УТВЕРЖДЕНО
решением Ученого совета ИЭиБ
от « 22 » июня 2023 г., протокол № 09 / 261
Председатель _____ И.Б.Романова
« 22 » июня 2023 г.

РАБОЧАЯ ПРОГРАММА ДИСЦИПЛИНЫ

Дисциплина	Машинное обучение
Факультет	Экономики
Кафедра	Цифровой экономики
Курс	1

Направление: 38.04.01 Экономика
Направленность (профиль): «Искусственный интеллект в финансово-экономических системах»
Форма обучения: очная

Дата введения в учебный процесс УлГУ: « 01 » сентября 2023 г.

Программа актуализирована на заседании кафедры: протокол №_____ от _____ 20_____г.
Программа актуализирована на заседании кафедры: протокол №_____ от _____ 20_____г.
Программа актуализирована на заседании кафедры: протокол №_____ от _____ 20_____г.

Сведения о разработчиках:

ФИО	Кафедра	Должность, ученая степень, звание
Сковиков А.Г.	Цифровой экономики	к.т.н., доцент

СОГЛАСОВАНО
Заведующий выпускающей кафедрой цифровой экономики _____ / Лутошкин И.В. / *Подпись* *ФИО* «21» июня 2023 г.

ВВЕДЕНИЕ

Настоящая программа учебной дисциплины устанавливает требования к образовательным результатам и определяет содержание и виды учебных занятий и отчетности.

По мере развития машинного обучения была выявлена проблема автоматического приобретения знаний, которая была сформулирована как проблема машинного обучения. Важность обучения понималась многими учёными в области искусственного интеллекта, но после исследований в области представления знаний стало ясно, насколько большой объём информации получает человек в процессе обучения, и насколько трудоёмко закладывать эти знания в машинные системы вручную. Машинное обучение стало центральным направлением исследований в области искусственного интеллекта, выделившись в самостоятельное направление в 1980-е годы. Учёные перестали рассматривать интеллект как некий готовый продукт, который можно воспроизвести, или как фиксированную способность к решению задач и манипулированию знаниями. В задачах поиска была сформулирована проблема автоматического построения эвристик поиска – оптимизации поиска, т.к. методы машинного обучения применяются к неопределённым и противоречивым наблюдаемым данным, из-за чего приобретённые знания не обладают полной достоверностью. Следствием этого является проблема представления нечётких знаний в условиях неопределённости.

Актуальность представленной образовательной программы «Машинное обучение» в рамках направления магистерской подготовки 38.04.01 Экономика обусловлена возросшей потребностью в специалистах, способных к решению профессиональных задач по модификации, внедрению и использованию интеллектуальных систем для решения задач информационного поиска, извлечения знаний и поддержки принятия решений.

Дисциплина «Машинное обучение» реализуется в рамках образовательной программы высшего образования – программы магистратуры 38.04.01 ЭКОНОМИКА, направленность (профиль): ИСКУССТВЕННЫЙ ИНТЕЛЛЕКТ В ФИНАНСОВО-ЭКОНОМИЧЕСКИХ СИСТЕМАХ.

Направленность курса «Машинное обучение» отвечает целям Федерального проекта «Кадры для цифровой экономики» Национальной программы «Цифровая экономика РФ».

Программа составлена в соответствии с требованиями Федеральных государственных образовательными стандартов высшего образования.

Прогностичность программы «Машинное обучение» заключается в том, что она отражает требования и актуальные тенденции не только сегодняшнего, но и завтрашнего дня, а также имеет междисциплинарный характер, что полностью отражает современные тенденции построения образования в целом. В процессе изучения машинного обучения и нейронных сетей, обучающиеся получают дополнительное образование в области биологии, физики, математики, информатики. В результате у них развиваются научно-исследовательские, технологические и гуманитарные компетенции.

1. ЦЕЛИ И ЗАДАЧИ ОСВОЕНИЯ ДИСЦИПЛИНЫ

Цель дисциплины является формирование у студентов теоретических знаний и практических навыков по основам машинного обучения, овладение обучающимися инструментарием, моделями и методами машинного обучения, а также приобретение навыков исследователя данных (data scientist) и разработчика математических моделей, методов и алгоритмов анализа данных.

Задачи дисциплины:

1. изучение понятийного аппарата дисциплины, основных теоретических положений и методов;

2. изучение основных принципов организации информационных процессов в нейрокомпьютерных система;
3. формирование логического мышления;
4. формирование навыков разработки и реализации программных моделей нейрокомпьютерных систем
5. приобретение навыков разработки систем на основе машинного обучения с целью закрепления полученных знаний.

В результате изучения курса обучающиеся должны освоить способы самоорганизации и саморазвития на основе комплексного представления о том, как новые технологии изменяют нашу жизнь и жизнь будущих поколений, как они преобразуют экономическую, социальную, культурную и гуманитарную среду нашего обитания.

2. МЕСТО ДИСЦИПЛИНЫ В СТРУКТУРЕ ОПОП

Дисциплина «Машинное обучение» относится к дисциплинам части, формируемой участниками образовательных отношений в системе подготовки магистров по направлению 38.04.01 Экономика, профиль «Искусственный интеллект в финансово-экономических системах».

Дисциплина рассчитана на обучающихся, имеющих начальную подготовку в области информационных технологий и систем, глобальных сетей, организации и инфраструктуры предпринимательской деятельности, коммерции, алгебры, теории вероятности, знакомых с основными понятиями физики, комбинаторики, информатики. Помимо этого, для успешного освоения данного курса магистранту необходимы навыки самостоятельной работы с различными источниками информации (интернет, печатные издания), умение обобщать информацию, полученную из разных источников, умение представлять результаты своих исследований.

Изучаемая дисциплина является основой для продолжения формирования указанных компетенций в следующих практиках и процедуре ГИА: Производственная практика. Практика по профилю профессиональной деятельности Преддипломная практика; Подготовка к процедуре защиты и защита выпускной квалификационной работы.

Дисциплина направлена на изучение основных теоретических положений и методов, формирование умений и привитие навыков применения теоретических знаний для решения прикладных задач, а также развитие новых подходов к применению интеллектуальных технологий в сфере экономики.

3. ПЕРЕЧЕНЬ ПЛАНИРУЕМЫХ РЕЗУЛЬТАТОВ ОБУЧЕНИЯ ПО ДИСЦИПЛИНЕ (МОДУЛЮ), СООТНЕСЕННЫХ С ПЛАНИРУЕМЫМИ РЕЗУЛЬТАТАМИ ОСВОЕНИЯ ОБРАЗОВАТЕЛЬНОЙ ПРОГРАММЫ

В результате освоения дисциплины формируются следующие компетенции:

Код и наименование реализуемой компетенции	Перечень планируемых результатов обучения по дисциплине (модулю), соотнесенных с индикаторами достижения компетенций
ПК-5 Способен адаптировать и применять методы и алгоритмы машинного обучения для решения прикладных задач в различных предметных областях	***знать:*** Классы методов и алгоритмов машинного обучения
	уметь: Ставить задачи и адаптировать методы и алгоритмы машинного обучения
	владеть: Способностью ставить задачи по адаптации или совершенствованию методов и алгоритмов для решения комплекса задач предметной области.

ПК-6 Способен руководить проектами по созданию систем искусственного интеллекта с применением новых методов и алгоритмов машинного обучения со стороны заказчика	***знать:*** Возможности современных инструментальных средств и систем программирования для решения задач машинного обучения; функциональность современных инструментальных средств и систем программирования в области создания моделей и методов машинного обучения; принципы построения систем искусственного интеллекта, методы и подходы к планированию и реализации проектов по созданию систем искусственного интеллекта, методы интеллектуального планирования экспериментов.
	уметь: Проводить сравнительный анализ и осуществлять выбор инструментальных средств для решения задач машинного обучения; применять современные инструментальные средства и системы программирования для разработки новых моделей и методов машинного обучения.
	владеть: Способностью руководить разработкой архитектуры комплексных систем искусственного интеллекта со стороны заказчика; Способностью осуществлять руководство коллективной проектной деятельностью для создания, поддержки и использования систем искусственного интеллекта.

4. ОБЩАЯ ТРУДОЕМКОСТЬ ДИСЦИПЛИНЫ

4.1. Объем дисциплины в зачётных единицах (всего): 3 з.е.

4.2. Объем дисциплины по видам учебной работы (в часах)

Вид учебной работы	Количество часов (форма обучения –**очная)**	
	Всего по плану	в т.ч. по семестрам
		2
Контактная работа обучающихся с преподавателем	36*	36*
Аудиторные занятия:		
лекции	18*	18*
практические и семинарские занятия	18*	18*
лабораторные работы, практикумы	-	-
Самостоятельная работа	72	72
Форма текущего контроля знаний и контроля самостоятельной работы	Тестирование, устный опрос, решение кейсов, оценивание реферата	Тестирование, устный опрос, решение кейсов, , оценивание реферата
Виды промежуточной аттестации	зачет	зачет
Всего часов по дисциплине	108	108

*В случае необходимости использования в учебном процессе частично/исключительно дистанционных образовательных технологий, указанные часы работы ППС с обучающимися проводятся в дистанционном формате с применением электронного обучения

4.3. Содержание дисциплины (модуля.) Распределение часов по темам и видам учебной работы:

Форма обучения: очная

Название и разделов и тем	Всего	Виды учебных занятий					Форма текущего контроля знаний
		Аудиторные занятия			Занятия в интерактивной форме	Самостоятельная работа	
		лекции	практические занятия, семинары	лабораторные работы			
1	2	3	4	5	6	7	
Модуль 1. Введение в машинное обучение	24	9	9		4	36	проверка тестовых заданий, устный опрос, решение кейсов, оценивание реферата
Модуль 2. Решение задач машинного обучения	18	9	9		4	36	проверка тестовых заданий, устный опрос, решение кейсов, оценивание реферата
Итого	108	18	18		8	72	

5. СОДЕРЖАНИЕ КУРСА

Модуль 1. Этапы развития систем искусственного интеллекта

1. Основные понятия и пирамида задач машинного обучения. Машинное обучение и ИИ. Архитектура систем ИИ.
2. Предмет и задачи машинного обучения и анализа данных. Основные принципы, задачи и подходы, использование в различных областях науки и индустрии. Основные этапы эволюции алгоритмов машинного обучения.
3. Постановки задач регрессионного анализа и классификации. Постановки задач анализа структуры данных и кластеризации. Примеры задач из области финансов и экономики. Подходы к их решению.
4. Сбор, фильтрация, подготовка и предварительный анализ данных. Разметка данных для последующего применения методов обучения с учителем.
5. Онтология предметной области. Графы знаний. Применение графов знаний для решения задач. Извлечение знаний из данных.
6. Области применения искусственных нейронных сетей. Биологический нейрон. Структура и свойства искусственного нейрона. Разновидности искусственных нейронов.
7. Типы задач машинного обучения.

Модуль 2. Системы аналитической обработки информации

1. Постановка и возможные пути решения задачи обучения искусственных нейронных сетей: обучение с учителем, алгоритм обратного распространения ошибки; обучение без учителя.
2. Решение задач регрессионного анализа.
3. Решение задач классификации.
4. Снижение размерности. Анализ структуры данных.

5. Алгоритмы кластеризации.
6. Деревья решений. Леса решающих деревьев.
7. Линейные классификаторы
8. Перцептрон и разделяющая гиперплоскость.
9. Нейронные сети и глубокое обучение.
10. Ансамблевые методы. Голосование. Бутстраппинг. Бустинг, адаптивный бустинг, градиентный бустинг.

6. ТЕМЫ ПРАКТИЧЕСКИХ И СЕМИНАРСКИХ ЗАНЯТИЙ

Цель проведения семинарских и практических занятий заключается в закреплении полученных теоретических знаний на лекциях и в процессе самостоятельного изучения студентами специальной литературы. Основной формой проведения семинарских и практических занятий является обсуждение наиболее проблемных и сложных вопросов по отдельным темам.

Часть практических занятий проводится в интерактивной форме: обсуждение проблемных вопросов на круглых столах.

В обязанности преподавателя входят оказание методической помощи и консультирование магистрантов по соответствующим темам курса.

№ п/п	№ модуля	Тематика практических занятий	Кол-во часов
1	1	Практическое занятие. Разбор примеров задач, возникающие при построении финансово-экономических систем	2
2	1	Практическое занятие. Знакомство со средой решения задач. Jupyter notebook. Google Colab. Использование готовых решений.	2
3	1	Практическое занятие. Изучаем базовые конструкции языка Python. Работа с коллекциями данных.	2
4	1	Практическое занятие. Чтение данных из различных источников, в том числе из репозиториев данных. Предварительный анализ данных. Библиотеки pandas и matplotlib.	3
5	2	Практическое занятие. Решение задач регрессионного анализа с использованием библиотеки scikit-learn	3
6	2	Практическое занятие. Решение задач классификации с использованием библиотеки scikit-learn	3
7	2	Практическое занятие. Решение задач снижения размерности и кластеризации с использованием библиотеки scikit-learn	3
		Итого:	**18**

МОДУЛЬ 1. ВВЕДЕНИЕ В МАШИННОЕ ОБУЧЕНИЕ

Практическое занятие №1. Разбор примеров задач, возникающие при построении финансово-экономических систем

Вопросы для рассмотрения на семинарских занятиях. *Изучение нижеперечисленных вопросов будет производиться на примере конкретных ситуаций, с целью формирования и развития профессиональных навыков обучающихся.*
Форма проведения занятия – практическое занятие.

Вопросы для самоконтроля и текущего контроля

1. Какую задачу можно назвать задачей машинного обучения?
2. Каким требованиям должна отвечать программная система, чтобы ее можно было отнести к реализации машинного обучения?
3. Чем отличается сильный ИИ от слабого ИИ?
4. Как можно протестировать сильный ИИ?
5. Перечислите иерархию задач машинного обучения
6. Приведите примеры задач, которые приходится решать прежде чем можно будет применить алгоритмы машинного обучения.
7. Какие задачи решает дата-инженер?
8. Какие задачи решает аналитик данных?
9. Кто такой дата-сайентист?

Что такое машинное обучение?

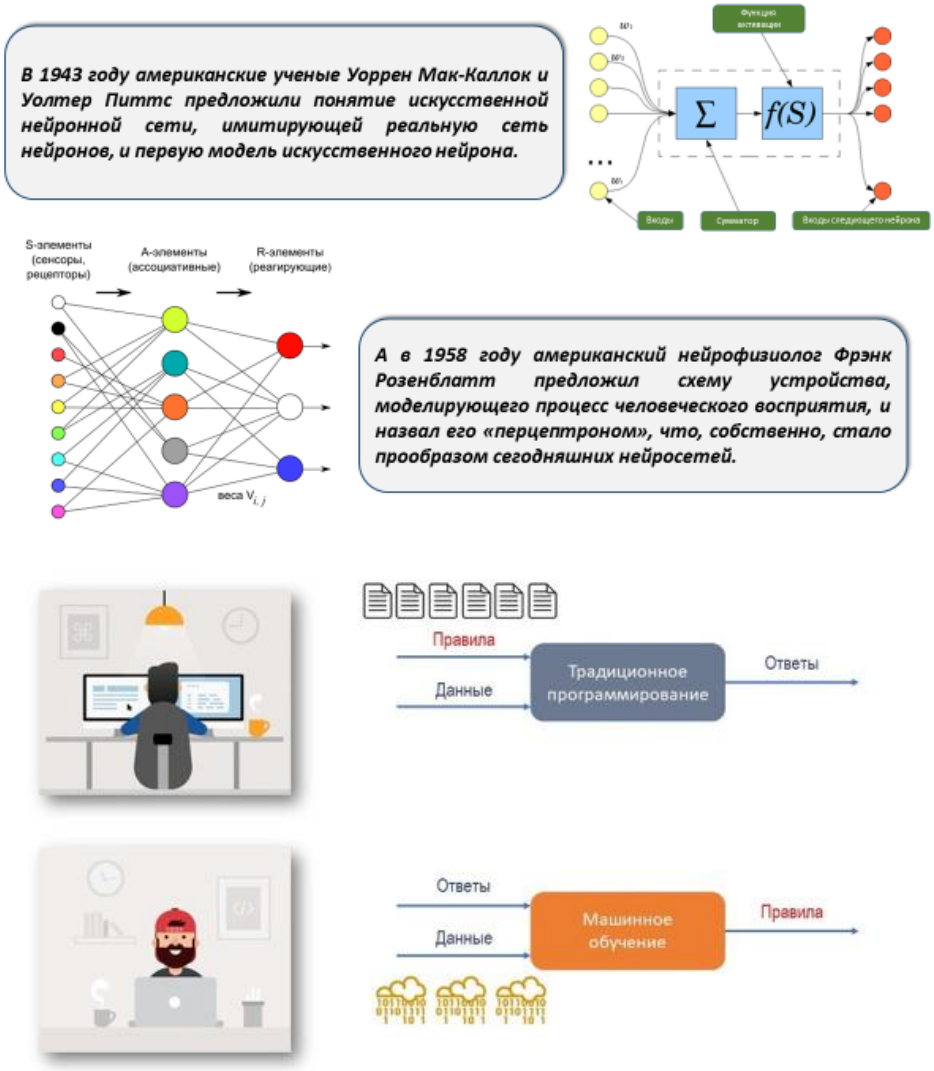
Первую программу на основе алгоритмов, способных самообучаться, разработал Артур Саму- эль (Arthur Samuel) в 1952 году, предназначена она была для игры в шашки. Самуэль дал и первое определение термину «машинное обучение»: это «область исследований разработки машин, не являющихся заранее запрограммированными». Более точное определение термину «обучение» дал намного позже Т. М. Митчелл.

Определение 1.1 (Машинное обучение по Т. М. Митчеллу). Будем считать, что компьютерная программа обучается на опыте (данных) E относительно некоторой задачи T и некоторого критерия качества работы P, если её работа по решению T, как измерено P, улучшается с опытом (увеличением данных) E.

Определение 1.2 (Задача машинного обучения, ML Problem). Пусть имеется:

- некоторая задача Task (например, улучшить продажи какого-то интернет-магазина);
- есть критерий качества работы Productivity нашей системы машинного обучения, решающей поставленную задачу Task (например, количество заходов на продающий сайт, количество продаж, суммарная прибыль за месяц и.т.п.);
- есть накапливаемый опыт (данные) Experience (сколько всего было заходов на сайт, сколько раз пользователь кликал на страницу с товаром, сколько товаров откладывал в корзину, сколько оплатил)

Тогда будем считать задачей машинного обучения - создание такой компьютерной программы, которая использует накапливаемые данные Experience для того, чтобы все лучше (с точки зрения выбранного критерия Productivity) решать некоторую практическую задачу Task.



Основное отличие между традиционным программированием и машинным обучением в том, что в машинном обучении нам не нужно строить модель самостоятельно. Эту задачу выполняют алгоритмы машинного обучения, с разве что небольшими правками, которые дата-инженер вносит в настройки алгоритма.

Другое важное отличие в количестве входных параметров, которое модель способна обработать. Для корректного прогноза погоды в той или иной локации, в теории понадобится ввести тысячи параметров, которые повлияют на результат. Человек априори не может построить алгоритм, который будет использовать все из них разумным образом. Для машинного обучения таких ограничений не существует.

Основные определения:

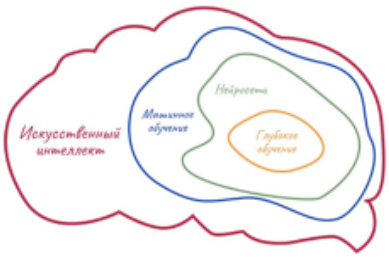
Искусственный интеллект (ИИ, AI) - это такая искусственно созданная система, которая способна имитировать интеллектуальную и творческую деятельность человека.

Data Mining (DM) – совокупность методов обнаружения в данных ранее неизвестных, нетривиальных, практически полезных и доступных интерпретации знаний, необходимых для принятия решений в различных сферах человеческой деятельности.

Машинное обучение (ML) предполагает самообучение и развитие компьютерной системы по подготовленным заранее данным.

Искусственные нейронные сети (ИНС, ANN) — вычислительные системы или машины, созданные для моделирования аналитических действий, совершаемых человеческим мозгом.

Глубокое обучение (DL) – это метод машинного обучения, который предполагает самостоятельное выстраивание (тренировку) общих правил в виде искусственной нейронной сети на примере данных во время процесса обучения. Обучение нейронной сети, особенно для систем машинного зрения, обычно проводится с учителем, т. е. на конкретных примерах данных с предварительно определенными для них результатами.



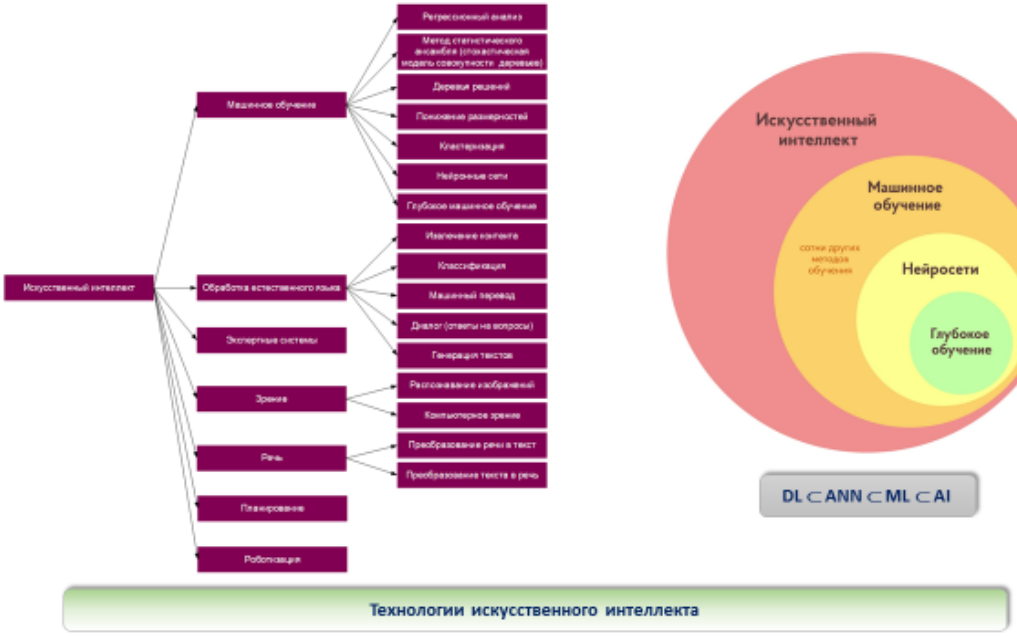
Машинное обучение — это раздел искусственного интеллекта

Искусственный интеллект (AI – artificial intelligence)

Сильный (общий) искусственный интеллект (strong AI) – компьютер\программа, способная решать любые интеллектуальные задачи не хуже человека.
Слабый (частный) искусственный интеллект (weak AI) – компьютер\программа, способная решать конкретный класс интеллектуальных задач.

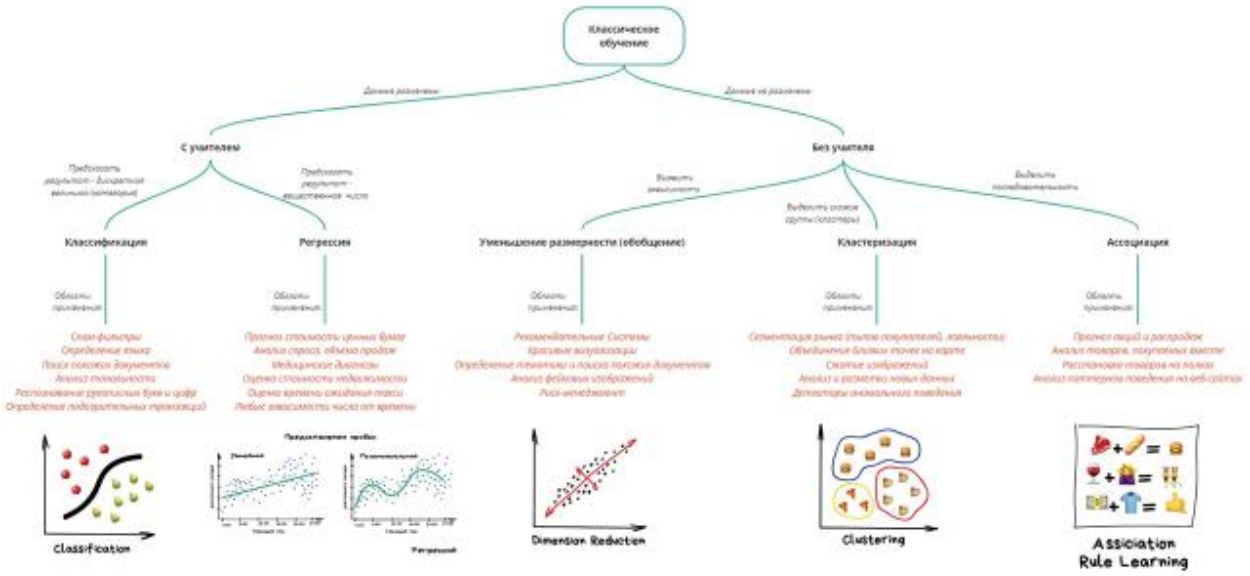
DM ≈ ML (в данном курсе)

Основные коммерческие сферы применения технологий искусственного интеллекта



Технологии искусственного интеллекта

Классификация машинного обучения



Примеры ML кейсов

Машинное обучение для создания новых вкусов

В 2017 году компания Coca-Cola запустила новый напиток со вкусом вишни – Cherry Sprite, идея для которого была определена на основе данных из автоматов самообслуживания. Эти машины предлагают покупателям самостоятельно выбирать вкусовые добавки к своим напиткам. Проанализировав клиентские предпочтения с помощью ML-алгоритмов кластеризации, Coca-Cola выбрала наиболее популярную комбинацию вкусов и превратила ее в готовый напиток для широкой аудитории.

Подобным образом, анализируя большие данные о потребностях локальных покупателей в более чем 200 странах мира, компания выявляет потребительские ожидания относительно тренда на ЗОЖ. Таким образом Кока-Кола компенсировала падение спроса на сладкие газированные напитки с помощью производства натуральных соков, продавая их под брендами Minute Maid и Simply Orange. При этом предприятие анализирует данные о погоде, спутниковые снимки, информацию об урожайности, показателях кислотности и сладости, а также ценообразовании, чтобы обеспечить оптимальное выращивание апельсиновых культур и сохранить оригинальный вкус.

Персонализация маркетинга

Чтобы лучше понять свою целевую аудиторию, таргетировать рекламные кампании и повысить их конверсию, Coca-Cola активно применяет технологии интеллектуального анализа данных (Data Mining). Так, еще в 2016 году корпорация продвигала свою марку холодного чая Gold Peak, ориентируя рекламные объявления на людей на основе анализа их фото в социальных сетях. С помощью ML-алгоритмов распознавания изображений, компания определяла фотографии людей со стаканами, банками и бутылками, в которых находился продукт Кока-Колы или ее конкурентов. Выявив таким образом любителей чая, компания таргетировала на них рекламные объявления на 40 различных мобильных сайтах, включая соцсети Instagram, Facebook и Twitter. В результате такой точечной рекламы конверсия продаж выросла более чем на 2%, что в 3-4 раза больше лучших показателей предприятий торговли в секторе Товары повседневного спроса. Дополнительным эффектом этого применения технологий Big Data стало множество детальных потребительских портретов (пол, возраст, регион, занятость, интересы, платежеспособность, семейное положение и

прочие характеристики клиента). После такого успеха собранная таким образом информация используется и в других маркетинговых задачах.

Искусственный интеллект в вендинговом автомате

Современный торговый автомат – это не просто немая витрина с продуктами, а полноценное средство общения с покупателем. Для интерактивного взаимодействия с клиентами Кока-Кола внедрила в свои вендинговые машины программные модули, которые меняют цвет сенсорного экрана в зависимости от предпочтений пользователя и окружающей обстановки. Это входит в программу персонализации продажи напитков, когда пользователь заказывает свою любимую смесь из любого торгового автомата. При этом вендинговая машина учитывает вкусы клиента при смешивании напитка, например, предлагая ему определенный ассортимент добавок. Также искусственный интеллект меняет поведение автомата в зависимости от его местоположения, добавляя больше интерактива в оживленных местах (торгово-развлекательные центры и пр.) или строгости в официальных учреждениях, таких как больницы, центры оказания госуслуг и т.д.

Тотальная цифровизация

Кока-Кола также использует Big Data и Machine Learning для оптимизации расходов, управление цепочками поставок, сканирование промо-кодов с использованием OCR-технологии для распознавания образов и множества других бизнес-задач. Для этого в компании построено современное корпоративное хранилище данных, которое аккумулирует всю многоканальную информацию о розничной торговле, чтобы быстро и точно реагировать на изменения рынка. Еще предприятие поддерживает процессы управления основными данными, чтобы стандартизовать все процедуры Data Governence для эффективного производства и продвижения продукции, а также для повышения качества обслуживания потребителей.

Предиктивная аналитика.

Сейчас на производстве часто внедряют IoT-системы: устанавливают датчики на оборудовании и в помещениях, а потом анализируют собранные ими данные. Эти данные и есть big data, их можно использовать для мониторинга состояния оборудования, моделирования производственных процессов, выявления и предотвращения сбоев.

Например, у «Газпром нефти» сбоил автоматический перезапуск насосов после аварийного отключения электричества. Разобраться в проблеме помогли большие данные. Аналитики собрали 200 миллионов записей с контроллеров систем управления, проанализировали их, смоделировали события и выявили неожиданные причинно-следственные связи. В итоге сбои удалось прекратить.

Снижение стоимости продукции и оптимизация производства.

Если собрать много данных о работе станков, проценте брака и каждом этапе производства, а потом их проанализировать, можно понять:

- при каких условиях чаще всего происходит брак;
- на какие этапы производства тратится больше всего времени и почему;
- какие тесты продукции малополезны и не дают новой информации;
- как можно оптимизировать и ускорить работу на отдельных этапах;
- как использовать меньше расходных материалов.

Все это помогает уменьшить издержки и снизить стоимость производства, а значит, повысить прибыль.

Например, компания Intel производит процессоры. Перед поставкой в магазин каждый процессор должен пройти примерно 19 000 тестов — это долго и дорого. Анализ данных всего производственного процесса помог понять, какие тесты избыточные. В итоге на них удалось сэкономить около 30 миллионов долларов.

ThyssenKrupp AG – один из ведущих мировых производителей лифтов, обслуживает более 1,1 млн. лифтов по всему миру. В партнерстве с Microsoft компания запустила систему MAX, которая через Интернет вещей собирает данные от множества датчиков, установленных в лифтах компании (отслеживают скорость кабины, функционирование дверей, температуру мотора и др.) и по ним строит предиктивные модели на платформе Azure Machine Learning. Модели позволяют предупредить инцидент до его возникновения и передать технику конкретный код поломки, один из 400 возможных, чтобы сократить время обслуживания. В результате сокращаются затраты на обслуживание и ремонт (одна поломка обходится минимум в 300$) и создается дополнительная ценность для клиентов: лифты становятся более надежными, безопасными, владельцы расположенных в зданиях магазинов, гостиниц и других организаций не несут убытки. Результат: время бесперебойной работы лифтов выросло в среднем на 50%.

Поиск новых месторождений.

При добыче природных ресурсов месторождения часто приходится искать почти вслепую. Однако с помощью анализа больших данных можно обнаруживать закономерности, изучать состояние почв, наличие подземных пустот, температуру пород — и таким образом эффективно искать перспективные месторождения, сравнивая новые участки с уже известными аналогами.

Так, ООО НПЦ «Геостра» занимается обработкой и интерпретацией больших объемов данных, полученных в ходе поиска нефтяных месторождений. В качестве пилотного проекта на облачную платформу Mail.ru Cloud Solutions перенесли геофизическое ПО для обработки высокоплотных геофизических данных. Проект оказался успешным: облачные вычислительные мощности справились с поставленной задачей.

Задание на самостоятельную работу

1. (Постановка задачи ML). Сформулируйте постановку задачи ML для концепции "Умный Дом". Постановка задачи должна содержать описание:
   – цели/задачи Task (например, улучшить продажи какого-то интернет-магазина);
   – критерия качества решения задачи Productivity (например, количество заходов на продающий сайт, количество продаж, суммарная прибыль за месяц и.т.п.);
   – используемых для решения задачи данных Experience, их происхождение, откуда они берутся и как размечаются.

   Согласуйте с преподавателем итоговую формултровку.
2. Ответьте на следующие вопросы:
   1. Дайте определение нейронной сети.
   2. Расскажите о нейросетевой парадигме.
   3. Опишите формальную модель искусственного нейрона.
   4. Определите известные Вам виды функций активации.
   5. Расскажите об общем построении нейронных сетей.
   6. Расскажите о применениях нейронных сетей.

   Задание для самостоятельной работы:
   - *Проработать лекционный материал, основную и дополнительную литературу с целью подготовки ответов на представленный перечень вопросов для устного опроса.*

Практическое занятие №2. Знакомство со средой решения задач. Jupyter notebook. Google Colab. Использование готовых решений.

Вопросы для рассмотрения на семинарских занятиях. *Изучение нижеперечисленных вопросов будет производиться на примере конкретных ситуаций, с целью формирования и развития профессиональных навыков обучающихся.*

Форма проведения занятия – практическое занятие.

Вопросы для самоконтроля и текущего контроля

1. Знакомство со средой решения задач.
2. Как работает Jupyter notebook.
3. Особенности Google Colab.
4. Использование готовых решений.

Работа с IPython и Jupyter Notebook

IPython представляет собой мощный инструмент для работы с языком Python. Базовые компоненты IPython – это интерактивная оболочка для с широким набором возможностей и ядро для Jupyter. Jupyter notebook является графической веб-оболочкой для IPython, которая расширяет идею консольного подхода к интерактивным вычислениям.

Основные отличительные особенности данной платформы – это комплексная интроспекция объектов, сохранение истории ввода на протяжении всех сеансов, кэширование выходных результатов, расширяемая система “магических” команд, логирование сессии, дополнительный командный синтаксис, подсветка кода, доступ к системной оболочке, стыковка с pdb отладчиком и Python профайлером.

IPython позволяет подключаться множеству клиентов к одному вычислительному ядру и, благодаря своей архитектуре, может работать в параллельном кластере.

В Jupyter notebook вы можете разрабатывать, документировать и выполнять приложения на языке Python, он состоит из двух компонентов: веб-приложение, запускаемое в браузере, и ноутбуки – файлы, в которых можно работать с исходным кодом программы, запускать его, вводить и выводить данные и т.п.

Веб приложение позволяет:

- редактировать Python код в браузере, с подсветкой синтаксиса, автоотступами и автодополнением;
- запускать код в браузере;
- отображать результаты вычислений с медиа представлением (схемы, графики);
- работать с языком разметки Markdown и LaTeX.

Ноутбуки – это файлы, в которых сохраняются исходный код, входные и выходные данные, полученные в рамках сессии. Фактически, он является записью вашей работы, но при этом позволяет заново выполнить код, присутствующий на нем. Ноутбуки можно экспортировать в форматы PDF, HTML.

Установка и запуск

Для запуска Jupyter Notebook перейдите в папку Scripts (она находится внутри каталога, в котором установлена Anaconda) и в командной строке наберите:

```
> ipython notebook
```

В результате будет запущена оболочка в браузере.

Примеры работы

Будем следовать правилу: лучше один раз увидеть… Рассмотрим несколько примеров, выполнив которые, вы сразу поймете принцип работы с Jupyter notebook.

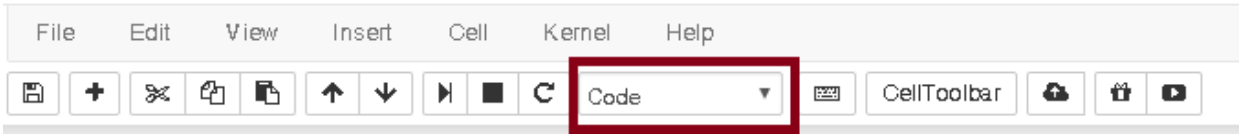
Запустите Jupyter notebook и создайте папку для наших примеров, для этого нажмите на New в правой части экрана и выберите в выпадающем списке Folder.

По умолчанию папке присваивается имя “Untitled folder”, переименуем ее в “notebooks”: поставьте галочку напротив имени папки и нажмите на кнопку “Rename”.

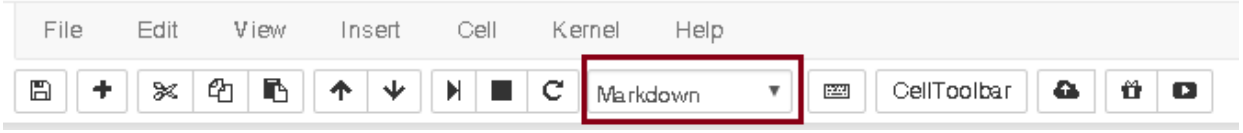
Зайдите в эту папку и создайте в ней ноутбук, воспользовавшись той же кнопкой New, только на этот раз нужно выбрать “Python [Root]”.

В результате будет создан ноутбук.
Код на языке Python или текст в нотации Markdown нужно вводить в ячейки:

Если это код Python, то на панели инструментов нужно выставить свойство “Code”.

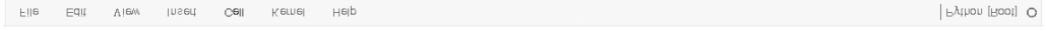


Если это Markdown текст – выставить “Markdown”.



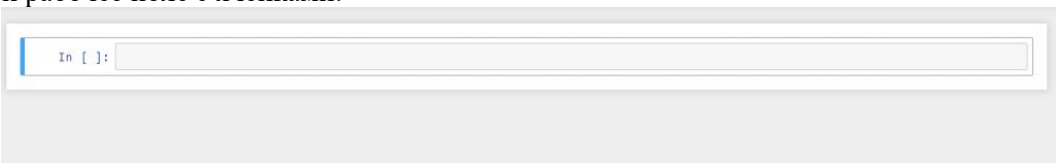
Основные элементы интерфейса Jupyter notebook
У каждого ноутбука есть имя, оно отображается в верхней части экрана. Для изменения имени нажмите на его текущее имя и введите новое.

Из элементов интерфейса можно выделить, панель меню:

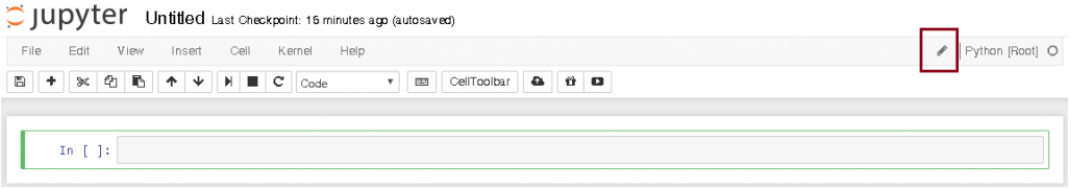


панель инструментов:

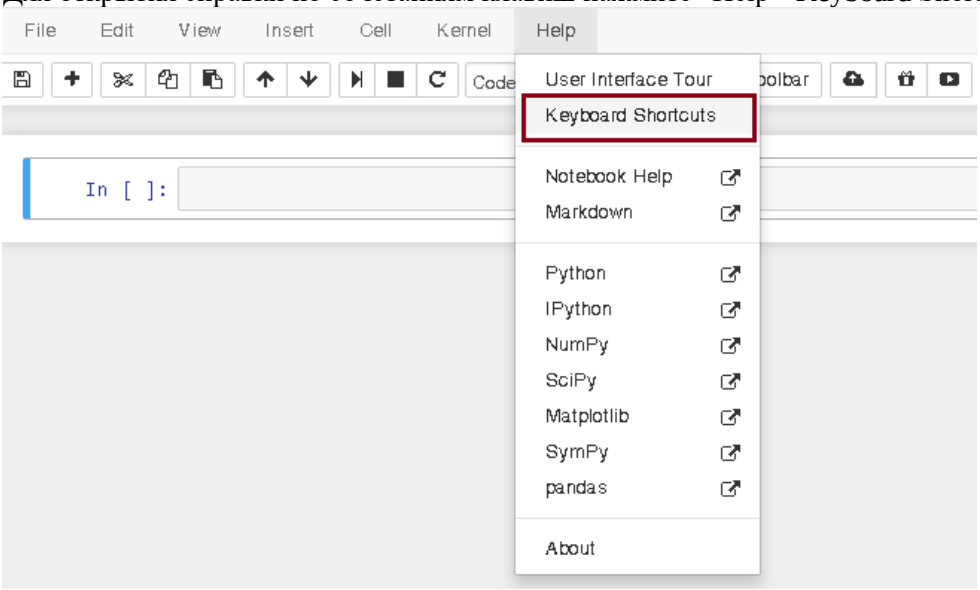
и рабочее поле с ячейками:



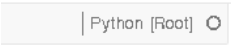
Ноутбук может находиться в одном из двух режимов – это режим правки (Edit mode) и командный режим (Command mode). Текущий режим отображается на панели меню в правой части, в режиме правки появляется изображение карандаша, отсутствие этой иконки значит, что ноутбук находится в командном режиме.



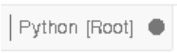
Для открытия справки по сочетаниям клавиш нажмите “Help->Keyboard Shortcuts”



В самой правой части панели меню находится индикатор загруженности ядра Python. Если ядро находится в режиме ожидания, то индикатор представляет собой окружность.



Если оно выполняет какую-то задачу, то изображение измениться на закрашенный круг.



Запуск и прерывание выполнения кода

Если ваша программа зависла, то можно прервать ее выполнение выбрав на панели меню пункт Kernel -> Interrupt.

Для добавления новой ячейки используйте Insert->Insert Cell Above и Insert->Insert Cell Below.

Для запуска ячейки используете команды из меню Cell, либо следующие сочетания клавиш:

Ctrl+Enter – выполнить содержимое ячейки.

Shift+Enter – выполнить содержимое ячейки и перейти на ячейку ниже.

Alt+Enter – выполнить содержимое ячейки и вставить новую ячейку ниже.

Как сделать ноутбук доступным для других людей?

Существует несколько способов поделиться своим ноутбуком с другими людьми, причем так, чтобы им было удобно с ним работать:

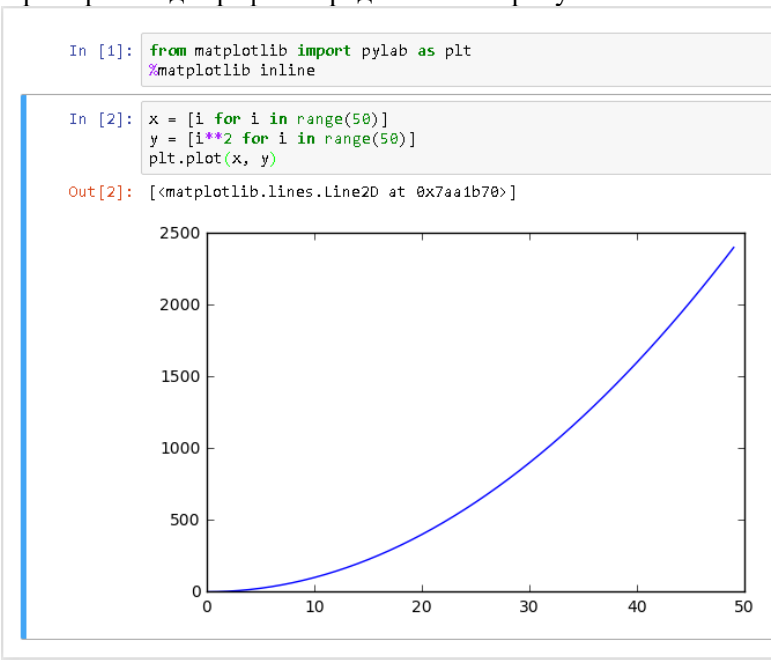
- передать непосредственно файл ноутбука, имеющий расширение “.ipynb”, при этом открыть его можно только с помощью Jupyter Notebook;
- сконвертировать ноутбук в html;
- использовать https://gist.github.com/;
- использовать http://nbviewer.jupyter.org/.

Вывод изображений в ноутбуке

Печать изображений может пригодиться в том случае, если вы используете библиотеку matplotlib для построения графиков. По умолчанию, графики не выводятся в рабочее поле ноутбука. Для того, чтобы графики отображались, необходимо ввести и выполнить следующую команду:

%matplotlib inline

Пример вывода графика представлен на рисунке ниже.



Задание на самостоятельную работу

1. Используя Google Colaboratory напишите код в jupyter notebook, решающий следующие задачи:

Задача 1

С помощью функции input считайте с входящей строки целое положительное число и сохраните его в переменную N. Используя конструкцию if...else, выведите на экран результат в зависимости от условий:

- *если N нечетное, то выведите на экран "N нечетное",*
- *если N четное и входит в интервал от 5 до 10 включительно, выведите на экран "N четное и принадлежит интервалу [5, 10]",*
- *если N четное и больше 10, выведите на экран "N четное и N > 10",*
- *если N четное и меньше 5, выведите на экран "N четное и N < 5".*

Задача 2

Используя цикл while, посчитайте и выведите на экран сумму квадратов всех целых чисел от 15 до 22 включительно.

Задача 3

Используя цикл for, найдите сумму всех элементов заданного списка (без использования встроенных функций sum, len, sort и т. д.).

```
my_list = [56, 23, 67, 45, 67, 2, 47, 158, 31, 34, 78, 23, 78, 23, 89, 23, 36]
```

Задача 4

Создайте словарь, соответствующий следующему описанию: в честь 8 Марта начальник отдела Валера решил принести на работу коробку конфет и угостить коллег :) Секретарша Наташа съела две конфеты, ее подруга Алина - целых три, разработчик Марат унес с собой в соседний опен-спейс целых пятнадцать, чтобы поделиться со своей командой, менеджер проекта Лев проходил мимо и съел одну конфету, а сам Валера, будучи сторонником здорового образа жизни, не съел ни одной.

2. Ответьте на следующие вопросы:
   1. Какие блокноты размещены в Google Colab?
   2. Какие типы ячеек доступны в Google Colab?
   3. Как вы подключаете свой Google Диск к Colab?
   4. Какой язык используется в ячейках кода Google Colab?

Задание для самостоятельной работы:

- *Проработать лекционный материал, основную и дополнительную литературу с целью подготовки ответов на представленный перечень вопросов для устного опроса.*

Практическое занятие №3. Изучаем базовые конструкции языка Python. Работа с коллекциями данных.

Изучение нижеперечисленных вопросов будет производиться на примере конкретных ситуаций, с целью формирования и развития профессиональных навыков обучающихся.
Форма проведения занятия – практическое занятие.

Вопросы для самоконтроля и текущего контроля

1. Типы и модель данных
2. Арифметические операции
3. Условные операторы и циклы
4. Работа со списками (list)
5. Кортежи (tuple)
6. Словари (dict)
7. Функции в Python

Условный оператор ветвления if

Оператор ветвления if позволяет выполнить определенный набор инструкций в зависимости от некоторого условия. Возможны следующие варианты использования.

1. Конструкция if

Синтаксис оператора if выглядит так.

```
if выражение:
    инструкция_1
    инструкция_2
    ...
    инструкция_n
```

После оператора if записывается выражение. Если это выражение истинно, то выполняются инструкции, определяемые данным оператором. Выражение является истинным, если его результатом

является число не равное нулю, непустой объект, либо логическое True. После выражения нужно поставить двоеточие “:”.

ВАЖНО: блок кода, который необходимо выполнить, в случае истинности выражения, отделяется четырьмя пробелами слева!

Примеры:

```
if 1:
  print("hello 1")
```

Напечатает: hello 1

```
a = 3
if a == 3:
   print("hello 2")
```

Напечатает: hello 2

```
a = 3
if a > 1:
  print("hello 3")
```

Напечатает: hello 3

```
lst = [1, 2, 3]
if lst :
  print("hello 4")
```

Напечатает: hello 4

2. Конструкция if – else

Бывают случаи, когда необходимо предусмотреть альтернативный вариант выполнения программы. Т.е. при истинном условии нужно выполнить один набор инструкций, при ложном – другой. Для этого используется конструкция if – else.

```
if выражение:
   инструкция_1
   инструкция_2
   ...
   инструкция_n
else:
   инструкция_a
   инструкция_b
   ...
   инструкция_x
```

Примеры.

```
a = 3
if a > 2:
  print("H")
else:
```

```
  print("L")
```
Напечатает: H

```
a = 1
if a > 2:
  print("H")
else:
  print("L")
```
Напечатает: L

Условие такого вида можно записать в строчку, в таком случае оно будет представлять собой тернарное выражение.

```
a = 17
b = True if a > 10 else False
print(b)
```
В результате выполнения такого кода будет напечатано: True

3. Конструкция if – elif – else
Для реализации выбора из нескольких альтернатив можно использовать конструкцию if – elif – else.

```
if выражение_1:
   инструкции_(блок_1)
elif выражение_2:
   инструкции_(блок_2)
elif выражение_3:
   инструкции_(блок_3)
else:
   инструкции_(блок_4)
```
Пример.

```
a = int(input("введите число:"))
if a < 0:
  print("Neg")
elif a == 0:
  print("Zero")
else:
  print("Pos")
```
Если пользователь введет число меньше нуля, то будет напечатано “Neg“, равное нулю – “Zero“, большее нуля – “Pos“.

Оператор цикла while
Оператор цикла while выполняет указанный набор инструкций до тех пор, пока условие цикла истинно. Истинность условия определяется также как и в операторе if. Синтаксис оператора while выглядит так.

```
while выражение:
```

```
  инструкция_1
  инструкция_2
  ...
  инструкция_n
```

Выполняемый набор инструкций называется телом цикла.

Пример.

```
a = 0
while a < 7:
  print("A")
  a += 1
```

Буква “A” будет выведена семь раз в столбик.

Пример бесконечного цикла.

```
a = 0
while a == 0:
  print("A")
```

Операторы break и continue

При работе с циклами используются операторы break и continue.

Оператор break предназначен для досрочного прерывания работы цикла while.

Пример.

```
a = 0
while a >= 0:
  if a == 7:
     break
  a += 1
  print("A")
```

В приведенном выше коде, выход из цикла произойдет при достижении переменной a значения 7. Если бы не было этого условия, то цикл выполнялся бы бесконечно.

Оператор continue запускает цикл заново, при этом код, расположенный после данного оператора, не выполняется.

Пример.

```
a = -1
while a < 10:
  a += 1
  if a >= 7:
     continue
  print("A")
```

При запуске данного кода символ “A” будет напечатан 7 раз, несмотря на то, что всего будет выполнено 11 проходов цикла.

Оператор цикла for

Оператор for выполняет указанный набор инструкций заданное количество раз, которое определяется количеством элементов в наборе.

Пример.

```
for i in range(5):
  print("Hello")
```

В результате “Hello” будет выведено пять раз.

Внутри тела цикла можно использовать операторы break и continue, принцип работы их точно такой же как и в операторе while.

Если у вас есть заданный список, и вы хотите выполнить над каждым элементом определенную операцию (возвести в квадрат и напечатать получившееся число), то с помощью for такая задача решается так.

```
lst = [1, 3, 5, 7, 9]
for i in lst:
  print(i ** 2)
```

Также можно пройти по всем буквам в строке.

```
word_str = "Hello, world!"
for l in word_str:
  print(l)
```

Строка “Hello, world!” будет напечатана в столбик.

Задание на самостоятельную работу

1. Заданы нечеткие истинностные значения: «истинно» = (0/0; 0/0,1; 0,25/0,3; 0,6/0,5; 0,7/0,8; 1/1); «более-менее истинно» = (0/0; 0,04/0,2; 0,6/0,5; 0,7/0,6; 0,95/0,8; 1/1); «почти истинно» = (0/0; 0,04/0,2; 0,47/0,4; 0,65/0,7; 0,95/0,8; 0,85/1). Найти нечеткую истинность выражения: «более-менее истинно» И «истинно». Сравнить результат с нечетким истинностным значением «почти истинно».

2. Решите несколько простых задач, используя ЯП Python 3:

Обратите внимание, что у любой задачи по программированию может быть несколько способов решения.

Задача 1

Есть список a = [1, 1, 2, 3, 5, 8, 13, 21, 34, 55, 89]. Выведите все элементы, которые меньше 5.

Пример решения:

```
for elem in a:
    if elem < 5:
        print(elem)
```

Также можно воспользоваться функцией filter, которая фильтрует элементы согласно заданному условию:

print(list(filter(lambda elem: elem < 5, a)))

И, вероятно, наиболее предпочтительный вариант решения этой задачи — списковое включение:

print([elem for elem in a if elem < 5])

Задача 2

Даны списки:

a = [1, 1, 2, 3, 5, 8, 13, 21, 34, 55, 89];

b = [1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13].

Нужно вернуть список, который состоит из элементов, общих для этих двух списков.

Пример решения:

Можем воспользоваться функцией filter:

result = list(filter(lambda elem: elem in b, a))

Или списковым включением:

result = [elem for elem in a if elem in b]

А можно привести оба списка к множествам и найти их пересечение:

result = list(set(a) & set(b))

Однако в таком случае каждый элемент встретится в результирующем списке лишь один раз, т.к. множество поддерживает уникальность входящих в него элементов. Первые два решения (с фильтрацией) оставят все дубли на своих местах.

Задача 3

Отсортируйте словарь по значению в порядке возрастания и убывания.

Задача 4

Напишите программу для слияния нескольких словарей в один.

Допустим, наши словари:

dict_a = {1:10, 2:20}

dict_b = {3:30, 4:40}

dict_c = {5:50, 6:60}

Задача 5

Найдите три ключа с самыми высокими значениями в словаре my_dict = {'a':500, 'b':5874, 'c': 560,'d':400, 'e':5874, 'f': 20}.

Задача 6

Напишите код, который переводит целое число в строку, при том что его можно применить в любой системе счисления.

Задача 7

Нужно вывести первые n строк треугольника Паскаля. В этом треугольнике на вершине и по бокам стоят единицы, а каждое число внутри равно сумме двух расположенных над ним чисел.

Задача 8

Напишите проверку на то, является ли строка палиндромом. Палиндром — это слово или фраза, которые одинаково читаются слева направо и справа налево.

Задача 9

Сделайте так, чтобы число секунд отображалось в виде дни:часы:минуты:секунды.

Задача 10

Вы принимаете от пользователя последовательность чисел, разделённых запятой. Составьте список и кортеж с этими числами.

Пример решения:

```
values = input('Введите числа через запятую: ')
ints_as_strings = values.split(',')
ints = map(int, ints_as_strings)
lst = list(ints)
tup = tuple(lst)
print('Список:', lst)
print('Кортеж:', tup)
```

Задача 11

Выведите первый и последний элемент списка.

Задача 12

Напишите программу, которая принимает имя файла и выводит его расширение. Если расширение у файла определить невозможно, сформируйте исключение.

Задача 13

При заданном целом числе n посчитайте n + nn + nnn.

Задача 14

Напишите программу, которая выводит чётные числа из заданного списка и останавливается, если встречает число 237.

Пример решения:

```
numbers = [
   386, 462, 47, 418, 907, 344, 236, 375, 823, 566, 597, 978, 328, 615, 953, 345,
   399, 162, 758, 219, 918, 237, 412, 566, 826, 248, 866, 950, 626, 949, 687, 217,
]
for x in numbers:
   if x == 237:
      break
   elif x % 2 == 0:
      print(x)
```

Задача 15

Напишите программу, которая принимает два списка и выводит все элементы первого, которых нет во втором.

Задача 16

Выведите список файлов в указанной директории.

Задача 17

Сложите цифры целого числа.

Пример решения:

```
def sum_digits(num):
   digits = [int(d) for d in str(num)]
```

return sum(digits)

print(sum_digits(5245))

Задача 18

Посчитайте, сколько раз символ встречается в строке.

Задача 19

Поменяйте значения переменных местами.

Пример решения:

В Python есть более удобный способ для решения этой задачи:

x = 5

y = 10

x, y = y, x

Задача 20

С помощью анонимной функции извлеките из списка числа, делимые на 15.

Задача 21

Нужно проверить, все ли числа в последовательности уникальны.

Задача 22

Напишите программу, которая принимает текст и выводит два слова: наиболее часто встречающееся и самое длинное.

3. Ответьте на следующие вопросы:

1. Как влияет использование в Python assert(True) на операторы, следующие за ним?
2. Как бы вы использовали Python для вызова исключения LastParamError?
3. Как бы вы использовали Python для вывода строки Hello?
4. Как с помощью Python возвести число 2 в степень 3?

Задание для самостоятельной работы:

- *Проработать лекционный материал, основную и дополнительную литературу с целью подготовки ответов на представленный перечень вопросов для устного опроса.*

Практическое занятие №4. Чтение данных из различных источников, в том числе из репозиториев данных. Предварительный анализ данных. Библиотеки pandas и matplotlib.

Изучение нижеперечисленных вопросов будет производиться на примере конкретных ситуаций, с целью формирования и развития профессиональных навыков обучающихся.
Форма проведения занятия – практическое занятие.

Вопросы для самоконтроля и текущего контроля

1. Ввод-вывод данных. Работа с файлами
2. Введение в pandas и его установка
3. Структуры данных Series и DataFrame
4. Доступ к данным в структурах pandas
5. Работа с пропусками в данных
6. Работа со структурами данных в pandas: удаление, объединение, расширение, группировка
7. Работа с внешними источниками данных

8. Операции над данными
9. Настройка pandas
10. Инструменты для работы с данными
11. Временные ряды
12. Визуализация данных

Вывод данных в консоль

Один из самых распространенных способов вывести данные в *Python* – это напечатать их в консоли. Если вы находитесь на этапе изучения языка, такой способ является основным для того, чтобы быстро просмотреть результат свой работы. Для вывода данных в консоль используется функция *print*.

Рассмотрим основные способы использования данной функции.

```
>>> print("Hello")
Hello
>>> print("Hello, " + "world!")
Hello, world!
>>> print("Age: " + str(23))
Age: 23
```

По умолчанию, для разделения элементов в функции *print* используется пробел.

```
>>> print("A", "B", "C")
A B C
```

Для замены разделителя необходимо использовать параметр *sep* функции *print*.

```
print("A", "B", "C", sep="#")
A#B#C
```

В качестве конечного элемента выводимой строки, используется символ перевода строки.

```
>>> for i in range(3):
    print("i: " + str(i))
i: 0
i: 1
i: 2
```

Для его замены используется параметр *end*.

```
>>> for i in range(3):
    print("[i: " + str(i) + "]", end=" -- ")
[i: 0] -- [i: 1] -- [i: 2] --
```

Ввод данных с клавиатуры

Для считывания вводимых с клавиатуры данных используется функция *input()*.

```
>>> input()
test
'test'
```

Для сохранения данных в переменной используется следующий синтаксис.

```
>>> a = input()
hello
>>> print(a)
hello
```

Если считывается с клавиатуры целое число, то строку, получаемую с помощью функции *input()*, можно передать сразу в функцию *int()*.

```
>>> val = int(input())
123
```

```
>>> print(val)
123
>>> type(val)
<class 'int'>
```

Для вывода строки-приглашения, используйте ее в качестве аргумента функции *input()*.

```
>>> tv = int(input("input number: "))
input number: 334
>>> print(tv)
334
```

Преобразование строки в список осуществляется с помощью метода *split()*, по умолчанию, в качестве разделителя, используется пробел.

```
>>> l = input().split()
1 2 3 4 5 6 7
>>> print(l)
['1', '2', '3', '4', '5', '6', '7']
```

Разделитель можно заменить, указав его в качестве аргумента метода *split()*.

```
>>> nl = input().split("-")
1-2-3-4-5-6-7
>>> print(nl)
['1', '2', '3', '4', '5', '6', '7']
```

Для считывания списка чисел с одновременным приведением их к типу *int* можно воспользоваться вот такой конструкцией.

```
>>> nums = map(int, input().split())
1 2 3 4 5 6 7
>>> print(list(nums))
[1, 2, 3, 4, 5, 6, 7]
```

Работа с файлами

Открытие и закрытие файла

Для открытия файла используется функция *open()*, которая возвращает файловый объект. Наиболее часто используемый вид данной функции выглядит так *open(имя_файла, режим_доступа)*.

Для указания режима доступа используется следующие символы:

‘*r*’ – открыть файл для чтения;

‘*w*’ – открыть файл для записи;

‘*x*’ – открыть файл с целью создания, если файл существует, то вызов функции *open* завершится с ошибкой;

‘*a*’ – открыть файл для записи, при этом новые данные будут добавлены в конец файла, без удаления существующих;

‘*b*’ – бинарный режим;

‘*t*’ – текстовый режим;

‘+’ – открывает файл для обновления.

По умолчанию файл открывается на чтение в текстовом режиме.

У файлового объекта есть следующие атрибуты.

file.closed – возвращает *true* если файл закрыт и *false* в противном случае;

file.mode – возвращает режим доступа к файлу, при этом файл должен быть открыт;

file.name – имя файла.

```
>>> f = open("test.txt", "r")
>>> print("file.closed: " + str(f.closed))
file.closed: False
>>> print("file.mode: " + f.mode)
file.mode: r
```

```
>>> print("file.name: " + f.name)
file.name: test.txt
```

Для закрытия файла используется метод *close()*.

Чтение данных из файла

Чтение данных из файла осуществляется с помощью методов *read(размер)* и *readline()*.

Метод *read(размер)* считывает из файла определенное количество символов, переданное в качестве аргумента. Если использовать этот метод без аргументов, то будет считан весь файл.

```
>>> f = open("test.txt", "r")
>>> f.read()
'1 2 3 4 5\nWork with file\n'
>>> f.close()
```

В качестве аргумента метода можно передать количество символом, которое нужно считать.

```
>>> f = open("test.txt", "r")
>>> f.read(5)
'1 2 3'
>>> f.close()
```

Метод *readline()* позволяет считать строку из открытого файла.

```
>>> f = open("test.txt", "r")
>>> f.readline()
'1 2 3 4 5\n'
>>> f.close()
```

Построчное считывание можно организовать с помощью оператора *for*.

```
>>> f = open("test.txt", "r")
>>> for line in f:
...     print(line)
...
1 2 3 4 5
Work with file

>>> f.close()
```

Запись данных в файл

Для записи данных файл используется метод *write(строка)*, при успешной записи он вернет количество записанных символов.

```
>>> f = open("test.txt", "a")
>>> f.write("Test string")
11
>>> f.close()
```

Дополнительные методы для работы с файлами

Метод *tell()* возвращает текущую позицию “условного курсора” в файле. Например, если вы считали пять символов, то “курсор” будет установлен в позицию 5.

```
>>> f = open("test.txt", "r")
>>> f.read(5)
'1 2 3'
>>> f.tell()
5
>>> f.close()
```

Метод *seek(позиция)* выставляет позицию в файле.

```
>>> f = open("test.txt", "r")
>>> f.tell()
```

```
0
>>> f.seek(8)
8
>>> f.read(1)
'5'
>>> f.tell()
9
>>> f.close()
```

Хорошей практикой при работе с файлами является применение оператора *with*. При его использовании нет необходимости закрывать файл, при завершении работы с ним, эта операция будет выполнена автоматически.

```
>>> with open("test.txt", "r") as f:
...     for line in f:
...         print(line)
...
1 2 3 4 5
Work with file
Test string
>>> f.closed
True
```

Задания на самостоятельную работу

1. Решите несколько простых задач, используя высокоуровневую Python библиотеку Pandas:

Задача 1.

Прочитать файл CSV и перевести его в DataFrame

Пример решения:

df = pd.read_csv('https://raw.githubusercontent.com/Grossmend/CSV/master/titanic/data.csv')

Задача 2.

Прочитать файл CSV (определенные столбцы и определенное кол-во строк) и перевести его в DataFrame

Задача 3.

Получить данные из DataFrame по условию

Задача 4.

Изменить данные столбца DataFrame по условию

Задача 5.

Проверить имеет ли DataFrame пропущенные значения

2. Ответьте на следующие вопросы:
    1. Для чего используются библиотека pandas?
    2. Как выполняется индексация и извлечение данных в Pandas?
    3. Как выполняется группировка данных в Pandas?
    4. Для чего применяются таблицы сопряженности в Pandas?
    5. Как выполняется загрузка данных в DataFramев в библиотеке Pandas?
    6. Как выполняется доступ к столбцам DataFramев в библиотеке Pandas?

Задание для самостоятельной работы:

- *Проработать лекционный материал, основную и дополнительную литературу с целью подготовки ответов на представленный перечень вопросов для устного опроса.*

Модуль 2. РЕШЕНИЕ ЗАДАЧ МАШИННОГО ОБУЧЕНИЯ

Практическое занятие №5. Решение задач регрессионного анализа с использованием библиотеки scikit-learn

Изучение нижеперечисленных вопросов будет производиться на примере конкретных ситуаций, с целью формирования и развития профессиональных навыков обучающихся. Форма проведения занятия – практическое занятие.

Вопросы для самоконтроля и текущего контроля

1. Постановка задач линейной регрессии и линейной классификации.
2. Метод наименьших квадратов в матричной форме.
3. Аналитическое решение.
4. Регуляризация в задаче регрессии.
5. Мультиколлинеарность и плохая обусловленность ковариационной матрицы.
6. Гребневая регрессия.
7. Логистическая регрессия.

Что такое регрессия?

Регрессия ищет отношения между переменными.

Для примера можно взять сотрудников какой-нибудь компании и понять, как значение зарплаты зависит от других **данных**, таких как опыт работы, уровень образования, роль, город, в котором они работают, и так далее.

Регрессия решает проблему единого **представления** данных анализа для каждого работника.

Причём опыт, образование, роль и город – это независимые переменные при зависимой от них зарплате.

Таким же способом можно установить математическую зависимость между ценами домов в определённой области, количеством комнат, расстоянием от центра и т. д.

Регрессия рассматривает некоторое явление и ряд наблюдений. Каждое наблюдение имеет две и более переменных. Предполагая, что одна переменная зависит от других, вы пытаетесь построить отношения между ними.

Другими словами, **вам нужно найти функцию, которая отображает зависимость одних переменных или данных от других**.

Зависимые данные называются **зависимыми переменными**, **выходами** или **ответами**.

Независимые данные называются **независимыми переменными, входами или предсказателями**.

Обычно в регрессии присутствует одна непрерывная и неограниченная зависимая переменная.

Входные переменные могут быть неограниченными, дискретными или категорическими данными, такими как пол, национальность, бренд, etc.

Общей практикой является обозначение данных на выходе – y, входных данных – x. В случае с двумя или более независимыми переменными, их можно представить в виде вектора $x = (x_1, \ldots, x_r)$, где r – количество входных переменных.

Когда вам нужна регрессия?

Регрессия полезна **для прогнозирования ответа** на новые условия. Можно угадать потребление электроэнергии в жилом доме из данных температуры, времени суток и количества жильцов.

Где она вообще нужна?

Регрессия используется во многих отраслях: экономика, компьютерные и социальные науки, прочее. Её важность растёт с доступностью больших данных.

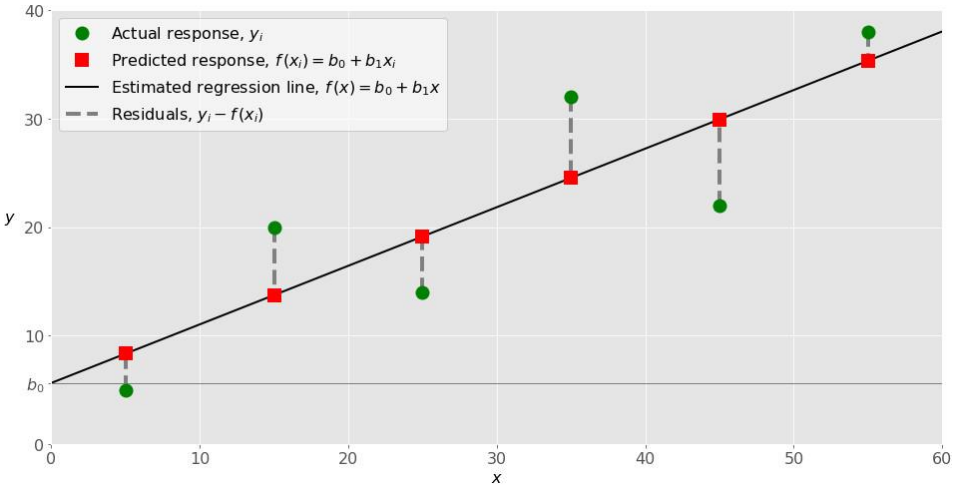
Линейная регрессия

Линейная регрессия – одна из важнейших и широко используемых техник регрессии. Эта самый простой метод регрессии. Одним из его достоинств является лёгкость интерпретации результатов.

Постановка проблемы

Линейная регрессия некоторой зависимой переменной y на набор независимых переменных $x = (x_1, \ldots, x_r)$, где r – это число предсказателей, предполагает, что линейное отношение между y и x: $y = \beta_0 + \beta_1 x_1 + \cdots + \beta_r x_r + \varepsilon$.

Это **уравнение регрессии**. β_0, β_1, …, β_r – **коэффициенты регрессии**, и ε – **случайная ошибка**.
Линейная регрессия вычисляет **оценочные функции** коэффициентов регрессии или просто **прогнозируемые весы измерения**, обозначаемые как b_0, b_1, …, b_r. Они определяют **оценочную функцию регрессии** $f(x) = b_0 + b_1x_1 + \cdots + b_rx_r$. Эта функция захватывает зависимости между входами и выходом достаточно хорошо.
Для каждого результата наблюдения i = 1, …, n, **оценочный** или **предсказанный ответ** $f(x_i)$ должен быть как можно ближе к соответствующему **фактическому ответу** y_i. Разницы $y_i - f(x_i)$ для всех результатов наблюдений называются **остатками**. Регрессия определяет лучшие **прогнозируемые весы измерения**, которые соответствуют наименьшим остаткам.
Для получения лучших **весов**, вам нужно **минимизировать сумму остаточных квадратов** (SSR) для всех результатов наблюдений: $SSR = \Sigma_i(y_i - f(x_i))^2$. Этот подход называется **методом наименьших квадратов**.
Простая линейная регрессия
Простая или одномерная линейная регрессия – случай линейной регрессии с единственной независимой переменной x.
А вот и она:



Реализация простой линейной регрессии начинается с заданным набором пар (зелёные круги) входов-выходов (x-y). Эти пары – результаты наблюдений. Наблюдение, крайнее слева (зелёный круг) имеет на входе x = 5 и соответствующий выход (ответ) y = 5. Следующее наблюдение имеет x = 15 и y = 20, и так далее.
Оценочная функция регрессии (чёрная линия) выражается уравнением $f(x) = b_0 + b_1x$. Нужно рассчитать оптимальные значения спрогнозированных весов b_0 и b_1 для минимизации SSR и определить оценочную функцию регрессии. Величина b_0, также называемая **отрезком**, показывает точку, где расчётная линия регрессии пересекает ось y. Это значение расчётного ответа f(x) для x = 0. Величина b_1 определяет **наклон** расчетной линии регрессии.
Предсказанные ответы (красные квадраты) – точки линии регрессии, соответствующие входным значениям. Для входа x = 5 предсказанный ответ равен f(5) = 8.33 (представленный крайним левыми квадратом).
Остатки (вертикальные пунктирные серые линии) могут быть вычислены как $y_i - f(x_i) = y_i - b_0 - b_1x_i$ для i = 1, …, n. Они представляют собой расстояния между зелёными и красными пунктами. При реализации линейной регрессии вы минимизируете эти расстояния и делаете красные квадраты как можно ближе к предопределённым зелёным кругам.
Реализуйте линейную регрессию в Python
Пришло время реализовать линейную регрессию в Python. Всё, что вам нужно, – подходящие пакеты, функции и классы.
Пакеты Python для линейной регрессии
NumPy – фундаментальный научный пакет для быстрых операций над одномерными и многомерными массивами. Он облегчает математическую рутину и, конечно, находится в open-source.

Пакет **scikit-learn** – это библиотека, широко используемая в машинном обучении. Она предоставляет значения для данных предварительной обработки, уменьшает размерность, реализует регрессию, классификацию, кластеризацию и т. д. Находится в open-source, как и NumPy.

Начните знакомство с линейными моделями и работой пакета на сайте scikit-learn.

Простая линейная регрессия со scikit-learn

Начнём с простейшего случая линейной регрессии.

Следуйте пяти шагам реализации линейной регрессии:

1. Импортируйте необходимые пакеты и классы.
2. Предоставьте данные для работы и преобразования.
3. Создайте модель регрессии и приспособьте к существующим данным.
4. Проверьте результаты совмещения и удовлетворительность модели.
5. Примените модель для прогнозов.

Это общие шаги для большинства подходов и реализаций регрессии.

Шаг 1: Импортируйте пакеты и классы

Первым шагом импортируем пакет NumPy и класс LinearRegressionиз sklearn.linear_model:

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

Теперь у вас есть весь функционал для реализации линейной регрессии.

Фундаментальный тип данных NumPy – это тип массива numpy.ndarray. Далее под **массивом** подразумеваются все экземпляры типа numpy.ndarray.

Класс sklearn.linear_model.LinearRegression используем для линейной регрессии и прогнозов.

Шаг 2 : Предоставьте данные

Вторым шагом определите данные, с которыми предстоит работать. Входы (регрессоры, x) и выход (предиктор, y) должны быть массивами (экземпляры класса numpy.ndarray) или похожими объектами. Вот простейший способ предоставления данных регрессии:

```
x = np.array([5, 15, 25, 35, 45, 55]).reshape((-1, 1))
y = np.array([5, 20, 14, 32, 22, 38])
```

Теперь у вас два массива: вход **x** и выход **y**. Вам нужно вызвать .reshape()на **x**, потому что этот массив должен быть **двумерным** или более точным – иметь **одну колонку и необходимое количество рядов**. Это как раз то, что определяет аргумент **(-1, 1)**.

Вот как **x** и **y** выглядят теперь:

```
>>> print(x)
[[ 5]
 [15]
 [25]
 [35]
 [45]
 [55]]
>>> print(y)
[ 5 20 14 32 22 38]
```

Шаг 3: Создайте модель

На этом шаге создайте и приспособьте модель линейной регрессии к существующим данным.

Давайте сделаем экземпляр класса LinearRegression, который представит модель регрессии:

```
model = LinearRegression()
```

Эта операция создаёт переменную model в качестве экземпляра LinearRegression. Вы можете предоставить несколько опциональных параметров классу LinearRegression:

- **fit_intercept** – логический (True по умолчанию) параметр, который решает, вычислять отрезок b_0 (True) или рассматривать его как равный нулю (False).
- **normalize** – логический (False по умолчанию) параметр, который решает, нормализовать входные переменные (True) или нет (False).
- **copy_X** – логический (True по умолчанию) параметр, который решает, копировать (True) или перезаписывать входные переменные (False).

- **n_jobs** – целое или None (по умолчанию), представляющее количество процессов, задействованных в параллельных вычислениях. None означает отсутствие процессов, при -1 используются все доступные процессоры.

Наш пример использует состояния параметров по умолчанию.
Пришло время задействовать model. Сначала вызовите .fit() на model:
model.fit(x, y)
С помощью .fit() вычисляются оптимальные значение весов b_0 и b_1, используя существующие вход и выход (**x** и **y**) в качестве аргументов. Другими словами, .fit() **совмещает модель**. Она возвращает self - переменную model. Поэтому можно заменить две последние операции на:
model = LinearRegression().fit(x, y)
Эта операция короче и делает то же, что и две предыдущие.

Шаг 4: Получите результаты

После совмещения модели нужно убедиться в удовлетворительности результатов для интерпретации.
Вы можете получить определения (R^2) с помощью .score(), вызванной на model:
```
>>> r_sq = model.score(x, y)
>>> print('coefficient of determination:', r_sq)
coefficient of determination: 0.715875613747954
```
.score() принимает в качестве аргументов предсказатель **x** и регрессор **y**, и возвращает значение R^2.
model содержит атрибуты .intercept_, который представляет собой коэффициент, и b_0 с .coef_, которые представляют b_1:
```
>>> print('intercept:', model.intercept_)
intercept: 5.633333333333329
>>> print('slope:', model.coef_)
slope: [0.54]
```
Код выше показывает, как получить b_0 и b_1. Заметьте, что .intercept_ – это скаляр, в то время как .coef_ – массив.
Примерное значение $b_0 = 5.63$ показывает, что ваша модель предсказывает ответ 5.63 при x, равном нулю. Равенство $b_1 = 0.54$ означает, что предсказанный ответ возрастает до 0.54 при x, увеличенным на единицу.
Заметьте, что вы можете предоставить **y** как двумерный массив. Тогда результаты не будут отличаться:
```
>>> new_model = LinearRegression().fit(x, y.reshape((-1, 1)))
>>> print('intercept:', new_model.intercept_)
intercept: [5.63333333]
>>> print('slope:', new_model.coef_)
slope: [[0.54]]
```
Как вы видите, пример похож на предыдущий, но в данном случае .intercept_ – одномерный массив с единственным элементом b_0, и .coef_ – двумерный массив с единственным элементом b_1.

Шаг 5: Предскажите ответ

Когда вас устроит ваша модель, вы можете использовать её для прогнозов с текущими или другими данными.
Получите предсказанный ответ, используя .predict():
```
>>> y_pred = model.predict(x)
>>> print('predicted response:', y_pred, sep='\n')
predicted response:
[ 8.33333333 13.73333333 19.13333333 24.53333333 29.93333333 35.33333333]
```
Применяя .predict(), вы передаёте регрессор в качестве аргумента и получаете соответствующий предсказанный ответ.
Вот почти идентичный способ предсказать ответ:
```
>>> y_pred = model.intercept_ + model.coef_ * x
>>> print('predicted response:', y_pred, sep='\n')
predicted response:
[[ 8.33333333]
```

[13.73333333]
[19.13333333]
[24.53333333]
[29.93333333]
[35.33333333]]

В этом случае вы умножаете каждый элемент массива **x** с помощью model.coef_ и добавляете model.intercept_ в ваш продукт.

Вывод отличается от предыдущего примера количеством измерений. Теперь предсказанный ответ – это двумерный массив, в отличии от предыдущего случая, в котором он одномерный.

Измените количество измерений **x** до одного, и увидите одинаковый результат. Для этого замените **x** на x.reshape(-1), x.flatten() или x.ravel() при умножении с помощью model.coef_.

На практике модель регрессии часто используется для прогнозов. Это значит, что вы можете использовать приспособленные модели для вычисления выходов на базе других, новых входов:

```
>>> x_new = np.arange(5).reshape((-1, 1))
>>> print(x_new)
[[0]
 [1]
 [2]
 [3]
 [4]]
>>> y_new = model.predict(x_new)
>>> print(y_new)
[5.63333333 6.17333333 6.71333333 7.25333333 7.79333333]
```

Здесь .predict() применяется на новом регрессоре x_new и приводит к ответу y_new. Этот пример удобно использует arange() из NumPy для генерации массива с элементами от 0 (включительно) до 5 (исключительно) – 0, 1, 2, 3, и 4.

Задания на самостоятельную работу

1. Создать модель линейной регрессии для прогнозирования содержания флавоноидов в вине от содержания яблочной кислоты.

Использовать dataset load_wine из библиотеки scikit-learn.

2. Ответьте на следующие вопросы:

1. Перечислите способы решения проблемы мультиколлинеарности в модели регрессии.
2. Укажите основные метрики для оценки прогнозной силы качества модели регрессии. В каком модуле они реализуются?
3. Обладает ли цифровая трансформация исключительно позитивным воздействием на инновационные фирмы? Обоснуйте свой ответ.
4. Каково влияние цифровизации на конкуренцию?
5. Предположите к входным или выходным характеристикам инновационной деятельности относится патентная активность предприятий;
6. Какая библиотека дает возможность провести статистический анализ значимости параметров регрессии?
7. Назовите критерии качества для оценки линейных моделей и модели бинарного выбора.
8. Перечислите критерии для проверки значимости каждого фактора и их совокупности в линейных моделях регрессии и логистической регрессии.

9. Перечислите основные методы подготовки данных для моделирования и анализа.

Задание для самостоятельной работы:

- *Проработать лекционный материал, основную и дополнительную литературу с целью подготовки ответов на представленный перечень вопросов для устного опроса.*

Практическое занятие №6. Решение задач классификации с использованием библиотеки scikit-learn

Изучение нижеперечисленных вопросов будет производиться на примере конкретных ситуаций, с целью формирования и развития профессиональных навыков обучающихся.
Форма проведения занятия – практическое занятие.

Вопросы для самоконтроля и текущего контроля

1. Метод k-ближайших соседей (K-Nearest Neighbors);
2. Метод опорных векторов (Support Vector Machines);
3. Классификатор дерева решений (Decision Tree Classifier) / Случайный лес (Random Forests);
4. Наивный байесовский метод (Naive Bayes);
5. Линейный дискриминантный анализ (Linear Discriminant Analysis);
6. Логистическая регрессия (Logistic Regression).

Основные термины

В системах машинного обучения или же системах нейросетей существуют входы и выходы. То, что подаётся на входы, принято называть признаками (англ. features).
Признаки по существу являются тем же, что и переменные в научном эксперименте — они характеризуют какой-либо наблюдаемый феномен и их можно как-то количественно измерить. Когда признаки подаются на входы системы машинного обучения, эта система пытается найти совпадения, заметить закономерность между признаками. На выходе генерируется результат этой работы.
Этот результат принято называть меткой (англ. label), поскольку у выходов есть некая пометка, выданная им системой, т. е. предположение (прогноз) о том, в какую категорию попадает выход после классификации.
В контексте машинного обучения классификация относится к обучению с учителем. Такой тип обучения подразумевает, что данные, подаваемые на входы системы, уже помечены, а важная часть признаков уже разделена на отдельные категории или классы. Поэтому сеть уже знает, какая часть входов важна, а какую часть можно самостоятельно проверить. Пример классификации — сортировка различных растений на группы, например «папоротники» и «покрытосеменные». Подобная задача может быть выполнена с помощью Дерева Решений — одного из типов классификатора в Scikit-Learn.
При обучении без учителя в систему подаются непомеченные данные, и она должна попытаться сама разделить эти данные на категории. Так как классификация относится к типу обучения с учителем, способ обучения без учителя в этой статье рассматриваться не будет.
Процесс обучения модели — это подача данных для нейросети, которая в результате должна вывести определённые шаблоны для данных. В процессе обучения модели с учителем на вход подаются признаки и метки, а при прогнозировании на вход классификатора подаются только признаки.
Принимаемые сетью данные делятся на две группы: набор данных для обучения и набор для тестирования. Не стоит проверять сеть на том же наборе данных, на которых она обучалась, т. к. модель уже будет «заточена» под этот набор.

Типы классификаторов

Scikit-Learn даёт доступ ко множеству различных алгоритмов классификации. Вот основные из них:

- Метод k-ближайших соседей (K-Nearest Neighbors);
- Метод опорных векторов (Support Vector Machines);
- Классификатор дерева решений (Decision Tree Classifier) / Случайный лес (Random Forests);
- Наивный байесовский метод (Naive Bayes);
- Линейный дискриминантный анализ (Linear Discriminant Analysis);
- Логистическая регрессия (Logistic Regression);

На сайте Scikit-Learn есть много литературы на тему этих алгоритмов с кратким пояснением работы каждого из них.

Обзор самых популярных алгоритмов машинного обучения

Метод k-ближайших соседей (K-Nearest Neighbors)

Этот метод работает с помощью поиска кратчайшей дистанции между тестируемым объектом и ближайшими к нему классифицированным объектами из обучающего набора. Классифицируемый объект будет относится к тому классу, к которому принадлежит ближайший объект набора.

Классификатор дерева решений (Decision Tree Classifier)

Этот классификатор разбивает данные на всё меньшие и меньшие подмножества на основе разных критериев, т. е. у каждого подмножества своя сортирующая категория. С каждым разделением количество объектов определённого критерия уменьшается.

Классификация подойдёт к концу, когда сеть дойдёт до подмножества только с одним объектом. Если объединить несколько подобных деревьев решений, то получится так называемый Случайный Лес (англ. Random Forest).

Наивный байесовский классификатор (Naive Bayes)

Такой классификатор вычисляет вероятность принадлежности объекта к какому-то классу. Эта вероятность вычисляется из шанса, что какое-то событие произойдёт, с опорой на уже на произошедшие события.

Каждый параметр классифицируемого объекта считается независимым от других параметров.

Линейный дискриминантный анализ (Linear Discriminant Analysis)

Этот метод работает путём уменьшения размерности набора данных, проецируя все точки данных на линию. Потом он комбинирует эти точки в классы, базируясь на их расстоянии от центральной точки.

Этот метод, как можно уже догадаться, относится к линейным алгоритмам классификации, т. е. он хорошо подходит для данных с линейной зависимостью.

Метод опорных векторов (Support Vector Machines)

Работа метода опорных векторов заключается в рисовании линии между разными кластерами точек, которые нужно сгруппировать в классы. С одной стороны линии будут точки, принадлежащие одному классу, с другой стороны — к другому классу.

Классификатор будет пытаться увеличить расстояние между рисуемыми линиями и точками на разных сторонах, чтобы увеличить свою «уверенность» определения класса. Когда все точки построены, сторона, на которую они падают — это класс, которому эти точки принадлежат.

Логистическая регрессия (Logistic Regression)

Логистическая регрессия выводит прогнозы о точках в бинарном масштабе — нулевом или единичном. Если значение чего-либо равно либо больше 0.5, то объект классифицируется в большую сторону (к единице). Если значение меньше 0.5 — в меньшую (к нулю).

У каждого признака есть своя метка, равная только 0 или только 1. Логистическая регрессия является линейным классификатором и поэтому используется, когда в данных прослеживается какая-то линейная зависимость.

Примеры задач классификации

Задача классификации — эта любая задача, где нужно определить тип объекта из двух и более существующих классов. Такие задачи могут быть разными: определение, кошка на изображении или собака, или определение качества вина на основе его кислотности и содержания алкоголя.

В зависимости от задачи классификации вы будете использовать разные типы классификаторов. Например, если классификация содержит какую-то бинарную логику, то к ней лучше всего подойдёт логистическая регрессия.

По мере накопления опыта вам будет проще выбирать подходящий тип классификатора. Однако хорошей практикой является реализация нескольких подходящих классификаторов и выбор наиболее оптимального и производительного.

Реализация классификатора

Первый шаг в реализации классификатора — его импорт в Python. Вот как это выглядит для логистической регрессии:

```
from sklearn.linear_model import LogisticRegression
```

Вот импорты остальных классификаторов, рассмотренных выше:

```
from sklearn.discriminant_analysis import LinearDiscriminantAnalysis
from sklearn.neighbors import KNeighborsClassifier
from sklearn.naive_bayes import GaussianNB
from sklearn.tree import DecisionTreeClassifier
from sklearn.svm import SVC
```

Однако, это не все классификаторы, которые есть в Scikit-Learn. Про остальные можно прочитать на соответствующей странице в документации.

После этого нужно создать экземпляр классификатора. Сделать это можно создав переменную и вызвав функцию, связанную с классификатором.

```
logreg_clf = LogisticRegression()
```

Теперь классификатор нужно обучить. Перед этим нужно «подогнать» его под тренировочные данные.

Обучающие признаки и метки помещаются в классификатор через функцию fit:

```
logreg_clf.fit(features, labels)
```

После обучения модели данные уже можно подавать в классификатор. Это можно сделать через функцию классификатора predict, передав ей параметр (признак) для прогнозирования:

```
logreg_clf.predict(test_features)
```

Эти этапы (создание экземпляра, обучение и классификация) являются основными при работе с классификаторами в Scikit-Learn. Но эта библиотека может управлять не только классификаторами, но и самими данными. Чтобы разобраться в том, как данные и классификатор работают вместе над задачей классификации, нужно разобраться в процессах машинного обучения в целом.

Процесс машинного обучения

Процесс содержит в себе следующие этапы: подготовка данных, создание обучающих наборов, создание классификатора, обучение классификатора, составление прогнозов, оценка производительности классификатора и настройка параметров.

Во-первых, нужно подготовить набор данных для классификатора — преобразовать данные в корректную для классификации форму и обработать любые аномалии в этих данных. Отсутствие значений в данных либо любые другие отклонения — все их нужно обработать, иначе они могут негативно влиять на производительность классификатора. Этот этап называется предварительной обработкой данных (англ. data preprocessing).

Следующим шагом будет разделение данных на обучающие и тестовые наборы. Для этого в Scikit-Learn существует отличная функция traintestsplit.

Как уже было сказано выше, классификатор должен быть создан и обучен на тренировочном наборе данных. После этих шагов модель уже может делать прогнозы. Сравнивая показания классификатора с фактически известными данными, можно делать вывод о точности классификатора.

Вероятнее всего, вам нужно будет «корректировать» параметры классификатора, пока вы не достигните желаемой точности (т. к. маловероятно, что классификатор будет соответствовать всем вашим требованиям с первого же запуска).

Ниже будет представлен пример работы машинного обучения от обработки данных и до оценки.

Реализация образца классификации

```
# Импорт всех нужных библиотек
import pandas as pd
from sklearn.metrics import classification_report
from sklearn.metrics import confusion_matrix
```

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

Поскольку набор данных iris достаточно распространён, в Scikit-Learn он уже присутствует, достаточно лишь заложить эту команду:

```
sklearn.datasets.load_iris
```

Тем не менее, тут ещё нужно подгрузить CSV-файл, который можно скачать здесь.
Этот файл нужно поместить в ту же папку, что и Python-файл. В библиотеке Pandas есть функция read_csv(), которая отлично работает с загрузкой данных.

```
data = pd.read_csv('iris.csv')

# Проверяем, всё ли правильно загрузилось
print(data.head(5))
```

Благодаря тому, что данные уже были подготовлены, долгой предварительной обработки они не требуют. Единственное, что может понадобиться — убрать ненужные столбцы (например ID) таким образом:

```
data.drop('Id', axis=1, inplace=True)
```

Теперь нужно определить признаки и метки. С библиотекой Pandas можно легко «нарезать» таблицу и выбрать определённые строки/столбцы с помощью функции iloc():

```
# ".iloc" принимает row_indexer, column_indexer
X = data.iloc[:,:-1].values
# Теперь выделим нужный столбец
y = data['Species']
```

Код выше выбирает каждую строку и столбец, обрезав при этом последний столбец.
Выбрать признаки интересующего вас набора данных можно также передав в скобках заголовки столбцов:

```
# Альтернативный способ выбора нужных столбцов:
X = data.iloc['SepalLengthCm', 'SepalWidthCm', 'PetalLengthCm']
```

После того, как вы выбрали нужные признаки и метки, их можно разделить на тренировочные и тестовые наборы, используя функцию train_test_split():

```
# test_size показывает, какой объем данных нужно выделить для тестового набора
# Random_state — просто сид для случайной генерации
# Этот параметр можно использовать для воссоздания определённого результата:
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.20, random_state=27)
```

Чтобы убедиться в правильности обработки данных, используйте:

```
print(X_train)
print(y_train)
```

Теперь можно создавать экземпляр классификатора, например метод опорных векторов и метод k-ближайших соседей:

```
SVC_model = svm.SVC()
# В KNN-модели нужно указать параметр n_neighbors
# Это число точек, на которое будет смотреть
# классификатор, чтобы определить, к какому классу принадлежит новая точка
KNN_model = KNeighborsClassifier(n_neighbors=5)
```

Теперь нужно обучить эти два классификатора:

```
SVC_model.fit(X_train, y_train)
KNN_model.fit(X_train, y_train)
```

Эти команды обучили модели и теперь классификаторы могут делать прогнозы и сохранять результат в какую-либо переменную.

```
SVC_prediction = SVC_model.predict(X_test)
KNN_prediction = KNN_model.predict(X_test)
```

Теперь пришло время оценить точности классификатора. Существует несколько способов это сделать.

Нужно передать показания прогноза относительно фактически верных меток, значения которых были сохранены ранее.

```
# Оценка точности — простейший вариант оценки работы классификатора
print(accuracy_score(SVC_prediction, y_test))
print(accuracy_score(KNN_prediction, y_test))
# Но матрица неточности и отчёт о классификации дадут больше информации о производительности
print(confusion_matrix(SVC_prediction, y_test))
print(classification_report(KNN_prediction, y_test))
```

Вот, к примеру, результат полученных метрик:

```
SVC accuracy: 0.9333333333333333
KNN accuracy: 0.9666666666666667
```

Поначалу кажется, что KNN работает точнее. Вот матрица неточностей для SVC:

```
[[ 7  0  0]
 [ 0 10  1]
 [ 0  1 11]]
```

Количество правильных прогнозов идёт с верхнего левого угла в нижний правый. Вот для сравнения метрики классификации для KNN:

	precision	recall	f1-score	support
Iris-setosa	1.00	1.00	1.00	7
Iris-versicolor	0.91	0.91	0.91	11
Iris-virginica	0.92	0.92	0.92	12
micro avg	0.93	0.93	0.93	30
macro avg	0.94	0.94	0.94	30
weighted avg	0.93	0.93	0.93	30

Оценка классификатора

Когда дело доходит до оценки точности классификатора, есть несколько вариантов.

Точность классификации

Точность классификации измерять проще всего, и поэтому этот параметр чаще всего используется. Значение точности — это число правильных прогнозов, делённое на число всех прогнозов или, проще говоря, отношение правильных прогнозов ко всем.

Хоть этот показатель и может быстро дать вам явное представление о производительности классификатора, его лучше использовать, когда каждый класс имеет хотя бы примерно одинаковое количество примеров. Так как такое будет случаться редко, рекомендуется использовать другие показатели классификации.

Логарифмические потери

Значение Логарифмических Потерь (англ. Logarithmic Loss) — или просто логлосс — показывает, насколько классификатор «уверен» в своём прогнозе. Логлосс возвращает вероятность принадлежности объекта к тому или иному классу, суммируя их, чтобы дать общее представление об «уверенности» классификатора.

Этот показатель лежит в промежутке от 0 до 1 — «совсем не уверен» и «полностью уверен» соответственно. Логлосс сильно падает, когда классификатор сильно «уверен» в неправильном ответе.

Площадь ROC-кривой (AUC)

Такой показатель используется только при бинарной классификации. Площадь под ROC-кривой представляет способность классификатора различать подходящие и не подходящие какому-либо классу объекты.

Значение 1.0: вся область, попадающая под кривую, представляет собой идеальный классификатор. Следовательно, 0.5 означает, что точность классификатора соответствует случайности. Кривая рассчитывается с учётом точности и специфичности модели. Подробнее о расчётах можно прочитать здесь.

Матрица неточностей
Матрица неточностей (англ. Confusion Matrix) — это таблица или диаграмма, показывающая точность прогнозирования классификатора в отношении двух и более классов. Прогнозы классификатора находятся на оси X, а результат (точность) — на оси Y.
Ячейки таблицы заполняются количеством прогнозов классификатора. Правильные прогнозы идут по диагонали от верхнего левого угла в нижний правый. Про это можно почитать в данной статье.
Отчёт о классификации
В библиотеке Scikit-Learn уже встроена возможность создавать отчёты о производительности классификатора. Эти отчёты дают интуитивно понятное представление о работе модели.

Задания на самостоятельную работу

1. Ответьте на следующие вопросы:
   1. В чем состоит смысл обучения в метрических методах?
   2. Для чего используется библиотека Scikit-learn?
   3. Для чего выполняют масштабирование признаков?
   4. Как обычно выполняется масштабирование количественных признаков?
   5. В каком классе Scikit-learn реализован метод kNN?
   6. Какой параметр метода k ближайших соседей, задает число соседей для построения прогноза?
   7. В чем смысл кроссвалидации?
   8. Как вычисляется весовая Евклидова метрика?

Задание для самостоятельной работы:

- *Проработать лекционный материал, основную и дополнительную литературу с целью подготовки ответов на представленный перечень вопросов для устного опроса.*

Практическое занятие №7. Решение задач снижения размерности и кластеризации с использованием библиотеки scikit-learn

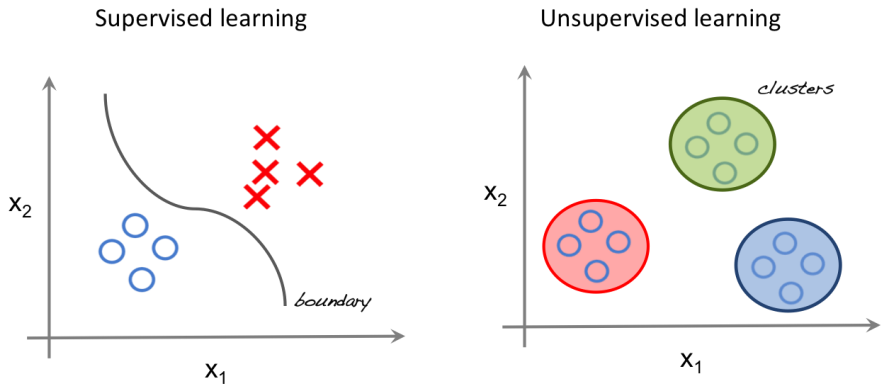
Изучение нижеперечисленных вопросов будет производиться на примере конкретных ситуаций, с целью формирования и развития профессиональных навыков обучающихся.
Форма проведения занятия – практическое занятие.

Вопросы для самоконтроля и текущего контроля

1. Обучение без учителя
2. Задачи кластеризации
3. Линейный факторный анализ. Метод главных компонент
4. Снижение размерности: метод t-SNE
5. Кластеризация объектов: метод k-means
6. Иерархическая кластеризация

Основные термины

Обучение без учителя
Обучение без учителя (unsupervised learning, неконтролируемое обучение) – класс методов машинного обучения для поиска шаблонов в наборе данных. Данные, получаемые на вход таких алгоритмов обычно не размечены, то есть передаются только входные переменные X без соответствующих меток y. Если в контролируемом обучении (обучении с учителем, supervised learning) система пытается извлечь уроки из предыдущих примеров, то в обучении без учителя – система старается самостоятельно найти шаблоны непосредственно из приведенного примера.



На левой части изображения представлен пример контролируемого обучения: здесь для того, чтобы найти лучшую функцию, соответствующую представленным точкам, используется метод регрессии. В то же время при неконтролируемом обучении входные данные разделяются на основе представленных характеристик, а предсказание свойств основывается на том, какому кластеру принадлежит пример.

Методы кластеризации данных являются одним из наиболее популярных семейств машинного обучения без учителя. Рассмотрим некоторые из них подробнее.

Важная терминология

- **Feature** (Особенности): входная переменная, используемая для создания прогнозов.
- **Predictions** (Прогнозы): выходные данные модели при наличии входного примера.
- **Example** (Пример): строка набора данных. Пример обычно содержит один или несколько объектов.
- **Label** (Метки): результат функции.

Подготовка выборки для кластеризации данных

Для составления прогнозов воспользуемся классическим набором данных ирисов Фишера. Датасет представляет набор из 150 записей с пятью атрибутами в следующем порядке: длина чашелистика (sepal length), ширина чашелистика (sepal width), длина лепестка (petal length), ширина лепестка (petal width) и класс, соответствующий одному из трех видов: Iris Setosa, Iris Versicolor или Iris Virginica, обозначенных соответственно 0, 1, 2. Наш алгоритм должен принимать четыре свойства одного конкретного цветка и предсказывать, к какому классу (виду ириса) он принадлежит. Имеющиеся в наборе данных метки можно использовать для оценки качества предсказания.

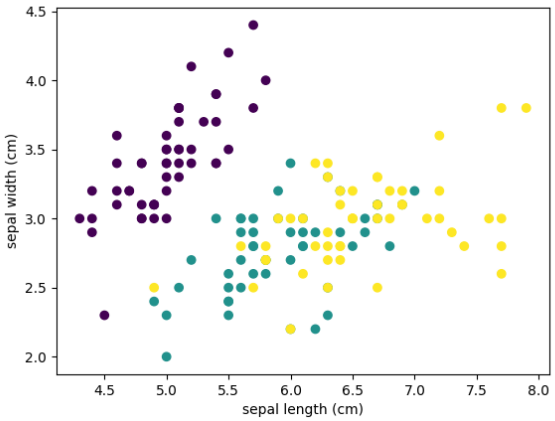
Для решения задач кластеризации данных в этой статье мы используем Python, библиотеку scikit-learn для загрузки и обработки набора данных и matplotlib для визуализации. Ниже представлен программный код для исследования исходного набора данных.

```
# Импортируем библиотеки
from sklearn import datasets
import matplotlib.pyplot as plt

# Загружаем набор данных
iris_df = datasets.load_iris()

# Методы, доступные для набора данных
print(dir(iris_df))

# Признаки
print(iris_df.feature_names)

# Метки
print(iris_df.target)

# Имена меток
print(iris_df.target_names)

# Разделение набора данных
x_axis = iris_df.data[:, 0]  # Sepal Length
```

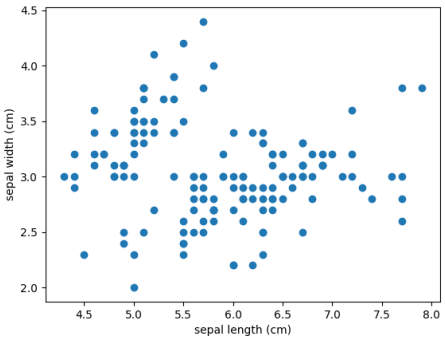
```
y_axis = iris_df.data[:, 1]  # Sepal Width

# Построение
plt.xlabel(iris_df.feature_names[0])
plt.ylabel(iris_df.feature_names[1])
plt.scatter(x_axis, y_axis, c=iris_df.target)
plt.show()
```

В результате запуска программы вы увидим следующие текст и изображение.

```
['DESCR', 'data', 'feature_names', 'target', 'target_names']
['sepal length (cm)', 'sepal width (cm)', 'petal length (cm)', 'petal width
(cm)']
[0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
 0 0 0 0 0 0 0 0 0 0 0 0 0 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 2 2 2 2 2 2 2 2 2 2 2 2 2
 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2
 2 2]
['setosa' 'versicolor' 'virginica']
```



На диаграмме фиолетовым цветом обозначен вид Setosa, зеленым – Versicolor и желтым – Virginica. При построении были взяты лишь два признака. Вы можете проанализировать как разделяются классы при других комбинациях параметров.



Цель кластеризации данных состоит в том, чтобы выделить группы примеров с похожими чертами и определить соответствие примеров и кластеров. При этом исходно у нас нет примеров такого разбиения. Это аналогично тому, как если бы в приведенном наборе данных у нас не было меток, как на рисунке ниже.

Наша задача – используя все имеющиеся данные, предсказать соответствие объектов выборки их классам, сформировав таким образом кластеры.

Метод k-средних

Наиболее популярным алгоритмом кластеризации данных является метод k-средних. Это итеративный алгоритм кластеризации, основанный на минимизации суммарных квадратичных отклонений точек кластеров от центроидов (средних координат) этих кластеров.

Первоначально выбирается желаемое количество кластеров. Поскольку нам известно, что в нашем наборе данных есть 3 класса, установим параметр модели **n_clusters** равный трем.
Теперь случайным образом из входных данных выбираются три элемента выборки, в соответствие которым ставятся три кластера, в каждый из которых теперь включено по одной точке, каждая при этом является центроидом этого кластера.
Далее ищем ближайшего соседа текущего центроида. Добавляем точку к соответствующему кластеру и пересчитываем положение центроида с учетом координат новых точек. Алгоритм заканчивает работу, когда координаты каждого центроида перестают меняться. Центроид каждого кластера в результате представляет собой набор значений признаков, описывающих усредненные параметры выделенных классов.

```
# Импортируем библиотеки
from sklearn import datasets
from sklearn.cluster import KMeans

# Загружаем набор данных
iris_df = datasets.load_iris()

# Описываем модель
model = KMeans(n_clusters=3)

# Проводим моделирование
model.fit(iris_df.data)

# Предсказание на единичном примере
predicted_label = model.predict([[7.2, 3.5, 0.8, 1.6]])

# Предсказание на всем наборе данных
all_predictions = model.predict(iris_df.data)

# Выводим предсказания
print(predicted_label)
print(all_predictions)
```

Результат:

```
[1]
[1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
 1 1 1 1 1 1 1 1 1 1 1 1 0 0 2 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
 0 0 0 2 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 2 0 2 2 2 2 0 2 2 2 2
 2 2 0 0 2 2 2 2 0 2 0 2 0 2 2 0 0 2 2 2 2 2 0 2 2 2 2 0 2 2 2 0 2 2 2 0 2
 2 0]
```

При выводе данных нужно понимать, что алгоритм не знает ничего о нумерации классов, и числа 0, 1, 2 – это лишь номера кластеров, определенных в результате работы алгоритма. Так как исходные точки выбираются случайным образом, вывод будет несколько меняться от одного запуска к другому.
Характерной особенностью набора данных ирисов Фишера является то, что один класс (Setosa) легко отделяется от двух остальных. Это заметно и в приведенном примере.

Задания на самостоятельную работу

1. На основании набора данных о покупках в розничном магазине, занимающемся продажей подарков и сувениров через интернет-сайт, осуществите кластеризацию заказов и покупателей. Источник данных: https://archive.ics.uci.edu/ml/datasets/online+retail.
2. Ответьте на следующие вопросы:
    1. Перечислите основные методы кластеризации.
    2. Укажите основные меры расстояний между кластерами.
    3. Определите ключевые причины, определяющие специфичные черты кластеров.
    4. Выделите классификационные признаки типов инновационного процесса.

5. В чем состоит отличие задач классификации и кластеризации?
6. Опишите способы определения числа кластеров.
7. Перечислите основные проблемы решения задачи кластеризации. Как обычно выполняется масштабирование количественных признаков?

Задание для самостоятельной работы:

- *Проработать лекционный материал, основную и дополнительную литературу с целью подготовки ответов на представленный перечень вопросов для устного опроса.*

Практические (семинарские занятия) представляют собой детализацию лекционного теоретического материала, проводятся в целях закрепления курса и охватывают основные разделы.

Основной формой проведения семинаров и практических занятий является обсуждение наиболее проблемных и сложных вопросов по отдельным темам, а также решение задач и разбор примеров и ситуаций в аудиторных условиях. В обязанности преподавателя входят: оказание методической помощи и консультирование обучающихся по соответствующим темам курса.

На семинаре каждый его участник должен быть готовым к выступлению по всем поставленным в плане вопросам, проявлять максимальную активность при их рассмотрении. Выступление должно строиться свободно, убедительно и аргументировано. Преподаватель следит, чтобы выступление не сводилось к репродуктивному уровню (простому воспроизведению текста), не допускается и простое чтение конспекта. Необходимо, чтобы выступающий проявлял собственное отношение к тому, о чем он говорит, высказывал свое личное мнение, понимание, обосновывал его и мог сделать правильные выводы из сказанного. При этом студент может обращаться к записям конспекта и лекций, непосредственно к первоисточникам, использовать знание художественной литературы и искусства, факты и наблюдения современной жизни и т. д. Вокруг такого выступления могут разгореться споры, дискуссии, к участию в которых должен стремиться каждый. Преподавателю необходимо внимательно и критически слушать, подмечать особенное в суждениях студентов, улавливать недостатки и ошибки, корректировать их знания, и, если нужно, выступить в роли рефери, обратить внимание на то, что еще не было сказано, или поддержать и развить интересную мысль, высказанную выступающим студентом. В заключение преподаватель, как руководитель семинара, подводит итоги семинара. Он может (выборочно) проверить конспекты студентов и, если потребуется, внести в них исправления и дополнения.

Активность на практических занятиях оценивается по следующим критериям:

- ответы на вопросы, предлагаемые преподавателем;
- участие в дискуссиях;
- выполнение проектных и иных заданий;
- ассистирование преподавателю в проведении занятий.

Доклады и оппонирование докладов проверяют степень владения теоретическим материалом, а также корректность и строгость рассуждений.

7. ЛАБОРАТОРНЫЕ РАБОТЫ (ЛАБОРАТОРНЫЙ ПРАКТИКУМ)

Не предусмотрены учебным планом.

8. ТЕМАТИКА КУРСОВЫХ, КОНТРОЛЬНЫХ РАБОТ, РЕФЕРАТОВ

Курсовые и контрольные работы не предусмотрены учебным планом.

Реферат это одна из форм текущего контроля знаний и контроля самостоятельной работы. Реферат – это самостоятельная исследовательская работа, в которой автор раскрывает суть исследуемой проблемы; приводит различные точки зрения, а также собственные взгляды не нее. Содержание реферата должно быть логичным; изложение материала должно носить проблемно-тематический характер.

Цель реферата как формы текущего контроля знаний и контроля самостоятельной работы - стимулировать раскрытие исследовательского потенциала учащегося, способность к творческому поиску, сотрудничеству, самораскрытию и проявлению возможностей.

Примерная тематика рефератов:

№ задания	Тема
1	Методы машинного обучения, распознавания образов и компьютерного зрения.
2	Сверточные нейронные сети
3	Рекуррентные нейронные сети
4	Архитектура искусственных нейронных сетей
5	Современные гибридные архитектуры глубоко обучения
6	Методы решения задач распознавания изображений
7	История развития машинного зрения
8	Использование AI в образовании
9	Использовании python для анализа данных
10	Алгоритм обратного распространения ошибки
11	Прикладной анализ данных
12	Технологии искусственного интеллекта для моделирования новых материалов
13	Машинное обучение для классификации текстов
14	Распознавание изображений на Python с помощью TensorFlow и Keras
15	Обнаружение и распознавание объектов в Python с помощью OpenCV
16	Задачи регрессии
17	Задачи классификации
18	Задачи кластеризации
19	Рекомендательные системы
20	Ансамблевые методы: стекинг, бэггинг, бустинг
21	Технологии обработки естественного языка (NLP) и интеллектуальный анализ документов

Формулировки приведенных выше тем являются примерными и могут быть изменены. Изменения согласуются с преподавателем, ведущим дисциплину. Кроме этого, обучающиеся могут предлагать собственные темы для исследования. Инициативные темы также согласуются с преподавателем.

В процессе изучения курса каждый должен подготовить реферат, который будет засчитан преподавателем, ведущим дисциплину.

Оценивая реферат, преподаватель обращает внимание на:

- соответствие содержания выбранной теме;
- отсутствие в тексте отступлений от темы;
- соблюдение структуры работы, четкость изложения и обоснованность выводов;
- умение работать с научной литературой - вычленять проблему из контекста;
- умение логически мыслить;
- культуру письменной речи;

- умение оформлять научный текст (правильное применение и оформление ссылок, составление библиографии и т.д.);
- умение правильно понять позицию авторов, работы которых использовались при написании реферата;
- способность верно, без искажения передать используемый авторский материал;
- соблюдение объема работы;
- соответствие установленным правилам оформления работы;
- аккуратность и правильность технического выполнения работы.

9. ПЕРЕЧЕНЬ ВОПРОСОВ К ЗАЧЕТУ

1. Нейрофизиологические данные об обработке информации в биологических системах.
2. Искусственный нейрон. Идея и техническая реализация.
3. Модели нейронов. Типичные виды функций активации нейрона.
4. Многослойный персептрон.
5. Однонаправленные многослойные сети. Алгоритм обратного распространения ошибки.
6. Вывод конкретных формул алгоритма обратного распространения ошибки для двухслойных сетей с малым числом нейронов (2-3).
7. Градиентные методы. Алгоритм наискорейшего спуска. Недостатки метода.
8. Рекуррентные сети. Ассоциативная сеть Хопфилда. Обучение. Распознавание образов.
9. Сеть встречного распространения.
10. Обучение слоя Кохонена. Решение задач кластеризации.
11. Статистический подход к обучению нейронной сети.
12. Применение нейронных сетей. Сбор данных для нейронных сетей.
13. Задача регрессии и прогнозирования временных рядов.
14. Структура и свойства искусственного нейрона.
15. Типы функций активации нейронов.
16. Архитектура, классификация и свойства нейронных сетей.
17. Нейронные сети Хопфилда и Хэмминга.
18. Основные понятия и определения гибридных сетей.
19. Алгоритмы обучения и использования гибридных сетей.
20. Нечеткие нейронные сети.
21. Программная реализация моделей гибридных нейронных сетей.
22. Нейронные сети для аппроксимации функций.
23. Создание и использование самоорганизующейся карты Кохонена.

10. САМОСТОЯТЕЛЬНАЯ РАБОТА ОБУЧАЮЩИХСЯ

Самостоятельная работа студентов является особой формой организации учебного процесса, представляющая собой планируемую, познавательно, организационно и методически направляемую деятельность студентов, ориентированную на достижение конкретного результата, осуществляемую без прямой помощи преподавателя. Самостоятельная работа студентов является составной частью учебной работы и имеет целью закрепление и углубление полученных знаний и навыков, поиск и приобретение новых знаний, а также выполнение учебных заданий, подготовку к предстоящим занятиям и зачету. Она предусматривает, как правило, разработку рефератов, написание докладов, выполнение творческих, индивидуальных заданий в соответствии с учебной программой (тематическим планом изучения дисциплины). Тема для такого выступления может быть

предложена преподавателем или избрана самим студентом, но материал выступления не должен дублировать лекционный материал. Реферативный материал служит дополнительной информацией для работы на практических занятиях. Основная цель данного вида работы состоит в обучении студентов методам самостоятельной работы с учебным материалом. Для полноты усвоения тем, вынесенных в практические занятия, требуется работа с первоисточниками. Курс предусматривает самостоятельную работу студентов со специальной литературой. Следует отметить, что самостоятельная работа студентов результативна лишь тогда, когда она выполняется систематически, планомерно и целенаправленно. Задания для самостоятельной работы предусматривают использование необходимых терминов и понятий по проблематике курса. Они нацеливают на практическую работу по применению изучаемого материала, поиск библиографического материала и электронных источников информации, иллюстративных материалов. Задания по самостоятельной работе даются по темам, которые требуют дополнительной проработки. Общий объем самостоятельной работы студентов по дисциплине включает аудиторную и внеаудиторную самостоятельную работу студентов в течение семестра. Аудиторная самостоятельная работа осуществляется в форме выполнения тестовых заданий, кейс-задач, письменных проверочных работ по дисциплине. Аудиторная самостоятельная работа обеспечена базой тестовых материалов, кейс-задач по разделам дисциплины.

Формирование внутренней потребности к самообучению становится и требованием времени, и условием реализации личностного потенциала. Способность человека состояться на уровне, адекватном его претензиям на положение в обществе, всецело зависит от его индивидуальной вовлеченности в самостоятельный процесс освоения новых знаний. Поэтому одной из целей профессиональной подготовки специалиста является необходимость дать студенту прочные фундаментальные знания, на основе которых он смог бы обучаться самостоятельно в нужном ему направлении. Решение задач современного образования невозможно без повышения роли самостоятельной работы студентов над учебным материалом, усиления ответственности преподавателей за развитие навыков самостоятельной работы, за стимулирование профессионального роста студентов, воспитание их творческой активности и инициативы. Методологическую основу самостоятельной работы студентов составляет деятельностный подход, который состоит в том, что цели обучения ориентированы на формирование умений решать типовые и нетиповые задачи, т. е. на реальные ситуации, где студентам надо проявить знание конкретной дисциплины.

Основная задача организации СРС заключается в создании психолого-дидактических условий развития интеллектуальной инициативы и мышления на занятиях любой формы. Основным принципом организации СРС должен стать перевод всех студентов на индивидуальную работу с переходом от формального пассивного выполнения определенных заданий к познавательной активности с формированием собственного мнения при решении поставленных проблемных вопросов и задач. Таким образом, в результате самостоятельной работы студент должен научиться осмысленно и самостоятельно работать сначала с учебным материалом, затем с научной информацией, использовать основы самоорганизации и самовоспитания с тем, чтобы развивать в дальнейшем умение непрерывно повышать свою квалификацию.

Виды самостоятельной работы студентов, обеспечивающие реализацию цели и решение задач данной рабочей программы:

- подготовка к практическим занятиям;
- изучение тем дисциплины, выносимых для самостоятельного изучения;
- подготовка реферата;
- подготовка к тестированию;

– выполнение самостоятельных практических, работ на занятиях;
– работа со справочной, методической и научной литературой;
– разбор конкретных ситуаций, в том числе углубляющих теоретические знания;
– участие в дискуссиях;
– работа над учебным материалом (учебника, первоисточника, дополнительной литературы, аудио- и видеозаписей);
– подготовка к проблемным семинарам, дискуссионным вопросам, «круглым столам» и др.;
– изучение отдельных тем (вопросов) учебных дисциплин, составление плана и тезисов ответа;
– ответы на контрольные вопросы;
– подготовка тезисов сообщений к выступлению на семинаре;
– экспресс-опросы по конкретным темам;
– подготовка к промежуточной аттестации.

Обучающиеся выполняют задания, самостоятельно обращаясь к учебной литературе. Проверка выполнения заданий осуществляется путем электронного тестирования и устного опроса на практических занятиях.

Материалы курса, выносимые студентам для самостоятельного изучения:

Форма обучения очная

Название разделов и тем	Вид самостоятельной работы	Объем в часах	Форма контроля
Модуль 1. Введение в машинное обучение	– Проработка учебного материала с использованием ресурсов учебно-методического и информационного обеспечения дисциплины; – Подготовка к тестированию; – Подготовка к решению кейсов; – Подготовка реферата; – Подготовка к сдаче зачета.	36	проверка тестовых заданий, устный опрос, решение кейсов
Модуль 2. Решение задач машинного обучения	– Проработка учебного материала с использованием ресурсов учебно-методического и информационного обеспечения дисциплины; – Подготовка к тестированию; – Подготовка к решению кейсов; – Подготовка реферата – Подготовка к сдаче зачета.	36	проверка тестовых заданий, устный опрос, решение кейсов

Методические указания для обучающихся по освоению дисциплины

Для качественного усвоения обучающимися материала курса при выполнении ими индивидуальных заданий необходимо, чтобы все работы выполнялись студентами после проработки соответствующего материала. Основная задача по организации учебного процесса по данной дисциплине сводится к обеспечению равномерной активной работы обучающихся над курсом в течение всего учебного семестра. Обучающиеся должны регулярно прорабатывать курс пройденных семинаров, готовиться к занятиям. Для контроля качества усвоения учебного материала следует проводить опросы по изученной теме. Для долговременного запоминания изученного материала следует увязывать вновь изучаемые вопросы с материалом предыдущих тем, добиваться преемственности знаний.

При выполнении заданий, вынесенных на самостоятельное изучение, необходимо наряду с библиотечным фондом пользоваться различными источниками знаний, размещенными в сети Интернет.

При изучении данного курса обучающимся предстоит выполнить следующие основные виды работ:

- Анализ теоретического материала;
- Проработка материала семинарских занятий;
- Выполнение практических заданий;
- Подготовка к семинару;
- Подготовка к тестированию.

Семинарские занятия желательно проводить с применением демонстрационного материала – презентации на ПК с проектором. С учетом современных возможностей, желательно обеспечивать слушателей раздаточным материалом на 1-2 занятия вперед. Материал этот должен носить иллюстративный характер (схемы, формулы, графики) и ни в коем случае не подменять конспекта, который слушатель должен составлять самостоятельно.

Текущий контроль

Для текущего контроля успеваемости (по отдельным разделам дисциплины) и промежуточной аттестации используется компьютерное тестирование, проверка реферата.

Методические указания для обучающихся по освоению отдельных дидактических единиц дисциплины

1. Планирование и организация времени, необходимого для самостоятельного изучения дисциплины.

Рекомендуется следующим образом организовать время, необходимое для изучения дисциплины:

- Изучение конспекта семинара в тот же день, после занятия: 1-2 часа.
- Подготовка к семинарскому занятию: 1-3 часа.
- Изучение теоретического материала по учебнику и конспекту (включая дополнительные источники, в том числе, в электронной форме): 3-4 часа в неделю.
- Подготовка к тестированию по текущей теме: 1-3 часа.
- Всего в неделю: 6–9 часов.

2. Методические рекомендации по подготовке к практическим занятиям.

По данному курсу предусмотрены практические занятия. При подготовке следует изучить соответствующий теоретический материал по цифровой экономике, электронной коммерции, электронному бизнесу или электронным платежным системам. Теоретический материал курса становится более понятным, когда дополнительно к обучению на семинарах и изучению конспекта, изучаются и книги по современным информационным технологиям.

Подготовка к практическому занятию включает 2 этапа: 1й - организационный; 2й - закрепление и углубление теоретических знаний. На первом этапе обучающийся планирует свою самостоятельную работу, которая включает: - уяснение задания на самостоятельную работу; - подбор рекомендованной литературы; - составление плана работы, в котором определяются основные пункты предстоящей подготовки. Составление плана дисциплинирует и повышает организованность в работе. Второй этап включает непосредственную подготовку студента к занятию. Начинать надо с изучения рекомендованной литературы. Необходимо помнить, что на семинаре обычно рассматривается не весь материал, а только его часть. Остальная его часть восполняется в процессе самостоятельной работы. В связи с этим работа с рекомендованной литературой обязательна. Особое внимание при этом необходимо обратить на содержание основных положений и выводов, объяснение явлений и фактов, уяснение практического приложения рассматриваемых теоретических вопросов. В процессе этой работы обучающийся должен стремиться понять и запомнить основные положения рассматриваемого материала, примеры,

поясняющие его, а также разобраться в иллюстративном материале. Заканчивать подготовку следует составлением плана (конспекта) по изучаемому материалу (вопросу). Это позволяет составить концентрированное, сжатое представление по изучаемым вопросам. В процессе подготовки к занятиям рекомендуется взаимное обсуждение материала, во время которого закрепляются знания, а также приобретается практика в изложении и разъяснении полученных знаний, развивается речь. При необходимости следует обращаться за консультацией к преподавателю. Идя на консультацию, необходимо хорошо продумать вопросы, которые требуют разъяснения. Записи имеют первостепенное значение для самостоятельной работы обучающихся. Они помогают понять построение изучаемого материала, выделить основные положения, проследить их логику и тем самым проникнуть в творческую лабораторию автора. Ведение записей способствует превращению чтения в активный процесс, мобилизует, наряду со зрительной, и моторную память. Следует помнить: у обучающегося, систематически ведущего записи, создается свой индивидуальный фонд подсобных материалов для быстрого повторения прочитанного, для мобилизации накопленных знаний. Особенно важны и полезны записи тогда, когда в них находят отражение мысли, возникшие при самостоятельной работе. Важно развивать у обучающихся умение сопоставлять источники, продумывать изучаемый материал. Большое значение имеет совершенствование навыков конспектирования у обучающихся. Преподаватель может рекомендовать студентам следующие основные формы записи: план (простой и развернутый), выписки, тезисы. Результаты конспектирования могут быть представлены в различных формах. План - это схема прочитанного материала, краткий (или подробный) перечень вопросов, отражающих структуру и последовательность материала. Подробно составленный план вполне заменяет конспект. Конспект - это систематизированное, логичное изложение материала источника. Различаются четыре типа конспектов:

— План-конспект - это развернутый детализированный план, в котором достаточно подробные записи приводятся по тем пунктам плана, которые нуждаются в пояснении.

— Текстуальный конспект - это воспроизведение наиболее важных положений и фактов источника.

— Свободный конспект - это четко и кратко сформулированные (изложенные) основные положения в результате глубокого осмысливания материала. В нем могут присутствовать выписки, цитаты, тезисы; часть материала может быть представлена планом.

— Тематический конспект - составляется на основе изучения ряда источников и дает более или менее исчерпывающий ответ по какой-то схеме (вопросу).

3. Групповая консультация

Разъяснение является основным содержанием данной формы занятий, наиболее сложных вопросов изучаемого программного материала. Цель - максимальное приближение обучения к практическим интересам с учетом имеющейся информации и является результативным материалом закрепления знаний. Групповая консультация проводится в следующих случаях:

— когда необходимо подробно рассмотреть практические вопросы, которые были недостаточно освещены или совсем не освещены в процессе лекции;

— с целью оказания помощи в самостоятельной работе (написание рефератов, выполнение курсовых работ, сдача экзаменов (зачетов), подготовка конференций);

— если обучающиеся самостоятельно изучают нормативный, справочный материал, инструкции, положения.

11. УЧЕБНО-МЕТОДИЧЕСКОЕ И ИНФОРМАЦИОННОЕ ОБЕСПЕЧЕНИЕ ДИСЦИПЛИНЫ

а) Список рекомендуемой литературы

Основная:

1. Платонов, А. В. Машинное обучение : учебное пособие для вузов / А. В. Платонов. — Москва : Издательство Юрайт, 2023. — 85 с. — (Высшее образование). — ISBN 978-5-534-15561-7. — Текст : электронный // Образовательная платформа Юрайт [сайт]. — URL: https://urait.ru/bcode/520544
2. Павлова, А. И. Искусственные нейронные сети : учебное пособие / А. И. Павлова. — Москва : Ай Пи Ар Медиа, 2021. — 190 с. — ISBN 978-5-4497-1165-6. — Текст : электронный // Цифровой образовательный ресурс IPR SMART : [сайт]. — URL: https://www.iprbookshop.ru/108228.html

Дополнительная:

1. Станкевич, Л. А. Интеллектуальные системы и технологии : учебник и практикум для вузов / Л. А. Станкевич. — 2-е изд., перераб. и доп. — Москва : Издательство Юрайт, 2023. — 495 с. — (Высшее образование). — ISBN 978-5-534-16238-7. — Текст : электронный // Образовательная платформа Юрайт [сайт]. — URL: https://urait.ru/bcode/530657
2. Вакуленко, С. А. Нейронные сети : учебное пособие / С. А. Вакуленко, А. А. Жихарева. — Санкт-Петербург : Санкт-Петербургский государственный университет промышленных технологий и дизайна, 2019. — 110 с. — Текст : электронный // Цифровой образовательный ресурс IPR SMART : [сайт]. — URL: https://www.iprbookshop.ru/102447.html

Учебно-методическая литература:

1. Сковиков А. Г. Машинное обучение : методические рекомендации для самостоятельной работы студентов направления 38.04.01 «Экономика». Профиль – искусственный интеллект в финансово-экономических системах / А. Г. Сковиков ; Ульян. гос. ун-т, Ин-т экономики и бизнеса. - 2023. - Неопубликованный ресурс. - URL: http://lib.ulsu.ru/MegaPro/Download/MObject/15135. - Режим доступа: ЭБС УлГУ. - Текст : электронный. URL: http://lib.ulsu.ru/MegaPro/UserEntry?Action=Link_FindDoc&id=511363&idb=0

Согласовано:

Гл.нав.библиотекарь	Голосова М.И		01.06.2023
Должность сотрудника научной библиотеки	ФИО	подпись	дата

б) Программное обеспечение

- Windows;
- Office;
- МойОфис Стандартный;
- Jupyter Notebook в составе дистрибутива Anacodnda;
- Онлайн-сервис Google Cplaboratory.

в) Профессиональные базы данных, информационно-справочные системы

1. Электронно-библиотечные системы:

1.1. Цифровой образовательный ресурс IPRsmart : электронно-библиотечная система : сайт / ООО Компания «Ай Пи Ар Медиа». - Саратов, [2023]. – URL: http://www.iprbookshop.ru. – Режим доступа: для зарегистрир. пользователей. - Текст : электронный.

1.2. Образовательная платформа ЮРАЙТ : образовательный ресурс, электронная библиотека : сайт / ООО Электронное издательство «ЮРАЙТ». – Москва, [2023]. - URL: https://urait.ru. – Режим доступа: для зарегистрир. пользователей. - Текст : электронный.

1.3. База данных «Электронная библиотека технического ВУЗа (ЭБС «Консультант студента») : электронно-библиотечная система : сайт / ООО «Политехресурс». – Москва, [2023]. – URL: https://www.studentlibrary.ru/cgi-bin/mb4x. – Режим доступа: для зарегистрир. пользователей. – Текст : электронный.

1.4. ЭБС Лань : электронно-библиотечная система : сайт / ООО ЭБС «Лань». – Санкт-Петербург, [2023]. – URL: https://e.lanbook.com. – Режим доступа: для зарегистрир. пользователей. – Текст : электронный.

1.5. ЭБС Znanium.com : электронно-библиотечная система : сайт / ООО «Знаниум». - Москва, [2023]. - URL: http://znanium.com . – Режим доступа : для зарегистрир. пользователей. - Текст : электронный.

2. КонсультантПлюс [Электронный ресурс]: справочная правовая система. / ООО «Консультант Плюс» - Электрон. дан. - Москва : КонсультантПлюс, [2023].

3. Базы данных периодических изданий:

3.1. eLIBRARY.RU: научная электронная библиотека : сайт / ООО «Научная Электронная Библиотека». – Москва, [2023]. – URL: http://elibrary.ru. – Режим доступа : для авториз. пользователей. – Текст : электронный

3.2. Электронная библиотека «Издательского дома «Гребенников» (Grebinnikon) : электронная библиотека / ООО ИД «Гребенников». – Москва, [2023]. – URL: https://id2.action-media.ru/Personal/Products. – Режим доступа : для авториз. пользователей. – Текст : электронный.

4. Федеральная государственная информационная система «Национальная электронная библиотека» : электронная библиотека : сайт / ФГБУ РГБ. – Москва, [2023]. – URL: https://нэб.рф. – Режим доступа : для пользователей научной библиотеки. – Текст : электронный.

5. Российское образование : федеральный портал / учредитель ФГАУ «ФИЦТО». – URL: http://www.edu.ru. – Текст : электронный.

6. Электронная библиотечная система УлГУ : модуль «Электронная библиотека» АБИС Мега-ПРО / ООО «Дата Экспресс». – URL: http://lib.ulsu.ru/MegaPro/Web. – Режим доступа : для пользователей научной библиотеки. – Текст : электронный.

СОГЛАСОВАНО:

Инженер ведущий	Изуленко Ю.В.		01.06.2023
Должность сотрудника УИТиТ	ФИО	подпись	дата

12 МАТЕРИАЛЬНО-ТЕХНИЧЕСКОЕ ОБЕСПЕЧЕНИЕ ДИСЦИПЛИНЫ

Помещения для проведения занятий лекционного типа, занятий практического/семинарского типа, групповых и индивидуальных консультаций, текущего контроля и промежуточной аттестации представляют собой учебные аудитории, оснащенные оборудованием и техническими средствами обучения. Аудитории укомплектованы специализированной мебелью и техническими средствами обучения.

Помещения для самостоятельной работы обучающихся оснащены компьютерной техникой с возможностью подключения к сети "Интернет" и обеспечением доступа к электронной информационно-образовательной среде Университета.

13 СПЕЦИАЛЬНЫЕ УСЛОВИЯ ДЛЯ ОБУЧАЮЩИХСЯ С ОГРАНИЧЕННЫМИ ВОЗМОЖНОСТЯМИ ЗДОРОВЬЯ

В случае необходимости, обучающимся из числа лиц с ограниченными возможностями здоровья (по заявлению обучающегося) могут предлагаться одни из следующих вариантов восприятия информации с учетом их индивидуальных психофизических особенностей:

– для лиц с нарушениями зрения: в печатной форме увеличенным шрифтом; в форме электронного документа; в форме аудиофайла (перевод учебных материалов в аудиоформат); в печатной форме на языке Брайля; индивидуальные консультации с привлечением тифлосурдопереводчика; индивидуальные задания и консультации;

– для лиц с нарушениями слуха: в печатной форме; в форме электронного документа; видеоматериалы с субтитрами; индивидуальные консультации с привлечением сурдопереводчика; индивидуальные задания и консультации;

– для лиц с нарушениями опорно-двигательного аппарата: в печатной форме; в форме электронного документа; в форме аудиофайла; индивидуальные задания и консультации.

В случае необходимости использования в учебном процессе частично/исключительно дистанционных образовательных технологий, организация работы ППС с обучающимися с ОВЗ и инвалидами предусматривается в электронной информационно-образовательной среде с учетом их индивидуальных психофизических особенностей

Разработчик _____ доцент кафедры ЦЭ А.Г. Сковиков

07.06.23