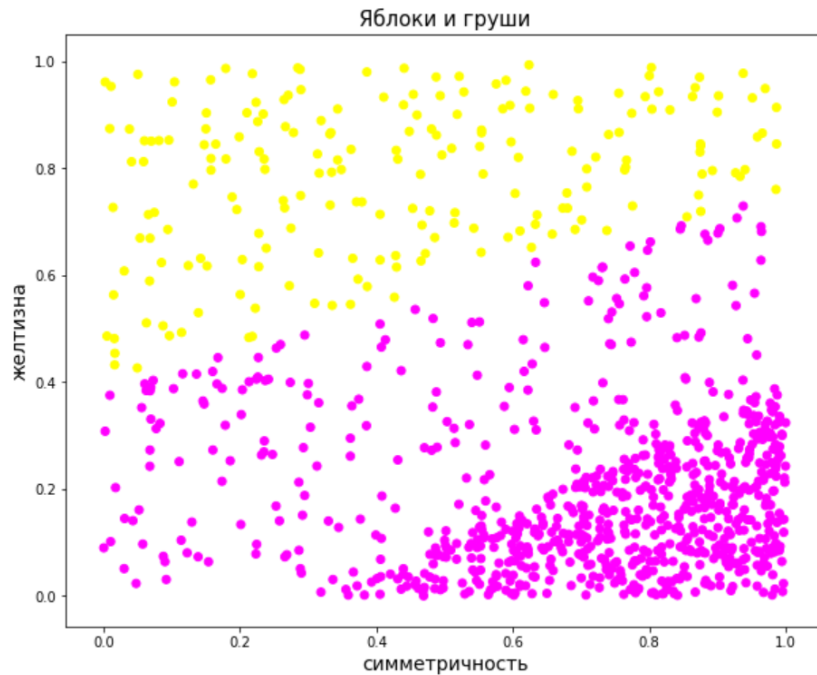
```
neuron(torch.autograd.Variable(torch.FloatTensor([1, 1])))

tensor([ 0.5060])
```

Реализуем предсказание с использованием необученного нейрона:

```
proba_pred = neuron(torch.autograd.Variable(torch.FloatTensor(X)))
y_pred = proba_pred > 0.5
y_pred = y_pred.data.numpy().reshape(-1)

plt.figure(figsize=(10, 8))
plt.scatter(data.iloc[:, 0], data.iloc[:, 1], c=y_pred, cmap='spring')
plt.title('Яблоки и груши', fontsize=15)
plt.xlabel('симметричность', fontsize=14)
plt.ylabel('желтизна', fontsize=14)
plt.show();
```



Как и ожидалось, ничего полезного. Давайте научим нейрон отличать груши от яблок по их симметричности и желтизне. Обернём данные в `torch.Tensor`, а тензоры в `torch.Variable`, чтобы можно было вычислять градиенты по весам:

```
X = torch.autograd.Variable(torch.FloatTensor(X))
y = torch.autograd.Variable(torch.FloatTensor(y))
```

Код обучения одного нейрона на PyTorch:

```
# квадратичная функция потерь (можно сделать другую)
loss_fn = torch.nn.MSELoss(size_average=False)

# шаг градиентного спуска (точнее -- метода оптимизации)
learning_rate = 0.001 # == 1e-3
# сам метод оптимизации нейросети (обычно лучше всего по-умолчанию работает Adam)
optimizer = torch.optim.SGD(neuron.parameters(), lr=learning_rate)

# итерируемся num_epochs раз, здесь 500
for t in range(500):
    # forward_pass() -- применение нейросети (этот шаг ещё называют inference)
    y_pred = neuron(X)

    # выведем Loss
    loss = loss_fn(y_pred, y)
    print('{} {}'.format(t, loss.data))

    # ВСЕГДА обнуляйте градиенты перед backward_pass'ом
    # подробнее: читайте документацию PyTorch
    optimizer.zero_grad()
```

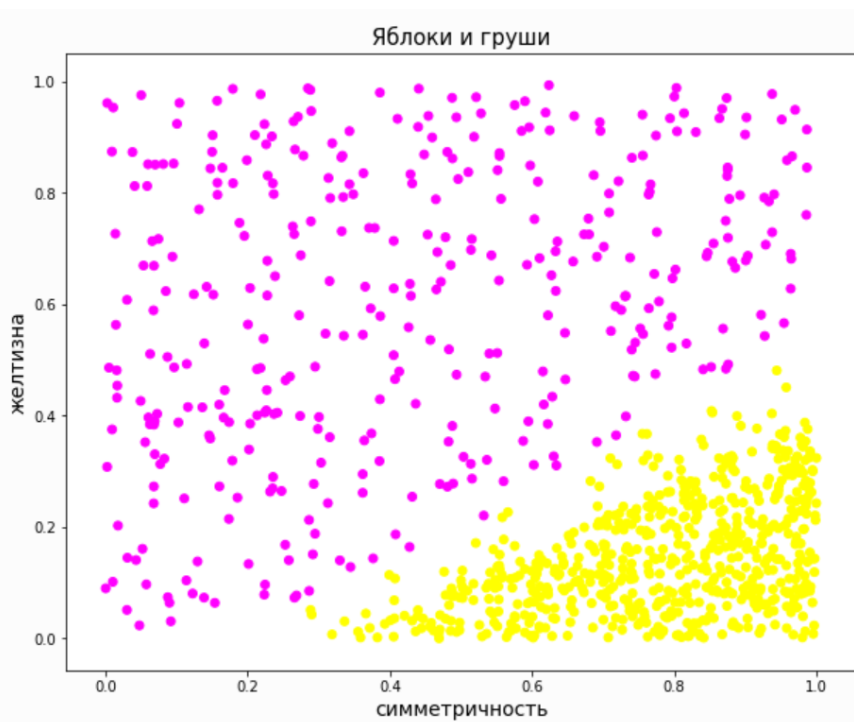
```
# backward_pass() -- вычисляем градиенты Loss'a по параметрам (весам) нейросети
# ВНИМАНИЕ! На этом шаге мы только вычисляем градиенты, но ещё не обновляем веса
loss.backward()

# А вот тут уже обновляем
optimizer.step()
```

После обучения можно выполнить предсказание:

```
proba_pred = neuron(X)
y_pred = proba_pred > 0.5
y_pred = y_pred.data.numpy().reshape(-1)

plt.figure(figsize=(10, 8))
plt.scatter(data.iloc[:, 0], data.iloc[:, 1], c=y_pred, cmap='spring')
plt.title('Яблоки и груши', fontsize=15)
plt.xlabel('симметричность', fontsize=14)
plt.ylabel('желтизна', fontsize=14)
plt.show();
```



Оборудование и материалы

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

Указания по технике безопасности

Студенты должны следовать общепринятой технике безопасности для пользователей персональных компьютеров. Не следует самостоятельно производить ремонт технических средств, установку и удаление программного обеспечения. В случае обнаружения неисправностей необходимо сообщить об этом администратору компьютерного класса (обслуживающему персоналу лаборатории).

Методика и порядок выполнения работы

Учебная задача

В разделе «Теоретическое обоснование» получен нейрон с неоптимальным качеством классификации. Добейтесь идеального качества (или очень близкого к идеальному), путём изменения `learning_rate` и количества итераций (возможно, ещё функции потерь и функции активации).

Используйте функцию потерь `torch.nn.NLLLoss()` (`== LogLoss`) или `torch.nn.CrossEntropy()`, сделайте вывод о работе нейрона.

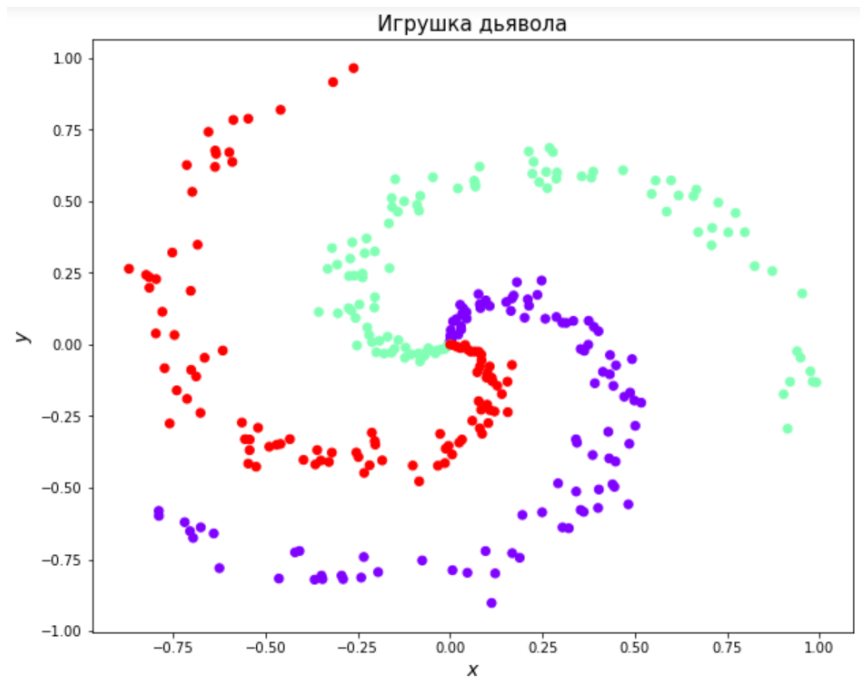
Индивидуальное задание

Протестируйте эффективность разработанной ИНС на основе одного нейрона на более сложном наборе данных (т.н. игрушка дьявола):

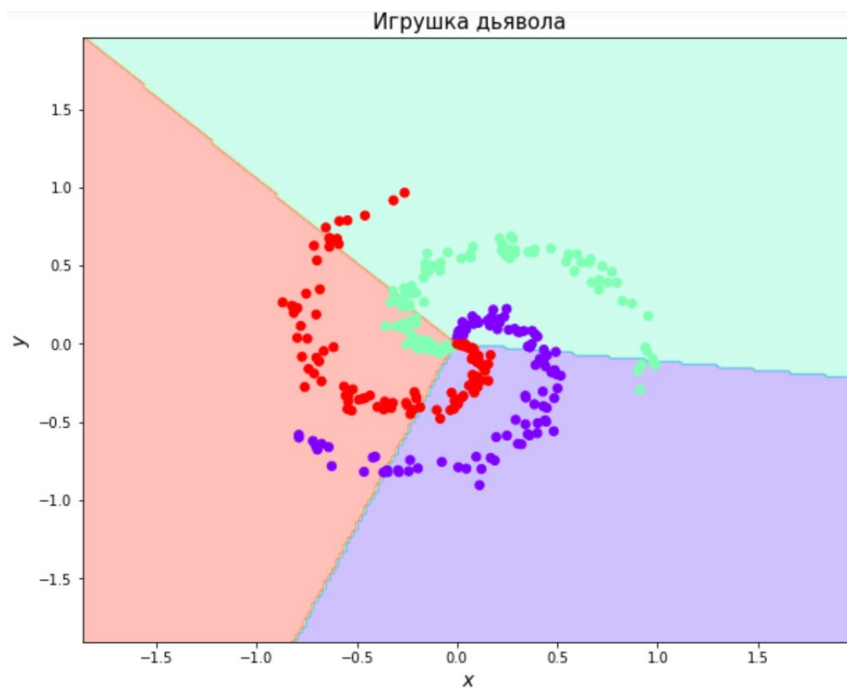
```
N = 100
D = 2
K = 3
X = np.zeros((N * K, D))
y = np.zeros(N * K, dtype='uint8')

for j in range(K):
    ix = range(N * j, N * (j + 1))
    r = np.linspace(0.0, 1, N)
    t = np.linspace(j * 4, (j + 1) * 4, N) + np.random.randn(N) * 0.2 # theta
    X[ix] = np.c_[r * np.sin(t), r * np.cos(t)]
    y[ix] = j

plt.figure(figsize=(10, 8))
plt.scatter(X[:, 0], X[:, 1], c=y, s=40, cmap=plt.cm.rainbow)
plt.title('Игрушка дьявола', fontsize=15)
plt.xlabel('$x$', fontsize=14)
plt.ylabel('$y$', fontsize=14)
plt.show();
```



Было получено следующее качество классификации:



Используя все доступные способы улучшения качества нейросетевой модели, получите более оптимального качества (при этом архитектуру менять нельзя).

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 аспекты работы с нейросетевыми моделями влияют на качество нейросетевых моделей, но не зависят от свойств набора данных?

2. Какие аспекты работы с нейросетевыми моделями влияют на качество нейросетевых моделей и зависят от свойств набора данных?

3. Какие стратегии определения нейронной сети существуют в PyTorch?

4. Какой код лучше использовать при загрузке данных из csv-файла?

а) `dataset = read_csv("data.csv")`

б) `dataset = import("data.csv")`

в) `dataset = read.csv("data.csv")`

г) `dataset = import.csv("data.csv")`

д) `dataset = read_xls("data.csv")`

Список литературы

Для выполнения лабораторной работы, при подготовке к защите, а также для ответа на контрольные вопросы рекомендуется использовать следующие источники: [1, 2, 5-7].

ЛАБОРАТОРНАЯ РАБОТА 15. МНОГОСЛОЙНЫЙ ПЕРСЕПТРОН НА PYTORCH

Цели и задачи

Цель лабораторной работы: реализация нейронной сети типа многослойный персептрон.

Основные задачи:

- реализовать многослойный персептрон средствами библиотеки PyTorch;
- получение навыков по обучению многослойного персептрона;
- исследование влияния различных гиперпараметров на обучение многослойного персептрона.

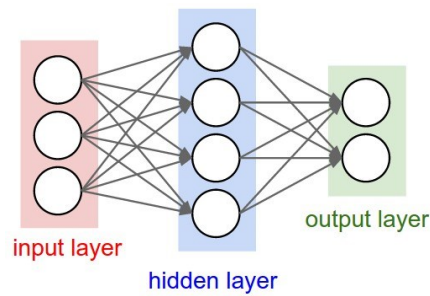
Теоретическое обоснование

В отличие от лабораторной работы №13, в данной работе не предполагается использования готовой модели из библиотеки sklearn. ИНС будет создаваться на основе библиотеки PyTorch.

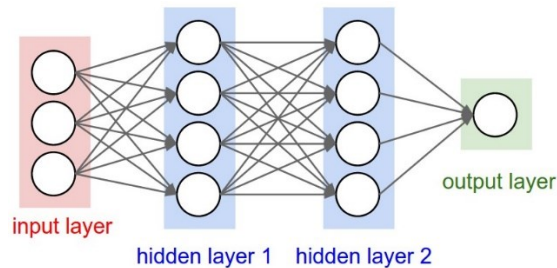
Как можно понять из названия, многослойная нейросеть состоит из нескольких слоёв. Каждый слой состоит из нейронов. Ранее мы уже писали свой нейрон, вот из таких нейронов и состоит MLP (Multi-Layer Perceptron). Ещё такую многослойную нейросеть, у которой каждый нейрон на предыдущем уровне соединён с нейроном на следующем уровне, называют Fully-Connected-сетью (или Dense-сетью).

Рассмотрим их устройство более подробно:

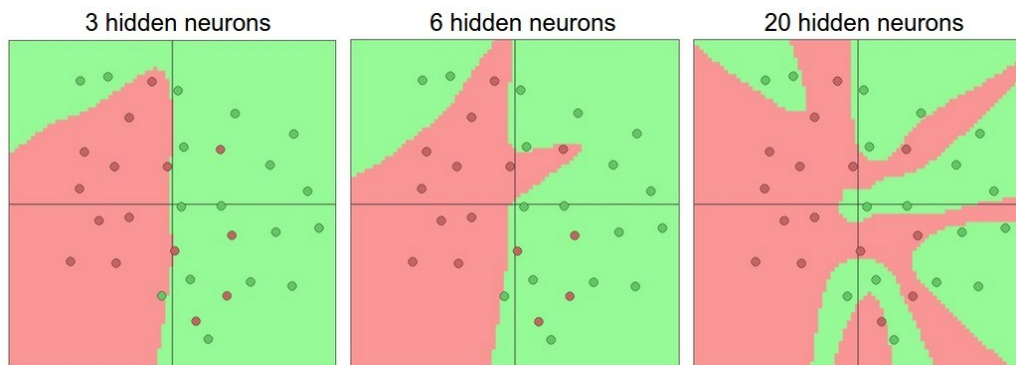
Вот так выглядит двухслойная нейросеть (первый слой – input layer – не считается, потому что это, по сути, не слой):



Так выглядит трёхслойная нейросеть:



Пример границ классификации при использовании MLP различной глубины показан на рисунке:



Обратите внимание: связи есть у нейронов со слоя L_{i-1} и нейронов L_i , но между нейронами в одном слое связей нет. Входной слой – это данные (матрица (n,m)). Слои, которые не являются входными или выходными, называются скрытыми слоями (hidden layers).

При решении задачи регрессии на выходном слое обычно один нейрон, который возвращает предсказанные числа (для каждого объекта по числу).

В случае задачи классификации на выходном слое обычно один нейрон, если задача бинарной классификации, и K нейронов, если задача K -класовой классификации.

Forward pass в MLP (прямое распространение сигнала в MLP)

Каждый слой многослойной нейросети – это матрица весов, строки которой – это нейроны (одна строка – один нейрон), а столбцы – веса каждого нейрона (то есть одна строка – это веса одного нейрона).

Допустим, мы решаем задачу K -классовой классификации (на последнем слое K нейронов). Рассмотрим, как в таком случае выглядит forward_pass нейросети:

Вход – это матрица (N, M) .

$$X = \begin{bmatrix} x_{11} & \cdots & x_{1M} \\ \vdots & \ddots & \vdots \\ x_{N1} & \cdots & x_{NM} \end{bmatrix}$$

Структура сети – это множество слоёв, в слоях много нейронов. Первый слой (после входного) выглядит так (матрица (M, L_1)):

$$W^1 = \begin{bmatrix} w_{11} & \cdots & w_{ML_1} \\ \vdots & \ddots & \vdots \\ w_{M1} & \cdots & w_{ML_1} \end{bmatrix}$$

То есть это в точности L_1 нейронов, каждый имеет свои собственные веса, их M штук.

Следует понимать, что нейрон – это линейное преобразование и потом нелинейная функция активации от этого преобразования. Однако в многослойных нейростях часто отделяют Linear часть и Activation, то есть слоем считаем набор весов нейронов, а следующий слой всегда функция активации (у всех нейронов из слоя она одна и та же, обычно фреймворки не позволяют задавать конкретному нейрону в слое отличную от других нейронов в этом слое функцию активации, однако это легко сделать, объявив слой из одного нейрона).

Другие слои выглядят точно так же, как первый слой. Например, второй слой будет такой (матрица (L_1, L_2)):

$$W^1 = \begin{bmatrix} w_{11} & \cdots & w_{1L_2} \\ \vdots & \ddots & \vdots \\ w_{L_11} & \cdots & w_{L_1L_2} \end{bmatrix}$$

То есть это в точности L_2 нейронов, каждый имеет свои собственные веса, их L_1 штук.

Выходной слой. Пусть в нейросети до выходного слоя идут t слоёв. Тогда выходной слой имеет форму:

$$W^{out} = \begin{bmatrix} w_{11} & \cdots & w_{1K} \\ \vdots & \ddots & \vdots \\ w_{L_t 1} & \cdots & w_{L_t K} \end{bmatrix}$$

То есть, выходной слой представляется матрицей (L_t, K) , где L_t – количество нейронов в t -ом слое, а K – количество классов.

В итоге для forward_pass нам нужно просто последовательно перемножить матрицы друг за другом, применяя после каждого умножения соответствующую функцию активации.

Примечание: можно думать об умножении на очередную матрицу весов как на переход в новое признаковое пространство. Действительно, когда подаём матрицу X и умножаем на матрицу первого слоя, мы получаем матрицу размера (N, L_1) , то есть как будто L_1 «новых» признаков (построенных как линейная комбинация старых до применения функции активации, и уже как нелинейная комбинация после активации).

Backward pass в MLP (обратное распространение)

Обучается с помощью метода «Error Backpropagation» (Обратное распространение ошибки), принцип распространения очень похож на то, как мы обучали один нейрон – это градиентный спуск, но по «всей нейросети» сразу.

Backpropagation работает корректно благодаря chain rule (по правилу взятия производной сложной функции):

Оборудование и материалы

Для выполнения лабораторной работы рекомендуется использовать персональный компьютер со следующими программными средствами

разработки (выбрать один или несколько программных продуктов для практической реализации задач лабораторной работы): MS Visual Studio 2015 и выше; среда разработки Java, интерпретатор Python (Anaconda) с библиотеками matplotlib, seaborn, numpy, PyTorch.

Указания по технике безопасности

Студенты должны следовать общепринятой технике безопасности для пользователей персональных компьютеров. Не следует самостоятельно производить ремонт технических средств, установку и удаление программного обеспечения. В случае обнаружения неисправностей необходимо сообщить об этом администратору компьютерного класса (обслуживающему персоналу лаборатории).

Методика и порядок выполнения работы

Перед выполнением индивидуального задания рекомендуется выполнить все пункты учебной задачи.

Учебная задача

Требуется спроектировать и обучить нейронную сеть на основе архитектуры MLP для решения задачи классификации на наборе данных «игрушка дьявола».

Пример реализации нейронной сети на языке Python:

```
import matplotlib.pyplot as plt
import numpy as np

import torch
```

```

N = 100
D = 2
K = 3
X = np.zeros((N * K, D))
y = np.zeros(N * K, dtype='uint8')

for j in range(K):
    ix = range(N * j, N * (j + 1))
    r = np.linspace(0.0, 1, N)
    t = np.linspace(j * 4, (j + 1) * 4, N) + np.random.randn(N) * 0.2 # theta
    X[ix] = np.c_[r * np.sin(t), r * np.cos(t)]
    y[ix] = j

```

Считаем градиенты:

```

X = torch.autograd.Variable(torch.FloatTensor(X))
y = torch.autograd.Variable(torch.LongTensor(y.astype(np.int64)))

```

```

print(X.data.shape, y.data.shape)

```

```

torch.Size([300, 2]) torch.Size([300])

```

Процесс обучения MLP:

```

# N - размер батча (batch_size, нужно для метода оптимизации);
# D_in - размерность входа (количество признаков у объекта);
# H - размерность скрытых слоёв;
# D_out - размерность выходного слоя (суть - количество классов)
N, D_in, H, D_out = 64, 2, 100, 3

# Use the nn package to define our model and loss function.
two_layer_net = torch.nn.Sequential(
    torch.nn.Linear(D_in, H),
    torch.nn.ReLU(),
    torch.nn.Linear(H, D_out),
).cuda()

loss_fn = torch.nn.CrossEntropyLoss(size_average=False)

learning_rate = 1e-4
optimizer = torch.optim.SGD(two_layer_net.parameters(), lr=learning_rate)
for t in range(500):
    # forward
    y_pred = two_layer_net(X)

    # loss
    loss = loss_fn(y_pred, y)
    print('{} {}'.format(t, loss.data))

```

```

optimizer.zero_grad()

```

```

# backward
loss.backward()

```

```

# ОБНОВЛЯЕМ!
optimizer.step()

```

Несмотря на то, что это задача 3-х классовой классификации и столбец *y* нужно по-хорошему кодировать посредством OneHotEncoding, в данном

случае использовали просто столбец из 0, 1 и 2 и всё отработало. Вывод – PyTorch сам делает OneHot в таком случае.

Проверяем, насколько хороша наша сеть из 100 нейронов:

```
# Обратно в NumPy для отрисовки
X = X.data.numpy()
y = y.data.numpy()

h = 0.02
x_min, x_max = X[:, 0].min() - 1, X[:, 0].max() + 1
y_min, y_max = X[:, 1].min() - 1, X[:, 1].max() + 1

xx, yy = np.meshgrid(np.arange(x_min, x_max, h),
                     np.arange(y_min, y_max, h))
grid_tensor = torch.FloatTensor(np.c_[xx.ravel(), yy.ravel()])

Z = two_layer_net(torch.autograd.Variable(grid_tensor))
Z = Z.data.numpy()
Z = np.argmax(Z, axis=1)
Z = Z.reshape(xx.shape)

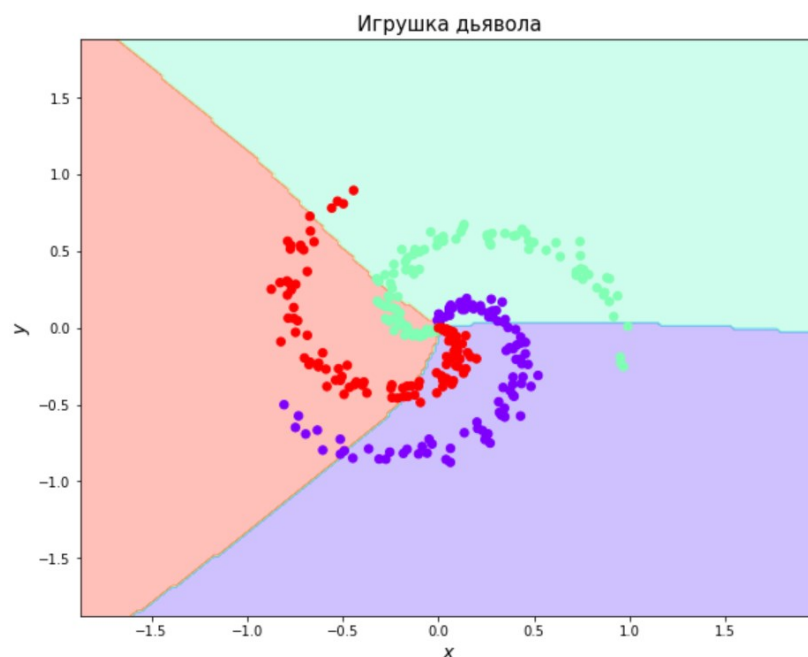
plt.figure(figsize=(10, 8))

plt.contourf(xx, yy, Z, cmap=plt.cm.rainbow, alpha=0.3)
plt.scatter(X[:, 0], X[:, 1], c=y, s=40, cmap=plt.cm.rainbow)

plt.xlim(xx.min(), xx.max())
plt.ylim(yy.min(), yy.max())

plt.title('Игрушка дьявола', fontsize=15)
plt.xlabel('$x$', fontsize=14)
plt.ylabel('$y$', fontsize=14)
plt.show();
```

Результат классификации с использованием MLP:



Индивидуальное задание

1. На основе учебной задачи разработайте нейронную сеть архитектуры MLP, но выполняющую классификацию более эффективно.
2. После выполнения задания 1 оцените эффективность спроектированной эффективной ИНС на наборе данных, выбранном в соответствии с тематикой магистерского исследования.

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 аспекты работы с нейросетевыми моделями влияют на качество нейросетевых моделей, но не зависят от свойств набора данных?
2. Какие аспекты работы с нейросетевыми моделями влияют на качество нейросетевых моделей и зависят от свойств набора данных?
3. Какие стратегии определения нейронной сети существуют в PyTorch?
4. Что такое прямое и обратное распространение в нейронных сетях?
5. Опишите архитектуру MLP.

6. Почему нельзя повышать качество модели бесконечно увеличивая количество слоев в MLP?

Список литературы

Для выполнения лабораторной работы, при подготовке к защите, а также для ответа на контрольные вопросы рекомендуется использовать следующие источники: [1, 2, 5-7].

ЛАБОРАТОРНАЯ РАБОТА 16. КЛАССИФИКАЦИЯ ИЗОБРАЖЕНИЙ С ИСПОЛЬЗОВАНИЕМ MLP

Цели и задачи

Цель лабораторной работы: получение навыков по применению MLP в задачах классификации.

Основные задачи:

- получение навыков предварительной обработки данных для нейронных сетей;
- получение навыков обучения и анализа эффективности процесса обучения нейросетевой модели;
- получение навыков участия в соревнованиях Kaggle.

Теоретическое обоснование

В данной работе используется набор данных Fashion MNIST (выборки данных скачиваются с Kaggle по ссылке:

<https://www.kaggle.com/zalando-research/fashionmnist>



Экземпляры набора данных
MNIST



Экземпляры набора данных
Fashion MNIST

Выше изображены примеры того, с чем мы будем работать – чёрно-белые изображения одежды. Слева более классический датасет – MNIST, он же датасет рукописных цифр. В данной работе рассматривается проблема

распознавания одежды (очевидно, что распознавание рукописных цифр также можно будет реализовать с рассматриваемой нейросетевой моделью).

Основная информация о структуре изображений

Как и вся информация, изображения представляются в компьютере числами. Стандартное цветовое пространство, с помощью которого декодируют и отрисовывают изображение – это RGB (Red, Green и Blue). Каждая комбинация трёх чисел от 0 до 255 задаёт какой-то цвет. Например, (255,255,255) задаёт белый цвет, (255,0,0) – красный. Также происходит и при загрузке картинок в Python, давайте посмотрим на примере:

Загрузим произвольную цветную картинку с помощью matplotlib:

```
import numpy as np
import matplotlib.pyplot as plt
```

executed in 12.7s, finished 00:35:07 2021-12-12

```
image_png = plt.imread('./ncfu_logo.png')
```

executed in 8ms, finished 00:37:13 2021-12-12

```
image_png
```

executed in 29ms, finished 00:37:15 2021-12-12

```
[[1., 1., 1.],
 [1., 1., 1.],
 [1., 1., 1.],
 ...,
 [1., 1., 1.],
 [1., 1., 1.],
 [1., 1., 1.]],
```

Проанализируем тип и размеры объекта:

```
image_png.shape
```

executed in 4ms, finished 00:37:26 2021-12-12

```
(100, 131, 3)
```

```
type(image_png)
```

executed in 4ms, finished 00:37:21 2021-12-12

```
numpy.ndarray
```

```
image_png[0].shape
```

executed in 5ms, finished 00:37:28 2021-12-12

```
(131, 3)
```

То есть это какая матрица, а точнее тензор (потому что есть третья размерность), у которого 100 строк, 131 столбец и 4 канала. Можно представлять это себе как 4 наложенных друг на друга матрицы, каждая из которых отвечает за один цвет – R, G и B. Внутри всех этих матриц лежат числа типа float32, то есть вещественные. Тут стоит сказать, что это просто тонкости загрузки в matplotlib – на самом деле это матрицы из целых чисел от 0 до 255 (включительно).

Но ведь каналов 4, а не 3? Да, четвёртый канал в данном случае – это альфа-канал, у .png картинок он обычно присутствует. Попробуем загрузить .jpg картинку:

```
image_jpg = plt.imread('./ncfu_logo.jpg')
```

executed in 540ms, finished 00:38:20 2021-12-12

```
image_jpg.shape
```

executed in 3ms, finished 00:38:21 2021-12-12

```
(609, 800, 3)
```

Теперь 3 канала – это три матрицы 609 на 800, каждая из которых отвечает за один цвет. Отрисуем две загруженные картинки с помощью matplotlib:

```
plt.imshow(image_png);
```

executed in 2.03s, finished 00:38:26 2021-12-12



```
plt.imshow(image_jpg);
```

executed in 229ms, finished 00:38:29 2021-12-12



В данном случае пиксель – это кортеж (упорядоченная последовательность чисел), состоящий из трёх чисел (как в примере раньше, например, (255,0,0) – полностью красный пиксель).

Итак, изображения – это матрицы, состоящие из чисел, которые характеризуют насыщенность данного пикселя определённым цветом цветом.

Аналогично и с чёрно-белыми изображениями – это просто матрица с одним каналом (то есть пксель – это просто число), например, 28 на 28, каждое число которой от 0 до 255 характеризует яркость пикселя (насыщенность белым). Например, 255 – это полностью белый пиксель, 0 – полностью чёрный.

Оборудование и материалы

Для выполнения лабораторной работы рекомендуется использовать персональный компьютер со следующими программными средствами разработки (выбрать один или несколько программных продуктов для практической реализации задач лабораторной работы): интерпретатор Python (Anaconda), библиотеки NumPy, Pandas, PyTorch.

Указания по технике безопасности

Студенты должны следовать общепринятой технике безопасности для пользователей персональных компьютеров. Не следует самостоятельно производить ремонт технических средств, установку и удаление программного обеспечения. В случае обнаружения неисправностей необходимо сообщить об этом администратору компьютерного класса (обслуживающему персоналу лаборатории).

Методика и порядок выполнения работы

Перед выполнением индивидуального задания рекомендуется выполнить все пункты учебной задачи.

Учебная задача

Задание. Реализовать MLP-классификатор на основе выборки Fashion MNIST.

Решение. Будем работать с картинками одежды (чёрно-белыми, то есть цветовых каналов не 3, а 1). По входной картинке нужно предсказать тип одежды. Давайте посмотрим, что представляет собой датасет Fashion MNIST:

```
import pandas as pd
```

executed in 12.0s, finished 00:25:44 2021-12-12

```
TRAIN_PATH = './fashionmnist/fashion-mnist_train.csv'  
TEST_PATH = './fashionmnist/fashion-mnist_test.csv'
```

executed in 4ms, finished 00:25:46 2021-12-12

```
train_df = pd.read_csv(TRAIN_PATH)  
test_df = pd.read_csv(TEST_PATH)
```

executed in 7.69s, finished 00:33:52 2021-12-12

```
test_df
```

executed in 807ms, finished 00:33:55 2021-12-12

label	pixel1	pixel2	pixel3	pixel4	pixel5	pixel6	pixel7
0	0	0	0	0	0	0	0
1	1	0	0	0	0	0	0
2	2	0	0	0	0	0	14
3	2	0	0	0	0	0	0
4	3	0	0	0	0	0	0

Проанализируем размеры обучающей и тестовой выборок:

<code>train_df.shape</code>
executed in 5ms, finished 00:34:14 2021-12-12
<code>(60000, 785)</code>
<code>test_df.shape</code>
executed in 4ms, finished 00:34:16 2021-12-12
<code>(10000, 785)</code>

Каждая строчка соответствует одной картинке, а столбцы – это значения в пикселях этой картинке. Первый столбец в `train_df` говорит о типе (классе) одежды (от 0 до 9).

Самый первый столбец – `label`. Каждая картинка имеет класс от 0 до 9, расшифровка меток класса:

class_id	class_name
0	T-shirt/top
1	Trouser
2	Pullover
3	Dress
4	Coat
5	Sandal
6	Shirt
7	Sneaker
8	Bag
9	Ankle boot

Итак, мы имеем 60000 картинок, у каждой известна метка класса (то есть что это за одежда). Отделим X (признаковое описание объектов) и y (метки классов):

```
X_train = train_df.values[:, 1:]
y_train = train_df.values[:, 0]

X_test = test_df.values[:, 1:] # удаляем столбец 'label'
```

executed in 4ms, finished 00:39:00 2021-12-12

```
print(X_train.shape, y_train.shape)
```

executed in 5ms, finished 00:39:02 2021-12-12

```
(60000, 784) (60000,)
```

```
print(X_test.shape)
```

executed in 3ms, finished 00:39:04 2021-12-12

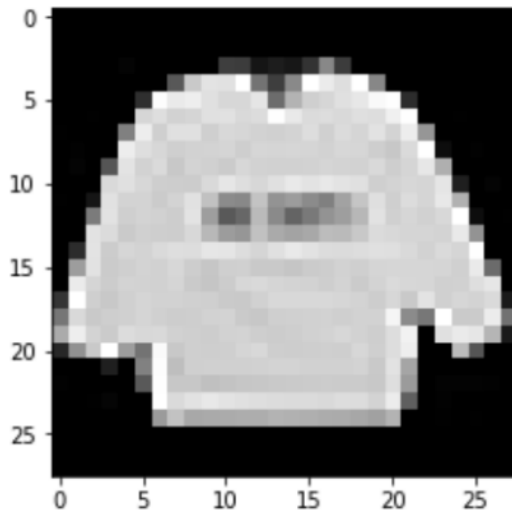
```
(10000, 784)
```

Следует обратить внимание, что изображения хранятся в строках текстового файла. На самом деле 784 пикселя – это 28*28, то есть это «развёрнутая в строку» чёрно-белая картинка 28 на 28 пикселей.

Давайте убедимся в этом, отрисовав несколько (можете менять индекс и смотреть на отрисовку):

```
plt.imshow(X_train[0].reshape(28, 28), cmap='gray');
```

executed in 202ms, finished 00:39:07 2021-12-12



Это изображение, относящееся к классу «пулlover». Просто если мы будем использовать изображения большего разрешения, нам понадобятся большие вычислительные мощности, поэтому пока что будем довольствоваться такими размерами. Мы убедились в том, что имеем 60000 картинок с метками для обучения, картинки «развёрнуты» в строку. Зачем разворачивать в строку? Потому что каждый пиксель в данном случае – это один признак, то есть всего 784 признака, и уже их мы будем взвешивать нашей нейросетью, то есть у одного нейрона на входном слое будет 784 веса (+ Bias, то есть 785 весов), на каждый пиксель по весу, и дальше уже будут второй слой, третий слой и так далее...

Реализуем нейронную сеть на PyTorch. Сначала обернём данные в тензоры пайторча (может занять некоторое время):

```
X_train_tensor = torch.FloatTensor(X_train)
y_train_tensor = torch.LongTensor(y_train.astype(np.int64))
```

executed in 662ms, finished 00:39:23 2021-12-12


```
print(X_train_tensor.shape, y_train_tensor.shape)
```

executed in 4ms, finished 00:39:25 2021-12-12

```
torch.Size([60000, 784]) torch.Size([60000])
```

```
y_train_tensor.unique()
```

executed in 885ms, finished 00:39:28 2021-12-12

```
tensor([2, 9, 6, 8, 0, 3, 4, 5, 7, 1])
```

Нельзя непосредственно использовать LogLoss (основная функция потерь для задачи классификации). Необходимо преобразовать метки классов из целых чисел в OneHot-кодированные метки:

```
length = y_train_tensor.shape[0]
num_classes = 10 # количество классов, в нашем случае 10 типов одежды

# закодированные OneHot-ом метки классов
y_onehot = torch.FloatTensor(length, num_classes)

y_onehot.zero_()
y_onehot.scatter_(1, y_train_tensor.view(-1, 1), 1)

print(y_train_tensor)
print(y_onehot)
```

executed in 328ms, finished 00:39:32 2021-12-12

```
tensor([2, 9, 6, ..., 8, 8, 7])
tensor([[0., 0., 1., ..., 0., 0., 0.],
        [0., 0., 0., ..., 0., 0., 1.],
        [0., 0., 0., ..., 0., 0., 0.],
        ...,
        [0., 0., 0., ..., 0., 1., 0.],
        [0., 0., 0., ..., 0., 1., 0.],
        [0., 0., 0., ..., 1., 0., 0.]])
```

Видим, что метки перешли в вид «единица там, где номер класса, а остальные нули». Возьмём два слоя – входной и один скрытый (выходной обычно не считают, но он тоже есть):

```
# N - размер батча (batch_size, нужно для метода оптимизации)
# D_in - размерность входа (количество признаков у объекта)
# H - размерность скрытых слоёв;
# D_out - размерность выходного слоя (суть - количество классов)
D_in, H, D_out = 784, 100, 10

# определим нейросеть:
net = torch.nn.Sequential(
    torch.nn.Linear(D_in, H),
    torch.nn.ReLU(),
    torch.nn.Linear(H, D_out),
    torch.nn.Softmax()
)
```

Следует обратить внимание:

D_{in} – это входная размерность (784 признака-пикселя);

D_{out} – выходная размерность (10 классов-типов одежды), то есть 10 нейронов на выходном слое;

H – количество нейронов в скрытом слое.

Осталось выбрать Loss (функцию потерь) и метод оптимизации, с помощью которого мы будем считать градиенты и обновлять с помощью них обновлять веса. Loss мы выберем CrossEntropy, то есть кросс-энтропию, этот лосс почти всегда используется в задаче многоклассовой классификации, а метод оптимизации выберем обычный SGD (Stochastic Gradient Descent, стохастический градиентный спуск).

Однако перед тем, как перейти к коду обучения нейросети, есть одна тонкость – батчи, а точнее мини-батчи.

Мини-батчи – это небольшие (обычно размера 16, 32 или 64) «фрагменты» выборки, то есть мини-батч размера 64 – это 64 объекта из датасета. Обычно мини-батч называют просто батч (batch).

Методы оптимизации по типу стохастического градиентного спуска часто считаются не под одному объекту (в этом случае оптимизация будет очень нестабильная, «шумная»), а по нескольким – по батчу. То есть в обычном градиентном спуске будет сумма по всей выборке, в стохастическом (чистом варианте) – по одному объекту, а «между ними» - мини-батч SGD, то есть подсчёт градиентов на небольшом кусочке данных.

Одна итерация (iteration) алгоритма оптимизации – это проход по одному батчу. Одна эпоха (epoch) алгоритма оптимизации – это проход по всей выборке. То есть, например, если выборка размера 60000, а батч размера 64, то одна эпоха занимает $60000 / 64 = 937,5 = 938$ итераций.

Напишем функцию, генерирующую батчи:

```
def generate_batches(X, y, batch_size=64):
    for i in range(0, X.shape[0], batch_size):
        X_batch, y_batch = X[i:i+batch_size], y[i:i+batch_size]
        yield X_batch, y_batch
```

Код обучения нейросети (обязательно убедитесь, что понимаете, что делает каждая строчка — это необходимо для ваших дальнейших экспериментов):

```
BATCH_SIZE = 64
NUM_EPOCHS = 100

loss_fn = torch.nn.CrossEntropyLoss(size_average=False)

learning_rate = 1e-4
optimizer = torch.optim.SGD(net.parameters(), lr=learning_rate)

for epoch_num in range(NUM_EPOCHS):
    iter_num = 0
    running_loss = 0.0
    for X_batch, y_batch in generate_batches(X_train_tensor, y_train_tensor, BATCH_SIZE):
        # forward (подсчёт ответа с текущими весами)
        y_pred = net(X_batch)

        # вычисляем Loss'ы
        loss = loss_fn(y_pred, y_batch)

        running_loss += loss.item()

    if iter_num % 100 == 99:
        print('[{epoch_num}, {iter_num}] current loss: {loss}'.format(epoch_num, iter_num + 1, running_loss / 2000))
        running_loss = 0.0

    # зануляем градиенты
    optimizer.zero_grad()

    # backward (подсчёт новых градиентов)
    loss.backward()

    # обновляем веса
    optimizer.step()

    iter_num += 1
```

мы получили обученную нейросеть. Давайте измерим качество на обучающей выборке:

```
class_correct = list(0. for i in range(10))
class_total = list(0. for i in range(10))

classes = ['T-shirt/top', 'Trouser', 'Pullover', 'Dress', 'Coat',
           'Sandal', 'Shirt', 'Sneaker', 'Bag', 'Ankle boot']

with torch.no_grad():
    for X_batch, y_batch in generate_batches(X_train_tensor, y_train_tensor, BATCH_SIZE):
        y_pred = net(X_batch)
        _, predicted = torch.max(y_pred, 1)
        c = (predicted == y_batch).squeeze()
        for i in range(len(y_pred)):
            label = y_batch[i]
            class_correct[label] += c[i].item()
            class_total[label] += 1

for i in range(10):
    print('Accuracy of %s : %2d %%' % (
        classes[i], 100 * class_correct[i] / class_total[i]))
```

```

Accuracy of T-shirt/top : 95 %
Accuracy of Trouser : 97 %
Accuracy of Pullover : 95 %
Accuracy of Dress : 94 %
Accuracy of Coat : 0 %
Accuracy of Sandal : 99 %
Accuracy of Shirt : 0 %
Accuracy of Sneaker : 95 %
Accuracy of Bag : 98 %
Accuracy of Ankle boot : 97 %

```

Очевидно, что сеть далеко не идеально решает задачу классификации — она предсказывает только 8 классов, а про некоторые просто «забывает».

Теперь предскажем на тестовой и сохраним предсказания в файл. Это ни что иное, как baseline, который вам необходимо превзойти, чтобы выполнить индивидуальное задание.

```
y_test_pred = net(torch.FloatTensor(X_test))
```

executed in 129ms, finished 00:42:41 2021-12-12

```
y_test_pred.shape
```

executed in 4ms, finished 00:42:43 2021-12-12

```
torch.Size([10000, 10])
```

```
y_test_pred[:5]
```

executed in 16ms, finished 00:42:45 2021-12-12

```

Out[43]: tensor([[1.0000, 0.0000, 0.0000, 0.0000, 0.0000, 0.0000, 0.0000, 0.0000, 0.0000,
                  0.0000],
                [0.0000, 1.0000, 0.0000, 0.0000, 0.0000, 0.0000, 0.0000, 0.0000, 0.0000,
                  0.0000],
                [0.0000, 0.0000, 1.0000, 0.0000, 0.0000, 0.0000, 0.0000, 0.0000, 0.0000,
                  0.0000],
                [1.0000, 0.0000, 0.0000, 0.0000, 0.0000, 0.0000, 0.0000, 0.0000, 0.0000,
                  0.0000],
                [0.0000, 0.0000, 0.0000, 1.0000, 0.0000, 0.0000, 0.0000, 0.0000, 0.0000,
                  0.0000]], grad_fn=<SliceBackward>)

```

Преобразуем OneHot'ы в числовые метки:

```
_, predicted = torch.max(y_test_pred, 1)
```

```
predicted
```

executed in 5ms, finished 00:42:49 2021-12-12

```
tensor([0, 1, 2, ..., 8, 8, 1])
```

Сохраним в датафрейм:

```
answer_df = pd.DataFrame(data=predicted.numpy(), columns=['Category'])
answer_df.head()
```

executed in 8ms, finished 00:42:52 2021-12-12

Out[45]:

Category
0
1
2
3
4

```
answer_df['Id'] = answer_df.index
```

executed in 4ms, finished 00:42:55 2021-12-12

```
answer_df.head()
```

executed in 8ms, finished 00:42:57 2021-12-12

Category	Id
0	0
1	1
2	2
3	0
4	3

```
answer_df.tail()
```

executed in 7ms, finished 00:42:59 2021-12-12

Category	Id
9995	0
9996	2
9997	8
9998	8
9999	1

Сохранение результатов в файл:

```
answer_df.to_csv('./baseline.csv', index=False)
```

executed in 234ms, finished 00:43:05 2021-12-12

Выполните индивидуальное задание.

Индивидуальное задание

Необходимо добиться как можно лучшего качества в соревновании по классификации Fashion Dataset. Используйте знания, полученные на занятиях и те, которые вы найдёте в интернете. Если у вас получится, можете использовать и свёрточные нейросети, а не только полносвязные. Вам нужно как минимум побить baseline.

Рекомендация: попробуйте поменять количество итераций, количество нейронов, количество слоёв, гиперпараметры сети (learning_rate, метод оптимизации вместо SGD можно взять другой).

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 аспекты работы с нейросетевыми моделями влияют на качество нейросетевых моделей, но не зависят от свойств набора данных?
2. Какие аспекты работы с нейросетевыми моделями влияют на качество нейросетевых моделей и зависят от свойств набора данных?
3. Какие стратегии определения нейронной сети существуют в PyTorch?
4. Что такое прямое и обратное распространение в нейронных сетях?

5. Опишите архитектуру MLP.
6. Почему нельзя повышать качество модели бесконечно увеличивая количество слоев в MLP?

Список литературы

Для выполнения лабораторной работы, при подготовке к защите, а также для ответа на контрольные вопросы рекомендуется использовать следующие источники: [1, 2, 5-7].

ЛАБОРАТОРНАЯ РАБОТА 17. СВЕРТОЧНАЯ НЕЙРОННАЯ СЕТЬ

Цели и задачи

Цель лабораторной работы: получение практических навыков по реализации пайплайна для решения задачи машинного обучения с использованием сверточной нейросетевой модели.

Основные задачи:

- получение навыков по выбору архитектуры сверточной ИНС для решения задачи классификации изображений;
- изучение принципов блочного построения сверточной нейронной сети с использованием фреймворка машинного обучения;
- реализация всех стадий жизненного цикла сверточной нейронной сети: проектирование, обучение, тюнинг, анализ результатов и развертывание.

Теоретическое обоснование

Для проектирования и разработки нейронной сети в рамках данной работы будет использоваться нейросетевой фреймворк PyTorch. На современном этапе развития инструментальных средств машинного обучения исследователю на рынке доступны следующие варианты программных средств:

1. TensorFlow. Фреймворк машинного обучения от компании Google (свободно распространяемое решение). TensorFlow предназначен для поддержки реализации всех стадий жизненного цикла нейросетевых моделей. TensorFlow представляет собой библиотеку крупномасштабного машинного обучения и численных расчетов. В качестве языка обращения к библиотеке используется Python, но само ядро библиотеки написано на C++. Библиотека позволяет проектировать, обучать и развертывать глубокие модели; именно deep learning является основным преимуществом TensorFlow. Библиотека представляет модели в виде вычислительного графа. Узлы графа (тензоры)

соответствуют обрабатываемым данным, а ребра графа – операциям над этими данными. Библиотека TensorFlow позволяет создавать модели глубокого обучения, которые могут быть развернуты на различных платформах: персональных компьютерах, смартфонах iOS и Android, CPU, GPU или TPU, облачные сервисы Google, Amazon или Microsoft. В настоящее время доступна версия библиотеки 2.0, которая сильно отличается от версий 1.x.

2. PyTorch. Фреймворк машинного обучения от компании Facebook, разработанный для языка Python. PyTorch во многом схож с TensorFlow: простое аппаратное ускорение с использованием GPU; простая система прототипирования моделей глубокого обучения; множество стандартных блоков вычислений, слоев ИНС, содержащихся в составе библиотеки. PyTorch рекомендуется использовать для небольших проектов, которые необходимо запустить в качестве узкоспециализированного веб-сервиса, тогда как TensorFlow предпочтительнее при обучении больших моделей, которые требуют использования больших вычислительных мощностей в виде облачных сервисов и кластеров.

TensorFlow и PyTorch – это наиболее популярные библиотеки в сфере машинного обучения. Работа с каждой из них возможна без непосредственной установки данных библиотек (что является достаточно трудоемким процессом): можно использовать бесплатный сервис Google Colab для обучения глубоких моделей. Но рассмотренные библиотеки не являются единственными на рынке.

CNTK (Microsoft Cognitive Toolkit) – библиотека машинного обучения от Microsoft. Как и TensorFlow данный программный продукт использует граф для представления потока вычислений, но CNTK является более специализированной на моделях глубокого обучения. CNTK отличается более высокой скоростью выполнения операций в рамках манипулирования данными при обучении нейронных сетей и предлагает различные языки в качестве клиентских (Python, C++, C#, Java). Основной недостаток CNTK – малое количество целевых платформ для развертывания.

Apache MXNet – фреймворк машинного обучения, используемый компанией Amazon в качестве библиотеки машинного обучения в рамках облака AWS. Отличается высокой степенью масштабируемости по отдельным GPU и компьютерам. Поддерживает большое количество различных языков разработки: Python, C++, Scala, R, JavaScript, Julia, Perl, Go. При этом API MXNet считается менее дружелюбным чем TensorFlow.

В рамках данной работы в качестве рабочего фреймворка рекомендуется использование PyTorch или TensorFlow. Основное преимущество TensorFlow – масштабируемость. При этом у PyTorch существует преимущество в виде удобного API и используемой идеи близости библиотеки к концепции языка Python (Python-like стиль).

Оборудование и материалы

Для выполнения лабораторной работы рекомендуется использовать персональный компьютер со следующими программными средствами разработки (выбрать один или несколько программных продуктов для практической реализации задач лабораторной работы интерпретатор Python (Anaconda).

Методика и порядок выполнения работы

Индивидуальное задание

1. Подберите набор данных на ресурсах [5-7] и согласуйте свой выбор с преподавателем. Студент может предложить набор данных в соответствии с тематикой магистерского исследования.

2. Реализуйте проектирование сверточной нейронной сети.

3. Реализуйте обучение сверточной нейронной сети на выбранном наборе данных.

3. Оцените результаты обучения, оцените характер изменения функции потерь и градиентов.

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. Что отличает сверточные нейронные сети от нейронных сетей типа многослойный персептрон?
2. Что такое глубокие нейронные сети? Какие типы слоев присутствуют в глубоких нейронных сетях?
3. Что такое сверточный слой? Как сверточный слой определяется в рамках библиотеки PyTorch? Поясните принцип свертки изображения, опишите соотношение следующих параметров: размер входного и выходного сигналов, отступ свертки, размер ядра свертки.
4. Что такое слои для понижения размерности? Опишите основные классы PyTorch, реализующие понижение размерности сигнала.
5. Опишите класс PyTorch для реализации полносвязного слоя.
6. Поясните смысл термина «метод оптимизации». Поясните возможные стратегии оптимизации модели нейронной сети в рамках библиотеки PyTorch.

7. Что такое функция потерь? Какие функции потерь доступны в библиотеке PyTorch?
8. Какой класс Pytorch применяется для составления последовательности слоев нейронной сети? Поясните принцип согласования соседних слоев.
9. Что такое регуляризация, применительно к обучению ИНС? Опишите возможные стратегии преодоления переобучения ИНС.
10. Опишите поведение модели ИНС в следующих ситуациях: модель переобучена; модель недообучена.

Список литературы

Для выполнения лабораторной работы, при подготовке к защите, а также для ответа на контрольные вопросы рекомендуется использовать следующие источники: [1, 2, 5-7].

ЗАКЛЮЧЕНИЕ

Учебное пособие (лабораторный практикум) по дисциплине «Методы искусственного интеллекта» для студентов направления 09.04.02 «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 методов 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 прошлые 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. Уэс, Маккинли. Python и анализ данных Электронный ресурс / Маккинли Уэс ; пер. А. А. Слинкин. - Python и анализ данных, 2025-04-19. - Саратов : Профобразование, 2017. - 482 с. - Книга находится в премиум-версии ЭБС IPR BOOKS. - ISBN 978-5-4488-0046-7, экземпляров неограниченно.
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 с. - Книга находится в базовой версии ЭБС IPRbooks. - ISBN 5-9556-0058-2, экземпляров неограниченно

Список дополнительной литературы

3. Стенли, Липпман. Язык программирования C++ Электронный ресурс : Полное руководство / Липпман Стенли, Лажойе Жози ; пер. А. Слинкин. - Язык программирования C++, 2025-04-19. - Саратов : Профобразование, 2017. - 1104 с. - Книга находится в премиум-версии ЭБС IPR BOOKS. - ISBN 978-5-4488-0136-5, экземпляров неограниченно
4. Седжвик, Р. Алгоритмы на C++ / Р. Седжвик. - 2-е изд., испр. - Москва : Национальный Открытый Университет «ИНТУИТ», 2016. - 1773 с., экземпляров неограниченно
5. <https://archive.ics.uci.edu/ml/index.html> – Репозиторий наборов данных для машинного обучения (Центр машинного обучения и интеллектуальных систем).
6. <https://www.kaggle.com> – Портал и система проведения соревнований по проблемам анализа данных.

7. <https://www.mockaroo.com> – Сайт для генерации наборов данных.

Методы искусственного интеллекта

Методические указания к лабораторным работам

Автор

Николаев Евгений Иванович

