

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

**УТВЕРЖДЕНО**

**Директор по цифровизации  
образования**

**Д.И. Гриц**

	<b>Рабочая программа дисциплины (модуля)</b>
<b>по дисциплине:</b>	Основы машинного и глубокого обучения
<b>по направлению:</b>	Материаловедение и технологии материалов
<b>профиль подготовки:</b>	Науки и цифровизация в культурном наследии Физтех-школа Электроники, Фотоники и Молекулярной Физики 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 составили:

А.О. Бугрий, старший методист  
Ж.И. Зубцова, канд. физ.-мат. наук, доцент  
Р.Г. Нейчев, старший преподаватель

Программа обсуждена на заседании центра дополнительного, дополнительного профессионального и онлайн-образования "Пуск" 27.02.2023

## Аннотация

Дисциплина состоит из двух модулей:

Модуль 1. Основы программирования на Python

Модуль 2. Основы машинного обучения

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

- совершенствование компетенций студентов в области решения профессиональных задач по машинному и глубокому обучению.

#### 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
УК-1 Способен осуществлять критический анализ проблемных ситуаций на основе системного подхода, вырабатывать стратегию действий	УК-1.1 Анализирует проблемную ситуацию как систему, выявляя ее составляющие и связи между ними

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

В результате освоения дисциплины обучающиеся должны знать:

- условия применения линейных моделей;
- основные подходы градиентной оптимизации;
- формальную постановку задачи машинного обучения с учителем;
- формальную постановку задачи линейной классификации;
- основные архитектуры;
- основные понятия линейной алгебры;
- основные понятия теории вероятностей;
- основные нелинейные функции активации;
- специфические свойства деревьев;
- формальную постановку задачи обработки естественного языка;
- интерфейс SK-learn;
- основные техники регуляризации в глубоком обучении;
- понятия Padding, Strides, Pooling;
- основные понятия машинного обучения;
- наивный Байесовский классификатор;
- линейные модели машинного обучения;
- какие функции потерь применимы для задачи классификации;
- теорему Байеса;
- теорему Гаусса-Маркова;
- теорему Экарта-Янга;
- принципы организации кода;
- понятия коллекций и функций в Python;
- назначение функций в языках программирования;
- сетевые термины (сокеты, клиент, сервер);
- базовые типы и конструкции Python;
- роль коллекций и функций в программировании на Python;
- принцип работы клиент-серверной архитектуры;
- понятия классов и объектов в Python, понимает их взаимосвязь;
- понятие наследования;
- механизмы наследования и его роль в программировании на Python;
- библиотеки Python для обработки данных;
- где находится каталог библиотек;
- понятия тестирования и отладки;
- способы синхронизации потоков;
- особенности работы с глобальным шлюзом GIL;
- несколько встроенных функций языка Python;
- понятие особых методов классов;
- роль особых классов в программировании на Python;
- принцип работы асинхронного взаимодействия;
- особенности объектно-ориентированной модели в Python;
- механизм формирования исключений;
- конструкции языка Python для создания потоков;
- примеры асинхронного программирования;
- принцип работы системы Git;
- понятия процессов и потоков,

уметь:

- выполнять практические задачи и проекты в команде;
- решать задачи классификации и регрессии методом ближайших соседей;
- находить производную функции, экстремумы и выпуклости функции, градиент;
- производить оптимизацию гладких функций и условную оптимизацию;
- улучшать качество предсказаний с помощью основных и продвинутых техник ансамблирования;
- находить вероятность и условную вероятность;
- оценивать качество модели для решения задачи классификации;
- выполнять базовые операции с векторами и матрицами;
- решать задачи линейной регрессии;
- производить матричное разложение SVD;
- решать задачи бинарной и мультиклассовой классификации;
- строить модель с регуляризацией в PyTorch;
- работать с векторами и матрицами с помощью NumPy;
- снижать размерность признакового пространства;
- оценивать подвыборку с помощью критерия мисклассификации, энтропийного критерия и критерия Джинни;
- предобрабатывать текстовые данные и извлекать из них признаки;
- загружать данные с помощью Pandas;
- реализовывать ML-pipeline;
- строить дерево решений;
- оценивать значимость признаков с помощью подходов Information gain, LIME, Shap;
- решать практические задачи с помощью наивного Байесовского классификатора;
- производить научные вычисления с помощью SciPy;
- строить графики с Matplotlib;
- реализовывать логистическую регрессию с помощью PyTorch;
- решать задачи классификации и регрессии с помощью решающих деревьев;
- реализовывать градиентный бустинг с помощью CatBoost;
- описывать линейные и нелинейные зависимости с помощью нейросетей;
- строить простейшую нейросеть на PyTorch;
- строить рекуррентную нейронную сеть;
- бороться с проблемой затухающих градиентов и проблемой взрывающихся градиентов;
- применять сверточные слои для обработки изображений;
- решать задачи классификации и регрессии с помощью градиентного бустинга;
- строить модели с помощью LSTM и GRU;
- бороться с переобучением нейросети с помощью регуляризации;
- строить информативные векторные представления слов;
- использовать базовые типы и конструкции Python для написания простых программ;
- сопоставить и выбрать необходимую структуру данных для конкретной практической задачи;
- составить иерархию классов и описать их методы для конкретной практической задачи;
- выделять в задаче на естественном языке необходимость применения многопоточности;
- выделять в задаче на естественном языке необходимость применения базовых конструкций языка;
- определить и спроектировать необходимые классы и методы классов для конкретной предметной области;
- создать программу, использующую несколько потоков, для конкретной практической задачи;
- написать простой код на Python;
- написать собственную функцию на языке Python;
- использовать магические методы в написании собственных программ на Python;
- писать многопоточный код на Python;
- создать программу по описанию задачи на естественном языке с использованием коллекций языка Python;
- использовать конструкции для генерации исключений на Python;
- установить интерпретатор Python себе на компьютер;
- установить среду разработки PyCharm;
- читать и записывать данные из файла;
- устанавливать внешние библиотеки в Python;
- использовать Git для отслеживания истории изменений версий кода;
- протестировать код на Python;
- вручную создавать или устанавливать библиотеки;
- проанализировать код на Python и исправить ошибки в программе;
- использовать возможности библиотеки asyncio для реализации асинхронности;
- использовать коллекции и функции для написания программ на Python;

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	Основы машинного обучения	30	45		60
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

##### 1. Введение в программирование на Python:

Лекция

Вводное видео к курсу

Введение в программирование на Python

О языке Python

Установка интерпретатора Python на Windows

Установка интерпретатора Python на Linux

Работа в IDE PyCharm. Первая программа

Введение в Python

Ввод и вывод данных

Примеры простейших программ

Итоги занятия

Практическая работа

Выполнение заданий на программирование по теме лекции

Самостоятельная работа

Самостоятельное выполнение заданий по теме лекции, изучение дополнительных материалов, тестирование

##### 2. Типы данных. Конструкции языка:

Лекция

Вводное видео

Числовые типы, операции

Строковый тип данных

Логический тип данных

Условный оператор

Инструкция if-elif-else

Цикл while

Цикл for

Оператор continue

Оператор break

Практическая работа

Выполнение заданий на программирование по теме лекции

Самостоятельная работа

Самостоятельное выполнение заданий по теме лекции, изучение дополнительных материалов, тестирование

### 3. Коллекции:

Лекция

Понятие структуры данных

Множества

Индексация строки (прямая, обратная)

Строка, срезы строк

Списки

Срезы списков

Кортежи

Словари

Практическая работа

Выполнение заданий на программирование по теме лекции

Самостоятельная работа

Самостоятельное выполнение заданий по теме лекции, изучение дополнительных материалов, тестирование

### 4. Функции. Работа с файлами:

Лекция

Именные функции, инструкция def

Функции, Аргументы функции

Лямбда-функция

Функция как аргумент. Декораторы

Исключения в python. Конструкция try – except

Работа с файлами: чтение из файла

Работа с файлами: запись в файл

Правила записи кода, PEP8

Практическая работа

Выполнение заданий на программирование по теме лекции

Самостоятельная работа

Самостоятельное выполнение заданий по теме лекции, изучение дополнительных материалов, тестирование

### 5. Классы и объекты:

Лекция

Введение в объектно-ориентированное программирование

Классы и экземпляры классов

Методы

Пример на классы (инкапсуляция, полиморфизм)

Практическая работа

Выполнение заданий на программирование по теме лекции

Самостоятельная работа

Самостоятельное выполнение заданий по теме лекции, изучение дополнительных материалов, тестирование

### 6. Наследование:

Лекция

Наследование в Python

Композиция классов, пример

Пример использования наследования

Практическая работа

Выполнение заданий на программирование по теме лекции

Самостоятельная работа

Самостоятельное выполнение заданий по теме лекции, изучение дополнительных материалов, тестирование

7. Особые методы классов. Механизм работы классов:

Лекция

Особые методы классов. Механизм работы классов

Магические (специальные) методы

Перегрузка операторов

Итераторы

Контекстные менеджеры

Дескрипторы

Практическая работа

Выполнение заданий на программирование по теме лекции

Самостоятельная работа

Самостоятельное выполнение заданий по теме лекции, изучение дополнительных материалов, тестирование

8. Работа с ошибками.

Лекция

Классы исключений и их обработка

Генерация исключений

Исключения в requests, пример

Практическая работа

Выполнение заданий на программирование по теме лекции

Самостоятельная работа

Самостоятельное выполнение заданий по теме лекции, изучение дополнительных материалов, тестирование

9. Установка внешних библиотек. Работа с Git.

Лекция

Подключение модуля из стандартной библиотеки

Установка внешних библиотек Python

Использование псевдонимов. Инструкция from

Создание своего модуля на Python

Установка python-пакетов с помощью pip

Отладка

Тестирование

Git, работа с распределёнными системами управления версиями

Практическая работа

Выполнение заданий на программирование по теме лекции

Самостоятельная работа

Самостоятельное выполнение заданий по теме лекции, изучение дополнительных материалов, тестирование

10. Процессы и потоки.

Лекция

Процессы и потоки

Процесс и его характеристики

Создание процессов

Создание потоков

Синхронизация потоков

Использование потоков threading

Многопроцессорная обработка multiprocessing

Практическая работа

Выполнение заданий на программирование по теме лекции

Самостоятельная работа

Самостоятельное выполнение заданий по теме лекции, изучение дополнительных материалов, тестирование

11. Работа с сетью. Сокеты.

Лекция

Работа с сетью, сокеты

Сокеты, программа клиент-сервер

Таймауты и обработка сетевых ошибок

Одновременная обработка нескольких соединений

Практическая работа

Выполнение заданий на программирование по теме лекции

Самостоятельная работа

Самостоятельное выполнение заданий по теме лекции, изучение дополнительных материалов, тестирование

12. Асинхронное программирование.

Лекция

Асинхронное программирование

Обработка запросов в один поток, модуль select

Итераторы и генераторы, в чем разница?

Генераторы и сопрограммы

Практическая работа

Выполнение заданий на программирование по теме лекции

Самостоятельная работа

Самостоятельное выполнение заданий по теме лекции, изучение дополнительных материалов, тестирование

13. Зачет

Семестр: 2 (Весенний)

2. Основы машинного обучения

1. Модуль 1. Математические основы машинного обучения

1.1. Линейная алгебра. Библиотека NumPy

Лекция

Цели и план занятия

Введение. Сферы применения машинного обучения

Векторы. Основные операции над векторами

Матрицы. Основные операции над матрицами

Инструкция по настройке локальной машины

Библиотека NumPy. Линейная алгебра в NumPy

Практическая работа

Выполнение задания на программирование

Самостоятельная работа

Самостоятельное выполнение заданий по теме лекции, изучение дополнительных материалов.

1.2. Метод ближайших соседей

Лекция

Цели и план занятия

Основные понятия машинного обучения

Формальная задача

Метод k ближайших соседей

Реализация kNN в Python



Практическая работа  
Выполнение задания на программирование  
Самостоятельная работа  
Самостоятельное выполнение заданий по теме лекции, изучение дополнительных материалов.

### 1.3. Случайность. Наивный Байесовский классификатор

Лекция  
Цели и план занятия  
Вероятность. Свойства вероятности  
Условная вероятность. Теорема Байеса  
Наивный Байесовский классификатор  
Реализация наивного байесовского классификатора  
Эмпирические функции распределения  
Практическая работа  
Выполнение задачи на программирование по теме урока, тестирование  
Самостоятельная работа  
Самостоятельное выполнение заданий по теме урока, изучение дополнительных материалов

### 1.4. Оптимизация

Лекция  
Цели и план занятия  
Производная и ее применения  
Градиентная оптимизация  
Условная оптимизация  
Решение задачи оптимизации градиентными методами  
Практическая работа  
Выполнение заданий по теме лекции  
Самостоятельная работа  
Самостоятельное выполнение заданий по теме лекции, изучение дополнительных материалов

### 1.5. Задача регрессии. Линейная регрессия

Лекция  
Цели и план занятия  
Линейные модели машинного обучения  
Линейная регрессия  
Теорема Гаусса-Маркова  
L1 и L2 регуляризация  
Решение линейной регрессии и анализ устойчивости решения  
Практическая работа  
Выполнение задачи на программирование по теме урока, тестирование  
Самостоятельная работа  
Самостоятельное выполнение заданий по теме урока, изучение дополнительных материалов

### 1.6. Обработка и визуализация данных. Матричные разложения

Лекция  
Цели и план занятия  
Задача снижения размерности  
Метод главных компонент  
Визуализация в Python на примере векторных представлений слов  
Практическая работа  
Выполнение задачи на программирование по теме урока, тестирование  
Самостоятельная работа  
Самостоятельное выполнение заданий по теме урока, изучение дополнительных материалов

## 1.7. Задача классификации. Логистическая регрессия

Лекция

Цели и план занятия

Задача линейной классификации

Правдоподобие

Логистическая регрессия

Мультиклассовая классификация

Метрики классификации

Базовое введение в PyTorch

Практическая работа

Выполнение задачи на программирование по теме урока

Самостоятельная работа

Самостоятельное выполнение заданий по теме урока, изучение дополнительных материалов

## 2. Модуль 2. Машинное обучение

### 2.1. Метод опорных векторов. Оценка качества классификации. Методы кросс-валидации

Лекция

Цели и план занятия

Метод опорных векторов

Нелинейный метод опорных векторов

Оценка качества классификации

Методы кросс-валидации

Cross-validation riddle

Практическая работа

Выполнение задачи на программирование по теме урока, тестирование

Самостоятельная работа

Самостоятельное выполнение заданий по теме урока, изучение дополнительных материалов

### 2.2. Решающие деревья и техники ансамблирования

Лекция

Цели и план занятия

Решающие деревья

Процедура построения дерева решений

Критерии информативности: Энтропия

Критерий Джини

Критерии в задаче регрессии. Усечение деревьев

Специфические свойства деревьев

Практическая работа

Выполнение заданий по теме лекции

Самостоятельная работа

Самостоятельное выполнение заданий по теме лекции, изучение дополнительных материалов

### 2.3. Seaborn и практика по деревьям

Лекция

Цели и план занятия

Seaborn и практика по деревьям

Практическая работа

Выполнение задачи на программирование по теме урока

Самостоятельная работа

Самостоятельное выполнение заданий по теме урока, изучение дополнительных материалов

### 2.4. Случайный лес. Продвинутое техники ансамблирования. Дилемма смещения-дисперсии.

Лекция

Цели и план занятия

Техника ансамблирования

RSM

Дилемма смещения

Смешивание

Стекинг

Практическая работа

Выполнение задачи на программирование по теме урока, тестирование

Самостоятельная работа

Самостоятельное выполнение заданий по теме урока, изучение дополнительных материалов

## 2.5. Градиентный бустинг.

Лекция

Цели и план занятия

Бустинг

Градиентный бустинг.

Визуализация градиентного бустинга

CatBoost

Практическая работа

Выполнение заданий по теме лекции

Самостоятельная работа

Самостоятельное выполнение заданий по теме лекции, изучение дополнительных материалов

## 2.6. Градиентный бустинг на практике

Лекция

Цели и план занятия

Градиентный бустинг. Примеры

Градиентный бустинг. Применение в Python

Практическая работа

Выполнение задачи на программирование по теме урока

Самостоятельная работа

Самостоятельное выполнение заданий по теме лекции, изучение дополнительных материалов

## 2.7. Оценка значимости признаков.

Лекция

Цели и план занятия

Information gain и прочие подходы для деревьев

LIME

Shar

Практическая работа

Выполнение задачи на программирование по теме урока, тестирование

Самостоятельная работа

Самостоятельное выполнение заданий по теме лекции, изучение дополнительных материалов

## 2.8. Введение в глубокое обучение.

Лекция

Цели и план занятия

История искусственных нейронных сетей

Нейронные сети

Механизм обратного распространения ошибки

Функции активации

Интерактивное демо

Нейронные сети. Итоги

Простейшая нейросеть на PyTorch

Практическая работа

Выполнение заданий по теме лекции

Самостоятельная работа

Самостоятельное выполнение заданий по теме лекции, изучение дополнительных материалов

### 3. Модуль 3. Глубокое обучение

#### 3.1. SGD доработки

Лекция

Цели и план занятия

SGD доработки

Регуляризация в DL

Проблема переобучения

Аугментация и итоги

PyTorch: модель с регуляризацией

Практическая работа

Выполнение задачи на программирование по теме урока

Самостоятельная работа

Самостоятельное выполнение заданий по теме лекции, изучение дополнительных материалов

#### 3.2. Векторные представления слов.

Лекция

Цели и план занятия

Введение в NLP

Предварительная обработка текста

Извлечение признаков

Векторное представление слов (Word Embeddings)

Векторные представления слов. Визуализация. (Word embeddings visualization)

Визуализация в Python

Практическая работа

Выполнение задачи на программирование по теме урока

Самостоятельная работа

Самостоятельное выполнение заданий по теме лекции, изучение дополнительных материалов

#### 3.3. Рекуррентные нейронные сети. Проблема затухающего градиента.

Лекция

Цели и план занятия

Языковое моделирование

Рекуррентные нейронные сети

RNN

LSTM

Проблема затухающих градиентов

Проблема взрывающихся градиентов

Языковое моделирование: реализация в Python

Рекуррентные нейронные сети: реализация в Python

Практическая работа

Выполнение задачи на программирование по теме урока

Самостоятельная работа

Самостоятельное выполнение заданий по теме лекции, изучение дополнительных материалов

#### 3.4. Обработка изображений. Сверточные нейронные сети. Часть 1.

Лекция

Цели и план занятия

Сверточные слои

Интерактивная демонстрация

Padding, Strides, Pooling

Практическая работа

Выполнение заданий по теме лекции

Самостоятельная работа

Самостоятельное выполнение заданий по теме лекции, изучение дополнительных материалов

### 3.5. Обработка изображений. Сверточные нейронные сети. Часть 2.

Лекция

Цели и план занятия

Обзор архитектур

Свертки для изображений. Базовый обзор, примеры

Практическая работа

Выполнение задачи на программирование по теме урока

Самостоятельная работа

Самостоятельное выполнение заданий по теме лекции, изучение дополнительных материалов

4. Зачет.

5. Итоговый проект

## 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 : в 3 т. Т. 1 : учебник для вузов : рек. М-вом образования Рос. Федерации / Г. М. Фихтенгольц ; пред. и прим. А. А. Флоринского .— 8-е изд. / .— М. : Физматлит, 2001, 2003, 2006, 2007 .— 680 с.
4. Курс аналитической геометрии и линейной алгебры [Текст], учебник для вузов /Д. В. Беклемишев. -СПб., Лань, 2019
5. Лекции по математическому анализу. В 3 частях, Часть 1, Функции одной переменной, учебник для вузов/Я. М. Дымарский , -Москва, МФТИ, 2020
6. Python и машинное обучение [Текст], крайне необходимое издание по новейшей предсказательной аналитике для более глубокого понимания методологии машинного обучения/С. Рашка, -М., ДМК Пресс, 2017

Дополнительная литература

1. 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.
2. Аналитическая геометрия и линейная алгебра [Текст] : в 2 ч. : учеб. пособие для вузов. Ч. 1 / А. Е. Умнов ; М-во образования и науки Рос. Федерации, Моск. физико-техн. ин-т (гос. ун-т .— 2-е изд., испр. и доп. — М. : Изд-во МФТИ, 2006 .— 272 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>	Материаловедение и технологии материалов
<b>профиль подготовки:</b>	Науки и цифровизация в культурном наследии Физтех-школа Электроники, Фотоники и Молекулярной Физики 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

**Разработчики:**

А.О. Бугрий, старший методист  
Ж.И. Зубцова, канд. физ.-мат. наук, доцент  
Р.Г. Нейчев, старший преподаватель

## 1. Компетенции, формируемые в процессе изучения дисциплины

Код и наименование компетенции	Индикаторы достижения компетенции
УК-1 Способен осуществлять критический анализ проблемных ситуаций на основе системного подхода, вырабатывать стратегию действий	УК-1.1 Анализирует проблемную ситуацию как систему, выявляя ее составляющие и связи между ними

## 2. Показатели оценивания компетенций

В результате изучения дисциплины «Основы машинного и глубокого обучения» обучающийся должен:

### знать:

- условия применения линейных моделей;
- основные подходы градиентной оптимизации;
- формальную постановку задачи машинного обучения с учителем;
- формальную постановку задачи линейной классификации;
- основные архитектуры;
- основные понятия линейной алгебры;
- основные понятия теории вероятностей;
- основные нелинейные функции активации;
- специфические свойства деревьев;
- формальную постановку задачи обработки естественного языка;
- интерфейс SK-learn;
- основные техники регуляризации в глубоком обучении;
- понятия Padding, Strides, Pooling;
- основные понятия машинного обучения;
- наивный Байесовский классификатор;
- линейные модели машинного обучения;
- какие функции потерь применимы для задачи классификации;
- теорему Байеса;
- теорему Гаусса-Маркова;
- теорему Экарта-Янга;
- принципы организации кода;
- понятия коллекций и функций в Python;
- назначение функций в языках программирования;
- сетевые термины (сокеты, клиент, сервер);
- базовые типы и конструкции Python;
- роль коллекций и функций в программировании на Python;
- принцип работы клиент-серверной архитектуры;
- понятия классов и объектов в Python, понимает их взаимосвязь;
- понятие наследования;
- механизмы наследования и его роль в программировании на Python;
- библиотеки Python для обработки данных;
- где находится каталог библиотек;
- понятия тестирования и отладки;
- способы синхронизации потоков;
- особенности работы с глобальным шлюзом GIL;
- несколько встроенных функций языка Python;
- понятие особых методов классов;
- роль особых классов в программировании на Python;
- принцип работы асинхронного взаимодействия;
- особенности объектно-ориентированной модели в Python;
- механизм формирования исключений;
- конструкции языка Python для создания потоков;
- примеры асинхронного программирования;
- принцип работы системы Git;
- понятия процессов и потоков,

### уметь:



- выполнять практические задачи и проекты в команде;
- решать задачи классификации и регрессии методом ближайших соседей;
- находить производную функции, экстремумы и выпуклости функции, градиент;
- производить оптимизацию гладких функций и условную оптимизацию;
- улучшать качество предсказаний с помощью основных и продвинутых техник ансамблирования;
- находить вероятность и условную вероятность;
- оценивать качество модели для решения задачи классификации;
- выполнять базовые операции с векторами и матрицами;
- решать задачи линейной регрессии;
- производить матричное разложение SVD;
- решать задачи бинарной и мультиклассовой классификации;
- строить модель с регуляризацией в PyTorch;
- работать с векторами и матрицами с помощью NumPy;
- снижать размерность признакового пространства;
- оценивать подвыборку с помощью критерия мисклассификации, энтропийного критерия и критерия Джинни;
- предобрабатывать текстовые данные и извлекать из них признаки;
- загружать данные с помощью Pandas;
- реализовывать ML-pipeline;
- строить дерево решений;
- оценивать значимость признаков с помощью подходов Information gain, LIME, Shap;
- решать практические задачи с помощью наивного Байесовского классификатора;
- производить научные вычисления с помощью SciPy;
- строить графики с Matplotlib;
- реализовывать логистическую регрессию с помощью PyTorch;
- решать задачи классификации и регрессии с помощью решающих деревьев;
- реализовывать градиентный бустинг с помощью CatBoost;
- описывать линейные и нелинейные зависимости с помощью нейросетей;
- строить простейшую нейросеть на PyTorch;
- строить рекуррентную нейронную сеть;
- бороться с проблемой затухающих градиентов и проблемой взрывающихся градиентов;
- применять сверточные слои для обработки изображений;
- решать задачи классификации и регрессии с помощью градиентного бустинга;
- строить модели с помощью LSTM и GRU;
- бороться с переобучением нейросети с помощью регуляризации;
- строить информативные векторные представления слов;
- использовать базовые типы и конструкции Python для написания простых программ;
- сопоставить и выбрать необходимую структуру данных для конкретной практической задачи;
- составить иерархию классов и описать их методы для конкретной практической задачи;
- выделять в задаче на естественном языке необходимость применения многопоточности;
- выделять в задаче на естественном языке необходимость применения базовых конструкций языка;
- определить и спроектировать необходимые классы и методы классов для конкретной предметной области;
- создать программу, использующую несколько потоков, для конкретной практической задачи;
- написать простой код на Python;
- написать собственную функцию на языке Python;
- использовать магические методы в написании собственных программ на Python;
- писать многопоточный код на Python;
- создать программу по описанию задачи на естественном языке с использованием коллекций языка Python;
- использовать конструкции для генерации исключений на Python;
- установить интерпретатор Python себе на компьютер;
- установить среду разработки PyCharm;
- читать и записывать данные из файла;
- устанавливать внешние библиотеки в Python;
- использовать Git для отслеживания истории изменений версий кода;
- протестировать код на Python;
- вручную создавать или устанавливать библиотеки;
- проанализировать код на Python и исправить ошибки в программе;
- использовать возможности библиотеки asyncio для реализации асинхронности;
- использовать коллекции и функции для написания программ на Python;

**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**

## Тестовые задания

1. Какая версия языка Python является актуальной в настоящее время?

- Python 3.X
- Python 2.6.X
- Python 1.X

2. Что такое инструкция в языках программирования?

- команда для выполнения какого-либо действия
- документ с перечислением всех команд языка
- данные, которые пользователь вводит с консоли в программу
- спецсимволы, используемые для написания кода

3. Какие элементы используют для формирования структуры программы в Python? Выберите все верные варианты.

- пробелы
- двоеточие
- табуляция
- фигурные скобки
- команды языка
- знак #

4. Людмила пишет программу на языке Python. Какую команду ей нужно записать на место пропуска, что вывести на экран сумму чисел a и b? Запишите ответ строчными буквами.

Код программы Людмилы.

a = 2

b = 4

## Практические задания

## Задание на программирование

## Цель заданий

В результате выполнения заданий вы сможете:

- разобраться с особенностями написания кода в LMS
- писать программы на Python с использованием ввода/вывода в консоль

## Задания:

## 1. Приветствие

На вход программа получает строку — имя пользователя. Считайте её и выведите на экран фразу: «Привет, [имя]!».

Обратите внимание, что выводимая строка должна быть написана в точности так, как в задании. Если будет отличаться регистр букв или будут отсутствовать знаки препинания, программа засчитана не будет.

Ввод    Вывод

Валентин Привет, Валентин!

Авторское решение:

```
name = input()
print('Привет, ', name, '!', sep="")
```

Тесты

№ теста Стандартный ввод Ожидаемый результат

- 1 Валентин Привет, Валентин!
- 2 Дмитрий Привет, Дмитрий!
- 3 Мария Привет, Мария!
- 4 АБВГД123 Привет, АБВГД123!

2. Вычисление значения предыдущего числа и следующего

На вход программа получает целое число  $a$ . Получите значения предыдущего числа и следующего. Выведите результат  $a-1$ ,  $a$ ,  $a+1$  по одному числу в строку.

Ввод Результат

5 4  
5  
6

Авторское решение

```
a = int(input())
print(a - 1)
print(a)
print(a + 1)
```

Тесты

Тест 1:

ввод Ожидаемый результат

5 4  
5  
6

Тест 2:

ввод Ожидаемый результат

0 -1  
0  
1

Тест 3:

ввод Ожидаемый результат

-300000 -300001  
-300000  
-299999

ввод	Ожидаемый результат
103 102	
103	
104	

[illegible]

Возраст питомца: 5

Возраст питомца: 5

британская длинношерстная

4 Ваш питомец: кошка Маруся  
Порода питомца: британская длинношерстная  
Возраст питомца: 4

Тест 3:

Ввод Вывод

Бонни

собака

бульдог

8 Ваш питомец: собака Бонни

Порода питомца: бульдог

Возраст питомца: 8

Примеры тестовых вопросов:

1. Объект описывается 10 числовыми признаками. Сколько параметров у линейной регрессионной модели (целевая переменная – скаляр)?
2. В каком случае предпочтительно выбирать MAE в качестве функции потерь в задаче регрессии?
3. Для каких случаев доступно аналитическое решение задачи регрессии?
4. Что общего имеют PCA и kNN?
5. Какой способ регуляризации в линейной регрессии имеет тенденцию к "отбору признаков"?

Пример задания на программирование

В данном задании вам необходимо реализовать функции ошибки для линейной регрессии и их производные по параметрам, не используя автоматическое дифференцирование. Все методы должны быть реализованы только с использованием библиотеки numpy.

Ваша основная задача: вывести формулы для производных MSE, MAE, L1 и L2 регуляризационных членов в векторном случае (т.е. когда и объект `_`, и целевое значение `_` являются векторами).

Для работы вновь обратимся к Boston housing prices dataset. Он был предобработан для вашего удобства и будет загружен ниже.

```
# Run some setup code for this notebook.
```

```
import random
import numpy as np
import matplotlib.pyplot as plt
import json
with open('boston_subset.json', 'r') as iofile:
    dataset = json.load(iofile)
feature_matrix = np.array(dataset['data'])
targets = np.array(dataset['target'])
```

Имплементация функций потерь и методов регуляризации.

Для того, чтобы решить задание, вам необходимо реализовать все методы в файле `loss_and_derivatives.py`. Внимание, в данном задании не требуется использовать свободный член (bias term), т.е. линейная модель примет простой вид

$\hat{y} =$

Единичный столбец также не добавляется к матрице .

Реализуйте методы для MSE, MAE, L1 и L2 регуляризации, а также вычисления их производных (опциональное задание) по параметрам линейной модели.

Для вашего удобства данные уже предобработаны, и использование линейной модели без свободного члена не является ошибкой. В данном задании он не должен быть использован.

```
import numpy as np
```

```
class LossAndDerivatives:
```

```
    @staticmethod
```

```
    def mse(X, Y, w):
```

```

"""
X : numpy array of shape (`n_observations`, `n_features`)
Y : numpy array of shape (`n_observations`, `target_dimensionality`) or (`n_observations`,)
w : numpy array of shape (`n_features`, `target_dimensionality`) or (`n_features`,)

Return : float
    single number with MSE value of linear model (X.dot(w)) with no bias term
    on the selected dataset.

Comment: If Y is two-dimensional, average the error over both dimensions.
"""

return np.mean((X.dot(w) - Y)**2)

@staticmethod
def mae(X, Y, w):
    """
    X : numpy array of shape (`n_observations`, `n_features`)
    Y : numpy array of shape (`n_observations`, `target_dimensionality`) or (`n_observations`,)
    w : numpy array of shape (`n_features`, `target_dimensionality`) or (`n_features`,)

    Return: float
        single number with MAE value of linear model (X.dot(w)) with no bias term
        on the selected dataset.

    Comment: If Y is two-dimensional, average the error over both dimensions.
    """

    # YOUR CODE HERE
    return

@staticmethod
def l2_reg(w):
    """
    w : numpy array of shape (`n_features`, `target_dimensionality`) or (`n_features`,)

    Return: float
        single number with sum of squared elements of the weight matrix (  $\sum_{ij} w_{ij}^2$  )

    Computes the L2 regularization term for the weight matrix w.
    """

    # YOUR CODE HERE
    return

@staticmethod
def l1_reg(w):
    """
    w : numpy array of shape (`n_features`, `target_dimensionality`)

    Return : float
        single number with sum of the absolute values of the weight matrix (  $\sum_{ij} |w_{ij}|$  )

    Computes the L1 regularization term for the weight matrix w.

```

```
"""
```

```
# YOUR CODE HERE
```

```
return
```

```
@staticmethod
```

```
def no_reg(w):
```

```
    """
```

```
    Simply ignores the regularization
```

```
    """
```

```
    return 0.
```

```
@staticmethod
```

```
def mse_derivative(X, Y, w):
```

```
    """
```

```
    X : numpy array of shape ('n_observations', 'n_features')
```

```
    Y : numpy array of shape ('n_observations', 'target_dimensionality') or ('n_observations',)
```

```
    w : numpy array of shape ('n_features', 'target_dimensionality') or ('n_features',)
```

```
    Return : numpy array of same shape as 'w'
```

```
    Computes the MSE derivative for linear regression (X.dot(w)) with no bias term  
    w.r.t. w weight matrix.
```

```
    Please mention, that in case 'target_dimensionality' > 1 the error is averaged along this  
    dimension as well, so you need to consider that fact in derivative implementation.
```

```
    """
```

```
# YOUR CODE HERE
```

```
return
```

```
@staticmethod
```

```
def mae_derivative(X, Y, w):
```

```
    """
```

```
    X : numpy array of shape ('n_observations', 'n_features')
```

```
    Y : numpy array of shape ('n_observations', 'target_dimensionality') or ('n_observations',)
```

```
    w : numpy array of shape ('n_features', 'target_dimensionality') or ('n_features',)
```

```
    Return : numpy array of same shape as 'w'
```

```
    Computes the MAE derivative for linear regression (X.dot(w)) with no bias term  
    w.r.t. w weight matrix.
```

```
    Please mention, that in case 'target_dimensionality' > 1 the error is averaged along this  
    dimension as well, so you need to consider that fact in derivative implementation.
```

```
    """
```

```
# YOUR CODE HERE
```

```
return
```

```
@staticmethod
```

```
def l2_reg_derivative(w):
```

```
    """
```

```
    w : numpy array of shape ('n_features', 'target_dimensionality') or ('n_features',)
```

Return : numpy array of same shape as `w`

Computes the L2 regularization term derivative w.r.t. the weight matrix w.

"""

# YOUR CODE HERE

return

@staticmethod

def l1\_reg\_derivative(w):

"""

Y : numpy array of shape (`n\_observations`, `target\_dimensionality`) or (`n\_observations`,)

w : numpy array of shape (`n\_features`, `target\_dimensionality`) or (`n\_features`,)

Return : numpy array of same shape as `w`

Computes the L1 regularization term derivative w.r.t. the weight matrix w.

"""

# YOUR CODE HERE

return

@staticmethod

def no\_reg\_derivative(w):

"""

Simply ignores the derivative

"""

return np.zeros\_like(w)

Обращаем ваше внимание, требуется реализовать решение в векторном виде (т.е. для каждого объекта предсказание  $\hat{y}$  является вектором с размерностью  $\geq 1$ . При подсчете ошибки она усредняется как по объектам, так и по размерности  $y$ .

Например, для вектора ошибок на одном объекте  $[1., 1., 1., 1.]$  значение функции ошибки будет равно  $\frac{1}{4} (1.+1.+1.+1.)$

Для вашего удобства метод `.mse` уже реализован и вы можете обращаться к нему за примером.

Для проверки своего кода вам доступно несколько assert'ов:

```
w = np.array([1., 1.])
```

```
x_n, y_n = feature_matrix, targets
```

```
# Repeating data to make everything multi-dimensional
```

```
w = np.vstack([w[None, :] + 0.27, w[None, :] + 0.22, w[None, :] + 0.45, w[None, :] + 0.1]).T
```

```
y_n = np.hstack([y_n[:, None], 2*y_n[:, None], 3*y_n[:, None], 4*y_n[:, None]])
```

```
reference_mse_derivative = np.array([
    [ 7.32890068, 12.88731311, 18.82128365, 23.97731238],
    [ 9.55674399, 17.05397661, 24.98807528, 32.01723714]
])
```

```
reference_l2_reg_derivative = np.array([
    [2.54, 2.44, 2.9, 2.2 ],
    [2.54, 2.44, 2.9, 2.2 ]
])
```

```
assert np.allclose(
    reference_mse_derivative,
    LossAndDerivatives.mse_derivative(x_n, y_n, w), rtol=1e-3
```



```

), 'Something wrong with MSE derivative'

assert np.allclose(
    reference_l2_reg_derivative,
    LossAndDerivatives.l2_reg_derivative(w), rtol=1e-3
), 'Something wrong with L2 reg derivative'

print(
    'MSE derivative:\n{} \n\nL2 reg derivative:\n{}'.format(
        LossAndDerivatives.mse_derivative(x_n, y_n, w),
        LossAndDerivatives.l2_reg_derivative(w))
)

reference_mae_derivative = np.array([
    [0.19708867, 0.19621798, 0.19621798, 0.19572906],
    [0.25574138, 0.25524507, 0.25524507, 0.25406404]
])
reference_l1_reg_derivative = np.array([
    [1., 1., 1., 1.],
    [1., 1., 1., 1.]
])

assert np.allclose(
    reference_mae_derivative,
    LossAndDerivatives.mae_derivative(x_n, y_n, w), rtol=1e-3
), 'Something wrong with MAE derivative'

assert np.allclose(
    reference_l1_reg_derivative,
    LossAndDerivatives.l1_reg_derivative(w), rtol=1e-3
), 'Something wrong with L1 reg derivative'

print(
    'MAE derivative:\n{} \n\nL1 reg derivative:\n{}'.format(
        LossAndDerivatives.mae_derivative(x_n, y_n, w),
        LossAndDerivatives.l1_reg_derivative(w))
)

```

Градиентный спуск для решения реальной задачи

Следующая функция позволяет найти оптимальные значения параметров с помощью градиентного спуска:

```

def get_w_by_grad(X, Y, w_0, loss_mode='mse', reg_mode=None, lr=0.05, n_steps=100,
reg_coeff=0.05):
    if loss_mode == 'mse':
        loss_function = LossAndDerivatives.mse
        loss_derivative = LossAndDerivatives.mse_derivative
    elif loss_mode == 'mae':
        loss_function = LossAndDerivatives.mae
        loss_derivative = LossAndDerivatives.mae_derivative
    else:
        raise ValueError('Unknown loss function. Available loss functions: `mse`, `mae`)

    if reg_mode is None:
        reg_function = LossAndDerivatives.no_reg

```

```

    reg_derivative = LossAndDerivatives.no_reg_derivative # lambda w: np.zeros_like(w)
elif reg_mode == 'l2':
    reg_function = LossAndDerivatives.l2_reg
    reg_derivative = LossAndDerivatives.l2_reg_derivative
elif reg_mode == 'l1':
    reg_function = LossAndDerivatives.l1_reg
    reg_derivative = LossAndDerivatives.l1_reg_derivative
else:
    raise ValueError('Unknown regularization mode. Available modes: `l1`, `l2`, None')

w = w_0.copy()

for i in range(n_steps):
    empirical_risk = loss_function(X, Y, w) + reg_coeff * reg_function(w)
    gradient = loss_derivative(X, Y, w) + reg_coeff * reg_derivative(w)
    gradient_norm = np.linalg.norm(gradient)
    if gradient_norm > 5.:
        gradient = gradient / gradient_norm * 5.
    w -= lr * gradient

    if i % 25 == 0:
        print('Step={}, loss={},\ngradient values={}\n'.format(i, empirical_risk, gradient))
return w

```

Рассмотрим простой пример:

```

# Initial weight matrix
w = np.ones((2,1), dtype=float)
y_n = targets[:, None]
w_grad = get_w_by_grad(x_n, y_n, w, loss_mode='mse', reg_mode='l2', n_steps=250)

```

Сравнение с sklearn

Сравним реализованную модель с версией из sklearn:

```

from sklearn.linear_model import Ridge
lr = Ridge(alpha=0.05)
lr.fit(x_n, y_n)
print('sklearn linear regression implementation delivers MSE = {}'.format(np.mean((lr.predict(x_n) -
y_n)**2)))
plt.scatter(x_n[:, -1], y_n[:, -1])
plt.scatter(x_n[:, -1], x_n.dot(w_grad)[:, -1], color='orange', label='Handwritten linear regression',
linewidth=5)
plt.scatter(x_n[:, -1], lr.predict(x_n), color='cyan', label='sklearn Ridge')
plt.legend()
plt.show()

```

Сдача задания

Сдайте в чекер реализованный класс LossAndDerivatives. Для этого можете скопировать всю ячейку с кодом (в том числе и импортирование numpy) в файл derivatives.py.

На этом задание завершено. Поздравляем!

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

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 измерявшимся показателям (включая пульс, давление, ЭКГ и т.д.) оценивался риск возникновения сердечного заболевания. Στα 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  $l_y$ ?
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?

## 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 30 баллов.

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

Во время проведения зачета обучающиеся могут пользоваться программой дисциплины.