

**Федеральное государственное автономное образовательное  
учреждение высшего образования  
«Московский физико-технический институт  
(национальный исследовательский университет)»**

**УТВЕРЖДЕНО**

**Директор по цифровизации  
образования**

**Д.И. Гриц**

<b>по дисциплине:</b>	<b>Рабочая программа дисциплины (модуля)</b> Машинное обучение и анализ больших данных
<b>по направлению:</b>	Бизнес-информатика
<b>профиль подготовки:</b>	Финансовые технологии и аналитика центр дополнительного, дополнительного профессионального и онлайн-образования "Пуск" центр дополнительного, дополнительного профессионального и онлайн-образования "Пуск"
<b>курс:</b>	1
<b>квалификация:</b>	магистр

Семестры, формы промежуточной аттестации:

1 (осенний) - Зачет  
2 (весенний) - Зачет

Аудиторных часов: 135 всего, в том числе:

лекции: 60 час.  
семинары: 75 час.  
лабораторные занятия: 0 час.

Самостоятельная работа: 135 час.

Всего часов: 270, всего зач. ед.: 6

Программу составил: Р.С. Кулиев, старший преподаватель

Программа обсуждена на заседании центра дополнительного, дополнительного профессионального и онлайн-образования "Пуск" 30.06.2022

## Аннотация

Дисциплина состоит из трех модулей:

Модуль 1. Основы программирования на Python

Модуль 2. Математика и Python для анализа данных

Модуль 3. Обучение на размеченных данных.

По итогам обучения обучающийся будет способен формализовать и алгоритмизировать поставленную задачу, написать программный код с использованием языков программирования, оформить код в соответствии с установленными требованиями.

### 1. Цели и задачи

#### Цель дисциплины

Целью реализации дисциплины является формирование/совершенствование компетенций слушателей в области решения профессиональных задач по машинному обучению и анализу больших массивов данных.

#### Задачи дисциплины

- сформировать умение использовать базовые типы и конструкции языка программирования Python;
- сформировать умение работать со стандартными структурами данных в Python, писать функции на Python, применять функциональные особенности языка, работать с файлами с помощью языка Python;
- сформировать умение применять механизмы наследования, создавать классы и работать с ними, обрабатывать исключения;
- сформировать умение искать и исправлять ошибки в программе на Python, тестировать программы на Python;
- сформировать умение писать многопоточный код на Python, писать асинхронный код на Python, работать с сетью, создать свое серверное сетевое приложение;
- сформировать умение пользоваться библиотеками Python для работы с данными;
- сформировать умение решать оптимизационные задачи с помощью Python;
- сформировать умение использовать математический аппарат для работы с данными;
- сформировать навыки построения предсказывающих моделей;
- сформировать умение оценивать качество построенных моделей;
- сформировать умение применять инструменты Python для решения задач машинного обучения.

### 2. Перечень формируемых компетенций

Освоение дисциплины направлено на формирование следующих компетенций:

Код и наименование компетенции	Индикаторы достижения компетенции
ОПК-1 Способен разрабатывать стратегию развития информационных технологий, инфраструктуры предприятия и управлять её реализацией	ОПК-1.1 Применяет на практике методики оценки качества ресурсов информационных технологий, управления активами и конфигурации информационных технологий, методики определения потребностей в уровне качества ресурсов ИТ

### 3. Перечень планируемых результатов обучения по дисциплине (модулю)

В результате освоения дисциплины обучающиеся должны знать:

- базовые сведения о языке, особенности организации кода на Python;
- стандартные структуры данных в Python;
- механизмы наследования, классы;
- особенности объектно-ориентированной модели в Python;
- процессы и потоки ОС;
- основные понятия анализа данных;
- основные математические объекты для работы с данными;
- принципы статистики и теории вероятностей;
- основные понятия машинного обучения;
- типы признаков в машинном обучении;
- метрики качества в задачах регрессии и классификации;
- свойства L1 и L2 регуляризации;
- методы предобработки данных;
- метрические методы машинного обучения.

уметь:

- использовать базовые типы и конструкции языка;
- работать со стандартными структурами данных в Python, писать функции на Python, применять функциональные особенности языка, работать с файлами с помощью языка Python;
- применять механизмы наследования, создавать классы и работать с ними, обрабатывать исключения;
- искать и исправлять ошибки в программе на Python, тестировать программы на Python;
- писать многопоточный код на Python, писать асинхронный код на Python, работать с сетью, создать своё серверное сетевое приложение;
- работать в команде.
- использовать математический аппарат для работы с данными;
- использовать основные инструменты Python для работы с данными;
- выбирать подходящий метод оптимизации для конкретной задачи;
- оценивать параметры модели;
- применять библиотеки Python для построения модели линейной регрессии, решающих деревьев и композиций алгоритмов;
- применять библиотеки Python для обучения метрических алгоритмов, SVM, байесовских моделей.

владеть:

- стандартными структурами данных в Python, умением писать функции на Python, применять функциональные особенности языка, работать с файлами с помощью языка Python;
- механизмами наследования, создавать классы и работать с ними, обрабатывать исключения;
- навыками выбора подходящего метода оптимизации для конкретной задачи;
- навыками применения библиотеки Python для построения модели линейной регрессии, решающих деревьев и композиций алгоритмов, для обучения метрических алгоритмов, SVM, байесовских моделей.

#### 4. Содержание дисциплины (модуля), структурированное по темам (разделам) с указанием отведенного на них количества академических часов и видов учебных занятий

##### 4.1. Разделы дисциплины (модуля) и трудоемкости по видам учебных занятий

№	Тема (раздел) дисциплины	Трудоемкость по видам учебных занятий, включая самостоятельную работу, час.			
		Лекции	Семинары	Лаборат. работы	Самост. работа
1	Основы программирования на Python	30	30		75
2	Математика и Python для анализа данных	10	15		6
3	Обучение на размеченных данных	20	30		54
Итого часов		60	75		135
Подготовка к экзамену		0 час.			

Общая трудоёмкость	270 час., 6 зач.ед.
--------------------	---------------------

#### 4.2. Содержание дисциплины (модуля), структурированное по темам (разделам)

Семестр: 1 (Осенний)

##### 1. Основы программирования на Python

Основы программирования на Python. Структуры данных и функции. Объектно-ориентированное программирование. Углубленный Python. Многопоточное и асинхронное программирование.

Семестр: 2 (Весенний)

##### 2. Математика и Python для анализа данных

Знакомство с анализом данных. Основные библиотеки Python для анализа данных — NumPy, SciPy, Pandas, Matplotlib. Математические объекты для изучения анализа данных. Матричные разложения. Элементы теории вероятности и статистики.

##### 3. Обучение на размеченных данных

Машинное обучение и линейные модели. Борьба с переобучением и оценивание качества. Линейные модели: классификация и практические аспекты. Решающие деревья и композиции алгоритмов. Нейронные сети и обзор методов.

#### 5. Описание материально-технической базы, необходимой для осуществления образовательного процесса по дисциплине (модулю)

Система дистанционного обучения:

Обучающемуся необходимо наличие доступа в сеть интернет, компьютер.

Преподавателю курса необходимо наличие доступа администратора курса и оборудование для проведения дистанционных семинаров (вебинаров), качественный отказоустойчивый доступ в сеть интернет.

#### 6. Перечень рекомендуемой литературы

Основная литература

1. Python и анализ данных, Электрон. версия печ. публикации / У. Маккини. — Москва, ДМК Пресс, 2020
2. Python и анализ данных, Первичная обработка данных с применением pandas, NumPy и IPython / У. Маккини. — Москва, ДМК Пресс, 2020.— URL: <https://e.lanbook.com/book/131721> (дата обращения: 26.01.2021). - Полный текст (Режим доступа : из сети МФТИ / Удаленный доступ)
3. Курс дифференциального и интегрального исчисления [Текст]. Т. 3, учебник для вузов /Г. М. Фихтенгольц. СПб., Лань, 2019
4. Курс аналитической геометрии и линейной алгебры [Текст], учебник для вузов /Д. В. Беклемишев. -СПб., Лань, 2019
5. Лекции по математическому анализу. В 2 частях. Часть 1, учебное пособие/Г. Е. Иванов , -Москва, МФТИ, 2019
6. Курс математического анализа [Текст] : [учеб. пособие для вузов] / А. М. Тер-Крикоров, М. И. Шабунин .— 4-е изд., испр. — М. : БИНОМ. Лаб. знаний, 2009, 2010, 2012, 2013 .— 672 с.
7. Машинное обучение: новый искусственный интеллект [Текст]/Э. Алпайдин, -М., Изд. группа "Точка", 2017

#### Дополнительная литература

1. Линейная алгебра и ее применения [Текст]/Г. Стренг , -М., Мир, 1980
2. Введение в теорию вероятностей и ее приложения [Текст] : в 2 т : учеб. пособие для вузов. Т. 1 / В.Феллер ; пер. с пересмотр. 3-го англ. изд. Ю. В. Прохорова ; [придесл. А. Н. Колмогорова] .— М. : Мир, 1984 .— 528 с.

### **7. Перечень ресурсов информационно-телекоммуникационной сети "Интернет", необходимых для освоения дисциплины (модуля)**

Think Python [Электронный ресурс] – Режим доступа - <https://greenteapress.com/wp/think-python-2e/>  
Automate the Boring Stuff with Python [Электронный ресурс] – Режим доступа - <https://automatetheboringstuff.com/>  
Dive Into Python 3 [Электронный ресурс] – Режим доступа - <http://diveintopython3.problemsolving.io/>  
Problem Solving with Algorithms and Data Structures using Python [Электронный ресурс] – Режим доступа - <https://runestone.academy/runestone/static/pythonds/index.html>  
Swaroop Chitlur. A Byte of Python [Электронный ресурс] – Режим доступа - <https://wombat.org.ua/AByteOfPython/AByteofPythonRussian-2.02.pdf> – 2020.  
Справочник по функциям DAX [Электронный ресурс] – Режим доступа - <https://docs.microsoft.com/ru-ru/dax/dax-function-reference>

### **8. Перечень информационных технологий, используемых при осуществлении образовательного процесса по дисциплине (модулю), включая перечень необходимого программного обеспечения и информационных справочных систем (при необходимости)**

Документация Postgres про сравнение строк - <https://postgrespro.ru/docs/postgrespro/9.5/functions-matching>  
Документация Postgres про другие функции работы со строками - <https://postgrespro.ru/docs/postgrespro/9.5/functions-string>  
Тестер регулярных выражений - <https://www.regextester.com>  
Интерактивный учебник по SQL - <http://www.sql-tutorial.ru/ru/content.html>  
Введение в анализ данных с помощью Pandas - <https://habr.com/ru/post/196980/>  
Начало работы с Power BI - <https://docs.microsoft.com/ru-ru/power-bi/fundamentals/desktop-getting-started>  
Профессиональный информационно-аналитический ресурс, посвященный машинному обучению, распознаванию образов и интеллектуальному анализу данных – <http://www.machinelearning.ru/>  
Информационная система «Единое окно доступа к образовательным ресурсам» (ИС «Единое окно») – <http://window.edu.ru/>  
Платформа открытое образование – <https://openedu.ru/>

### **9. Методические указания для обучающихся по освоению дисциплины (модуля)**

Самостоятельная работа подразделяется на аудиторную и внеаудиторную. Аудиторную самостоятельную работу составляют практические задания, которые выполняются слушателями во время учебных занятий, результаты ее выполнения проверяются и оцениваются преподавателем в учебном процессе.

Внеаудиторная самостоятельная работа включает формы: изучение дополнительной литературы, подготовка итоговых проектов по модулям, подготовка проекта.

Основными критериями качества организации самостоятельной работы служит наличие контроля результатов самостоятельной работы.

Основными современными формами организации самостоятельной работы являются творческие работы и работа с информационными компьютерными технологиями.

**ОЦЕНОЧНЫЕ МАТЕРИАЛЫ ПО ДИСЦИПЛИНЕ (МОДУЛЮ)**

<b>по направлению:</b>	Бизнес-информатика		
<b>профиль подготовки:</b>	Финансовые технологии и аналитика	▲	▲
	онлайн-образования "Пуск"	▲	▲
	онлайн-образования "Пуск"		
<b>курс:</b>	1		
<b>квалификация:</b>	магистр		
Семестры, формы промежуточной аттестации:			
	1 (осенний) - Зачет		
	2 (весенний) - Зачет		
<b>Разработчик:</b>	Р.С. Кулиев, старший преподаватель		

## 1. Компетенции, формируемые в процессе изучения дисциплины

Код и наименование компетенции	Индикаторы достижения компетенции
ОПК-1 Способен разрабатывать стратегию развития информационных технологий, инфраструктуры предприятия и управлять её реализацией	ОПК-1.1 Применяет на практике методики оценки качества ресурсов информационных технологий, управления активами и конфигурации информационных технологий, методики определения потребностей в уровне качества ресурсов ИТ

## 2. Показатели оценивания компетенций

В результате изучения дисциплины «Машинное обучение и анализ больших данных» обучающийся должен:

### знать:

- базовые сведения о языке, особенности организации кода на Python;
- стандартные структуры данных в Python;
- механизмы наследования, классы;
- особенности объектно-ориентированной модели в Python;
- процессы и потоки ОС;
- основные понятия анализа данных;
- основные математические объекты для работы с данными;
- принципы статистики и теории вероятностей;
- основные понятия машинного обучения;
- типы признаков в машинном обучении;
- метрики качества в задачах регрессии и классификации;
- свойства L1 и L2 регуляризации;
- методы предобработки данных;
- метрические методы машинного обучения.

### уметь:

- использовать базовые типы и конструкции языка;
- работать со стандартными структурами данных в Python, писать функции на Python, применять функциональные особенности языка, работать с файлами с помощью языка Python;
- применять механизмы наследования, создавать классы и работать с ними, обрабатывать исключения;
- искать и исправлять ошибки в программе на Python, тестировать программы на Python;
- писать многопоточный код на Python, писать асинхронный код на Python, работать с сетью, создать своё серверное сетевое приложение;
- работать в команде.
- использовать математический аппарат для работы с данными;
- использовать основные инструменты Python для работы с данными;
- выбирать подходящий метод оптимизации для конкретной задачи;
- оценивать параметры модели;
- применять библиотеки Python для построения модели линейной регрессии, решающих деревьев и композиций алгоритмов;
- применять библиотеки Python для обучения метрических алгоритмов, SVM, байесовских моделей.

### владеть:

- стандартными структурами данных в Python, умением писать функции на Python, применять функциональные особенности языка, работать с файлами с помощью языка Python;
- механизмами наследования, создавать классы и работать с ними, обрабатывать исключения;
- навыками выбора подходящего метода оптимизации для конкретной задачи;
- навыками применения библиотеки Python для построения модели линейной регрессии, решающих деревьев и композиций алгоритмов, для обучения метрических алгоритмов, SVM, байесовских моделей.

## 3. Перечень типовых (примерных) вопросов, заданий, тем для подготовки к текущему контролю

#### **4. Перечень типовых (примерных) вопросов и тем для проведения промежуточной аттестации обучающихся**

##### **Критерии оценивания**

Форма аттестации предусматривает зачет в форме тестирования.

Оценка «зачтено» выставляется студенту, если он показал всесторонние, систематизированные, глубокие знания учебной программы дисциплины и умение уверенно применять их на практике при решении конкретных задач, свободное и правильное обоснование принятых решений.

Оценка «не зачтено» выставляется студенту, который не знает большей части основного содержания учебной программы дисциплины, допускает грубые ошибки в формулировках основных понятий дисциплины и не умеет использовать полученные знания при решении типовых практических задач.

Максимальная сумма, которую можно набрать, успешно выполнив все контрольные мероприятия, составляет 100 баллов. Для получения положительной оценки «зачтено» необходимо набрать не менее 50 баллов.

#### **5. Методические материалы, определяющие процедуры оценивания знаний, умений, навыков и (или) опыта деятельности**

Во время проведения зачета обучающиеся могут пользоваться программой дисциплины.



## Модуль 1

### Тема 1. Введение в Python

Написать программу (скрипт), которая будет запускаться из командной строки. Программа принимает в качестве аргумента строку, состоящую из цифр. Гарантируется, что других символов в переданном параметре нет и на вход всегда подается не пустая строка. Программа должна вычислить сумму цифр из которых состоит строка и вывести полученный результат на печать в стандартный вывод.

### Тема 2. Структуры данных и функции

Реализовать собственное **key-values** хранилище. Данные будут сохраняться в файле **storage.data**. Добавление новых данных в хранилище и получение текущих значений осуществляется с помощью утилиты командной строки **storage.py**.

### Тема 3. Объектно-ориентированное программирование

Написать python-модуль **solution.py**, внутри которого необходимо поместить код класса **FileReader**. Конструктор этого класса принимает один параметр: путь до файла на диске. В классе **FileReader** должен быть реализован метод **read**, возвращающий строку - содержимое файла, путь к которому был указан при создании экземпляра класса. Python модуль должен быть написан таким образом, чтобы импорт класса **FileReader** из него не вызвал ошибок.

При написании реализации метода **read**, вам нужно учитывать случай, когда при инициализации был передан путь к несуществующему файлу. Требуется обработать возникающее при этом исключение **FileNotFoundException** и вернуть из метода **read** пустую строку.

### Тема 4. Углубленный Python

Создать интерфейс для работы с файлами. Интерфейс должен предоставлять следующие возможности по работе с файлами:

- чтение из файла, метод **read** возвращает строку с текущим содержанием файла
  - запись в файл, метод **write** принимает в качестве аргумента строку с новым содержанием файла
  - сложение объектов типа **File**, результатом сложения является объект класса **File**, при этом создается новый файл и файловый объект, в котором содержимое второго файла добавляется к содержимому первого файла. Новый файл должен создаваться в директории, полученной с помощью функции **tempfile.gettempdir**. Для получения нового пути можно использовать **os.path.join**.
  - возвращать в качестве строкового представления объекта класса **File** полный путь до файла
  - поддерживать протокол итерации, причем итерация проходит по строкам файла
- При создании экземпляра класса **File** в конструктор передается полный путь до файла на файловой системе. Если файла с таким путем не существует, он должен быть создан при инициализации.

### Тема 5. Многопоточное и асинхронное программирование

В крупных проектах, с большим количеством пользователей, необходимо тщательно наблюдать за всеми процессами, происходящими в нем. Информация о процессах может быть представлена различными численными показателями, например: количество запросов к вашему приложению, время ответа вашего сервиса на каждый

запрос, количество пользователей в сутки и другие. Эти различные численные показатели мы будем называть метриками.

Для сбора, хранения и отображения подобных метрик существуют готовые решения, например Graphite, InfluxDB. Мы в рамках курса разработаем свою систему для сбора и хранения метрик, основанную на клиент-серверной архитектуре.

На этой неделе мы начнем с разработки клиента для отправки и получения метрик. На следующей неделе, в качестве финального задания, вам будет предложено реализовать сервер для хранения метрик.

### ***Протокол взаимодействия***

Прежде, чем приступить к описанию задания, рассмотрим протокол взаимодействия, по которому будет происходить обмен данными между клиентом и сервером.

Клиент и сервер взаимодействуют между собой по простому текстовому протоколу через TCP сокет. Текстовый протокол имеет главное преимущество – наглядность, можно просматривать диалог взаимодействия клиентской и серверной стороны без использования дополнительных инструментов.

Общий формат запросов и ответов.

Протокол поддерживает два вида запросов к серверу со стороны клиента:

- отправка данных для сохранения их на сервере
- получения сохраненных данных

Общий формат запроса клиента:

`<команда> <данные запроса><\n>`

где:

- `<команда>` - команда сервера (команда может принимать одно из двух значений: `put` — сохранить данные на сервере, `get` — вернуть сохраненные данные с сервера),

- `<данные запроса>` - данные запроса (их формат мы подробно разберем ниже в примере),

- `<\n>` - символ переноса строки.

Обратим ваше внимание на пробел между командой и данными запроса и его отсутствием между данными и символом перевода на новую строку.

Общий формат ответов сервера:

`<статус ответа><\n><данные ответа><\n\n>`

где:

- `<статус ответа>` - статус выполнения команды, допустимы два варианта: «`ok`» - команда успешно выполнена на сервере и «`error`» - выполнение команды завершилось ошибкой

- `<данные ответа>` - не обязательное поле (формат ответа и случаи его отсутствия будут рассмотрены в примере ниже)

- `<\n\n>` - два символа переноса строки.

Обратите внимание, что статус ответа и данные ответа разделены символом перевода строки `<\n>`.

### ***Метод put***

Метод **put** принимает в качестве параметров: название метрики, численное значение и необязательный именованный параметр **timestamp**. Если пользователь вызвал метод **put** без аргумента **timestamp**, то клиент автоматически должен подставить значение временной отметки, полученное с помощью вызова **int(time.time())**.

Метод **put** не возвращает ничего в случае успешной отправки и выбрасывает пользовательское исключение **ClientError** в случае не успешной.

### *Method get*

Метод **get** принимает в качестве параметра имя метрики, значения которой мы хотим получить. В качестве имени метрики можно использовать символ «\*», о котором мы упоминали в описании протокола.

Метод **get** возвращает словарь с метриками (смотрите пример ниже) в случае успешного получения ответа от сервера и выбрасывает исключение **ClientError** в случае не успешного.

Клиент получает данные от сервера в текстовом виде, метод **get** должен обработать строку ответа и вернуть словарь с полученными ключами с сервера. Значением ключей в словаре является список кортежей:

```
[(timestamp1, metric_value1), (timestamp2, metric_value2), ...]
```

Значение **timestamp** и **metric\_value** должны быть преобразованы соответственно к типам **int** и **float**. Список должен быть отсортирован по значению **timestamp** (по возрастанию).

## **Модуль 2**

### **Тема 1. Знакомство с анализом данных**

Ваше первое задание заключается в установке Python и библиотек. Как вы могли убедиться из демонстрационного видео, это очень просто и займет совсем немного времени. Тем не менее, важно выполнить задание своевременно, потому что эти инструменты понадобятся нам для дальнейшего прохождения курса. Убедитесь, что у вас не возникло проблем с установкой и использованием Python, и потренируйтесь запускать простенькие команды – будет интересно!

#### **Задание 1**

1. Перейдите на сайт [continuum.io](https://www.anaconda.com/products/individual) по ссылке <https://www.anaconda.com/products/individual>
2. Выберите подходящую вам операционную систему и перейдите в соответствующий раздел сайта.
3. Следуя инструкциям на сайте, установите Python 2.7.

#### **Задание 2**

1. Запустите `ipython notebook`. Команда запуска может несколько отличаться от команды запуска в инструкции из видео. Например, команда может быть такой: `ipython-2.7 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.

## Тема 2. Основные библиотеки Python для анализа данных — NumPy, SciPy, Pandas, Matplotlib

В этом задании вы познакомитесь с некоторыми базовыми методами из линейной алгебры, реализованными в пакете `SciPy`: в частности, с методами подсчета косинусного расстояния и решения систем линейных уравнений. Обе эти задачи еще много раз встретятся нам в специализации. Так, на решении систем линейных уравнений основана настройка линейных моделей — очень большого и важного класса алгоритмов машинного обучения. Косинусное расстояние же часто используется в анализе текстов для измерения сходства между ними.

### Задача 1: сравнение предложений

Дан набор предложений, скопированных из Википедии. Каждое из них имеет «кошачью тему» в одном из трех смыслов:

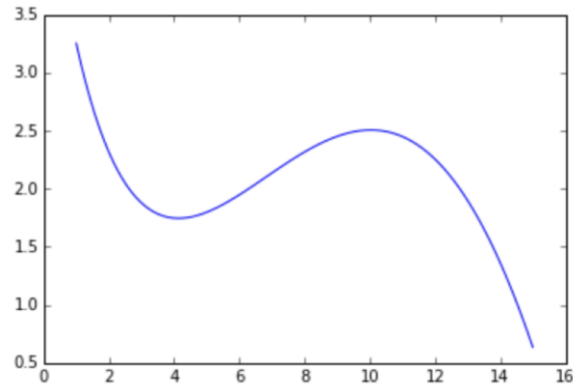
- кошки (животные),
- UNIX-утилита `cat` для вывода содержимого файлов,
- версии операционной системы `OS X`, названные в честь семейства кошачьих.

Ваша задача — найти два предложения, которые ближе всего по смыслу к расположенному в самой первой строке. В качестве меры близости по смыслу мы будем использовать косинусное расстояние.

### Задача 2: аппроксимация функции

Рассмотрим сложную математическую функцию на отрезке  $[1, 15]$ :

$$f(x) = \sin(x / 5) * \exp(x / 10) + 5 * \exp(-x / 2)$$



Она может описывать, например, зависимость оценок, которые выставляют определенному сорту вина эксперты, в зависимости от возраста этого вина. По сути, задача машинного обучения состоит в том, чтобы приблизить сложную зависимость с помощью функции из определенного семейства. В этом задании мы будем приближать указанную функцию с помощью многочленов.

Как известно, многочлен степени  $n$  (то есть  $w_0 + w_1 x + w_2 x^2 + \dots + w_n x^n$ ) однозначно определяется любыми  $n + 1$  различными точками, через которые он проходит. Это значит, что его коэффициенты  $w_0, \dots, w_n$  можно определить из следующей системы линейных уравнений:

$$\begin{cases} w_0 + w_1 x_1 + w_2 x_1^2 + \dots + w_n x_1^n = f(x_1) \\ \dots \\ w_0 + w_1 x_{n+1} + w_2 x_{n+1}^2 + \dots + w_n x_{n+1}^n = f(x_{n+1}) \end{cases}$$

где через  $x_1, \dots, x_n, x_{n+1}$  обозначены точки, через которые проходит многочлен, а через  $f(x_1), \dots, f(x_n), f(x_{n+1})$  — значения, которые он должен принимать в этих точках.

Воспользуемся описанным свойством и будем находить приближение функции многочленом, решая систему линейных уравнений.

1. Сформируйте систему линейных уравнений (то есть задайте матрицу коэффициентов  $A$  и свободный вектор  $b$ ) для многочлена первой степени, который должен совпадать с функцией  $f$  в точках 1 и 15. Решите данную систему с помощью функции `scipy.linalg.solve`. Нарисуйте функцию  $f$  и полученный многочлен. Хорошо ли он приближает исходную функцию?

2. Повторите те же шаги для многочлена второй степени, который совпадает с функцией  $f$  в точках 1, 8 и 15. Улучшилось ли качество аппроксимации?

3. Повторите те же шаги для многочлена третьей степени, который совпадает с функцией  $f$  в точках 1, 4, 10 и 15. Хорошо ли он аппроксимирует функцию? Коэффициенты данного многочлена (четыре числа в следующем порядке:  $w_0, w_1, w_2, w_3$ ) являются ответом на задачу. Округлять коэффициенты не обязательно, но при желании можете произвести округление до второго знака (т.е. до числа вида 0.42).

4. Запишите полученные числа в файл, разделив пробелами. Обратите внимание, что файл должен состоять из одной строки, в конце которой не должно быть переноса.

### Тема 3. Математические объекты для изучения анализа данных

В этом задании вы научитесь решать задачи оптимизации с помощью библиотеки SciPy. Сначала вы решите задачу поиска минимума функции с помощью одного из градиентных методов оптимизации, затем увидите отличия в работе градиентного метода и одного из методов глобальной оптимизации, а в заключение – найдете глобальный минимум негладкой функции, т.е. функции, у которой не всегда определен градиент.

Понимание задачи глобальной оптимизации и отличий градиентных методов, от методов, не использующих градиент, очень полезно в задачах анализа данных, в частности, для подбора параметров алгоритмов.

#### Задача 1. Минимизация гладкой функции

1. Рассмотрим все ту же функцию из задания по линейной алгебре:  $f(x) = \sin(x / 5) * \exp(x / 10) + 5 * \exp(-x / 2)$ , но теперь уже на промежутке  $[1, 30]$

2. В первом задании будем искать минимум этой функции на заданном промежутке с помощью `scipy.optimize`. Разумеется, в дальнейшем вы будете использовать методы оптимизации для более сложных функций, а  $f(x)$  мы рассмотрим как удобный учебный пример.

3. Напишите на Питоне функцию, вычисляющую значение  $f(x)$  по известному  $x$ . Будьте внимательны: не забывайте про то, что по умолчанию в питоне целые числа делятся нацело, и о том, что функции `sin` и `exp` нужно импортировать из модуля `math`.

4. Изучите примеры использования `scipy.optimize.minimize` в документации SciPy (см. "Материалы")

5. Попробуйте найти минимум, используя стандартные параметры в функции `scipy.optimize.minimize` (т.е. задав только функцию и начальное приближение). Попробуйте менять начальное приближение и изучить, меняется ли результат.

6. Укажите в `scipy.optimize.minimize` в качестве метода BFGS (один из самых точных в большинстве случаев градиентных методов оптимизации), запустите из начального приближения  $x=2$ . Градиент функции при этом указывать не нужно – он будет оценен численно. Полученное значение функции в точке минимума - ваш первый ответ по заданию 1, его надо записать с точностью до 2 знака после запятой.

7. Теперь измените начальное приближение на  $x=30$ . Значение функции в точке минимума - ваш второй ответ по заданию 1, его надо записать через пробел после первого, с точностью до 2 знака после запятой.

8. Стоит обдумать полученный результат. Почему ответ отличается в зависимости от начального приближения? Если нарисовать график функции (например, как это делалось в видео, где мы знакомились с Numpy, Scipy и Matplotlib), можно увидеть, в какие именно минимумы мы попали. В самом деле, градиентные методы обычно не решают задачу глобальной оптимизации, поэтому результаты работы ожидаемые и вполне корректные.

#### Задача 2. Глобальная оптимизация

9. Теперь попробуем применить к той же функции  $f(x)$  метод

глобальной оптимизации — дифференциальную эволюцию.

10. Изучите документацию и примеры использования функции `scipy.optimize.differential_evolution`.

11. Обратите внимание, что границы значений аргументов функции представляют собой список кортежей (`list`, в который помещены объекты типа `tuple`). Даже если у вас функция одного аргумента, возьмите границы его значений в квадратные скобки, чтобы передавать в этом параметре список из одного кортежа, т.к. в реализации `scipy.optimize.differential_evolution` длина этого списка используется чтобы определить количество аргументов функции.

12. Запустите поиск минимума функции  $f(x)$  с помощью дифференциальной эволюции на промежутке  $[1, 30]$ . Полученное значение функции в точке минимума - ответ в задаче 2. Запишите его с точностью до второго знака после запятой. В этой задаче ответ - только одно число.

13. Заметьте, дифференциальная эволюция справилась с задачей поиска глобального минимума на отрезке, т.к. по своему устройству она предполагает борьбу с попаданием в локальные минимумы.

14. Сравните количество итераций, потребовавшихся BFGS для нахождения минимума при хорошем начальном приближении, с количеством итераций, потребовавшихся дифференциальной эволюции. При повторных запусках дифференциальной эволюции количество итераций будет меняться, но в этом примере, скорее всего, оно всегда будет сравнимым с количеством итераций BFGS. Однако в дифференциальной эволюции за одну итерацию требуется выполнить гораздо больше действий, чем в BFGS. Например, можно обратить внимание на количество вычислений значения функции (`nfev`) и увидеть, что у BFGS оно значительно меньше. Кроме того, время работы дифференциальной эволюции очень быстро растет с увеличением числа аргументов функции.

Задача 3. Минимизация негладкой функции

15. Теперь рассмотрим функцию  $h(x) = \text{int}(f(x))$  на том же отрезке  $[1, 30]$ , т.е. теперь каждое значение  $f(x)$  приводится к типу `int` и функция принимает только целые значения.

16. Такая функция будет негладкой и даже разрывной, а ее график будет иметь ступенчатый вид. Убедитесь в этом, построив график  $h(x)$  с помощью `matplotlib`.

17. Попробуйте найти минимум функции  $h(x)$  с помощью BFGS, взяв в качестве начального приближения  $x=30$ . Получившееся значение функции – ваш первый ответ в этой задаче.

18. Теперь попробуйте найти минимум  $h(x)$  на отрезке  $[1, 30]$  с помощью дифференциальной эволюции. Значение функции  $h(x)$  в точке минимума – это ваш второй ответ в этом задании. Запишите его через пробел после предыдущего.

19. Обратите внимание на то, что полученные ответы различаются. Это ожидаемый результат, ведь BFGS использует градиент (в одномерном случае – производную) и явно не пригоден для

минимизации рассмотренной нами разрывной функции. Попробуйте понять, почему минимум, найденный BFGS, именно такой (возможно в этом вам поможет выбор разных начальных приближений).

20. Выполнив это задание, вы увидели на практике, чем поиск минимума функции отличается от глобальной оптимизации, и когда может быть полезно применить вместо градиентного метода оптимизации метод, не использующий градиент. Кроме того, вы попрактиковались в использовании библиотеки SciPy для решения оптимизационных задач, и теперь знаете, насколько это просто и удобно.

#### **Тема 4. Матричные разложения. Элементы теории вероятности и статистики.**

В этом задании вам предстоит проверить работу центральной предельной теоремы, а также поработать с генерацией случайных чисел и построением графиков в Питоне.

Выберите ваше любимое непрерывное распределение (чем меньше оно будет похоже на нормальное, тем интереснее; попробуйте выбрать какое-нибудь распределение из тех, что мы не обсуждали в курсе). Сгенерируйте из него выборку объёма 1000, постройте гистограмму выборки и нарисуйте поверх неё теоретическую плотность распределения вашей случайной величины (чтобы величины были в одном масштабе, не забудьте выставить у гистограммы значение параметра `normed=True`).

Ваша задача — оценить распределение выборочного среднего вашей случайной величины при разных объёмах выборок. Для этого при трёх и более значениях  $n$  (например, 5, 10, 50) сгенерируйте 1000 выборок объёма  $n$  и постройте гистограммы распределений их выборочных средних. Используя информацию о среднем и дисперсии исходного распределения (её можно без труда найти в википедии), посчитайте значения параметров нормальных распределений, которыми, согласно центральной предельной теореме, приближается распределение выборочных средних. Обратите внимание: для подсчёта значений этих параметров нужно использовать именно теоретические среднее и дисперсию вашей случайной величины, а не их выборочные оценки. Поверх каждой гистограммы нарисуйте плотность соответствующего нормального распределения (будьте внимательны с параметрами функции, она принимает на вход не дисперсию, а стандартное отклонение).

Опишите разницу между полученными распределениями при различных значениях  $n$ . Как меняется точность аппроксимации распределения выборочных средних нормальным с ростом  $n$ ?

### **Модуль 3**

#### **Тестовые задания**

1. Рассмотрим признак "Число обращений клиента в службу поддержки банка". Он принимает только целые неотрицательные значения. Какой тип имеет данный признак?

2. В чём заключается отличие градиентного спуска от стохастического градиентного спуска?

3. Каково значение точности (`precision`) для алгоритма со следующей матрицей ошибок?

	$y = 1$	$y = -1$
--	---------	----------



$a(x) = 1$	20	20
$a(x) = -1$	5	80

4. Что делает функция `LeaveOneOut()` из модуля `cross_validation`?
5. Какой вид регуляризатора позволяет проводить отбор признаков?
6. Какие средства помогают от переобучения линейных моделей?
7. Какая проблема, связанная с несбалансированными выборками, может возникнуть в подходе "один-против-всех" для многоклассовой классификации?
8. Можно ли решать задачу регрессии с помощью решающих деревьев?
9. Почему деревья большой глубины имеют высокий разброс?
10. Градиент какой функции вычисляется на каждой итерации градиентного бустинга?
11. По какому правилу байесовский классификатор относит объект к некоторому классу?
12. Как работает центроидный классификатор?
13. Что называют формулой обращения условной плотности?

### **Практические задания**

#### **Тема 1. Общее представление о WEB**

В этом практическом задании Вы будете прогнозировать выручку компании в зависимости от уровня ее инвестиций в рекламу по TV, в газетах и по радио.

1. Загрузите данные из файла `advertising.csv` в объект `pandas DataFrame`. [Источник данных](#).

```
In [ ]:
import pandas as pd
adver_data = pd.read_csv('advertising.csv')
```

Посмотрите на первые 5 записей и на статистику признаков в этом наборе данных.

```
In [ ]:
# Ваш код здесь
In [ ]:
# Ваш код здесь
```

Создайте массивы NumPy `X` из столбцов `TV`, `Radio` и `Newspaper` и `y` - из столбца `Sales`. Используйте атрибут `values` объекта `pandas DataFrame`.

```
In [ ]:
X = # Ваш код здесь
y = # Ваш код здесь
```

Отмасштабируйте столбцы матрицы `X`, вычтя из каждого значения среднее по соответствующему столбцу и поделив результат на стандартное отклонение. Для определенности, используйте методы `mean` и `std` векторов NumPy (реализация `std` в Pandas может отличаться). Обратите внимание, что в numpy вызов функции `.mean()` без параметров возвращает среднее по всем элементам массива, а не по столбцам, как в pandas. Чтобы произвести вычисление по столбцам, необходимо указать параметр `axis`.

```
In [ ]:
```

```
means, stds = # Ваш код здесь
```

```
In [ ]:
```

```
X = # Ваш код здесь
```

Добавьте к матрице X столбец из единиц, используя методы `hstack`, `ones` и `reshape` библиотеки NumPy. Вектор из единиц нужен для того, чтобы не обрабатывать отдельно коэффициент  $w_0$  линейной регрессии.

```
In [ ]:
```

```
import numpy as np
```

```
X = np.hstack # Ваш код здесь
```

2. Реализуйте функцию `mserror` - среднеквадратичную ошибку прогноза. Она принимает два аргумента - объекты `Series` `y` (значения целевого признака) и `y_pred` (предсказанные значения). Не используйте в этой функции циклы - тогда она будет вычислительно неэффективной.

```
In [ ]:
```

```
def mserror(y, y_pred):
```

```
    # Ваш код здесь
```

Какова среднеквадратичная ошибка прогноза значений `Sales`, если всегда предсказывать медианное значение `Sales` по исходной выборке? Полученный результат, округленный до 3 знаков после запятой, является ответом на '1 задание'.

```
In [ ]:
```

```
answer1 = # Ваш код здесь
```

```
print(round(answer1, 3))
```

3. Реализуйте функцию `normal_equation`, которая по заданным матрицам (массивам NumPy) `X` и `y` вычисляет вектор весов `w` согласно нормальному уравнению линейной регрессии.

```
In [ ]:
```

```
def normal_equation(X, y):
```

```
    return np.linalg.inv # Ваш код здесь
```

```
In [ ]:
```

```
norm_eq_weights = normal_equation(X, y)
```

```
print(norm_eq_weights)
```

Какие продажи предсказываются линейной моделью с весами, найденными с помощью нормального уравнения, в случае средних инвестиций в рекламу по ТВ, радио и в газетах? (то есть при нулевых значениях масштабированных признаков `TV`, `Radio` и `Newspaper`). Полученный результат, округленный до 3 знаков после запятой, является ответом на '2 задание'.

```
In [ ]:
```

```
answer2 = # Ваш код здесь
```

```
print(round(answer2, 3))
```

4. Напишите функцию `linear_prediction`, которая принимает на вход матрицу `X` и вектор весов линейной модели `w`, а возвращает вектор прогнозов в виде линейной комбинации столбцов матрицы `X` с весами `w`.

```
In [ ]:
```

```
def linear_prediction(X, w):
```

```
    # Ваш код здесь
```

Какова среднеквадратичная ошибка прогноза значений `Sales` в виде линейной модели с весами, найденными с помощью нормального уравнения? Полученный результат, округленный до 3 знаков после запятой, является ответом на '3 задание'

```
In [ ]:
```

```
answer3 = # Ваш код здесь
```

```
print(round(answer3, 3))
```

5. Напишите функцию `stochastic_gradient_step`, реализующую шаг стохастического градиентного спуска для линейной регрессии. Функция должна принимать матрицу `X`, вектора `y` и `w`, число `train_ind` - индекс объекта обучающей выборки (строки матрицы `X`), по которому считается изменение весов, а также число  $\eta$  (`eta`) - шаг градиентного спуска (по умолчанию `eta=0.01`). Результатом будет вектор обновленных весов. Наша реализация функции будет явно написана для данных с 3 признаками, но несложно модифицировать для любого числа признаков, можете это сделать.

```
In [ ]:
def stochastic_gradient_step(X, y, w, train_ind, eta=0.01):
    grad0 = # Ваш код здесь
    grad1 = # Ваш код здесь
    grad2 = # Ваш код здесь
    grad3 = # Ваш код здесь
    return w - eta * np.array([grad0, grad1, grad2, grad3])
```

6. Напишите функцию `stochastic_gradient_descent`, реализующую стохастический градиентный спуск для линейной регрессии. Функция принимает на вход следующие аргументы:

- `X` - матрица, соответствующая обучающей выборке
- `y` - вектор значений целевого признака
- `w_init` - вектор начальных весов модели
- `eta` - шаг градиентного спуска (по умолчанию 0.01)
- `max_iter` - максимальное число итераций градиентного спуска (по умолчанию 10000)
- `max_weight_dist` - максимальное евклидово расстояние между векторами весов на соседних итерациях градиентного спуска, при котором алгоритм прекращает работу (по умолчанию  $1e-8$ )
- `seed` - число, используемое для воспроизводимости сгенерированных псевдослучайных чисел (по умолчанию 42)
- `verbose` - флаг печати информации (например, для отладки, по умолчанию `False`)

На каждой итерации в вектор (список) должно записываться текущее значение среднеквадратичной ошибки. Функция должна возвращать вектор весов `w`, а также вектор (список) ошибок.

```
In [ ]:
def stochastic_gradient_descent(X, y, w_init, eta=1e-2, max_iter=1e4,
                               min_weight_dist=1e-8, seed=42, verbose=False):
    # Инициализируем расстояние между векторами весов на соседних
    # итерациях большим числом.
    weight_dist = np.inf
    # Инициализируем вектор весов
    w = w_init
    # Сюда будем записывать ошибки на каждой итерации
    errors = []
    # Счетчик итераций
    iter_num = 0
    # Будем порождать псевдослучайные числа
    # (номер объекта, который будет менять веса), а для воспроизводимости
    # этой последовательности псевдослучайных чисел используем seed.
```

```
np.random.seed(seed)
```

```
# Основной цикл
```

```
while weight_dist > min_weight_dist and iter_num < max_iter:
```

```
    # порождаем псевдослучайный
```

```
    # индекс объекта обучающей выборки
```

```
    random_ind = np.random.randint(X.shape[0])
```

```
    # Ваш код здесь
```

```
return w, errors
```

Запустите  $10^5$  итераций стохастического градиентного спуска. Укажите вектор начальных весов `w_init`, состоящий из нулей. Оставьте параметры `eta` и `seed` равными их значениям по умолчанию (`eta=0.01`, `seed=42` - это важно для проверки ответов).

```
In [ ]:
```

```
%%time
```

```
stoch_grad_desc_weights, stoch_errors_by_iter = # Ваш код здесь
```

Посмотрим, чему равна ошибка на первых 50 итерациях стохастического градиентного спуска. Видим, что ошибка не обязательно уменьшается на каждой итерации.

```
In [ ]:
```

```
%pylab inline
```

```
plot(range(50), stoch_errors_by_iter[:50])
```

```
xlabel('Iteration number')
```

```
ylabel('MSE')
```

Теперь посмотрим на зависимость ошибки от номера итерации для  $10^5$  итераций стохастического градиентного спуска. Видим, что алгоритм сходится.

```
In [ ]:
```

```
%pylab inline
```

```
plot(range(len(stoch_errors_by_iter)), stoch_errors_by_iter)
```

```
xlabel('Iteration number')
```

```
ylabel('MSE')
```

Посмотрим на вектор весов, к которому сошелся метод.

```
In [ ]:
```

```
stoch_grad_desc_weights
```

Посмотрим на среднеквадратичную ошибку на последней итерации.

```
In [ ]:
```

```
stoch_errors_by_iter[-1]
```

Какова среднеквадратичная ошибка прогноза значений Sales в виде линейной модели с весами, найденными с помощью градиентного спуска? Полученный результат, округленный до 3 знаков после запятой, является ответом на '4 задание'.

```
In [ ]:
```

```
answer4 = # Ваш код здесь
```

```
print(round(answer4, 3))
```

## Тема 2. Сбор данных со сторонних сайтов

В этом задании мы на примерах увидим, как переобучаются линейные модели, разберем, почему так происходит, и выясним, как диагностировать и контролировать переобучение.

Во всех ячейках, где написан комментарий с инструкциями, нужно написать код, выполняющий эти инструкции. Остальные ячейки с кодом (без комментариев) нужно просто выполнить. Кроме того, в задании требуется отвечать на вопросы; ответы нужно вписывать после выделенного слова "Ответ:".

Напоминаем, что посмотреть справку любого метода или функции (узнать, какие у нее аргументы и что она делает) можно с помощью комбинации Shift+Tab. Нажатие Tab после имени объекта и точки позволяет посмотреть, какие методы и переменные есть у этого объекта.

In [ ]:

```
import pandas as pd
import numpy as np
from matplotlib import pyplot as plt
%matplotlib inline
```

Мы будем работать с датасетом **"bikes\_rent.csv"**, в котором по дням записаны календарная информация и погодные условия, характеризующие автоматизированные пункты проката велосипедов, а также число прокатов в этот день. Последнее мы будем предсказывать; таким образом, мы будем решать задачу регрессии.

### Знакомство с данными

Загрузите датасет с помощью функции **pandas.read\_csv** в переменную **df**. Выведите первые 5 строчек, чтобы убедиться в корректном считывании данных:

In [ ]:

```
# (0 баллов)
```

```
# Считайте данные и выведите первые 5 строк
```

Для каждого дня проката известны следующие признаки (как они были указаны в источнике данных):

- *season*: 1 - весна, 2 - лето, 3 - осень, 4 - зима
- *yr*: 0 - 2011, 1 - 2012
- *mnth*: от 1 до 12
- *holiday*: 0 - нет праздника, 1 - есть праздник
- *weekday*: от 0 до 6
- *workingday*: 0 - нерабочий день, 1 - рабочий день
- *weathersit*: оценка благоприятности погоды от 1 (чистый,

ясный день) до 4 (ливень, туман)

- *temp*: температура в Цельсиях
- *atemp*: температура по ощущениям в Цельсиях
- *hum*: влажность
- *windspeed(mph)*: скорость ветра в милях в час
- *windspeed(ms)*: скорость ветра в метрах в секунду
- *cnt*: количество арендованных велосипедов (это целевой

признак, его мы будем предсказывать)

Итак, у нас есть вещественные, бинарные и номинальные (порядковые) признаки, и со всеми из них можно работать как с вещественными. С номинальными признаками тоже можно работать как с вещественными, потому что на них задан порядок. Давайте посмотрим на графиках, как целевой признак зависит от остальных

In [ ]:

```
fig, axes = plt.subplots(nrows=3, ncols=4, figsize=(15, 10))
```

```
for idx, feature in enumerate(df.columns[:-1]):
```

```
    df.plot(feature, "cnt", subplots=True, kind="scatter", ax=axes[idx // 4, idx % 4])
```

**Блок 1. Ответьте на вопросы (каждый 0.5 балла):**

1. Каков характер зависимости числа прокатов от месяца?  
○ ответ:
2. Укажите один или два признака, от которых число прокатов скорее всего зависит линейно  
○ ответ:

Давайте более строго оценим уровень линейной зависимости между признаками и целевой переменной. Хорошей мерой линейной зависимости между двумя векторами является корреляция Пирсона. В pandas ее можно посчитать с помощью двух методов датафрейма: `corr` и `corrwith`. Метод `df.corr` вычисляет матрицу корреляций всех признаков из датафрейма. Методу `df.corrwith` нужно подать еще один датафрейм в качестве аргумента, и тогда он посчитает попарные корреляции между признаками из `df` и этого датафрейма.

In [ ]:

*# Код 1.1 (0.5 балла)*

*# Посчитайте корреляции всех признаков, кроме последнего, с последним с помощью метода corrwith:*

В выборке есть признаки, коррелирующие с целевым, а значит, задачу можно решать линейными методами.

По графикам видно, что некоторые признаки похожи друг на друга. Поэтому давайте также посчитаем корреляции между вещественными признаками.

In [ ]:

*# Код 1.2 (0.5 балла)*

*# Посчитайте попарные корреляции между признаками temp, atemp, hum, windspeed(mph), windspeed(ms) и cnt*

*# с помощью метода corr:*

На диагоналях, как и полагается, стоят единицы. Однако в матрице имеются еще две пары сильно коррелирующих столбцов: `temp` и `atemp` (коррелируют по своей природе) и два `windspeed` (потому что это просто перевод одних единиц в другие). Далее мы увидим, что этот факт негативно сказывается на обучении линейной модели.

Напоследок посмотрим средние признаков (метод `mean`), чтобы оценить масштаб признаков и доли 1 у бинарных признаков.

In [ ]:

*# Код 1.3 (0.5 балла)*

*# Выведите средние признаков*

Признаки имеют разный масштаб, значит для дальнейшей работы нам лучше нормировать матрицу объекты-признаки.

### **Проблема первая: коллинеарные признаки**

Итак, в наших данных один признак дублирует другой, и есть еще два очень похожих. Конечно, мы могли бы сразу удалить дубликаты, но давайте посмотрим, как бы происходило обучение модели, если бы мы не заметили эту проблему.

Для начала проведем масштабирование, или стандартизацию признаков: из каждого признака вычтем его среднее и поделим на стандартное отклонение. Это можно сделать с помощью метода `scale`.

Кроме того, нужно перемешать выборку, это потребуется для кросс-валидации.

In [ ]:

**from** sklearn.preprocessing **import** scale

**from** sklearn.utils **import** shuffle

In [ ]:

df\_shuffled = shuffle(df, random\_state=123)

```
X = scale(df_shuffled[df_shuffled.columns[:-1]])
```

```
y = df_shuffled["cnt"]
```

Давайте обучим линейную регрессию на наших данных и посмотрим на веса признаков.

```
In [ ]:
```

```
from sklearn.linear_model import LinearRegression
```

```
In [ ]:
```

```
# Код 2.1 (1 балл)
```

```
# Создайте объект линейного регрессора, обучите его на всех данных и выведите веса модели
```

```
# (веса хранятся в переменной coef_ класса регрессора).
```

```
# Можно выводить пары (название признака, вес), воспользовавшись функцией zip, встроенной в язык python
```

```
# Названия признаков хранятся в переменной df.columns
```

Мы видим, что веса при линейно-зависимых признаках по модулю значительно больше, чем при других признаках.

Чтобы понять, почему так произошло, вспомним аналитическую формулу, по которой вычисляются веса линейной модели в методе наименьших квадратов:

$$w = (X^T X)^{-1} X^T y.$$

Если в  $X$  есть коллинеарные (линейно-зависимые) столбцы, матрица  $X^T X$  становится вырожденной, и формула перестает быть корректной. Чем более зависимы признаки, тем меньше определитель этой матрицы и тем хуже аппроксимация  $Xw \approx y$ . Такая ситуацию называют *проблемой мультиколлинеарности*, вы обсуждали ее на лекции.

С парой `temp-atemp` чуть менее коррелирующих переменных такого не произошло, однако на практике всегда стоит внимательно следить за коэффициентами при похожих признаках.

**Решение** проблемы мультиколлинеарности состоит в *регуляризации* линейной модели. К оптимизируемому функционалу прибавляют L1 или L2 норму весов, умноженную на коэффициент регуляризации  $\alpha$ . В первом случае метод называется Lasso, а во втором --- Ridge. Подробнее об этом также рассказано в лекции.

Обучите регрессоры Ridge и Lasso с параметрами по умолчанию и убедитесь, что проблема с весами решилась.

```
In [ ]:
```

```
from sklearn.linear_model import Lasso, Ridge
```

```
In [ ]:
```

```
# Код 2.2 (0.5 балла)
```

```
# Обучите линейную модель с L1-регуляризацией и выведите веса
```

```
In [ ]:
```

```
# Код 2.3 (0.5 балла)
```

```
# Обучите линейную модель с L2-регуляризацией и выведите веса
```

### **Проблема вторая: неинформативные признаки**

В отличие от L2-регуляризации, L1 обнуляет веса при некоторых признаках. Объяснение данному факту дается в одной из лекций курса.

Давайте пронаблюдаем, как меняются веса при увеличении коэффициента регуляризации  $\alpha$  (в лекции коэффициент при регуляризаторе мог быть обозначен другой буквой).

```
In [ ]:
```

```
# Код 3.1 (1 балл)
```



```

alphas = np.arange(1, 500, 50)
coefs_lasso = np.zeros((alphas.shape[0], X.shape[1])) # матрица весов размера
(число регрессоров) x (число признаков)
coefs_ridge = np.zeros((alphas.shape[0], X.shape[1]))
# Для каждого значения коэффициента из alphas обучите регрессор Lasso
# и запишите веса в соответствующую строку матрицы coefs_lasso (вспомните
встроенную в python функцию enumerate),
# а затем обучите Ridge и запишите веса в coefs_ridge.
Визуализируем динамику весов при увеличении параметра регуляризации:
In [ ]:
plt.figure(figsize=(8, 5))
for coef, feature in zip(coefs_lasso.T, df.columns):
    plt.plot(alphas, coef, label=feature, color=np.random.rand(3))
plt.legend(loc="upper right", bbox_to_anchor=(1.4, 0.95))
plt.xlabel("alpha")
plt.ylabel("feature weight")
plt.title("Lasso")

```

```

plt.figure(figsize=(8, 5))
for coef, feature in zip(coefs_ridge.T, df.columns):
    plt.plot(alphas, coef, label=feature, color=np.random.rand(3))
plt.legend(loc="upper right", bbox_to_anchor=(1.4, 0.95))
plt.xlabel("alpha")
plt.ylabel("feature weight")
plt.title("Ridge")

```

Ответы на следующие вопросы можно давать, глядя на графики или выводя коэффициенты на печать.

## Блок 2. Ответьте на вопросы (каждый 0.25 балла):

1. Какой регуляризатор (Ridge или Lasso) агрессивнее уменьшает веса при одном и том же  $\alpha$ ?
  - Ответ:
2. Что произойдет с весами Lasso, если  $\alpha$  сделать очень большим? Поясните, почему так происходит.
  - Ответ:
3. Можно ли утверждать, что Lasso исключает один из признаков `windspeed` при любом значении  $\alpha > 0$ ? А Ridge? Считается, что регуляризатор исключает признак, если коэффициент при нем  $< 1e-3$ .
  - Ответ:
4. Какой из регуляризаторов подойдет для отбора неинформативных признаков?
  - Ответ:

Далее будем работать с Lasso.

Итак, мы видим, что при изменении  $\alpha$  модель по-разному подбирает коэффициенты признаков. Нам нужно выбрать наилучшее  $\alpha$ .

Для этого, во-первых, нам нужна метрика качества. Будем использовать в качестве метрики сам оптимизируемый функционал метода наименьших квадратов, то есть Mean Square Error.



Во-вторых, нужно понять, на каких данных эту метрику считать. Нельзя выбирать  $\alpha$  по значению MSE на обучающей выборке, потому что тогда мы не сможем оценить, как модель будет делать предсказания на новых для нее данных. Если мы выберем одно разбиение выборки на обучающую и тестовую (это называется holdout), то настроимся на конкретные "новые" данные, и вновь можем переобучиться. Поэтому будем делать несколько разбиений выборки, на каждом пробовать разные значения  $\alpha$ , а затем усреднять MSE. Удобнее всего делать такие разбиения кросс-валидацией, то есть разделить выборку на  $K$  частей, или блоков, и каждый раз брать одну из них как тестовую, а из оставшихся блоков составлять обучающую выборку.

Делать кросс-валидацию для регрессии в `sklearn` совсем просто: для этого есть специальный регрессор, **LassoCV**, который берет на вход список из  $\alpha$  и для каждого из них вычисляет MSE на кросс-валидации. После обучения (если оставить параметр `cv=3` по умолчанию) регрессор будет содержать переменную `mse_path_`, матрицу размера  $\text{len}(\alpha) \times k$ ,  $k = 3$  (число блоков в кросс-валидации), содержащую значения MSE на тесте для соответствующих запусков. Кроме того, в переменной `alpha_` будет храниться выбранное значение параметра регуляризации, а в `coef_`, традиционно, обученные веса, соответствующие этому `alpha_`.

Обратите внимание, что регрессор может менять порядок, в котором он проходит по `alphas`; для сопоставления с матрицей MSE лучше использовать переменную регрессора `alphas_`.

```
In [ ]:
```

```
from sklearn.linear_model import LassoCV
```

```
In [ ]:
```

```
# Код 3.2 (1 балл)
```

```
# Обучите регрессор LassoCV на всех параметрах регуляризации из alpha
```

```
# Постройте график _усредненного_ по строкам MSE в зависимости от alpha.
```

```
# Выведите выбранное alpha, а также пары "признак-коэффициент" для  
обученного вектора коэффициентов
```

```
alphas = np.arange(1, 100, 5)
```

Итак, мы выбрали некоторый параметр регуляризации. Давайте посмотрим, какие бы мы выбирали  $\alpha$ , если бы делили выборку только один раз на обучающую и тестовую, то есть рассмотрим траектории MSE, соответствующие отдельным блокам выборки.

```
In [ ]:
```

```
# Код 3.3 (1 балл)
```

```
# Выведите значения alpha, соответствующие минимумам MSE на каждом  
разбиении (то есть по столбцам).
```

```
# На трех отдельных графиках визуализируйте столбцы .mse_path_
```

На каждом разбиении оптимальное значение  $\alpha$  свое, и ему соответствует большое MSE на других разбиениях. Получается, что мы настраиваемся на конкретные обучающие и контрольные выборки. При выборе  $\alpha$  на кросс-валидации мы выбираем нечто "среднее", что будет давать приемлемое значение метрики на разных разбиениях выборки.

Наконец, как принято в анализе данных, давайте проинтерпретируем результат.

### Блок 3. Ответьте на вопросы (каждый 0.5 балла):

1. В последней обученной модели выберите 4 признака с наибольшими (положительными) коэффициентами (и выпишите их), посмотрите на визуализации зависимостей `cnt` от этих признаков, которые мы

рисовали в блоке "Знакомство с данными". Видна ли возрастающая линейная зависимость `cnt` от этих признаков по графикам? Логично ли утверждать (из здравого смысла), что чем больше значение этих признаков, тем больше людей захотят взять велосипеды?

○ Ответ:

2. Выберите 3 признака с наибольшими по модулю отрицательными коэффициентами (и выпишите их), посмотрите на соответствующие визуализации. Видна ли убывающая линейная зависимость? Логично ли утверждать, что чем больше величина этих признаков, тем меньше людей захотят взять велосипеды?

○ Ответ:

3. Выпишите признаки с коэффициентами, близкими к нулю ( $< 1e-3$ ). Как вы думаете, почему модель исключила их из модели (вновь посмотрите на графики)? Верно ли, что они никак не влияют на спрос на велосипеды?

○ Ответ:

#### Тема 4. Хранение данных. SQL / NoSQL

Загрузите датасет `digits` с помощью функции `load_digits` из `sklearn.datasets` и подготовьте матрицу признаков `X` и ответы на обучающей выборке `y` (вам потребуются поля `data` и `target` в объекте, который возвращает `load_digits`).

Для оценки качества далее нужно будет использовать `cross_val_score` из `sklearn.model_selection` с параметром `cv=10`. Эта функция реализует `k-fold cross validation` с `k` равным значению параметра `cv`. Мы предлагаем использовать `k=10`, чтобы полученные оценки качества имели небольшой разброс, и было проще проверить полученные ответы. На практике же часто хватает и `k=5`. Функция `cross_val_score` будет возвращать `numpy.ndarray`, в котором будет `k` чисел - качество в каждом из `k` экспериментов `k-fold cross validation`. Для получения среднего значения (которое и будет оценкой качества работы) вызовите метод `.mean()` у массива, который возвращает `cross_val_score`.

С небольшой вероятностью вы можете натолкнуться на случай, когда полученное вами качество в каком-то из пунктов не попадет в диапазон, заданный для правильных ответов - в этом случае попробуйте перезапустить ячейку с `cross_val_score` несколько раз и выбрать наиболее «типичное» значение. Если это не помогает, то где-то была допущена ошибка.

Если вам захочется ускорить вычисление `cross_val_score` - можете попробовать использовать параметр `n_jobs`, но будьте осторожны: в одной из старых версий `sklearn` была ошибка, которая приводила к неверному результату работы `cross_val_score` при задании `n_jobs` отличным от 1. Сейчас такой проблемы возникнуть не должно, но проверить, что все в порядке, не будет лишним.

1. Создайте `DecisionTreeClassifier` с настройками по умолчанию и измерьте качество его работы с помощью `cross_val_score`. Эта величина и будет ответом в пункте 1.

2. Воспользуйтесь `BaggingClassifier` из `sklearn.ensemble`, чтобы обучить бэггинг над `DecisionTreeClassifier`. Используйте в `BaggingClassifier` параметры по умолчанию, задав только количество деревьев равным 100. Качество классификации новой модели - ответ в пункте 2. Обратите внимание, как соотносится качество работы композиции решающих деревьев с качеством работы одного решающего дерева.

3. Теперь изучите параметры `BaggingClassifier` и выберите их такими, чтобы каждый базовый алгоритм обучался не на всех  $d$  признаках, а на  $d\sqrt{d}$  случайных признаков. Качество работы получившегося классификатора - ответ в пункте 3. Корень из числа признаков - часто используемая эвристика в задачах классификации, в задачах регрессии же часто берут число признаков, деленное на три. Но в общем случае ничто не мешает вам выбирать любое другое число случайных признаков.

4. Наконец, давайте попробуем выбирать случайные признаки не один раз на все дерево, а при построении каждой вершины дерева. Сделать это несложно: нужно убрать выбор случайного подмножества признаков в `BaggingClassifier` и добавить его в `DecisionTreeClassifier`. Какой параметр за это отвечает, можно понять из документации `sklearn`, либо просто попробовать угадать (скорее всего, у вас сразу получится). Попробуйте выбирать опять же  $d\sqrt{d}$  признаков. Качество полученного классификатора на контрольной выборке и будет ответом в пункте 4.

5. Полученный в пункте 4 классификатор - бэггинг на рандомизированных деревьях (в которых при построении каждой вершины выбирается случайное подмножество признаков и разбиение ищется только по ним). Это в точности соответствует алгоритму `Random Forest`, поэтому почему бы не сравнить качество работы классификатора с `RandomForestClassifier` из `sklearn.ensemble`. Сделайте это, а затем изучите, как качество классификации на данном датасете зависит от количества деревьев, количества признаков, выбираемых при построении каждой вершины дерева, а также ограничений на глубину дерева. Для наглядности лучше построить графики зависимости качества от значений параметров, но для сдачи задания это делать не обязательно. На основе наблюдений выпишите через пробел номера правильных утверждений из приведенных ниже в порядке возрастания номера (это будет ответ в п.5)

1) Случайный лес сильно переобучается с ростом количества деревьев

2) При очень маленьком числе деревьев (5, 10, 15), случайный лес работает хуже, чем при большем числе деревьев

3) С ростом количества деревьев в случайном лесе, в какой-то момент деревьев становится достаточно для высокого качества классификации, а затем качество существенно не меняется.

4) При большом количестве признаков (для данного датасета - 40, 50) качество классификации становится хуже, чем при малом количестве признаков (5, 10). Это связано с тем, что чем меньше признаков выбирается в каждом узле, тем более различными получаются деревья (ведь деревья сильно неустойчивы к изменениям в обучающей выборке), и тем лучше работает их композиция.

5) При большом количестве признаков (40, 50, 60) качество классификации лучше, чем при малом количестве признаков (5, 10). Это связано с тем, что чем больше признаков - тем больше информации об объектах, а значит алгоритм может делать прогнозы более точно.

6) При небольшой максимальной глубине деревьев (5-6) качество работы случайного леса намного лучше, чем без ограничения глубины, т.к. деревья получаются не переобученными. С ростом глубины деревьев качество ухудшается.

7) При небольшой максимальной глубине деревьев (5-6) качество работы случайного леса заметно хуже, чем без ограничений, т.к. деревья получаются недообученными. С ростом глубины качество сначала улучшается, а затем не меняется существенно, т.к. из-за усреднения прогнозов и различий деревьев их переобученность в бэггинге не сказывается на итоговом качестве (все деревья преобучены по-разному, и

при усреднении они компенсируют переобученность друг-друга).

## Тема 5. Веб интерфейсы с Django и Bootstrap

В этом задании будет использоваться датасет `digits` из `sklearn.datasets`. Оставьте последние 25% объектов для контроля качества, разделив  $X$  и  $y$  на  $X_{\text{train}}$ ,  $y_{\text{train}}$  и  $X_{\text{test}}$ ,  $y_{\text{test}}$ .

Целью задания будет реализовать самый простой метрический классификатор — метод ближайшего соседа, а также сравнить качество работы реализованного вами 1NN с `RandomForestClassifier` из `sklearn` на 1000 деревьях.

### Задание 1

Реализуйте самостоятельно метод одного ближайшего соседа с евклидовой метрикой для задачи классификации. Можно не извлекать корень из суммы квадратов отклонений, т.к. корень — монотонное преобразование и не влияет на результат работы алгоритма.

Никакой дополнительной работы с признаками в этом задании делать не нужно — мы еще успеем этим заняться в других курсах. Ваша реализация может быть устроена следующим образом: можно для каждого классифицируемого объекта составлять список пар (расстояние до точки из обучающей выборки, метка класса в этой точке), затем сортировать этот список (по умолчанию сортировка будет сначала по первому элементу пары, затем по второму), а затем брать первый элемент (с наименьшим расстоянием).

Сортировка массива длиной  $N$  требует порядка  $N \log N$  сравнений (строже говоря, она работает за  $O(N \log N)$ ). Подумайте, как можно легко улучшить получившееся время работы. Кроме простого способа найти ближайший объект всего за  $N$  сравнений, можно попробовать придумать, как разбить пространство признаков на части и сделать структуру данных, которая позволит быстро искать соседей каждой точки. За выбор метода поиска ближайших соседей в `KNeighborsClassifier` из `sklearn` отвечает параметр `algorithm` — если у вас уже есть некоторый бэкграунд в алгоритмах и структурах данных, вам может быть интересно познакомиться со структурами данных `ball tree` и `kd tree`.

Доля ошибок, допускаемых 1NN на тестовой выборке, — ответ в задании 1.

### Задание 2

Теперь обучите на обучающей выборке `RandomForestClassifier(n_estimators=1000)` из `sklearn`. Сделайте прогнозы на тестовой выборке и оцените долю ошибок классификации на ней. Эта доля — ответ в задании 2. Обратите внимание на то, как соотносится качество работы случайного леса с качеством работы, пожалуй, одного из самых простых методов — 1NN. Такое различие — особенность данного датасета, но нужно всегда помнить, что такая ситуация тоже может иметь место, и не забывать про простые методы.

## Тест

1. Что такое производная функции?
2. Объект `frame` имеет тип `pandas.DataFrame()`. К чему приведет применение функции `fillna()` к объекту `frame` в случае вызова функции со следующими параметрами: `frame.fillna("", inplace=True)`?
3. Как с матричными разложениями связана задача рекомендации фильмов пользователям?
4. Произведением каких матриц представляется исходная при сингулярном разложении?
5. Как себя должен вести шаг градиентного спуска?
6. Почему при поиске минимума методом имитации отжига допускаются переходы в точки, в которых функция принимает большие значения, нежели в текущей?
7. В каких случаях стоит применять методы оптимизации, не использующие градиент?
8. Предположим, что в некоторой популяции вероятность дожить до 60 лет равна 0.5, а вероятность дожить до 80 лет — 0.2. Какова вероятность, что случайно выбранный шестидесятилетний представитель популяции доживёт до восьмидесяти? Запишите ответ с точностью до одного знака после десятичной точки (задавать разделитель в этой задаче и следующих нужно именно в виде точки).
9. Какова вероятность того, что при независимом подбрасывании двух симметричных шестигранных кубиков хотя бы на одном из них выпадет больше трёх очков? Запишите точный ответ в виде десятичной дроби.
10. Для продвижения вашего продукта рекламный отдел предлагает использовать новый видеоролик. На фокус-группе вы показываете 40 испытуемым новый и старый видеоролики и спрашиваете, какой из них им нравится больше; 62.5% испытуемых выбирают новый. Используя центральную предельную теорему и правило двух сигм, постройте 95% доверительный интервал для доли членов целевой аудитории, предпочитающих новый видеоролик. Выберите вывод, соответствующий построенному интервалу.
11. Выберите признаки, которые могут рассматриваться только как категориальные (и не могут рассматриваться как бинарные или вещественные)
12. Как можно избавиться от свободного члена в линейных моделях?
13. Какая из метрик качества в задачах регрессии является несимметричной?
14. Как может быть интерпретирован коэффициент детерминации?
15. Какая функция из модуля `cross_validation` делает стратифицированное разбиение выборки по фолдам?
16. Предположим, вы оцениваете качество работы алгоритма при помощи кросс-валидации с разбиением на  $k$  блоков. Сколько раз будет проведено обучение модели?
17. При комплексном обследовании нескольких тысяч человек по измерившимся показателям (включая пульс, давление, ЭКГ и т.д.) оценивался риск возникновения сердечного заболевания. Ста пациентам с самым высоким риском была предложена оздоровительная программа, включающая диету, упражнения и приём профилактических препаратов. Через несколько месяцев после окончания программы пациенты снова прошли диспансеризацию; средний оцениваемый риск возникновения сердечного заболевания существенно уменьшился. Что можно сказать об эффективности оздоровительной программы?

18. В каком случае может понадобиться переход в спрямляющее признаковое пространство?

19. Мы работаем с классификатором `classifier = linear_model.SGDClassifier()`. Мы хотим подобрать параметры модели с помощью поиска по сетке, а для этого нам хочется предварительно получить набор доступных параметров модели с их значениями. Какая команда позволяет это сделать?

20. Какую форму будет иметь разделяющая поверхность, построенная деревом с условиями вида  $[x^j < t]$  в вершинах? Считайте, что в выборке два признака.

21. В чём заключается переподбор прогнозов в листьях дерева в градиентном бустинге?

22. Как можно оценить по обучающей выборке априорную вероятность класса  $P(y)$ , если количество объектов в обучающей выборке  $\ell$ , из них  $k$  к классу  $y$  относятся  $I_y$ ?

23. Каким получится оптимальное значение количества соседей  $k$  в методе ближайших соседей, если настраивать его по качеству работы алгоритма на обучающей выборке?

24. Что является результатом оценивания параметра  $\theta$  по выборке  $X$  при байесовском подходе?

В зачетно-экзаменационную ведомость оценка выставляется в соответствии с нижеприведенной таблицей 1.

Таблица 1

Сумма баллов	Оценка
50-100	Зачтено
менее 50	Не зачтено

Составляющие процесса обучения, которые оцениваются в ходе обучения, и их вклад в итоговую оценку представлены в таблице 2.

Таблица 2

№ п/п	Основные показатели оценки	Вклад в итоговую оценку
1	Практические занятия	30%
2	Выполнение домашних заданий	40%
3	Промежуточная аттестация	30%

### 3.2. Оценочные материалы

Таблица 3

Наименование модуля	Формы и методы контроля и оценки	Вес задания
Основы программирования на Python	Практические задания по темам лекций Домашние задания в системе дистанционного обучения по темам лекций	14
Структуры данных и функции	Практические задания по темам лекций Домашние задания в системе дистанционного обучения по темам лекций	14
Объектно-ориентированное программирование	Практические задания по темам лекций Домашние задания в системе дистанционного обучения по темам лекций	14
Углубленный Python	Практические задания по темам лекций Домашние задания в системе дистанционного обучения по темам лекций	14
Многопоточное и асинхронное программирование	Практические задания по темам лекций Домашние задания в системе дистанционного обучения по темам лекций	14
Промежуточная аттестация	Итоговое тестирование	30