ЛЕКЦИЯ 3
Программирование на Python для решения задач искусственного интеллекта

Цель лекции

Цель лекции — сформировать у обучающихся представление о роли программирования в разработке систем искусственного интеллекта, показать возможности языка Python для анализа и подготовки данных, познакомить с базовыми принципами работы библиотек NumPy и Pandas и подготовить обучающихся к практической реализации этапов машинного обучения в среде Google Colab.

Роль программирования в задачах искусственного интеллекта

Методы искусственного интеллекта реализуются в виде программных процедур. Независимо от того, решается ли задача классификации, регрессии, кластеризации, анализа текста или распознавания изображений, разработчику необходимо получить данные, представить их в форме, пригодной для вычислений, выполнить необходимые преобразования, запустить алгоритм и проанализировать результат.

Поэтому программирование в искусственном интеллекте выполняет несколько функций. Оно обеспечивает загрузку и хранение данных, автоматизацию вычислений, реализацию алгоритмов, проведение серии экспериментов, сравнение результатов и интеграцию обученной модели в прикладную систему.

Python как язык для анализа данных и машинного обучения

Python является универсальным языком программирования, который широко применяется при решении задач искусственного интеллекта и машинного обучения. Его практическая ценность связана с тем, что один и тот же язык можно использовать для загрузки данных, численных вычислений, статистического анализа, визуализации, подготовки признаков, обучения моделей и оценки результатов.

По искусственному интеллекту среди основных библиотек для работы с данными на Python выделяются NumPy, SciPy, Pandas, Matplotlib и scikit-learn. В данной лекции основное внимание уделяется NumPy и Pandas, поскольку именно они составляют базу для последующей работы с данными и моделями машинного обучения.

При этом важно различать язык программирования и библиотеку. Python задаёт синтаксис и общие средства написания программы, а библиотека предоставляет готовые структуры данных и функции для специализированных задач. Например, NumPy предназначен для эффективной работы с числовыми массивами, а Pandas — для хранения, преобразования и анализа табличных данных.

Организация вычислительного эксперимента

При изучении искусственного интеллекта программный код удобно организовывать как последовательность этапов. Такая последовательность соответствует общему процессу машинного обучения: сбор данных → исследование данных → подготовка данных → обучение модели → оценка модели → улучшение модели.

На практике каждый из этих этапов может быть представлен отдельным блоком программы. Сначала подключаются библиотеки и загружается набор данных. Затем исследуется его структура, вычисляются статистические характеристики, выявляются отсутствующие значения и категориальные признаки. После этого данные преобразуются в форму, пригодную для обучения модели. Только после подготовки выполняется обучение и оценка результата.

Такой порядок важен не только с методической точки зрения. Он делает эксперимент воспроизводимым: можно проследить, какие данные использовались, какие преобразования выполнялись и как получен итоговый результат.

Google Colab и формат Notebook

Для учебных работ удобно использовать среду Google Colab, работающую с документами типа Jupyter Notebook. Notebook состоит из ячеек, в которых можно размещать программный код, текстовые пояснения и результаты вычислений. Это позволяет выстраивать эксперимент пошагово и сразу видеть результат выполнения каждой операции.

Например, первая ячейка может содержать подключение библиотек, вторая — загрузку файла, третья — вывод первых строк таблицы, четвёртая — вычисление статистики, пятая — обработку пропущенных значений. В последующих ячейках выполняются подготовка признаков, обучение модели и оценка качества.

Для образовательной работы такой формат удобен тем, что объединяет код и пояснения в одном документе. Однако облачная среда требует осознанного отношения к данным: в учебных заданиях целесообразно использовать открытые или специально подготовленные наборы данных и не размещать в публичных блокнотах конфиденциальную информацию.

Базовые элементы программы на Python

Любая программа состоит из данных и операций над ними. В Python данные могут храниться в переменных. Переменная связывает имя с определённым значением. Для задач искусственного интеллекта переменные могут содержать значения признаков, параметры алгоритма, результаты измерений и метрики качества.

```
accuracy = 0.94
model_name = "kNN"
k = 5
```

В Python используются числовые значения, строки, логические значения и различные коллекции. Для хранения последовательности элементов можно применять список. Для хранения пар «ключ — значение» используется словарь. Эти структуры удобны при работе с параметрами эксперимента, небольшими наборами значений и результатами вычислений.

```
classes = ["normal", "attack"]
result = {"prediction": "attack", "probability": 0.91}
```

Для выполнения разных действий в зависимости от условия используются условные операторы. Для многократного выполнения операции применяются циклы. Повторяющиеся действия целесообразно оформлять как функции. Однако при обработке больших числовых наборов данных вместо ручного перебора элементов часто используются векторные операции NumPy.

Подключение библиотек

Перед использованием специализированной библиотеки её необходимо подключить. В Python для этого используется команда import. В учебных и исследовательских проектах приняты стандартные сокращения названий библиотек.

```
import numpy as np
import pandas as pd
```

После такого подключения обращение к функциям NumPy выполняется через префикс np, а к функциям Pandas — через pd. Это упрощает код и соответствует распространённой практике работы с данными.

NumPy и числовое представление данных

Большинство алгоритмов машинного обучения оперирует числовыми данными. Даже если исходный объект является изображением, текстом или сетевым событием, перед вычислениями его необходимо представить в виде чисел. Поэтому важным элементом программирования в искусственном интеллекте является работа с векторами и матрицами.

NumPy предоставляет структуру многомерного массива, предназначенную для эффективного хранения чисел и выполнения операций над ними. Одномерный массив

можно рассматривать как вектор, двумерный — как матрицу. Набор объектов машинного обучения обычно удобно представлять в виде матрицы, где строки соответствуют объектам, а столбцы — признакам.

```
x = np.array([2.4, 35, 5200, 0])
```

В приведённом примере один объект описывается четырьмя числовыми признаками. Если таких объектов много, формируется двумерный массив. Именно в такой форме данные могут передаваться алгоритмам классификации или регрессии.

В методических материалах ИТМО NumPy разрешается использовать при программной реализации классических алгоритмов машинного обучения. Например, в заданиях по k-ближайшим соседям и логистической регрессии предлагается реализовывать вычисления самостоятельно, используя NumPy и Pandas как базовые инструменты. Это подчёркивает роль NumPy не как готового классификатора, а как средства выполнения математических операций.

Основные операции с массивами NumPy

При исследовании данных необходимо уметь получать размер массива, выбирать его элементы, вычислять агрегированные характеристики и выполнять преобразования. Наиболее часто используются сумма, среднее значение, минимум, максимум и стандартное отклонение.

```
values = np.array([10, 15, 12, 18, 20])
values.mean()
values.min()
values.max()
values.std()
```

Такие операции необходимы на этапе первичного анализа. Например, при исследовании характеристик сетевого трафика можно вычислить среднее число пакетов, минимальный и максимальный объём передачи данных, а также оценить разброс значений.

Важным преимуществом массивов NumPy является возможность выполнять одну операцию сразу над множеством значений. Такой подход называется векторизованными вычислениями. Он делает код компактнее и обычно эффективнее, чем последовательная обработка каждого числа в цикле Python.

Pandas и табличные наборы данных

В прикладных задачах машинного обучения данные часто представлены в табличном виде. Каждая строка соответствует отдельному объекту наблюдения, а каждый столбец — определённой характеристике. Для работы с такой структурой используется библиотека Pandas.

Основной объект Pandas — DataFrame. Его можно представить как таблицу с именованными столбцами. В отличие от обычной числовой матрицы DataFrame может одновременно содержать числовые и категориальные данные, например возраст, количество событий, тип протокола и текстовую метку класса.

Методические материалы, использованные для подготовки лекции, связывают Pandas с хранением и анализом данных, предварительной обработкой и подготовкой набора для последующего обучения модели. Поэтому работа с DataFrame является одним из центральных навыков при переходе от исходного файла к алгоритму машинного обучения.

Загрузка данных в Pandas

Одним из наиболее распространённых форматов для учебных наборов данных является CSV. Загрузка такого файла выполняется функцией read_csv.

```
data = pd.read_csv("dataset.csv")
```

После загрузки не следует сразу переходить к построению модели. Сначала необходимо проверить структуру таблицы. Полезно вывести несколько первых строк,

определить количество объектов и признаков, посмотреть названия столбцов и типы данных.

```
data.head()
data.shape
data.columns
data.info()
```

Эти операции позволяют убедиться, что файл загружен корректно и что столбцы имеют ожидаемый смысл. Например, числовой признак не должен случайно интерпретироваться как текст из-за ошибок в исходном файле.

Описательная статистика

В лабораторных заданиях ИТМО при работе с наборами данных предлагается получать и визуализировать статистику, включая количество наблюдений, среднее значение, стандартное отклонение, минимум, максимум и квантили. Такой анализ позволяет получить первое представление о диапазоне и распределении значений.

```
data.describe()
```

Если, например, среднее значение некоторого признака невелико, а максимальное значение в сотни раз больше, необходимо проверить, является ли это реальным наблюдением или потенциальной ошибкой. Статистический анализ не заменяет содержательного понимания данных, но помогает обнаружить признаки, требующие дополнительного внимания.

Пропущенные значения

Реальные наборы данных часто содержат пропуски. Они могут возникать из-за отсутствия измерения, ошибки регистрации или неполного объединения данных из разных источников. До обучения модели необходимо определить, в каких столбцах имеются пропущенные значения и насколько их много.

```
data.isna().sum()
```

Дальнейшее решение зависит от конкретной задачи. Некоторые строки можно удалить, если пропусков немного и удаление не искажает выборку. В других случаях пропуски заменяются некоторым значением, например медианой или другим обоснованным показателем. Важно не применять один и тот же способ автоматически ко всем признакам.

В методических заданиях ИТМО обработка отсутствующих значений прямо входит в этап предварительной подготовки данных перед построением моделей линейной регрессии и k-ближайших соседей.

Категориальные признаки

Не все признаки представлены числами. В наборе данных могут встречаться категории: тип протокола, название сервиса, регион, класс устройства и другие текстовые значения. Многие алгоритмы требуют числового представления таких признаков, поэтому перед обучением выполняется кодирование категорий.

Например, значения протокола TCP, UDP и ICMP нельзя интерпретировать как обычный числовой ряд 1, 2, 3, если такое упорядочивание не имеет содержательного смысла. Для категориальных данных применяются специальные способы кодирования. Конкретный метод выбирается с учётом типа признака и последующего алгоритма.

Масштабирование и нормализация

В задачах, где расстояние между объектами играет существенную роль, признаки с разными масштабами могут влиять на результат неравномерно. Например, один признак может изменяться от 0 до 1, а другой — от 0 до 100 000. В этом случае второй признак способен доминировать при вычислении расстояния.

Именно поэтому в лабораторных заданиях по методу k-ближайших соседей отдельно указано масштабирование данных, а в заданиях по линейной регрессии — нормализация. Эти операции приводят числовые признаки к более сопоставимым диапазонам и являются частью предварительной обработки.

Разделение на обучающую и тестовую выборки

После подготовки данных необходимо отделить данные для обучения от данных для проверки. Обучающая выборка используется для настройки параметров модели. Тестовая выборка предназначена для оценки того, насколько хорошо модель работает на объектах, которые не участвовали в обучении.

Такое разделение присутствует в методических заданиях ИТМО для линейной и логистической регрессии. Смысл разделения заключается в том, что качество нельзя оценивать только по тем данным, на которых алгоритм обучался. Иначе полученная оценка не показывает способность модели обобщать закономерности на новые наблюдения.

Признаки и целевая переменная

При подготовке данных для обучения с учителем таблица логически разделяется на входные признаки и целевую переменную. Признаки описывают объект, а целевая переменная содержит значение, которое требуется прогнозировать.

Например, в задаче определения вредоносного сетевого соединения признаками могут быть длительность соединения, количество пакетов, объём переданных данных и тип протокола. Целевая переменная указывает класс объекта — нормальное соединение или атака.

```
X = data[["duration", "packets", "bytes"]]
y = data["label"]
```

Такое разделение подготавливает набор данных к передаче в алгоритм машинного обучения.

Визуализация данных

Исследование набора данных не ограничивается числовой статистикой. Графическое представление помогает увидеть форму распределений, выбросы, дисбаланс классов и возможные зависимости между признаками.

Рабочая программа дисциплины относит Matplotlib к базовым библиотекам анализа данных на Python, а методические задания требуют графически визуализировать статистику и признаки. Поэтому визуализация должна рассматриваться как часть исследовательского анализа перед обучением модели, а не только как способ оформления итогового отчёта.

Например, гистограмма позволяет оценить распределение числового признака, диаграмма рассеяния — связь между двумя признаками, а столбчатая диаграмма — соотношение классов.

От данных к модели машинного обучения

После загрузки, анализа и подготовки данных начинается собственно этап машинного обучения. Методическое пособие ИТМО рассматривает линейную регрессию, метод k-ближайших соседей, деревья решений и логистическую регрессию. В лабораторных заданиях часть алгоритмов предлагается реализовать без готовых библиотек машинного обучения, используя NumPy и Pandas.

Такой подход имеет учебную ценность: обучающийся видит внутреннюю структуру вычислений и понимает, как исходные данные преобразуются в прогноз. Например, при реализации k-ближайших соседей необходимо вычислять расстояния до объектов обучающей выборки и анализировать классы ближайших наблюдений. При реализации логистической регрессии требуется задать сигмоидальную функцию, функцию потерь и процедуру оптимизации параметров.

В дальнейшем, когда принцип метода понятен, для прикладной разработки могут использоваться готовые реализации из специализированных библиотек. Однако способность объяснить этапы вычислений остаётся важной профессиональной компетенцией.

Типичная структура Python-программы для AI-задачи

Обобщённую структуру учебного проекта можно представить следующим образом.

1. Подключить необходимые библиотеки.
2. Загрузить набор данных.
3. Проверить размер, типы данных и состав признаков.
4. Получить описательную статистику и визуализировать данные.
5. Обработать пропущенные значения и категориальные признаки.
6. При необходимости выполнить масштабирование или нормализацию.
7. Выделить признаки X и целевую переменную y.
8. Разделить данные на обучающую и тестовую части.
9. Реализовать или подключить алгоритм машинного обучения.
10. Выполнить обучение и сформировать прогноз.
11. Оценить качество модели соответствующими метриками.
12. Сравнить варианты модели и сделать выводы.

Эта последовательность соответствует логике учебной программы и практических заданий: от исследования исходного набора данных к построению и оценке модели.

Пример: ***анализ данных информационной безопасности***

Рассмотрим учебный набор данных о сетевых соединениях. Пусть каждая строка содержит длительность соединения, число пакетов, объём переданных данных, тип протокола и метку класса. Цель состоит в подготовке данных для последующего построения классификатора.

Сначала таблица загружается в Pandas. Затем проверяются размер набора и типы столбцов. С помощью описательной статистики анализируются числовые признаки. Проверяется наличие пропусков. Категориальный признак «протокол» преобразуется в числовую форму. Далее данные разделяются на признаки и целевой класс. Если используемый алгоритм чувствителен к масштабу, числовые признаки масштабируются. После этого формируются обучающая и тестовая выборки.

Таким образом, даже до выбора конкретного классификатора необходимо выполнить значительный объём программной работы. Качество подготовки набора данных непосредственно влияет на возможность корректно обучить и оценить модель.

Воспроизводимость и документирование эксперимента

При выполнении учебной или исследовательской работы важно сохранять последовательность действий, параметры эксперимента и полученные результаты. Notebook хорошо подходит для этой задачи, поскольку в одном документе можно оставить код, пояснения, промежуточные таблицы и графики.

Если проводится сравнение нескольких вариантов модели, необходимо фиксировать, какие признаки использовались, как выполнялась подготовка данных, какие параметры были выбраны и какие метрики получены. В лабораторных заданиях ИТМО также предусматривается подготовка кратких и расширенных отчётов по результатам практических работ, что подчёркивает значение документирования вычислительного эксперимента.

Типичные ошибки при работе с данными

При программировании AI-задач необходимо избегать нескольких распространённых ошибок. Первая — переход к обучению модели без предварительного исследования

данных. Вторая — механическое удаление всех строк с пропусками без анализа причин их появления. Третья — использование категориальных признаков без корректного преобразования. Четвёртая — сравнение признаков с резко различающимися масштабами без учёта особенностей выбранного алгоритма.

Ещё одна принципиальная ошибка — оценивать модель на тех же данных, на которых она обучалась. Именно поэтому учебные задания предусматривают разделение на обучающий и тестовый наборы. Корректная организация эксперимента является не менее важной, чем написание синтаксически правильного кода.

Связь лекции с последующими темами курса

Освоение Python, NumPy и Pandas создаёт основу для следующих тем курса. При изучении подготовки и предобработки данных эти инструменты будут использоваться более детально. При изучении классификации и регрессии подготовленные массивы и таблицы станут входом для моделей. При переходе к нейронным сетям числовое представление данных будет использоваться уже в виде многомерных тензоров.

Поэтому цель данной лекции — не изучить весь язык Python, а сформировать набор программных навыков, необходимых для дальнейшего освоения искусственного интеллекта: понимать структуру данных, уметь выполнить первичный анализ, подготовить признаки и организовать вычислительный эксперимент.

Заключение

Python является удобным инструментом программной реализации задач искусственного интеллекта и машинного обучения. В учебном AI-проекте он используется для загрузки, исследования и преобразования данных, выполнения вычислений, реализации алгоритмов и анализа результатов.

NumPy обеспечивает работу с числовыми массивами и математическими операциями. Pandas предоставляет табличную структуру DataFrame и средства анализа структурированных данных. Эти библиотеки позволяют реализовать ключевые этапы, предусмотренные учебными программами и практическими заданиями: анализ статистики, обработку пропусков, кодирование категориальных признаков, нормализацию и масштабирование, разделение на обучающую и тестовую выборки.

Основная логика программирования для задач искусственного интеллекта может быть представлена как последовательность: данные → исследование → подготовка → модель → прогноз → оценка. Понимание этой последовательности позволяет перейти от простого запуска готовых средств к осознанной разработке и исследованию интеллектуальных систем.

Контрольные вопросы

1. Какую роль программирование выполняет в разработке систем искусственного интеллекта?
2. Почему Python удобно использовать для анализа данных и машинного обучения?
3. Какие основные библиотеки Python для анализа данных указаны в учебной программе?
4. Какие этапы включает типичный процесс решения задачи машинного обучения?
5. В чём преимущество организации эксперимента в формате Jupyter Notebook или Google Colab?
6. Что представляет собой библиотека NumPy?
7. Чем одномерный массив отличается от двумерного массива?
8. Почему данные для большинства алгоритмов машинного обучения должны быть представлены в числовой форме?
9. Что понимается под векторизованными вычислениями?
10. Что представляет собой DataFrame в Pandas?

11. Какие действия необходимо выполнить сразу после загрузки табличного набора данных?

12. Для чего используется описательная статистика?

13. Почему пропущенные значения требуют отдельного анализа?

14. Что такое категориальный признак и почему его часто необходимо кодировать?

15. Для чего применяются масштабирование и нормализация признаков?

16. Зачем разделять данные на обучающую и тестовую выборки?

17. В чём различие между признаками и целевой переменной?

18. Какую роль играет визуализация данных перед обучением модели?

19. Почему полезно реализовывать некоторые алгоритмы машинного обучения самостоятельно с использованием NumPy и Pandas?

20. Какие ошибки при подготовке данных могут привести к некорректной оценке модели?

Список литератур:

1. Бессмертный И.А., Авдюшина А.Е., Кугаевских А.В., Королева Ю.А., Рущенко Н.Г. Системы искусственного интеллекта. Методическое пособие. Санкт-Петербург: Университет ИТМО, 2024.

2. Рабочая программа дисциплины «Искусственный интеллект и машинное обучение». Направление подготовки 09.03.01 «Информатика и вычислительная техника». Москва, 2024.

3. Сидоркина И.Г. Системы искусственного интеллекта. Учебное пособие. Москва: КНОРУС, 2015.

4. Бондарев В.Н., Аде Ф.Г. Искусственный интеллект. Учебное пособие. Севастополь, 2002.