МИНИСТЕРСТВО ОБРАЗОВАНИЯ И НАУКИ РОССИЙСКОЙ ФЕДЕРАЦИИ
Федеральное государственное автономное образовательное учреждение высшего образования
«Дальневосточный федеральный университет»
(ДВФУ)

ШКОЛА ЕСТЕСТВЕННЫХ НАУК

«СОГЛАСОВАНО»
Руководитель образовательной программы

_____ А.С. Величко

«30» июня 2016 г.

«УТВЕРЖДАЮ»
Врио заведующего кафедрой
математических методов в экономике

_____ А.С. Величко

«30» июня 2016 г.

РАБОЧАЯ ПРОГРАММА УЧЕБНОЙ ДИСЦИПЛИНЫ

Машинное обучение и анализ данных

Направление подготовки 01.03.04 Прикладная математика

Форма подготовки очная

курс 3 семестр 5, 6
лекции 36 час.
практические занятия 0 час.
лабораторные работы 54 час.
в том числе с использованием МАО лек. 36 час. / пр. 0 час. / лаб. 54 час.
всего часов аудиторной нагрузки 90 час.
в том числе с использованием МАО 90 час.
самостоятельная работа 90 час.
в том числе на подготовку к экзамену 27 час.
контрольные работы (количество) 6
курсовая работа / курсовой проект не предусмотрены
зачет 5 семестр
экзамен 6 семестр

Рабочая программа учебной дисциплины (РПУД) составлена в соответствии с требованиями образовательного стандарта по направлению 01.03.04 «Прикладная математика», самостоятельно устанавливаемого ДВФУ, утвержденного приказом ректора от 18.02.2016 № 12-13-235

Рабочая программа обсуждена на заседании кафедры математических методов в экономике, протокол № 12 от «30» июня 2016 г.

Врио заведующего кафедрой математических методов в экономике, к.ф.-м.н., доцент А.С. Величко

Составитель:
старший преподаватель кафедры математических методов в экономике Е.А. Воронцова

АННОТАЦИЯ

Дисциплина «Машинное обучение и анализ данных» предназначена для студентов направления подготовки 01.03.04 «Прикладная математика».

Общая трудоемкость освоения дисциплины составляет 5 зачётных единиц (180 часов). Дисциплина реализуется на 3 курсе в 5-м и в 6-м семестрах. Дисциплина входит в обязательные дисциплины базовой части блока «Дисциплины (модули)».

Особенности построения курса: лекции (36 часов), лабораторные работы (54 часа), самостоятельная работа (63 часа), подготовка к экзамену (27 часов).

Основная тематика курса определяется потребностями в базовых теоретических и прикладных знаниях студентов в предметной области. Содержание практических занятий и лабораторных работ призвано сформировать у студентов полноценное и единое представление о предмете.

Цели освоения дисциплины: знакомство студентов с методами решения задач с помощью машинного обучения, изучение принципов функционирования систем машинного обучения, в том числе нейронных сетей, получение практических навыков по написанию программ на языке программирования Python; изучение видов искуственных нейронных сетей; получение практических навыков по графической компьютерной визуализации научных данных и их предварительной обработки, в том числе с помощью методов машинного обучения. Основным языком для практических занятий и лабораторных работ выбран язык программирования Python как наиболее современный язык программирования, для которого существуют все необходимые библиотеки машинного обучения.

Курс «Машинное обучение и анализ данных» охватывает следующие разделы: теория и практика машинного обучения, язык программирования Python, нейронные сети, визуализация данных.

Задачи:

- развитие способности осваивать современные технологии программирования на языке программирования Python;

- развитие способность демонстрировать знание основ организации систем машинного обучения и построения искусственных нейронных сетей;
- развитие способности разрабатывать программы, использующие возможности современных библиотек машинного обучения;
- развитие способности отлаживать и тестировать прикладное программное обеспечение;
- развитие способность настраивать, тестировать и осуществлять проверку вычислительной техники и программных средств;
- развитие готовности настраивать, тестировать и осуществлять проверку вычислительной техники и программных средств; .
- развитие способности демонстрировать знания современных языков программирования;
- развитие готовности демонстрировать знания современных языков программирования.

Для успешного изучения дисциплины «Машинное обучение и анализ данных» у обучающихся должны быть сформированы следующие предварительные компетенции:

- способность выявить сущность проблем, возникающих в ходе профессиональной деятельности, готовность использовать для их решения соответствующий аппарат;
- способность использовать для работы операционную систему ЭВМ Microsoft Windows, информационно-телекоммуникационную сеть "Интернет"
- знание одного из современных языков программирования.

Уровень подачи материала курса в достаточной степени опирается на следующие предметы, изученные студентами на 1 и 2 курсах: программные и аппаратные средства информатики, программирование для ЭВМ, модели и методы прикладной математики, теория вероятности и математическая статистика, инструментальные средства обработки и управления данными.

В результате изучения данной дисциплины у обучающихся формируются следующие общепрофессиональные, профессиональные компетенции (элементы компетенций).

Код и формулировка компетенции	**Этапы формирования компетенции**	
ОПК-2 - способность использовать современные математические методы и современные прикладные программные средства и осваивать современные технологии программирования	Знает	современные технологии программирования на языке Python; современные математические методы машинного обучения
	Умеет	разрабатывать программные продукты с помощью современных технологий программирования
	Владеет	эффективными методами разработки программных продуктов с помощью современных технологий программирования
ПК-2 - способность и готовность настраивать, тестировать и осуществлять проверку вычислительной техники и программных средств	Знает	способы тестирования программных средств, в том числе, разработанных самостоятельно
	Умеет	находить и исправлять ошибки в тестируемых программных продуктах
	Владеет	навыками настройки вычислительной техники для работы в интегрированной среде разработки, навыками тестирования и проверки программных продуктов
ПК-3 – способность и готовность демонстрировать знания современных языков программирования, операционных систем, офисных приложений, информационно-телекоммуникационной сети "Интернет", способов и	Знает	принципы работы современных операционных систем, современные языки программирования (в частности, язык Python и его библиотеки NumPy, SciPy, Mathplotlib, Pandas, Sklearn)
	Умеет	применять знания современных языков программирования и

механизмов управления данными, принципов организации, состава и схемы работы операционных систем		принципов работы современных систем машинного обучения для разработки программных продуктов
	Владеет	навыками разработки программ и реализации методы обработки научных данных на современных языках программирования

Для формирования вышеуказанных компетенций в рамках дисциплины «Машинное обучение и анализ данных» применяются следующие методы активного/интерактивного обучения: лекция-беседа, работа в малых группах, занятие-дискуссия, обсуждение и разрешение проблем при создании программного обеспечения с использованием компьютерных технологий.

I. СТРУКТУРА И СОДЕРЖАНИЕ ТЕОРЕТИЧЕСКОЙ ЧАСТИ КУРСА

Лекции (36 часов, в том числе 36 часов с использованием методов активного обучения)

МОДУЛЬ 1 (18 ЧАСОВ, В ТОМ ЧИСЛЕ 18 ЧАСОВ С ИСПОЛЬЗОВАНИЕМ МЕТОДОВ АКТИВНОГО ОБУЧЕНИЯ)

Тема 1. Язык программирования Python и компьютерная графика (12 часов)

Интерпретируемые языки программирования. Основные конструкции языка программирования Python. Синтаксис, семантика. Функции. Списки, словари, кортежи, множества, их особенности, правила использования и отличия. Библиотека для работы с массивами NumPy. Библиотека для научных вычислений SciPy. Библиотека для обработки данных Pandas.

Тема 2. Визуализация данных (6 часов)

Компьютерная графическая визуализация данных в Mathplotlib.

МОДУЛЬ 2 (18 ЧАСОВ, В ТОМ ЧИСЛЕ 18 ЧАСОВ С ИСПОЛЬЗОВАНИЕМ МЕТОДОВ АКТИВНОГО ОБУЧЕНИЯ)

Тема 1. Обработка данных с помощью методов машинного обучения (10 часов) (10 часов)

Функции ошибки и регуляризации. Распознавание спама в почтовых сообщениях. Предобработка данных в Pandas. Решающие деревья. Выбор метрики. Нормализация признаков. Метод опорных векторов. Анализ текстов. Логистическая регрессия.

Тема 2. Нейронные сети (8 часов)

Искусственный интеллект и машинное обучение. Как работает перцептрон. Функции активации. Многослойный перцептрон. Метод моментов. Свёрточные нейронные сети и автокодировщики.

II. СТРУКТУРА И СОДЕРЖАНИЕ ПРАКТИЧЕСКОЙ ЧАСТИ КУРСА

Лабораторные работы (54 часа, в том числе 54 часа с использованием методов активного обучения)

Модуль 1 (36 часов, в том числе 36 часов с использованием методов активного обучения)

Лабораторная работа № 1. Язык программирования Python. Задача о пенсионных накоплениях. Задача «Диофантовы уравнения и Макдональдс» (4 часа)

Лабораторная работа № 2. Адаптивный тренажёр Python (8 часов)

Лабораторная работа № 3. Словари, кортежи, списки, множества (4 часа)

Лабораторная работа № 4. Работы с библиотекой NumPy (4 часа)

Лабораторная работа № 5. Работа с библиотекой SciPy (4 часа)

Лабораторная работа № 6. Компьютерная графика и Mathplotlib (4 часа)

Лабораторная работа № 7. Защита проектов — приложений для анализа данных, использующих методы машинного обучения (8 часов)

Модуль 2 (18 часов, в том числе 18 часов с использованием методов активного обучения)

Лабораторная работа № 1. Предобработка данных в Pandas и важность признаков (4 часа, в том числе 4 часа с использованием методов активного обучения)

Лабораторная работа № 2. Выбор числа соседей (2 часа, в том числе 2 часа с использованием методов активного обучения)

Лабораторная работа № 3. Выбор метрики (2 часа, в том числе 2 часа с использованием методов активного обучения)

Лабораторная работа № 4. Нормализация признаков (2 часа, в том числе 8 часов с использованием методов активного обучения)

Лабораторная работа № 5. Метод опорных векторов (2 часа, в том числе 2 часа с использованием методов активного обучения)

Лабораторная работа № 6. Логистическая регрессия (4 часа, в том числе 4 часа с использованием методов активного обучения)

Лабораторная работа № 7. Искусственные нейронные сети. Построение и анализ (6 часов, в том числе 6 часов с использованием методов активного обучения)

III. УЧЕБНО-МЕТОДИЧЕСКОЕ ОБЕСПЕЧЕНИЕ САМОСТОЯТЕЛЬНОЙ РАБОТЫ ОБУЧАЮЩИХСЯ

Учебно-методическое обеспечение самостоятельной работы обучающихся по дисциплине «Машинное обучение и анализ данных» представлено в Приложении 1 и включает в себя:

план-график выполнения самостоятельной работы по дисциплине, в том числе примерные нормы времени на выполнение по каждому заданию;

характеристика заданий для самостоятельной работы обучающихся и методические рекомендации по их выполнению;

требования к представлению и оформлению результатов самостоятельной работы;

критерии оценки выполнения самостоятельной работы.

IV. КОНТРОЛЬ ДОСТИЖЕНИЯ ЦЕЛЕЙ КУРСА

Контролируемые разделы дисциплины, этапы формирования компетенций, виды оценочных средств, зачетно-экзаменационные материалы, комплекты оценочных средств для текущей аттестации, описание показателей и критериев оценивания компетенций на различных этапах их формирования, методические материалы, определяющие процедуры оценивания знаний, умений и навыков и (или) опыта деятельности, а также критерии и показатели, необходимые для оценки знаний, умений, навыков и характеризующие этапы формирования компетенций в процессе освоения образовательной программы, представлены в Приложении 2.

V. СПИСОК УЧЕБНОЙ ЛИТЕРАТУРЫ И ИНФОРМАЦИОННО-МЕТОДИЧЕСКОЕ ОБЕСПЕЧЕНИЕ ДИСЦИПЛИНЫ

Основная литература

(электронные и печатные издания)

1. Маккинли У. Python и анализ данных [Электронный ресурс]. — М.: ДМК Пресс, 2015. — 482 с. — Режим доступа: http://e.lanbook.com/books/element.php?pl1_id=73074.

2. Саммерфилд М. Python на практике [Электронный ресурс]. — М.: ДМК Пресс, 2016. — 338 с. — Режим доступа: http://e.lanbook.com/books/element.php?pl1_id=66480.

3. Сузи Р.А. Язык программирования Python [Электронный ресурс]. М.: Интуит, 2007. — 328 с. — Режим доступа: http://www.iprbookshop.ru/22450.

Дополнительная литература

(печатные и электронные издания)

1. Кручинин А.Ю. Операционные системы [Электронный ресурс] : учебное пособие. — Оренбург: изд-во «Бибком», 2009. — Режим доступа: http://www.iprbookshop.ru/30115.

2. Одиноков, В. В. Операционные системы и сети [Электронный ресурс] : учебное пособие. - Томск: Томский государственный университет систем управления и радиоэлектроники, 2007. - 391 с. — Режим доступа: http://www.iprbookshop.ru/13951.

3. Перемитина, Т. О. Компьютерная графика [Электронный ресурс] : учебное пособие. - Томск: изд-во «Эль Контент», Томский государственный университет систем управления и радиоэлектроники, 2012. - 144 с. — Режим доступа: http://www.iprbookshop.ru/13940.

Перечень ресурсов информационно-телекоммуникационной сети «Интернет»

1. Linux. URL: http://www.linux.org/
2. The UNIX System. URL: http://www.unix.org/
3. Чем отличается устройство Windows - Microsoft Windows. URL: http://windows.microsoft.com/ru-ru/windows-8/compare

4. Онлайн-курс «Математика и Python для анализа данных». МФТИ, Академия Яндекса. https://www.coursera.org/learn/mathematics-and-python
5. Онлайн-курс «Введение в машинное обучение». Высшая школа экономики. https://www.coursera.org/learn/vvedenie-mashinnoe-obuchenie

Перечень дополнительных информационно-методических материалов

1. Таненбаум Э. Современные операционные системы. 3-е издание. СПб.: Питер, 2010.
2. Илюшечкин В.М. Операционные системы: учебное пособие. М.: Бином, Лаборатория знаний, 2011.
3. Бендел Д., Нейпир Р. Использование Linux. 6-е издание. М.: Вильямс, 2003.
4. Стивенс У. UNIX: взаимодействие процессов. СПб.: Питер, 2003.
5. Брейман А.Д. Сети ЭВМ и телекоммуникации. Глобальные сети. Учебное пособие. М.: МГУПИ, 2006.
6. Алиев Т.И. Сети ЭВМ и телекоммуникации. СПб.: СПБГУ ИТМО, 2011.
7. Silbershatz A., Galvin P. B., Gagne G. Operating System Concepts. 9th Edition International Student Version edition. John Wiley & Sons, 2013.
8. Цикритзис Д., Бернстайн Ф. Операционные системы. М.: Мир, 1977.

Перечень информационных технологий и программного обеспечения

При осуществлении образовательного процесса по дисциплине используется набор библиотек языка Python (Anaconda одной из последних версий), IPython Notebook. Обязателен компьютерный класс с доступом в Интернет.

VI. МЕТОДИЧЕСКИЕ УКАЗАНИЯ ПО ОСВОЕНИЮ ДИСЦИПЛИНЫ

Рекомендации по планированию и организации времени, отведенного на изучение дисциплины, описание последовательности действий обучающихся

Освоение дисциплины следует начинать с изучения рабочей учебной программы, которая содержит основные требования к знаниям, умениям и навыкам. Обязательно следует учитывать рекомендации преподавателя, данные в ходе установочных занятий. Затем – приступать к изучению отдельных разделов и тем в порядке, предусмотренном программой.

Получив представление об основном содержании раздела, темы, необходимо изучить материал с помощью рекомендуемой основной литературы. Целесообразно составить краткий конспект или схему, отображающую смысл и связи основных понятий данного раздела и включенных в него тем. Обязательно следует записывать возникшие вопросы, на которые не удалось ответить самостоятельно.

Подготовку к началу обучения включает несколько необходимых пунктов:

1) Необходимо создать для себя рациональный и эмоционально достаточный уровень мотивации к последовательному и планомерному изучению дисциплины.

2) Необходимо изучить список рекомендованной основной и дополнительной литературы и убедиться в её наличии у себя дома или в библиотеке в бумажном или электронном виде.

3) Необходимо иметь «под рукой» специальные и универсальные словари, справочники и энциклопедии, для того, чтобы постоянно уточнять значения используемых терминов и понятий. Пользование словарями и справочниками необходимо сделать привычкой. Опыт показывает, что неудовлетворительное усвоение предмета зачастую коренится в неточном, смутном или неправильном понимании и употреблении понятийного аппарата учебной дисциплины.

4) Желательно в самом начале периода обучения возможно тщательнее спланировать время, отводимое на работу с источниками и литературой по дисциплине, представить этот план в наглядной форме (график работы с датами) и в дальнейшем его придерживаться, не допуская срывов графика индивидуальной работы и «аврала» в предсессионный период. Пренебрежение этим пунктом приводит к переутомлению и резкому снижению качества усвоения учебного материала.

Рекомендации по работе с литературой

1) Всю учебную литературу желательно изучать «под конспект». Чтение литературы, не сопровождаемое конспектированием, даже пусть самым кратким – бесполезная работа. Цель написания конспекта по дисциплине – сформировать навыки по поиску, отбору, анализу и формулированию учебного материала. Эти навыки обязательны для любого специалиста с высшим образованием независимо от выбранной специальности.

2) Написание конспекта должно быть творческим – нужно не переписывать текст из источников, но пытаться кратко излагать своими словами содержание ответа, при этом максимально структурируя конспект, используя символы и условные обозначения. Копирование и «заучивание» неосмысленного текста трудоемко и по большому счету не имеет большой познавательной и практической ценности.

3) При написании конспекта используется тетрадь, поля в которой обязательны. Страницы нумеруются, каждый новый вопрос начинается с нового листа, для каждого экзаменационного вопроса отводится 1-2 страницы конспекта. На полях размещается вся вспомогательная информация – ссылки, вопросы, условные обозначения и т.д.

4) В итоге данной работы «идеальным» является полный конспект по программе дисциплины, с выделенными определениями, узловыми пунктами, примерами, неясными моментами, проставленными на полях вопросами.

5) При работе над конспектом обязательно выявляются и отмечаются трудные для самостоятельного изучения вопросы, с которыми уместно обратиться к преподавателю при посещении установочных лекций и консультаций, либо в индивидуальном порядке.

6) При чтении учебной и научной литературы всегда следить за точным и полным пониманием значения терминов и содержания понятий, используемых в тексте. Всегда следует уточнять значения по словарям или энциклопедиям, при необходимости записывать.

7) При написании учебного конспекта обязательно указывать все прорабатываемые источники, автор, название, дата и место издания, с указанием использованных страниц.

Подготовка к промежуточной аттестации по дисциплине: экзамену (зачету)

К аттестации допускаются студенты, которые систематически в течение всего семестра посещали и работали на занятиях и показали уверенные знания в ходе выполнении практических заданий и лабораторных работ.

Непосредственная подготовка к аттестации осуществляется по вопросам, представленным в рабочей учебной программе. Тщательно изучите формулировку каждого вопроса, вникните в его суть, составьте план ответа. Обычно план включает в себя:

— определение сущности рассматриваемого вопроса, основных положений, утверждений, определение необходимости их доказательства;

— запись обозначений, формул, необходимых для полного раскрытия вопроса;

— графический материал (таблицы, рисунки, графики), необходимые для раскрытия сущности вопроса;

— роль и значение рассматриваемого материала для практической деятельности, примеры использования в практической деятельности.

VII. МАТЕРИАЛЬНО-ТЕХНИЧЕСКОЕ ОБЕСПЕЧЕНИЕ ДИСЦИПЛИНЫ

Для осуществления образовательного процесса по дисциплине необходим компьютерный класс с персональными компьютерами с доступом в сеть «Интернет» и установленным пакетом программ Anaconda (continuum.io) – полный набор библиотек для работы с языком программирования Python. В компьютерном классе должно быть оборудование мультимедийного типа (мультимедийный проектор, настенный экран) и пластиковая доска.

Приложение 1

МИНИСТЕРСТВО ОБРАЗОВАНИЯ И НАУКИ РОССИЙСКОЙ ФЕДЕРАЦИИ
Федеральное государственное автономное образовательное учреждение высшего образования
«Дальневосточный федеральный университет»
(ДВФУ)

ШКОЛА ЕСТЕСТВЕННЫХ НАУК

УЧЕБНО-МЕТОДИЧЕСКОЕ ОБЕСПЕЧЕНИЕ САМОСТОЯТЕЛЬНОЙ РАБОТЫ ОБУЧАЮЩИХСЯ

по дисциплине «Машинное обучение и анализ данных»
Направление подготовки 01.03.04 Прикладная математика

Форма подготовки очная

Владивосток
2016

План-график выполнения самостоятельной работы по дисциплине

№ п/п	Дата/сроки выполнения	Вид самостоятельной работы	Примерные нормы времени на выполнение	Форма контроля
1	Модуль 1. 12 неделя	Повторение теоретического и практического материала дисциплины, заслушиваемого и конспектируемого в ходе аудиторных занятий; изучение основной и дополнительной литературы, указанной в рабочей учебной программе дисциплины, самоконтроль ответов на основные проблемные вопросы по темам лекций; самостоятельный разбор заданий и задач, решаемых на практических занятиях; самостоятельный повтор действий, осуществляемых в ходе выполнения практических заданий, в том числе при работе со специальным программным обеспечением	10 часов	Собеседование, вопросы 1-7; защита проекта, темы 1-6
2	Модуль 2. 15 неделя	Повторение теоретического и практического материала дисциплины, заслушиваемого и конспектируемого в ходе аудиторных занятий; изучение основной и дополнительной литературы, указанной в рабочей учебной программе дисциплины, самоконтроль ответов на основные проблемные вопросы по темам лекций; самостоятельный разбор заданий и задач, решаемых на практических занятиях; самостоятельный повтор действий, осуществляемых в ходе выполнения практических заданий, в том числе при работе со	8 часов	Собеседование, вопрос 13, защита проекта, темы 7-8

		специальным программным обеспечением		
3	Модуль 2. 12 неделя	Повторение теоретического и практического материала дисциплины, заслушиваемого и конспектируемого в ходе аудиторных занятий; изучение основной и дополнительной литературы, указанной в рабочей учебной программе дисциплины, самоконтроль ответов на основные проблемные вопросы по темам лекций; самостоятельный повтор действий, осуществляемых в ходе выполнения лабораторных работ, в том числе при работе со специальным программным обеспечением	25 часов	Собеседование, вопросы 8-16; защита проекта, темы 9-17
4	Модуль 2. 16 неделя	Повторение теоретического и практического материала дисциплины, заслушиваемого и конспектируемого в ходе аудиторных занятий; изучение основной и дополнительной литературы, указанной в рабочей учебной программе дисциплины, самоконтроль ответов на основные проблемные вопросы по темам лекций	20 часов	Собеседование, вопросы 17-20; защита проектов, темы 18-20
5	Модуль 2. Сессия	Подготовка к экзамену	27 часов	Экзамен

Характеристика заданий для самостоятельной работы обучающихся и методические рекомендации по их выполнению

Задания и методические рекомендации по теме «Язык программирования Python»

Источник: Маккинли У. Python и анализ данных. — М.: ДМК Пресс, 2015.

Интерпретатор Python

Python – *интерпретируемый* язык. Интерпретатор Python исполняет программу по одному предложению за раз. Стандартный интерактивный интерпретатор Python запускается из командной строки командой `python`:

```
$ python
Python 2.7.2 (default, Oct 4 2011, 20:06:09)
[GCC 4.6.1] on linux2
Type "help", "copyright", "credits" or "license" for more information.
>>> a = 5
>>> print a
5
```

Строка `>>>` – это приглашение к вводу выражения. Для выхода из интерпретатора Python и возврата в командную строку нужно либо ввести команду `exit()`, либо нажать `Ctrl-D`.

Для выполнения Python-программы нужно просто набрать команду `python`, указав в качестве первого аргумента имя файла с расширением `.py`. Допустим, вы создали файл `hello_world.py` с таким содержимым:

```
print 'Hello world'
```

Чтобы выполнить его, достаточно ввести следующую команду:

```
$ python hello_world.py
Hello world
```

Многие программисты выполняют свой код на Python именно таким образом, но в мире *научных* приложений принято использовать IPython, улучшенный и дополненный интерпретатор Python. Ему посвящена вся глава 3. С помощью команды `%run` IPython исполняет код в указанном файле в том же процессе, что позволяет интерактивно изучать результаты по завершении выполнения.

```
$ ipython
Python 2.7.2 |EPD 7.1-2 (64-bit)| (default, Jul 3 2011, 15:17:51)
Type "copyright", "credits" or "license" for more information.
```

```
IPython 0.12 -- An enhanced Interactive Python.
?         -> Introduction and overview of IPython's features.
%quickref -> Quick reference.
help      -> Python's own help system.
object?   -> Details about 'object', use 'object??' for extra details.

In [1]: %run hello_world.py
```

```
Hello world

In [2]:
```

По умолчанию приглашение IPython содержит не стандартную строку `>>>`, а строку вида `In [2]:`, включающую порядковый номер предложения.

Источник: Онлайн-курс на Курсере «Математика и Python для анализа данных». МФТИ, Академия Яндекса. https://www.coursera.org/learn/mathematics-and-python

Задание 1

1. Перейдите на сайт continuum.io по ссылке https://www.continuum.io/downloads
2. Выберите подходящую Вам операционную систему и перейдите в соответствующий раздел сайта.
3. Следуя инструкциям на сайте, установите Python 3.5.

Задание 2

1. Запустите ipython notebook. Команда запуска может несколько отличаться от команды запуска в инструкции из видео. Например, команда может быть такой: ipython-3.5 notebook
2. Создайте новый файл типа .ipynb
3. Создайте новую ячейку. В ней импортируйте библиотеку numpy (import numpy) и выведите на экран версию библиотеки (numpy.__version__)
4. Создайте новую ячейку. В ней импортируйте библиотеку scipy (import scipy) и выведите на экран версию библиотеки (scipy.__version__)
5. Создайте новую ячейку. В ней импортируйте библиотеку pandas (import pandas) и выведите на экран версию библиотеки (pandas.__version__)
6. Создайте новую ячейку. В ней импортируйте библиотеку matplotlib (import matplotlib) и выведите на экран версию библиотеки (matplotlib.__version__)
7. Сделайте скриншот №1, на котором будет хорошо видно результаты вашей работы, и загрузите его в форму.

Задание 3

1. Запустите ipython notebook. Команда запуска может несколько отличаться от команды запуска в инструкции, например, команда может быть такой ipython-2.7 notebook
2. Создайте новый файл типа .ipynb
3. В файле создайте новую ячейку и измените её тип на Markdown
4. В первой строке созданной ячейки наберите название специализации “Машинное обучение и анализ данных” и сделайте эту строку заголовком уровня 1.
5. В следующей строке созданной ячейки наберите название нашего курса “Математика и Python” и сделайте строку заголовком уровня 2.
6. В третьей строке ячейки наберите текст "Задание 1".
7. Запустите выполнение ячейки.
8. Сделайте скриншот №2, на котором будет хорошо видно результаты вашей работы, и загрузите его в форму.

В результате работы вы установите на компьютер Python и библиотеки, необходимые для дальнейшего прохождения курса.

Вы также научитесь запускать iPython notebook и выполнять в нем простые команды.

Выполнение задания будет проверяться на основе двух скриншотов, которые вы прикрепите к заданию.

Программирование на Python

1. Типы данных в Python

Настало время познакомиться с типами данных в Python. Числовые типы `int` и `float` уже были упомянуты ранее:

```
x = 3.5
print type(x) % OUT: <type 'float'>

y = 4
print type(y) % OUT: <type 'int'>
```

С помощью функции `type` можно узнать код произвольного объекта. Строки, например, могут быть двух типов:

```
print type("abs")  % OUT: <type 'str'>
print type(u"abs") % OUT: <type 'unicode'>
```

Отличия между ними будут подробнее обсуждаться позднее.

Упорядоченный набор значений в Python можно представить с помощью списка:

```
w = [1, 2, 3]
print type(w) % OUT: <type 'list'>
```

Списки в Python очень похожи на знакомые по другим языкам программирования массивы, но, в отличие от массивов, могут содержать элементы разных типов.

Python является языком с утиной типизацией. В языках с динамической типизацией, частным случаем которой является утиная типизация, тип переменной определяется не в момент её объявления, а в момент присваивания значения. Следовательно, в разных участках кода переменная может принимать значения разных типов. В случае утиной типизации границы применимости объекта определяются не столько конкретным типом и иерархией наследования, сколько наличием и отсутствия у него определенных методов и свойств. Термин происходит от шуточного «утиного теста»: «Если нечто выглядит, плавает и крякает как утка, то это, вероятно, и есть утка».

Другой способ представить упорядоченный набор значений состоит в определении кортежа (`tuple`). В Python кортежи записываются в круглых скобках:

```
print type((1, 2, 3)) % OUT: <type 'tuple'>
```

Кортеж относится к так называемым неизменяемым типам данных, чем существенно отличается от списка. Попытка изменить кортеж или добавить в него еще один элемент приведет к возникновению ошибки. Эта особенность позволяет использовать кортежи в качестве ключей словаря.

Неупорядоченный набор значений представим в виде множеств, которые записываются в фигурных скобках:

```
print type({1, 2, 3}) % OUT: <type 'set'>
```

Еще один тип данных, словарь, позволяет хранить в себе пары вида ключ-значение. Создать словарь можно с помощью конструктора `dict()`:

```
e = dict()      % Создан пустой словарь
e['abc'] = 3.5  % Строке-ключу 'abc' поставлено в
                %    соответствие значение 3.5 типа float
```

Ключами в словаре могут быть не только строки. В качестве ключа допустимо использовать, например, числовые значения и кортежи:

```
e[3.5]       = 'abc'
e[(1, 2, 3)] = 4.5
```


Но использование списка в качестве ключа недопустимо, так как список является изменяемым.

Синтаксис для чтения данных из словаря по ключу аналогичен синтаксису для получения данных по индексу списка. Грубо говоря, ключ есть обобщение понятия индекса списка: ключ может принимать значение произвольного хэшируемого типа. К слову, список не является хэшируемым, и именно поэтому не может быть выбран в качестве ключа.

В Python словарь использует хэш-таблицы. Идея заключается в том, чтобы заменить сравнение ключей более быстрым сравнением их хэш-значений. В таком случае важно, чтобы все ключи были хэшируемы, то есть хэш-значение не менялось во время исполнения программы, а равные ключи-объекты имели одинаковое хэш-значение.

Ранее было сказано, что множества являются изменяемыми объектами, а значит не могут выступать в роли ключей. Но в качестве ключей можно использовать неизменяемые множества, которые можно получить используя функцию `frozenset()`:

```
s = frozenset(1,2,3)
e[s] = 3.76
```

Таким образом, к данному моменту были изучены основные типы данных в Python и подробно было рассказано о существенных отличиях изменяемых и неизменяемых типов. На следующем занятии будут рассмотрены основные управляющие конструкции языка Python.

2. Синтаксис языка Python

В Python оператор условного перехода выглядит привычным образом. Например, пусть дан следующий код:

```
if x:
   print "OK"
else:
   print "NOT OK"
```

Если переменная `x` была равна `True`, то исполнится код в теле инструкции `if` и на экран будет выведено OK. В противном случае, например если `x` определена как `False`, исполнится код в теле инструкции `else`. В Python для выделения блоков кода используются отступы: отступы инструкций в пределах одного блока должны совпадать по величине.

Другой пример. Следующий код выводит числа от 0 до 9:

```
for i in range(10):
    print i
```

При этом в `range()` можно указать начальное значение i. Например, вот этот код выводит числа от 1 до 9.

```
for i in range(1,10):
```

```
    print i
```

В данных примерах каждое число выводится на отдельной строке, поскольку `print` по умолчанию начинает новую строчку после вывода требуемого значения. Чтобы все числа выводились в одной строке, необходимо добавить запятую в конец строчки с `print` следующим образом:

```
for i in range(1,10):
    print i,
```

Теперь числа от 1 до 9 выводятся в одной строке через пробел.

На самом деле `range()` это функция, которая возвращает список натуральных чисел в определяемом её аргументами диапазоне:

```
print range(2,5)
OUT: [2,3,4]
```

Использовать `range()` совершенно не обязательно. Циклы в Python перебирают по очереди элементы списка, стоящего после `in`. Например, можно написать так:

```
for i in [2,3,4]:
    print i,
```


В Python 2 кроме функции `range()` существует и функция `xrange()`, которую также можно использовать в цикле:

```
for i in xrange(2,5):
    print i,
OUT: 2, 3, 4
```

Использовать `range()` совершенно не обязательно. Циклы в Python перебирают по очереди элементы списка, стоящего после `in`. Например, можно написать так:

```
for i in [2,3,4]:
    print i,
```


В Python 2 кроме функции `range()` существует и функция `xrange()`, которую также можно использовать в цикле:

```
for i in xrange(2,5):
    print i,
OUT: 2, 3, 4
```

Результат выполнения не отличается от такового при использовании `range()`. Но можно убедиться, что `range` и `xrange` возвращают объекты разных типов:

```
print type(range(2,5))
OUT: list
print type(xrange(2,5))
OUT: iter
```

Функция `xrange` возвращает не список, а так называемый генератор. В то время, как при использовании `range()` сначала создается список, а только потом происходит итерирование по этому списку, при использовании генераторов список никогда не создается и не занимает память. Это особенно актуально, когда необходимо итерировать в очень большом диапазоне значений.

```
print xrange(2,500000000)
OUT:  xrange(2,500000000)
```

В Python существует способ удобно и компактно задавать некоторые списки с помощью так называемого конструктора списка (анг. list comprehension.):

```
w = [ x ** 2 for x in range(1,11) ]
print w
OUT: [ 1, 4, 9, 16, 25, 36, 49, 64, 81, 100 ]
print type(w)
OUT: <type 'list'>
```

В этом примере создан список квадратов чисел от 1 до 10. Если необходимо построить список, например, квадратов только четных чисел из этого диапазона, конструктор допускает использование условия:

```
w = [ x ** 2 for x in range(1,11) if x % 2 == 0]
OUT: [ 4, 16, 36, 64, 100 ]
```

Аналогично выглядит конструктор генераторов:

```
w = ( x ** 2 for x in range(1,11) if x % 2 == 0 )
OUT: ( 4, 16, 36, 64, 100 )
print type(w)
OUT: <type 'generator'>
```

В этом случае весь список не будет храниться в памяти во время работы цикла.

Также в Python доступен цикл **while**, который продолжает выполнение, пока выполнено указанное условие, а также операторы **break**, который досрочно прерывает цикл, и **continue**, который начинает следующий проход цикла, минуя оставшееся тело цикла. Например:

```
s = 0
while True: % Это условие всегда выполняется
    s += 1
    if s % 2 ==0: % Через раз
        print "Continue" % вместо s будет выведено "continue"
        continue % и код ниже в таком случае не будет исполнен в этой итерации
    print s
    if s > 10: % С помощью этой конструкции
        break  % <<бесконечный цикл досрочно прерывается>>
```

Важно уметь определять свои функции. Например, следующий код представляет собой альтернативную реализацию уже известной функции range:


```
def myrange(a,b):
    res = [ ]
    s = a
    while s!=b:
      res.append(s)
      s+=1
    return res
```

Этот пример демонстрирует синтаксис для создания функций. На самом деле при создании функции необходимо обработать особые случаи, но это не является текущей целью.

Функции в Python являются объектами первого класса: их можно передавать в качестве аргументов, присваивать их переменным и так далее. Например, функция `map()` позволяет обрабатывать одну или несколько последовательностей с помощью переданной в качестве аргумента функции:

```
def sq(x):
    return x ** 2

print map(sq, range(10))
OUT: [ 1, 4, 9, 16, 25, 36, 49, 64, 81]
```

Функция в Python может быть определена как с помощью оператора `def`, так и с помощью лямбда-выражения. Следующие операторы эквивалентны:

```
def sq(x):
    return x ** 2

sq = lambda x: x**2
```

Задание 4

Реализовать клиент-серверное приложение с использованием API GoogleMaps, позволяющее пользователю искать географические данные по введенному названию. Для этого воспользоваться сетевой библиотекой языка Python urllib.

Задания и методические рекомендации по теме «Компьютерная графическая визуализация данных в Mathplotlib»

1. Изучить рекомендуемую литературу по данной теме.
2. Изучить библиотеку mathplotlib для графической визуализации данных.
3. Применить методы из библиотеки mathplotlib для обработки и анализа данных.
4. Ознакомиться с другими средствами визуализации (см. ниже).

Источник: Маккинли У. Python и анализ данных. — М.: ДМК Пресс, 2015.

Инструментальная экосистема визуализации для Python

Как обычно бывает в проектах с открытым исходным кодом, в средствах создания графики для Python нехватки не ощущается (их слишком много, чтобы все перечислить). Помимо открытого кода, существуют также многочисленные коммерческие библиотеки с интерфейсом к Python.

В этой главе и в книге в целом я использую, в основном пакет matplotlib, потому что это наиболее популярное средство построения графиков в Python. Но хотя matplotlib является важной частью научной экосистемы Python, у него обнаруживается немало недостатков, когда дело доходит до создания и отображения статистических графиков. Пользователям MATLAB пакет matplotlib покажется знакомым, тогда как пользователи R (особенно работающие с великолепными пакетами `ggplot2` и `trellis`), наверное, испытают разочарование (по крайней мере, на момент написания этой книги). С помощью matplotlib можно создать отличные графики для отображения в веб, но для этого обычно приходится прикладывать немало усилий, потому что библиотека проектировалась для полиграфических изданий. Но если не обращать внимания на эстетические нюансы, то для большинства задач этого достаточно. В pandas я, наряду с другими разработчиками, стремился предложить пользователям удобный интерфейс, позволяющий упростить построение большинства графиков, применяемых в анализе данных.

Но существуют и активно используются и другие средства визуализации. Ниже я перечислю некоторых из них и призываю вас самостоятельно исследовать экосистему.

Chaco

Пакет Chaco (*http://code.enthought.com/chaco/*), разработанный компанией Enthought, представляет собой инструментарий для построения графиков. Он

годится как для рисования статических графиков, так и для интерактивной визуализации. Особенно хорош он, когда нужно выразить сложные визуализации, характеризуемые наличием взаимозависимостей между данными. По сравнению с matplotlib Chaco гораздо лучше поддерживает взаимодействие с элементами графика, а отрисовка производится очень быстро, так что этот пакет будет хорошим выбором для построения интерактивных приложений с графическим интерфейсом.

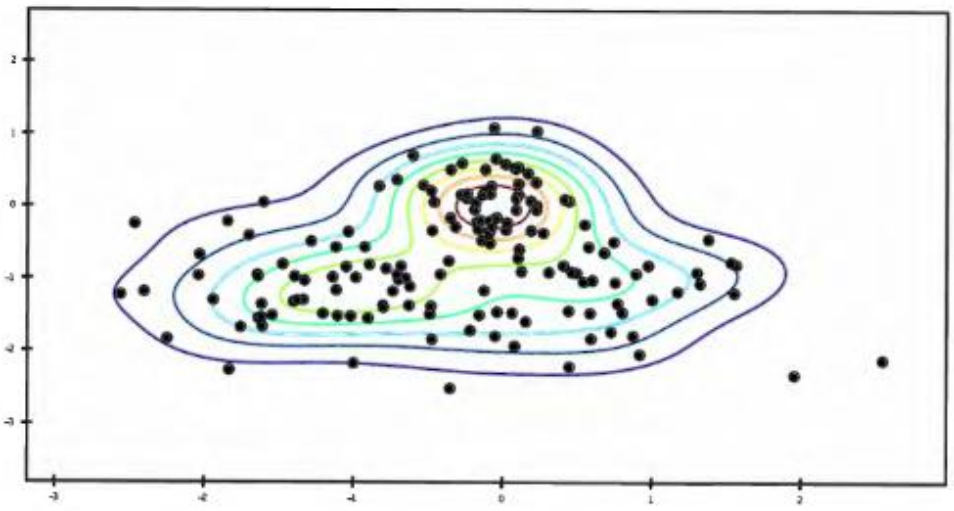

Рис. 8.26. Пример графика, построенного Chaco

mayavi

Проект mayavi, разработанный Прабху Рамачандраном (Prabhu Ramachandran) Гаэлем Вароко (Gaël Varoquaux) и другими, – это библиотека трехмерной графики, построенная поверх написанной на C++ библиотеки VTK с открытым исходным кодом. mayavi, как и matplotlib, интегрирована с IPython, поэтому с ней легко работать интерактивно. Графики можно панорамировать, поворачивать и масштабировать с помощью мыши и клавиатуры. Я использовал mayavi для создания од-

ной иллюстрации укладывания в главе 12. И хотя я не привожу здесь свой код работы с mayavi, в сети достаточно документации и примеров. Я полагаю, что во многих случаях это неплохая альтернатива таким технологиям, как WebGL, хотя интерактивными графиками труднее поделиться.

Прочие пакеты

Разумеется, для Python хватает и других библиотек и приложений для визуализации: PyQwt, Veusz, gnuplot-py, biggles и т. д. Я видел, как удачное применение PyQwt в графических приложениях, написанных с помощью каркаса Qt и надстройки PyQt над ним. Многие библиотеки продолжают активно развиваться (некоторые являются частью гораздо более крупных приложений), но в последние годы наблюдается общая тенденция к дрейфу в сторону веб-технологий и отхода от графики для настольных компьютеров. Я еще скажу об этом несколько слов в следующем разделе.

Будущее средств визуализации

Похоже, будущее за веб-технологиями (на основе JavaScript), и от этого никуда не деться. Без сомнения, за прошедшие годы вы встречались с самыми разными видами статических и интерактивных визуализаций, написанных на Flash или JavaScript. Постоянно появляются все новые и новые инструменты (например, d3.js и многочисленные отпочковавшиеся от него проекты). Напротив, разработка средств визуализации на платформах, отличных от веб, в последние годы резко замедлилась. Это относится как к Python, так и к другим средам для анализа данных

Требования к представлению и оформлению результатов самостоятельной работы

Самостоятельная работа включает в себя повторение теоретического и практического материала дисциплины, заслушиваемого и конспектируемого в ходе аудиторных занятий; изучение основной и дополнительной литературы, указанной в рабочей учебной программе дисциплины, самоконтроль ответов на основные проблемные вопросы по темам занятий; самостоятельный повтор действий, осуществляемых в ходе выполнения лабораторных работ и практических занятий, в том числе при работе со специальным программным обеспечением — средой разработки для языка программирования Python.

Результаты самостоятельной работы представляются и оформляются в виде ответов на основные положения теоретического и практического материала дисциплины по темам; работающего исходного кода результата решения практических заданий и задач; собственных действий, осуществляемых в ходе выполнения лабораторных работ.

Критерии оценки выполнения самостоятельной работы

Общие критерии оценки выполнения самостоятельной работы – правильность ответов на вопросы по темам теоретической части дисциплины, верность получаемых ответов в ходе решения практических заданий и задач, достижение правильного результата при осуществлении собственных действий по лабораторным работам.

Оценивание знаний в форме собеседования проводится по критериям:

- логичность изложения, знание и понимание основных теоретических и практических положений и концепций по теме;
- владение теоретическими и практическими приёмами написания, отладки и тестирования программ на языке программирования Python.

Оценивание знаний в форме защиты проекта проводится по критериям:

- завершенность и полнота выполненных заданий в рамках данного проекта;
- владение методами и приемами решения конкретных задач и самостоятельность использования специализированного программного обеспечения;
- качество оформления отчёта в соответствии с правилами и стандартами оформления документации разработанных программ.

Приложение 2

МИНИСТЕРСТВО ОБРАЗОВАНИЯ И НАУКИ РОССИЙСКОЙ ФЕДЕРАЦИИ
Федеральное государственное автономное образовательное учреждение высшего образования
«Дальневосточный федеральный университет»
(ДВФУ)

ШКОЛА ЕСТЕСТВЕННЫХ НАУК

ФОНД ОЦЕНОЧНЫХ СРЕДСТВ

по дисциплине «Машинное обучение и анализ данных»
Направление подготовки 01.03.04 Прикладная математика

Форма подготовки очная

Владивосток
2016

Паспорт
фонда оценочных средств
по дисциплине «Машинное обучение и анализ данных»

Код и формулировка компетенции	Этапы формирования компетенции	
ОПК-2 - способность использовать современные математические методы и современные прикладные программные средства и осваивать современные технологии программирования	Знает	современные технологии программирования на языке Python; современные математические методы машинного обучения
	Умеет	разрабатывать программные продукты с помощью современных технологий программирования
	Владеет	эффективными методами разработки программных продуктов с помощью современных технологий программирования
ПК-2 - способность и готовность настраивать, тестировать и осуществлять проверку вычислительной техники и программных средств	Знает	способы тестирования программных средств, в том числе, разработанных самостоятельно
	Умеет	находить и исправлять ошибки в тестируемых программных продуктах
	Владеет	навыками настройки вычислительной техники для работы в интегрированной среде разработки, навыками тестирования и проверки программных продуктов
ПК-3 – способность и готовность демонстрировать знания современных языков программирования, операционных систем, офисных приложений, информационно-телекоммуникационной сети "Интернет", способов и механизмов управления данными, принципов организации, состава и схемы работы операционных систем	Знает	принципы работы современных механизмов работы с данными, современные языки программирования (в частности, язык Python и его библиотеки NumPy, SciPy, Mathplotlib, Pandas, Sklearn)
	Умеет	применять знания современных языков программирования и принципов работы современных механизмов работы с данными для разработки программных продуктов
	Владеет	навыками разработки программ и реализации методы обработки научных данных на современных языках программирования

№ п/п	Контролируемые разделы дисциплины	Коды и этапы формирования компетенций		Оценочные средства - наименование	
				текущий контроль	промежуточная аттестация
1	Язык программирования Python	ОПК-2	Знает	Собеседование (УО-1)	Зачёт (вопросы 1-7), экзамен (вопросы 1-7)
			Умеет	Проект (ПР-9)	Зачёт (проекты 1-6), экзамен (проект 8)
			Владеет	Проект (ПР-9)	Зачёт (проекты 1-6), экзамен (проект 8)
		ПК-2	Знает	Собеседование (УО-1)	Зачёт (вопросы 1-7), экзамен (вопросы 1-7)
			Умеет	Проект (ПР-9)	Зачёт (проекты 1-6), экзамен (проект 8)
			Владеет	Проект (ПР-9)	Зачёт (проекты 1-6), экзамен (проект 8)
		ПК-3	Знает	Собеседование (УО-1)	Зачёт (вопросы 1-7), экзамен (вопросы 1-7)
			Умеет	Проект (ПР-9)	Зачёт (проекты 1-6), экзамен (проект 8)
			Владеет	Проект (ПР-9)	Зачёт (проекты 1-6), экзамен (проект 8)
2	Визуализация данных	ОПК-2	Знает	Собеседование (УО-1)	Зачёт (вопрос 4, 13), экзамен (вопрос 4, 13)
			Умеет	Проект (ПР-9)	Зачёт (проект 7), экзамен (проект 20)
			Владеет	Проект (ПР-9)	Зачёт (проект 7), экзамен (проект 20)
		ПК-2	Знает	Собеседование (УО-1)	Зачёт (вопрос 4, 13), экзамен (вопрос 4, 13)
			Умеет	Проект (ПР-9)	Зачёт (проект 7), экзамен (проект 20)
			Владеет	Проект (ПР-9)	Зачёт (проект 7), экзамен (проект 20)
		ПК-3	Знает	Собеседование (УО-1)	Зачёт (вопрос 4, 13), экзамен (вопрос 4, 13)
			Умеет	Проект (ПР-9)	Зачёт (проект 7), экзамен (проект 20)

			Владеет	Проект (ПР-9)	Зачёт (проект 7), экзамен (проект 20)
3	Методы машинного обучения	ОПК-2	Знает	Собеседование (УО-1)	Экзамен (вопросы 8-12)
			Умеет	Проект (ПР-9)	Экзамен (проекты 9-17)
			Владеет	Проект (ПР-9)	Экзамен (проекты 9-17)
		ПК-2	Знает	Собеседование (УО-1)	Экзамен (вопросы 8-12)
			Умеет	Проект (ПР-9)	Экзамен (проекты 9-17)
			Владеет	Проект (ПР-9)	Экзамен (проекты 9-17)
		ПК-3	Знает	Собеседование (УО-1)	Экзамен (вопросы 8-12)
			Умеет	Проект (ПР-9)	Экзамен (проекты 9-17)
			Владеет	Проект (ПР-9)	Экзамен (проекты 9-17)
4	Нейронные сети	ОПК-2	Знает	Собеседование (УО-1)	Экзамен (вопросы 14-20)
			Умеет	Проект (ПР-9)	Экзамен (проекты 18-20)
			Владеет	Проект (ПР-9)	Экзамен (проекты 18-20)
		ПК-2	Знает	Собеседование (УО-1)	Экзамен (вопросы 14-20)
			Умеет	Проект (ПР-9)	Экзамен (проекты 18-20)
			Владеет	Проект (ПР-9)	Экзамен (проекты 18-20)
		ПК-3	Знает	Собеседование (УО-1)	Экзамен (вопросы 14-20)
			Умеет	Проект (ПР-9)	Экзамен (проекты 18-20)
			Владеет	Проект (ПР-9)	Экзамен (проекты 18-20)

Зачётно-экзаменационные материалы

Вопросы для подготовки к зачёту

по дисциплине «Машинное обучение и анализ данных»

1. Язык программирования Python. Основные операторы. Синтаксис, семантика.
2. Язык программирования Python. Списки, словари, кортежи, множества.
3. Язык программирования Python. Функции.
4. Язык программирования Python. Работа с компьютерной графикой.
5. Язык программирования Python. Библиотека Numpy.
6. Язык программирования Python. Библиотека Scipy.
7. Язык программирования Python. Библиотека Pandas.

Вопросы для подготовки к экзамену

по дисциплине «Машинное обучение и анализ данных»

1. Язык программирования Python. Основные операторы. Синтаксис, семантика.
2. Язык программирования Python. Списки, словари, кортежи, множества.
3. Язык программирования Python. Функции.
4. Язык программирования Python. Работа с компьютерной графикой.
5. Язык программирования Python. Библиотека Numpy.
6. Язык программирования Python. Библиотека Scipy.
7. Язык программирования Python. Библиотека Pandas.
8. Язык программирования Python. Реализация основных методов машинного обучения.
9. Решающие деревья.
10. Метод опорных векторов.
11. Выбор метрики.
12. Логистическая регрессия.

13. Основные способы графического представления численных данных (визуализация).

14. Искусственный интеллект и машинное обучение.

15. Как работает перцептрон.

16. Функции активации.

17. Многослойный перцептрон.

18. Метод моментов.

19. Свёрточные нейронные сети.

20. Автокодировщики.

Комплекты оценочных средств для текущей аттестации

Вопросы для собеседования

по дисциплине «Машинное обучение и анализ данных»

1. Язык программирования Python. Основные операторы. Синтаксис, семантика.
2. Язык программирования Python. Списки, словари, кортежи, множества.
3. Язык программирования Python. Функции.
4. Язык программирования Python. Работа с компьютерной графикой.
5. Язык программирования Python. Библиотека Numpy.
6. Язык программирования Python. Библиотека Scipy.
7. Язык программирования Python. Библиотека Pandas.
8. Язык программирования Python. Реализация основных методов машинного обучения.
9. Решающие деревья.
10. Метод опорных векторов.
11. Выбор метрики.
12. Логистическая регрессия.
13. Основные способы графического представления численных данных (визуализация).
14. Искусственный интеллект и машинное обучение.
15. Как работает перцептрон.
16. Функции активации.
17. Многослойный перцептрон.
18. Метод моментов.
19. Свёрточные нейронные сети.
20. Автокодировщики.

Критерии оценки:

✓ 100-86 баллов - если ответ показывает глубокое и систематическое знание всего программного материала и структуры конкретного вопроса, а также основного содержания и новаций лекционного курса по сравнению с учебной литературой. Студент демонстрирует отчетливое и свободное владение концептуально-понятийным аппаратом, научным языком и терминологией соответствующей научной области. Знание основной литературы и знакомство с дополнительно рекомендованной литературой. Логически корректное и убедительное изложение ответа.

✓ 85-76 - баллов - знание узловых проблем программы и основного содержания лекционного курса; умение пользоваться концептуально-понятийным аппаратом в процессе анализа основных проблем в рамках данной темы; знание важнейших работ из списка рекомендованной литературы. В целом логически корректное, но не всегда точное и аргументированное изложение ответа.

✓ 75-61 - балл – фрагментарные, поверхностные знания важнейших разделов программы и содержания лекционного курса; затруднения с использованием научно-понятийного аппарата и терминологии учебной дисциплины; неполное знакомство с рекомендованной литературой; частичные затруднения с выполнением предусмотренных программой заданий; стремление логически определенно и последовательно изложить ответ.

✓ 60-50 баллов – незнание, либо отрывочное представление о данной проблеме в рамках учебно-программного материала; неумение использовать понятийный аппарат; отсутствие логической связи в ответе.

Итоговая шкала оценивания

Менее 60 баллов	незачтено	неудовлетворительно
От 61 до 75 баллов	зачтено	удовлетворительно
От 76 до 85 баллов	зачтено	хорошо
От 86 до 100 балов	зачтено	отлично

Составитель _____Воронцова Е.А._____
(подпись)

«____»_____2016 г.

Темы проектов

по дисциплине «Машинное обучение и анализ данных»

1. Язык программирования Python. Расчёт пенсионных накоплений.
2. Язык программирования Python. Проект «Диофантовы уравнения и MacDonalds».
3. Адаптивный тренажёр Python.
4. Линейная алгебра и язык скриптов Python.
5. Работа с массивами в NumPy.
6. Возможности библиотеки SciPy.
7. Графическая обработка и визуализация данных (Mathplotlib).
8. Сетевой веб-сервис на Python.
9. Задача распознавания спама в текстовых сообщениях
10. Предобработка данных в Pandas
11. Важность признаков
12. Выбор числа соседей
13. Выбор метрики
14. Нормализация признаков
15. Метод опорных векторов
16. Анализ текстов
17. Логистическая регрессия
18. Перцептрон
19. Многослойные перцептрон
20. Свёрточные нейронные сети

Критерии оценки:

✓ 100-86 баллов - если ответ показывает глубокое и систематическое знание всего программного материала и структуры конкретного вопроса, а также основного содержания и новаций лекционного курса по сравнению с учебной литературой. Студент демонстрирует отчетливое и свободное владение концептуально-понятийным аппаратом, научным языком и терминологией соответствующей научной области. Знание основной литературы и знакомство с дополнительно рекомендованной литературой. Логически корректное и убедительное изложение ответа.

✓ 85-76 - баллов - знание узловых проблем программы и основного содержания лекционного курса; умение пользоваться концептуально-понятийным аппаратом в процессе анализа основных проблем в рамках данной темы; знание важнейших работ из списка рекомендованной литературы. В целом логически корректное, но не всегда точное и аргументированное изложение ответа.

✓ 75-61 балл – фрагментарные, поверхностные знания важнейших разделов программы и содержания лекционного курса; затруднения с использованием научно-понятийного аппарата и терминологии учебной дисциплины; неполное знакомство с рекомендованной литературой; частичные затруднения с выполнением предусмотренных программой заданий; стремление логически определенно и последовательно изложить ответ.

✓ 60-50 баллов – незнание, либо отрывочное представление о данной проблеме в рамках учебно-программного материала; неумение использовать понятийный аппарат; отсутствие логической связи в ответе.

Итоговая шкала оценивания

Менее 60 баллов	незачтено	неудовлетворительно
От 61 до 75 баллов	зачтено	удовлетворительно
От 76 до 85 баллов	зачтено	хорошо
От 86 до 100 балов	зачтено	отлично

Составитель _____Воронцова Е.А.
(подпись)

«____»_____2016 г.

Описание показателей и критериев оценивания компетенций, шкал оценивания

Критерии оценки собеседования

✓ 100-86 баллов - если ответ показывает глубокое и систематическое знание всего программного материала и структуры конкретного вопроса, а также основного содержания и новаций курса по сравнению с учебной литературой. Студент демонстрирует отчетливое и свободное владение концептуально-понятийным аппаратом, научным языком и терминологией соответствующей научной области. Знание основной литературы и знакомство с дополнительно рекомендованной литературой. Логически корректное и убедительное изложение ответа.

✓ 85-76 - баллов - знание узловых проблем программы и основного содержания курса; умение пользоваться концептуально-понятийным аппаратом в процессе анализа основных проблем в рамках данной темы; знание важнейших работ из списка рекомендованной литературы. В целом логически корректное, но не всегда точное и аргументированное изложение ответа.

✓ 75-61 - балл – фрагментарные, поверхностные знания важнейших разделов программы и содержания курса; затруднения с использованием научно-понятийного аппарата и терминологии учебной дисциплины; неполное знакомство с рекомендованной литературой; частичные затруднения с выполнением предусмотренных программой заданий; стремление логически определенно и последовательно изложить ответ.

✓ 60-50 баллов – незнание, либо отрывочное представление о данной проблеме в рамках учебно-программного материала; неумение использовать понятийный аппарат; отсутствие логической связи в ответе.

Итоговая шкала оценивания

Менее 60 баллов	незачтено	неудовлетворительно
От 61 до 75 баллов	зачтено	удовлетворительно
От 76 до 85 баллов	зачтено	хорошо
От 86 до 100 балов	зачтено	отлично

Методические рекомендации, определяющие процедуры оценивания результатов освоения дисциплины

Текущая аттестация студентов. Текущая аттестация студентов по дисциплине «Машинное обучение и анализ данных» проводится в соответствии с локальными нормативными актами ДВФУ и является обязательной.

Текущая аттестация по дисциплине «Машинное обучение и анализ данных» проводится в форме собеседования и выполнения проектных работ, практических заданий, лабораторных работ и осуществляется ведущим преподавателем.

Объектами оценивания выступают:

– степень усвоения теоретических знаний - оценивается в форме собеседования;

– уровень овладения практическими умениями и навыками – оценивается в форме выполнения лабораторных работ, практических заданий, работы над проектами.

Промежуточная аттестация студентов. Промежуточная аттестация студентов по дисциплине «Машинное обучение и анализ данных» проводится в соответствии с локальными нормативными актами ДВФУ и является обязательной.

По дисциплине предусмотрены экзамен в 6 семестре и зачёт в 5 семестре, которые проводится в устной форме.

Критерии выставления оценки студенту на зачёте/ экзамене по дисциплине «Машинное обучение и анализ данных»

Баллы (рейтинговой оценки)	**Оценка зачета/ экзамена** (стандартная)	**Требования к сформированным компетенциям**

	«зачтено»/ «отлично»	Оценка «отлично» выставляется студенту, если он глубоко и прочно усвоил программный материал, исчерпывающе, последовательно, четко и логически стройно его излагает, умеет тесно увязывать теорию с практикой, свободно справляется с задачами, вопросами и другими видами применения знаний, причем не затрудняется с ответом при видоизменении заданий, правильно обосновывает принятое решение, владеет разносторонними навыками и приёмами выполнения лабораторных работ; знает современные технологии программирования на языке Python; умеет разрабатывать программные продукты с помощью современных технологий программирования; владеет эффективными методами разработки программных продуктов с помощью современных технологий программирования; знает технологии отладки и тестирования прикладного программного обеспечения; умеет отлаживать и тестировать прикладное программное обеспечение; владеет методами отладки и тестирования прикладного программного обеспечения; знает способы тестирования программных средств, в том числе, разработанных самостоятельно; умеет находить и исправлять ошибки в тестируемых программных продуктах; владеет навыками настройки вычислительной техники для работы в интегрированной среде разработки, навыками тестирования и проверки программных продуктов; знает современные языки программирования (в частности, язык Python); умеет применять знания современных языков программирования для разработки программных продуктов; владеет навыками разработки программ на современных языках программирования.
	«зачтено»/ «хорошо»	Оценка «хорошо» выставляется студенту, если он твердо знает материал, грамотно и по существу излагает его, не допуская существенных неточностей в ответе на вопрос, правильно применяет теоретические положения при решении практических вопросов и задач, владеет необходимыми навыками и приемами их выполнения; знает современные технологии программирования на языке Python; умеет разрабатывать программные продукты с помощью

		современных технологий программирования; владеет эффективными методами разработки программных продуктов с помощью современных технологий программирования; умеет находить и исправлять ошибки в тестируемых программных продуктах; владеет навыками настройки вычислительной техники для работы в интегрированной среде разработки; знает современные языки программирования (в частности, язык программирования Python); умеет применять знания современных языков программирования для разработки программных продуктов; владеет навыками разработки программ на современных языках программирования.
	зачтено»/ «удовлетворительно»	Оценка «удовлетворительно» выставляется студенту, если он имеет знания только основного материала, но не усвоил его деталей, допускает неточности, недостаточно правильные формулировки, нарушения логической последовательности в изложении программного материала, испытывает затруднения при выполнении лабораторных работ; частично владеет навыками разработки программ на современных языках программирования.
	«не зачтено»/ «неудовлетворительно»	Оценка «неудовлетворительно» выставляется студенту, который не знает значительной части программного материала, допускает существенные ошибки, неуверенно, с большими затруднениями выполняет лабораторные работы, практически не владеет навыками разработки программ на современных языках программирования. Как правило, оценка «неудовлетворительно» ставится студентам, которые не могут продолжить обучение без дополнительных занятий по соответствующей дисциплине.