

МЕЖГОСУДАРСТВЕННОЕ ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ
ВЫСШЕГО ОБРАЗОВАНИЯ
«БЕЛОРУССКО-РОССИЙСКИЙ УНИВЕРСИТЕТ»

Кафедра «Автоматизированные системы управления»

ОСНОВЫ МАШИННОГО ОБУЧЕНИЯ

*Методические рекомендации к лабораторным работам
для студентов специальностей*

6-05-0612-03 «Системы управления информацией»

и 6-05-0611-04 «Электронная экономика»

очной и заочной форм обучения



УДК 004.85
ББК 16.63я73
О75

Рекомендовано к изданию
учебно-методическим отделом
Белорусско-Российского университета

Одобрено кафедрой «Автоматизированные системы управления»
«24» марта 2026 г., протокол № 8

Составитель ст. преподаватель И. А. Беккер

Рецензент канд. техн. наук, доц. В. В. Кутузов

Методические рекомендации к лабораторным работам содержат краткие теоретические сведения, задания, контрольные вопросы. Предназначены для студентов специальностей 6-05-0612-03 «Системы управления информацией» и 6-05-0611-04 «Электронная экономика» очной и заочной форм обучения.

Учебное издание

ОСНОВЫ МАШИННОГО ОБУЧЕНИЯ

Ответственный за выпуск	А. И. Якимов
Корректор	А. А. Подошевка
Компьютерная верстка	Н. П. Полевничая

Подписано в печать . Формат 60×84/16. Бумага офсетная. Гарнитура Таймс.
Печать трафаретная. Усл. печ. л. . Уч.-изд. л. . Тираж 21 экз. Заказ №

Издатель и полиграфическое исполнение:
Межгосударственное образовательное учреждение высшего образования
«Белорусско-Российский университет».
Свидетельство о государственной регистрации издателя,
изготовителя, распространителя печатных изданий
№ 1/156 от 07.03.2019.
Пр-т Мира, 43, 212022, г. Могилев.

© Белорусско-Российский
университет, 2026

Содержание

Введение	4
1 Лабораторная работа № 1. Основы работы с данными в программной среде.....	5
2 Лабораторная работа № 2. Предварительная обработка данных. Отбор признако	8
3 Лабораторная работа № 3. Построение модели линейной регрессии	11
4 Лабораторная работа № 4. Реализация логистической регрессии, оценка качества модели	12
5 Лабораторная работа № 5. Метод ближайших соседей	14
6 Лабораторная работа № 6. Построение моделей деревьев решений и случайного леса	15
7 Лабораторная работа № 7. Метод опорных векторов	17
8 Лабораторная работа № 8. Работа с различными алгоритмами машинного обучения	21
Список литературы	23

Введение

В результате изучения дисциплины «Основы машинного обучения» у студентов формируются знания терминов, основных фактов и закономерностей машинного обучения, умения работать с алгоритмами машинного обучения на языке программирования Python. Основная цель учебной дисциплины – это обзорное изучение современных методов автоматического обнаружения и описания скрытых закономерностей в данных.

Процесс выполнения и защиты лабораторной работы студентом состоит в том, что следует успешно пройти следующие этапы учебной деятельности:

- выполнить согласно варианту, выданному преподавателем, задание;
- составить отчет с описанием хода работы и результатов, сделать вывод по заданию;
- предъявить разработанное программное обеспечение на языке программирования Python в применении к набору данных;
- продемонстрировать знание теоретического материала применительно к заданию лабораторной работы (знать все термины, свойства, закономерности анализа данных, а также команды Python, встречающиеся в работе; ответить на контрольные вопросы, приведенные в конце лабораторной работы).

1 Лабораторная работа № 1. Основы работы с данными в программной среде

Цель работы: научиться выполнять загрузку датасета в рабочую среду и основные действия с ним.

Теоретические сведения

В Python есть много библиотек для работы с данными.

NumPy – это библиотека для научных вычислений в Python, она содержит множество функций и операций для работы с массивами.

Pandas – библиотека для обработки и анализа данных в Python, предоставляющая высокоуровневые структуры данных DataFrame и Series для удобного и эффективного манипулирования табличными данными. Работа Pandas с данными строится поверх библиотеки NumPy.

Series – это проиндексированный одномерный массив значений. Он похож на простой словарь типа Dictionary, где имя элемента будет соответствовать индексу, а значение – значению записи.

DataFrame – это проиндексированный многомерный массив значений, соответственно каждый столбец DataFrame является структурой Series.

Pandas позволяет импортировать и экспортировать данные из различных источников, выполнять операции фильтрации, сортировки, агрегации и многое другое.

Matplotlib – мощная библиотека для создания графиков и визуализации данных в Python. Matplotlib.pyplot является подмодулем Matplotlib, гибким инструментом для визуализации данных и используется во многих областях, включая научные исследования, анализ данных и машинное обучение.

Scikit-learn (также известная как sklearn) – это библиотека машинного обучения, которая дает широкий спектр инструментов и алгоритмов для задач обучения с учителем и без учителя, включая классификацию, регрессию, кластеризацию, снижение размерности, выбор моделей и многое другое.

Для загрузки файла в среду выполнения Google CoLab воспользуемся методом `read_csv` из библиотеки Pandas. В методе требуется указать первым параметром путь до имени файла, далее по необходимости параметры:

`sep` – задаёт символ-разделитель полей в файле (по умолчанию разделитель запятая);

`names` – список названий колонок, если он не задан в файле;

`index_col` – номер колонки с индексом;

`decimal` – символ-разделитель для знаков после запятой;

`encoding` – кодировку файла и т. д.

```
import pandas
df = pandas.read_csv('D:/Dataset/titanic.csv')
```

Данные будут представлены в виде DataFrame. Чтобы выбрать один из столбцов фрейма данных, следует написать название столбца в квадратных

скобках, например, для печати `print(df['Pclass'])`.

Для быстрого просмотра первых пяти строк набора данных можно использовать метод `head(): df.head()`

Для получения последних пяти строк – метод `tail(): df.tail()`

Для получения определённого количества случайных строк из набора данных используют метод `sample(количество): df.sample(2)`

Для вывода определённого количества строк можно указать их число: `df[:3]`

Для ввода данных из определённого столбца можно указать его название и количество выводимых значений: `df['Berri 1'][:10]`

Размерность фрейма данных можно получить, используя свойство фрейма данных `shape: df.shape`

Количество строк можно найти, применив функцию `len()`, к индексу фрейма данных или к фрейму: `len(df.index) len(df)`

Для получения информации о типах столбцов фрейма данных используется свойство `dtypes: df.dtypes`

Для получения информации о статистических оценках значений фрейма используют метод `describe(): df.describe(include='all')`

Некоторые из признаков, отличных от целевой переменной, могут оказывать чересчур значимое влияние на прогноз, поскольку по своему смыслу содержат большую долю информации о значении целевой переменной. Нерелевантные признаки обнаруживаются ручным отбором и оценкой значимости. Если признак не имеет значения для проекта, его нужно исключить.

Произвести отбор признаков можно с помощью метода корреляции.

Для фильтрации данных необходимо сформировать условие (условие накладывается на столбцы) и указать его внутри квадратных скобок `DataFrame[условие]`, например: `fix_df[fix_df['Berri 1'] > 600]`

Для получения количества уникальных значений в столбце можно использовать метод `unique()`:

```
# Подсчет уникальных значений в столбце
len(fix_df['Berri 1'].unique())
```

Данные датафрейма могут быть изменены в процессе обработки и их потребуется сохранить в табличный файл. Для этого можно использовать команду: `dataframe.to_csv('file_name.csv')`

Метод `.to_csv` имеет ряд входных параметров, которые могут указывать формат выходного файла `csv`:

- кодировка, например, `encoding='utf-8'`;
- разделитель значений `sep` (по умолчанию `' '`);
- запись имен строк (индексы), по умолчанию `True`.

Задание

1 Ознакомьтесь со структурой портала data.worldbank.org и системой поиска датасетов по определенной тематике.

2 Загрузите с портала data.worldbank.org набор данных в формате `csv` в кодировке Windows по выданной преподавателем теме и исследуйте структуру

данных: количество записей, признаков, типы данных, имеющиеся пропуски.

3 Выполните базовые статистические операции: частота события, среднее значение, медиана, мода, стандартное отклонение, процентиля, минимальное и максимальное значение величины.

4 Выполните выборку и фильтрацию данных по условию.

5 Выполните экспорт изменённого набора данных в новый csv.

6 Изучите данные о пассажирах «Титаника» из датасета `titanic.csv` (ссылка: <https://github.com/tpo9hbi4/MachineDeepLearning>) и дайте ответы на следующие вопросы:

Сколько мужчин, женщин, детей было на корабле?

Какая доля пассажиров выжила?

Какая доля пассажиров путешествовала в каком классе?

Чему равно среднее и медиана возраста всех людей на корабле?

Коррелирует ли число братьев/сестер с числом родителей/детей?

Контрольные вопросы

1 Что такое файл csv и как его воспринимает библиотека Pandas?

2 Какова общая структура датафрейма?

3 Опишите структуру Series.

4 Как средствами Python можно узнать количество записей, признаков?

5 Как можно выяснить типы данных?

6 Какими способами можно видеть имеющиеся пропуски?

7 Как сделать выборку данных по условию?

8 Какие данные выведет метод `describe`?

9 Метод `describe` выполнится только при подключении библиотеки Pandas?

10 Как просмотреть весь столбец датафрейма или 100 значений из этого столбца?

11 Как вычислить корреляцию между двумя признаками?

12 Для чего вычисляют корреляцию между признаками в наборе данных?

13 Можно ли просмотреть в Python набор уникальных значений столбца?

14 Каким образом экспортировать данные в новый датасет?

2 Лабораторная работа № 2. Предварительная обработка данных. Отбор признаков

Цель работы: научиться улучшать качество данных и исключать различного рода факторы, снижающие качество данных, которые будут мешать работе аналитических алгоритмов.

Теоретические сведения

Пустое значение для обработки данных является особым случаем, требующим исключения или заполнения. Если пропущенных значений не очень много, можно их заполнить. Если пропущенных значений в каком-то столбце много, то следует исключить столбец из рассмотрения, т. к. данных по нему недостаточно.

Для работы с пропущенными значениями в библиотеке Pandas есть методы: `isnull()` – для генерации булевой маски для отсутствующих значений;

`dropna()` – для фильтрации данных по отсутствующим значениям;

`fillna()` – для замены пропусков, его аргументы `method='ffill'` и `method='bfill'` определяют, какими значениями будут заполняться пропуски (предыдущими или последующими в массиве).

Для получения количества пустых значений в столбцах можно использовать метод `isnull()`:

```
#Возвращает количество пропущенных значений,  
#содержащихся в каждом столбце: df.isnull ()  
df.isnull().sum()
```

Метод `fillna()` не изменяет текущую структуру, он возвращает структуру `DataFrame`, созданную на базе существующей, с заменой `NaN` значений на те, что переданы в метод в качестве аргумента. В качестве значения для заполнения могут использоваться ближайшее ненулевое значение, скользящее среднее, следующее или предыдущее значение и др.

Пример:

```
#заполнение нулями  
df.fillna(0)  
  
#заполнение нулями (изменяем исходный датасет)  
df.fillna(0,inplace=True)  
  
#заполнение средним значением  
df.fillna(df.mean())  
df.Age [df['Age'].isnull()] = age_mean  
  
# Используйте экстремальное значение, среднее, медианное и т. д. из разных столбцов для пропущенных ячеек  
df.fillna({'Age':df.Age.mean(), 'weight':df.Weight.max()},inplace=True)
```

Для удаления объектов, которые содержат значения NaN, используется метод `dropna()`, в зависимости от параметров метода можно удалять весь столбец с пустыми значениями, только строки с пустыми значениями, указывать ограничение на возможное число пустых значений в строке.

```
# Удалить строки с пропущенными значениями напрямую
df.dropna()
# Прямое удаление столбцов с пропущенными значениями
df.dropna(axis = 1)
# Удалять только строки с пропущенными значениями
df.dropna(how = 'all')
```

Когда признаков в наборе не очень много, удобно визуализировать пропущенные значения с помощью тепловой карты:

```
cols = df.columns[:30] # первые 30 колонок
# определяем цвета пропущенные данные/не пропущенные
colours = ['#000055', '#ffff07']
sns.heatmap(df[cols].isnull(), cmap=sns.color_palette(colours))
```

Вот как можно увидеть, являются ли столбцы числовыми или категориальными признаками:

```
# отбор числовых колонок
df_numeric = df.select_dtypes(include=[np.number])
numeric_cols = df_numeric.columns.values
print(numeric_cols)
# отбор нечисловых колонок
df_non_numeric = df.select_dtypes(exclude=[np.number])
non_numeric_cols = df_non_numeric.columns.values
print(non_numeric_cols)
```

Еще одна хорошая техника визуализации для наборов с большим количеством признаков – построение гистограммы для числа отсутствующих значений в записи.

```
# сначала создаем индикатор для признаков с пропущенными данными
for col in df.columns:
    missing = df[col].isnull()
    num_missing = np.sum(missing)

    if num_missing > 0:
        print('created missing indicator for: {}'.format(col))
        df['{}_ismissing'.format(col)] = missing

# затем на основе индикатора строим гистограмму
ismissing_cols = [col for col in df.columns if 'ismissing' in col]
df['num_missing'] = df[ismissing_cols].sum(axis=1)
```

```
df['num_missing'].value_counts().reset_index().sort_values(by='index').plot.bar(x='index', y='num_missing')
```

Изменить значение данных в столбце можно с помощью явного присвоения (=) или с помощью метода `apply()`. Чтобы изменить данные с учётом некоторых условий, можно использовать вызов функции или лямбда-функции.

Пример:

```
df['romanic'].apply(lambda x: 1 if x == 'yes' else 0)
df['maximum'] = df[['c1', 'c2']].max(axis = 1)
```

Преобразовать данные можно также с помощью метода `map()`, необходимо определить словарь, карту, в котором «ключами» являются старые значения, а «значениями» – новые значения.

Пример:

```
level_map = {1: 'high', 2: 'medium', 3: 'low'}
df['c_level'] = df['c'].map(level_map)
```

Чтобы изучить нетипичные особенности данных и обнаружить выбросы, строят коробчатую диаграмму: `df.boxplot(column=['life_sq'])`.

Задание

- 1 Изучите структуру набора данных.
- 2 Обнаружьте и заполните пропуски разными методами.
- 3 Проведите кодирование категориальных признаков методами one-hot, label encoding.
- 4 Выполните масштабирование данных (Min-Max, StandardScaler).
- 5 Изучите описание датасета и исключите признаки, сильно коррелирующие с ответами. Ваш выбор признаков для исключения из выборки обоснуйте.
- 6 Используйте методы поиска и исправления нетипичных данных – выбросов.
- 7 Проведите обработку неинформативных данных – дубликатов.
- 8 Обработайте несогласованные данные – одни и те же данные, представленные в разных регистрах или форматах.

Контрольные вопросы

- 1 Для чего служит преобразование one-hot-encoding?
- 2 Всегда ли использование этого метода будет повышать качество работы модели машинного обучения?
- 3 Как выявить наличие дубликатов?
- 4 Как избавиться от одинаковых строк?
- 5 Как устранить выбросы?
- 6 Почему регуляризатор L1 (Lasso) обнуляет веса при линейно зависимых признаках?

7 Для чего исключают коррелирующие признаки, обязательно ли это делать?

8 С какой целью в предобработке используют карту `map`?

3 Лабораторная работа № 3. Построение модели линейной регрессии

Цель работы: научиться строить линейную регрессию.

Теоретические сведения

Линейная регрессия – базовый метод предсказания численных значений на основе линейной зависимости признаков и целевой переменной. В линейной регрессии рассматривается связь между одной переменной, называемой зависимой переменной, и несколькими другими, называемыми независимыми переменными. Эта связь представляется с помощью математической модели, т. е. уравнения вида $y_i = w_0 + w_1 x_i + e_i$. Величины w_0 и w_1 являются неизвестными параметрами, а $e_1, \dots, e_N$ суть некоррелированные ошибки, которые можно считать выборкой случайной величины e со средним 0 и неизвестной дисперсией: $E\{e\} = 0$, $\text{Var}\{e\} = \sigma^2$.

Таким образом, у линейной регрессии есть три параметра: w_0 – пересечение (свободный член), w_1 – угловой коэффициент (коэффициент регрессии), σ^2 – дисперсия остатков.

При построении модели линейной регрессии ставится задача нахождения оценок параметров $\hat{w}_0$, $\hat{w}_1$, $\hat{\sigma}^2$. В результате будет получена подогнанная линейная функция $f(x) = \hat{w}_0 + \hat{w}_1 x$, которая позволит получить оценку зависимой переменной y_i на основании значения независимой переменной x_i : $\hat{y}_i = f(x_i)$.

Зная оценки y_i можно вычислить остатки $e_i = y_i - \hat{y}_i = y_i - (\hat{w}_0 + \hat{w}_1 x_i)$.

При получении (обучении) линейной регрессии важную роль играет параметр суммы квадратов отклонений или остатков (RSS – residual sums of squares): $RSS = \sum e_i^2$.

Задание

- 1 Загрузите датасет для регрессии.
- 2 Выполните предобработку данных, приводя данные к виду, пригодному для обучения линейных моделей. Для этого вещественные признаки надо отмасштабировать, а категориальные привести к числовому виду. Также надо устранить пропуски в данных.
- 3 Разделите данные на обучающую и тестовую выборки в отношении 70/30 (для этого можно использовать функцию `train_test_split`).
- 4 Обучите модель линейной регрессии.
- 5 Оцените качество модели с помощью метрик MSE и R^2 .
- 6 Обучите линейную регрессию на 200 объектах из обучающей выборки и

выведите значения MSE и R^2 и на этой подвыборке и контрольной выборке. Проинтерпретируйте полученный результат, насколько качественные прогнозы строит полученная модель.

7 Постройте графики реальных и предсказанных значений.

4 Лабораторная работа № 4. Реализация логистической регрессии, оценка качества модели

Цель работы: научиться строить модель логистической регрессии.

Теоретические сведения

Логистическая регрессия используется для решения задач бинарной классификации, моделируя вероятность принадлежности к одному из классов.

Предположим, имеется набор данных о студентах, и нужно предсказать, поступят ли они в университет на основе результатов вступительных экзаменов.

```
import numpy as np
from sklearn.linear_model import LogisticRegression

# Обучающие данные
X = np.array([[6, 180, 12],
              [5.5, 160, 8],
              [7, 190, 11],
              [5.9, 170, 9],
              [6.2, 175, 10]])
y = np.array([1, 0, 1, 0, 1]) # 1 - поступил, 0 - не поступил
# Создание и обучение модели логистической регрессии
model = LogisticRegression()
model.fit(X, y)

# Предсказание
new_student = np.array([[6.1, 178, 11]])
prediction = model.predict(new_student)
if prediction[0] == 1:
    print("Студент поступит в университет")
else:
    print("Студент не поступит в университет")
```

В этом примере импортируется класс `LogisticRegression` из библиотеки `Scikit-learn`. Затем создается обучающий набор данных X и соответствующие метки y_i для каждого элемента списка X . Таким образом создается экземпляр модели логистической регрессии и обучается на основе обучающих данных с помощью метода `fit`. После этого используется обученная модель для предсказания поступления нового студента на основе его результатов на вступительных экзаменах с помощью метода `predict`. В зависимости от предсказанного класса, выводится соответствующее сообщение.

В этом примере используется только три признака для простоты, но в реальных задачах может быть больше признаков. Кроме того, перед использованием модели логистической регрессии важно предварительно обработать данные и масштабировать их, чтобы достичь лучшей производительности модели.

Модуль `roc_auc_score` из `scikit-learn` вычисляет площадь под кривой ROC (Receiver Operating Characteristic).

ROC-кривая используется для оценки производительности классификатора, основываясь на его чувствительности и специфичности.

Площадь под кривой ROC (AUC) является метрикой качества классификатора, где значение 1 соответствует идеальному классификатору, а значение 0,5 – случайному классификатору.

Задание

- 1 Загрузите датасет, подходящий для бинарной классификации.
- 2 Проведите предобработку данных: заполните пропущенные значения, преобразуйте значения необходимых столбцов в числовые и выполните другие необходимые преобразования данных.
- 3 Разделите данные на обучающую и тестовую выборки, выбрав соотношение 70 % на 30 %.
- 4 Используйте алгоритм логистической регрессии (например, `LogisticRegression` из библиотеки `Scikit-learn`) для обучения моделей. Для сравнения также обучите модель при помощи алгоритма случайного леса (`RandomForestRegressor`).
- 5 Оцените точность модели на тестовой выборке, используя подходящую метрику для классификации: точность (accuracy), матрицу ошибок (confusion matrix) или площадь под ROC-кривой (AUC-ROC).
- 6 Создайте модель случайного леса (`RandomForestClassifier`) и оцените ее точности также, как и для логистической регрессии.

Контрольные вопросы

- 1 Что такое логистическая регрессия?
- 2 Перечислите основные этапы применения логистической регрессии.
- 3 Какие предположения делаются при использовании логистической регрессии?
- 4 Какие типы задач можно решать с помощью логистической регрессии?
- 5 Что такое функция потерь в логистической регрессии?
- 6 Чем классификация отличается от линейной регрессии?
- 7 Какие два основных типа классификации вы знаете? Приведите примеры задач для каждого типа.
- 8 Что такое сигмоида и какую роль она играет в логистической регрессии?
- 9 Что такое матрица ошибок (confusion matrix) и какие показатели на её основе рассчитываются?
- 10 Как вычисляются точность (precision), полнота (recall) и F-мера?

5 Лабораторная работа № 5. Метод ближайших соседей

Цель работы: изучить алгоритм метода К ближайших соседей в Python.

Метод KNN (K Nearest Neighbours – К ближайших соседей) – это модель машинного обучения, которая основана на следующей идее: если объекты похожи по одним характеристикам, то будут похожи и по другим.

Например, если две квартиры расположены в одном районе, построены в одно и то же время и имеют схожую по размеру жилплощадь, то с большой вероятностью их стоимость будет почти одинаковой.

KNN часто применяется в системах рекомендаций: для пользователя находят других похожих на него пользователей и рекомендуют ему те же товары, фильмы или услуги.

Также этот метод используется в некоторых экономических задачах и в медицине.

Метод KNN решает задачи классификации и регрессии.

Пример задачи классификации: определить породу собаки по набору её характеристик.

Пример задачи регрессии: по описанию квартиры предсказать её стоимость.

В обеих задачах выделяют объект – по которому делается предсказание (собака или квартира) и целевую переменную – то, что надо предсказать (порода, цена на квартиру).

Согласно данному методу, если объекты «похожи», то их целевые переменные принимают те же (в случае классификации) или близкие (в случае регрессии) значения.

При сравнении объектов сопоставляется набор характеристик, которые выражаются через числа.

Для классификации нового объекта алгоритм выполняет следующие шаги.

Шаг 1. Находит К ближайших соседей в обучающей выборке.

Шаг 2. Смотрит, к какому классу принадлежат эти соседи.

Шаг 3. Присваивает новому объекту класс большинства среди соседей.

Для определения близости объектов выбирается метрика расстояния, чаще всего это евклидово расстояние.

Преимущества алгоритма – его простота в реализации и интерпретации, высокая точность прогнозов при правильном подборе К и метрики расстояния.

Недостатками алгоритма являются большое потребление памяти и низкая скорость работы из-за хранения и вычисления расстояний между всеми обучающими и тестовыми образцами.

Задание

1 Импортировать необходимые библиотеки и модули.

2 Загрузить датасет (MNIST): <http://yann.lecun.com/exdb/mnist/> (или другой по варианту).

3 Отобразить данные с помощью методов класса DataFrame. Провести предварительное изучение данных.

4 Визуализировать данные, проанализировать их.

5 Реализовать классификацию методом К ближайших соседей при разных К.

6 Сравнить качество моделей, вычислив оценки hold-out для разных К и различных долей обучающей и тестирующей подвыборок, cross validation для разных К и для различных значений количества подмножеств при кросс-валидации.

7 Вычислить оптимальные значения К. Продемонстрировать использование классификатора.

Контрольные вопросы

1 Каким образом следует выбирать порог разделения для иерархической кластеризации?

2 Как можно улучшить инициализацию центров кластеров в алгоритме k-means с целью нахождения наиболее оптимальных начальных приближений?

3 Реализуйте алгоритм DBSCAN на данных из лабораторной работы и сравните его качество с качеством работы алгоритма K-means и метода linkage.

6 Лабораторная работа № 6. Построение моделей деревьев решений и случайного леса

Цель работы: научиться строить модели деревьев решений и случайного леса.

Теоретические сведения

Деревья решений являются эффективным инструментом интеллектуального анализа данных и предсказательной аналитики, который позволяет решать задачи классификации и регрессии.

Они представляют собой иерархические древовидные структуры, состоящие из решающих правил вида «Если ..., то ...».

Правила автоматически генерируются в процессе обучения на обучающем множестве.

В обучающем множестве для примеров должно быть задано целевое значение, т. к. деревья решений являются моделями, строящимися на основе обучения с учителем.

Метод деревьев решений для задачи классификации состоит в том, чтобы осуществлять процесс деления исходных данных на группы, пока не будут получены однородные (или почти однородные) их множества.

Совокупность правил, которые дают такое разбиение, позволит затем делать прогноз (т. е. определить наиболее вероятный номер класса) для новых данных. Среди задач, успешно решаемых с помощью этого метода, можно назвать скоринговые модели кредитования (от англ. credit scoring models), диагностику медицинскую и техническую, где по набору значений факторов нужно поставить

диагноз.

Модуль `DecisionTreesClassifier` из библиотеки `Scikit-learn` реализует алгоритм дерева решений для задач классификации, создавая модель дерева решений как экземпляр соответствующего класса с определенным набором полей. `DecisionTreeClassifier` – это класс, способный выполнять многоклассовую классификацию на наборе данных.

Случайный лес – это ансамбль деревьев решений, где каждое дерево обучается на подмножестве данных и случайном подмножестве признаков. В результате получается ансамбль деревьев, который может использоваться для прогнозирования классов на новых данных.

Модуль `RandomForestClassifier` из библиотеки `Scikit-learn` реализует алгоритм случайного леса для задач классификации, создавая модель случайного леса как экземпляр соответствующего класса с большим набором полей, поскольку случайный лес объединяет несколько деревьев решений.

В случайных лесах класса `RandomForestClassifier` каждое дерево в ансамбле строится на основе выборки, взятой с заменой (по типу бутстрап выборки) из обучающего набора.

Задание

- 1 Загрузите выбранный датасет.
- 2 Проведите предобработку данных: заполните пропущенные значения, преобразуйте значения необходимых столбцов в числовые и выполните другие необходимые преобразования данных.
- 3 Разделите данные на обучающую и тестовую выборки.
- 4 Используйте модель дерева решений (`DecisionTrees`) для того, чтобы предсказать значение итогового столбца выборки данных.
- 5 Оцените точность модели на тестовой выборке, используя подходящую метрику для классификации, например, точность (accuracy), матрицу ошибок (confusion matrix) или площадь под ROC-кривой (AUC-ROC).
- 6 Создайте модель случайного леса (`RandomForestClassifier`) и настройте количество деревьев.
- 7 Оцените точности также, как и для дерева решений. Сравните результаты.

Контрольные вопросы

- 1 Что такое случайный лес?
- 2 Какие преимущества имеет случайный лес по сравнению с другими методами машинного обучения?
- 3 Что такое дерево решений?
- 4 Как выполняется процесс принятия решений с использованием дерева решений?
- 5 Как можно избежать переобучения при использовании деревьев решений?

7 Лабораторная работа № 7. Метод опорных векторов

Цель работы: изучить метод опорных векторов, ознакомиться с библиотеками Python для использования; научиться применять метод на практике.

Теоретические сведения

Основной задачей алгоритма SVM является найти наиболее правильную гиперплоскость, разделяющую данные на два класса.

SVM – это алгоритм, который получает на входе данные и возвращает такую разделяющую линию.

Одна из главных идей SVM заключается в том, чтобы преобразовать данные в пространство большей размерности, где объекты могут быть более легко разделены гиперплоскостью.

Это достигается через ядро, которое позволяет выполнить нелинейное преобразование данных, сохраняя при этом вычислительную эффективность.

Ядро определяет функцию сходства между объектами в новом пространстве признаков.

Вот четыре основных ядра:

1) линейное ядро (Linear kernel) – это самое простое ядро, которое строит гиперплоскость для разделения данных. Оно часто используется в задачах с линейно разделимыми данными и в математическом смысле вычисляет скалярное произведение между векторами признаков объектов;

2) ядро Radial Basis Function (RBF) – это наиболее распространенное ядро, которое может разделять данные, не являющиеся линейно разделимыми. Оно создает границу принятия решений в виде радиально-симметричного колокола;

3) ядро с полиномиальной функцией (Polynomial kernel), которое вводит полиномиальную функцию в пространство признаков для разделения данных, что может быть полезно для данных, которые не могут быть разделены гиперплоскостью;

4) ядро с сигмоидной функцией (Sigmoid kernel), которое используется для моделирования нейронных сетей. Оно может работать с нелинейными данными, но не так эффективно, как RBF-ядро.

Каждое ядро может иметь свои параметры, которые могут быть настроены для улучшения производительности модели.

Пусть есть набор данных и нужно его классифицировать, отделив квадраты от кругов.

Основной целью в данной задаче будет найти «идеальную» линию, которая разделит эти два класса (рисунок 1).

Алгоритм SVM устроен таким образом, что он ищет точки на графике, которые расположены непосредственно к линии разделения ближе всего.

Эти точки называются опорными векторами.

Затем, алгоритм вычисляет расстояние между опорными векторами и разделяющей плоскостью.

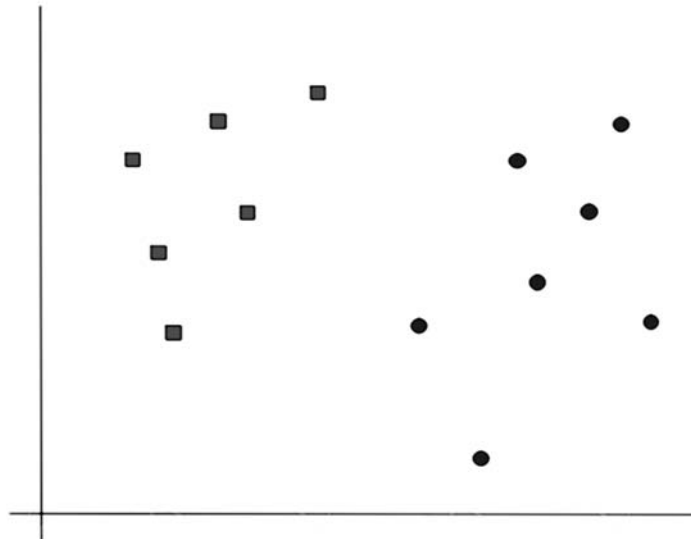


Рисунок 1

Это расстояние называется зазором.

Основная цель алгоритма – максимизировать расстояние зазора.

Лучшей гиперплоскостью считается такая гиперплоскость, для которой этот зазор является максимально большим (рисунок 2).

Импорт модуля SVM из библиотеки Scikit-learn (sklearn) позволяет использовать функции и классы, связанные с методами опорных векторов (SVM) для задач машинного обучения.

Функция `svm.LinearSVC()` предоставляет набор параметров, которые можно использовать для настройки линейного классификатора методом опорных векторов (Linear Support Vector Classifier).

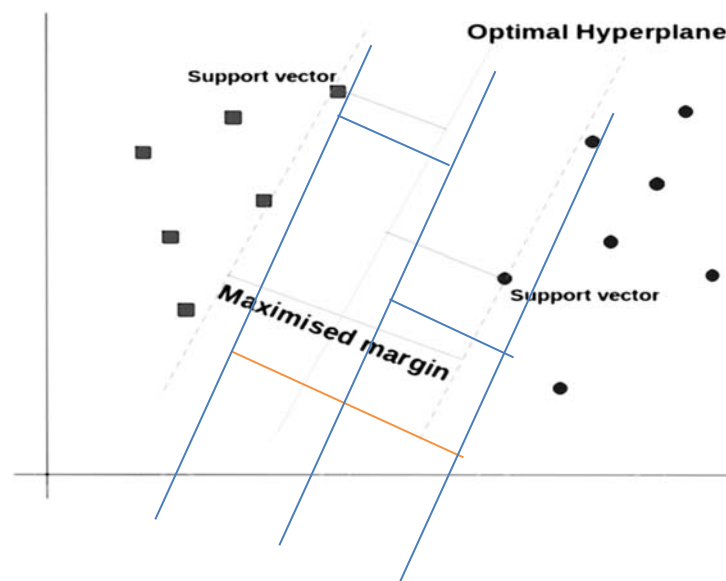


Рисунок 2

Листинг

```
import numpy as np
import matplotlib.pyplot as plt
from sklearn import svm, datasets
```

```

from sklearn.metrics import accuracy_score
from sklearn.datasets import make_classification

# Генерируем данные для обучения
X, y = make_classification(n_samples=1000, n_features=2, n_redundant=0,
n_informative=2, random_state=1, n_clusters_per_class=1)

# Создаем экземпляр SVM и обучаем модель с использованием линейного ядра
C = 1.0 # параметр регуляризации SVM
linear_svc = svm.SVC(kernel='linear', C=C).fit(X, y)

# Создаем экземпляр SVM и обучаем модель с использованием RBF-ядро
rbf_svc = svm.SVC(kernel='rbf', gamma=0.7, C=C).fit(X, y)

# Создаем экземпляр SVM и обучаем модель с использованием полиномиального
ядра
poly_svc = svm.SVC(kernel='poly', degree=2, C=C).fit(X, y)

# Создаем экземпляр SVM и обучаем модель с использованием сигмоидного
ядра
sig_svc = svm.SVC(kernel='sigmoid', C=C).fit(X, y)

# оцениваем качество моделей
print('Accuracy of linear kernel:', accuracy_score(y,
linear_svc.predict(X)))
print('Accuracy of polynomial kernel:', accuracy_score(y,
poly_svc.predict(X)))
print('Accuracy of RBF kernel:', accuracy_score(y, rbf_svc.predict(X)))
print('Accuracy of sigmoid kernel:', accuracy_score(y, sig_svc.predict(X)))
Вывод:

```

```

OUT:
Accuracy of linear kernel: 0.898
Accuracy of polynomial kernel: 0.924
Accuracy of RBF kernel: 0.922
Accuracy of sigmoid kernel: 0.815

```

#строим график для каждой из моделей, чтобы увидеть, как они разделяют данные

```

# Создаем сетку для графиков
h = 0.02 # шаг в сетке
x_min, x_max = X[:, 0].min() - 0.5, X[:, 0].max() + 0.5
y_min, y_max = X[:, 1].min() - 0.5, X[:, 1].max() + 0.5
xx, yy = np.meshgrid(np.arange(x_min, x_max, h), np.arange(y_min, y_max, h))

# Заголовки графиков
titles = ['SVM with linear kernel',
'SVM with RBF kernel',
'SVM with polynomial (degree 2) kernel',
'SVM with sigmoid kernel']

```

```
# Создаем график
plt.figure(figsize=(12, 8))

for i, clf in enumerate((linear_svc, rbf_svc, poly_svc, sig_svc)):

    # Рисуем границы принятия решений на графике
    plt.subplot(2, 2, i+1)
    Z = clf.predict(np.c_[xx.ravel(), yy.ravel()])
    Z = Z.reshape(xx.shape)
    plt.contourf(xx, yy, Z, cmap=plt.cm.Set1, alpha=0.8)

    # Рисуем точки данных
    plt.scatter(X[:, 0], X[:, 1], c=y, cmap=plt.cm.Set1, edgecolor='k')
    plt.title(titles[i])

plt.show()
```

Результат классификации отличается (рисунок 3).

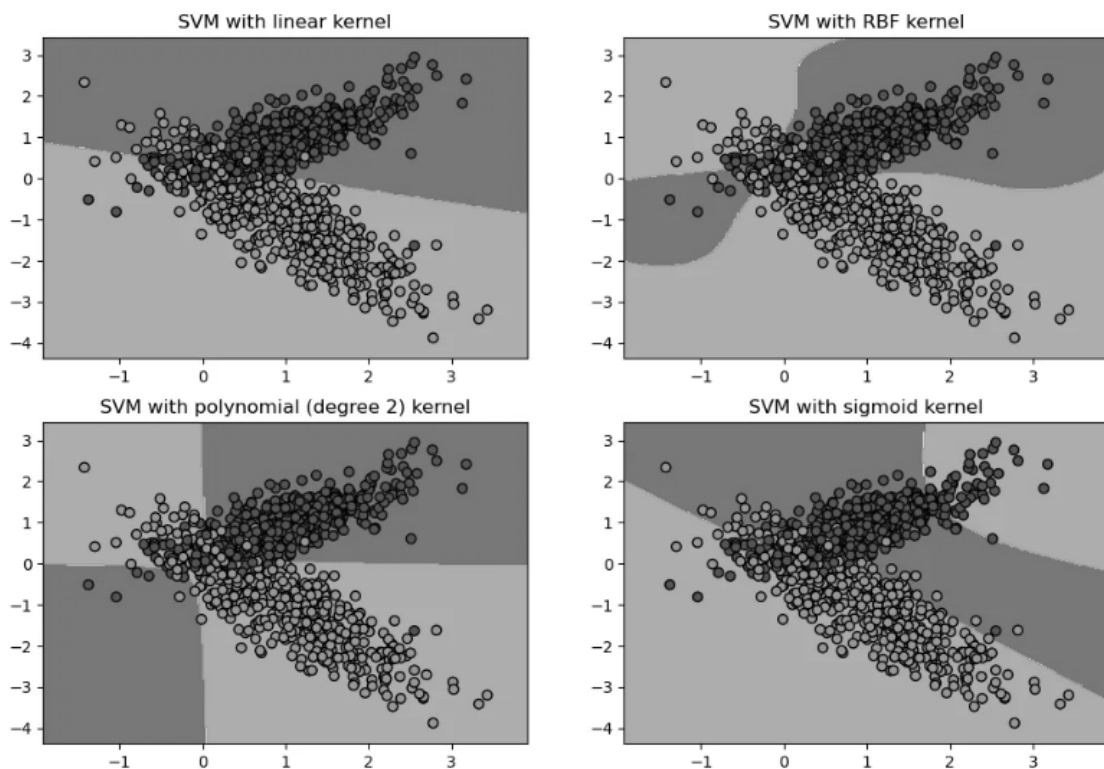


Рисунок 3

Задание

- 1 Разберите теоретическую часть и листинг на языке Python.
- 2 Подключите необходимые библиотеки и модули.
- 3 Загрузите датасет с двумя классами, подходящий для классификации.
- 4 Обучите модель SVM с линейным ядром.

- 5 Выполните алгоритм SVM с RBF-ядром.
- 6 Обучите модель SVM с полиномиальным ядром.
- 7 Постройте модель с сигмоидным ядром.
- 8 Сравните полученные результаты.
- 9 Визуализируйте разделение классов.

8 Лабораторная работа № 8. Работа с различными алгоритмами машинного обучения

Цель работы: изучить метод опорных векторов, ознакомиться с библиотеками Python для использования, научиться применять метод на практике.

Теоретические сведения

Рассмотрим Наивный Байес – семейство вероятностных алгоритмов, широко используемых в машинном обучении для задач классификации.

Основанный на статистических принципах, он применяет теорему Байеса с сильным (или «наивным») предположением о независимости между признаками.

Несмотря на простоту, этот метод весьма эффективен для категоризации данных, особенно в сценариях с наборами данных высокой размерности, таких как текст.

Он служит фундаментальным строительным блоком в области обучения с учителем, обеспечивая баланс между вычислительной эффективностью и прогностической производительностью.

Алгоритм предсказывает вероятность того, что данная точка данных принадлежит к определенному классу.

«Наивный» аспект проистекает из предположения, что наличие конкретного признака в классе не связано с наличием любого другого признака.

Например, фрукт можно считать яблоком, если он красный, круглый и имеет диаметр около 3 дюймов.

Классификатор Наивный Байес рассматривает каждый из этих пунктов извлечения признаков независимо, чтобы рассчитать вероятность того, что фрукт является яблоком, независимо от любых возможных корреляций между цветом, округлостью и размером.

Это упрощение резко снижает вычислительную мощность, необходимую для обучения модели, что делает алгоритм исключительно быстрым.

Однако, поскольку реальные данные часто содержат зависимые переменные и сложные взаимосвязи, это предположение иногда может ограничивать производительность модели по сравнению с более сложными архитектурами.

Наивный Байес отлично подходит для приложений, где скорость критически важна, а предположение о независимости выполняется достаточно хорошо.

Одно из самых известных применений Наивного Байеса – это обработка

естественного языка (NLP) для фильтрации электронной почты.

Классификатор анализирует частоту слов (токенов) в письме, чтобы определить, является ли оно «спамом» или «хэмом» (легитимным сообщением).

Он рассчитывает вероятность того, что сообщение является спамом при наличии таких слов, как «бесплатно», «победитель» или «срочно».

Это приложение в значительной степени полагается на методы текстовой классификации для поддержания чистоты входящих сообщений.

Компании используют этот алгоритм для оценки общественного мнения путем анализа отзывов клиентов или публикаций в социальных сетях.

Ассоциируя определенные слова с положительными или отрицательными тональностями, модель может быстро классифицировать огромные объемы обратной связи.

Это позволяет компаниям проводить масштабный анализ тональности, чтобы понять восприятие бренда без ручного чтения каждого комментария.

Пример реализации Наивного Байеса приведен в листинге.

Листинг

```
import numpy as np
from sklearn.naive_bayes import GaussianNB

# Sample training data: [height (cm), weight (kg)] and Labels (0: Cat A,
# 1: Cat B)
X = np.array([[175, 70], [180, 80], [160, 50], [155, 45]])
y = np.array([0, 0, 1, 1])

# Initialize and train the classifier
model = GaussianNB()
model.fit(X, y)

# Predict class for a new individual [172 cm, 75 kg]
# Returns the predicted class label (0 or 1)
print(f"Predicted Class: {model.predict([[172, 75]])[0]}")
```

Задание

1. Выбрать универсальный датасет.
2. Обучить модели: логистическая регрессия, KNN, дерево решений, случайный лес, SVM, гауссов Наивный Байес.
3. Сравнить метрики качества каждой модели.
4. Построить сводный отчет.
5. Сделать вывод о выборе алгоритма.

Контрольные вопросы

1. Как выбрать подходящий алгоритм машинного обучения для конкретной задачи: случайный лес, логистическая регрессия или дерево решений?

2 Как сравнить результаты работы двух алгоритмов машинного обучения на одном датасете?

Список литературы

1 **Лимановская, О. В.** Основы машинного обучения : учеб. пособие / О. В. Лимановская, Т. И. Алферьева. – Екатеринбург : Урал. ун-т, 2020. – 88 с.

2 Алгоритмы искусственного интеллекта на Python : метод. рекомендации для семинарских (практических) занятий, лабораторного практикума и самостоятельной работы для студентов направления подготовки 11.04.02 «Инфокоммуникационные технологии и системы связи» / сост. С. В. Липатова. – Ульяновск : Ульянов. гос. ун-т, 2019. – 106 с.

3 Практикум по статистике / под ред. Е. Г. Борисовой. – 4-е изд., перераб. и доп. – М. : МГИМО-Университет, 2020. – 166 с.