Федеральное государственное бюджетное образовательное учреждение высшего образования
«САМАРСКИЙ ГОСУДАРСТВЕННЫЙ МЕДИЦИНСКИЙ УНИВЕРСИТЕТ»
Министерства здравоохранения Российской Федерации
Институт профессионального образования
Передовая медицинская инженерная школа

Утверждено
на заседании
Координационного совета ПМИШ
протокол № 2 от «06» марта 2023 г.

Дополнительная профессиональная программа повышения квалификации инженеров по направлению подготовки 09.04.01 Информатика и вычислительная техника «Инженерия искусственного интеллекта в медицине» со сроком освоения 144 часа (форма обучения очно-заочная)

Самара, 2023

Разработчики:

Иващенко Антон Владимирович – доктор технических наук, профессор, директор Передовой медицинской инженерной школы

Палевская Светлана Александровна – доктор медицинских наук, MBA, проректор по профессиональному образованию и межрегиональному взаимодействию – директор ИПО

Чертыковцева Наталья Валерьевна – кандидат технических наук, заместитель директора Передовой медицинской инженерной школы

ДОПОЛНИТЕЛЬНАЯ ПРОГРАММА
ПОВЫШЕНИЯ КВАЛИФИКАЦИИ
«ИНЖЕНЕРИЯ ИСКУССТВЕННОГО ИНТЕЛЛЕКТА В МЕДИЦИНЕ»

Дополнительная профессиональная программа повышения квалификации «Инженерия искусственного интеллекта в медицине» (далее – программа) разработана на основе ФГОС ВО по направлению подготовки 09.04.01 Информатика и вычислительная техника (уровень магистратуры), утвержденного приказом Минобрнауки России от 19 сентября 2017 г. № 918, а также профессионального стандарта «Руководитель разработки программного обеспечения» утвержденного приказом Министерства труда и социальной защиты РФ от 22 июля 2022 г. № 423н.

I. ОБЩИЕ ПОЛОЖЕНИЯ

Цель Программы: подготовка инженерных кадров, способных решать задачи разработки новых моделей нейронных сетей, адаптации и внедрения элементов искусственного интеллекта в прикладных решениях; задачи разработки систем диагностики на основе анализа больших данных и искусственного интеллекта.

Направленность Программы: совершенствование, повышение профессионального уровня в рамках имеющейся квалификации, а также качественное изменение профессиональных компетенций в области информационных технологий; обучение навыкам разработки программного обеспечения на основе анализа больших данных и искусственного интеллекта для реализации проектов по цифровой трансформации отрасли здравоохранения.

Задачи Программы:

- Обновление существующих теоретических знаний, методик и изучение передового практического опыта в области применения современных информационных технологий, технологий искусственного интеллекта и робототехники в медицинских приложениях.
- Усвоение и закрепление на практике профессиональных знаний, умений и навыков, обеспечивающих совершенствование профессиональных компетенций по вопросам разработки программного обеспечения на основе анализа больших данных и искусственного интеллекта для реализации проектов по цифровой трансформации отрасли здравоохранения.

Трудоемкость освоения: 144 академических часов.

Форма обучения: очно-заочная, с применением ДОТ

Категория обучающихся:

К обучению по программе профессиональной переподготовки допускаются лица, получившие высшее образование по очной (очно-заочной) форме, лица, освоившие основную образовательную программу бакалавриата, специалитета, а также магистратуры по инженерным специальностям.

Основными компонентами Программы являются:

- цель программы;
- планируемые результаты обучения;
- учебный план;
- требования к итоговой аттестации обучающихся;
- календарный учебный график;
- организационно-педагогические условия реализации Программы;
- оценочные материалы и иные компоненты.

Содержание Программы построено в соответствии с модульным принципом, структурными единицами модуля являются разделы. Каждый раздел дисциплины подразделяется на темы, каждая тема - на элементы, каждый элемент - на подэлементы. Для удобства пользования программой в учебном процессе каждая его структурная единица кодируется. На первом месте ставится код раздела дисциплины (например, 1), на втором - код темы (например, 1.1), далее - код элемента (например, 1.1.1), затем – код подэлемента (например, 1.1.1.1). Кодировка вносит определенный порядок в перечень вопросов, содержащихся в программе, что, в свою очередь, позволяет кодировать контрольно-измерительные (тестовые) материалы в учебно-методическом комплексе (далее - УМК).

Учебный план определяет состав изучаемых дисциплин с указанием их трудоемкости, объема, последовательности и сроков изучения, устанавливает формы организации учебного процесса и их соотношение (лекции, лабораторные и практические занятия), конкретизирует формы контроля знаний и умений обучающихся.

Результаты обучения: в программу включены планируемые результаты обучения. Планируемые результаты обучения направлены на совершенствование профессиональных компетенций, профессиональных знаний, умений, навыков в области применения современных информационных технологий, технологий искусственного интеллекта и робототехники в медицинских приложениях. В планируемых результатах отражается преемственность с профессиональным стандартом «Руководитель разработки программного обеспечения».

В Программе содержатся требования к аттестации обучающихся. Итоговая аттестация по Программе осуществляется посредством проведения зачета и выявляет теоретическую и практическую подготовку слушателя в соответствии с целями и содержанием программы.

Организационно-педагогические условия реализации программы.

Условия реализации Программы включают:

а) учебно-методическую документацию и материалы по всем разделам (модулям) курса;

б) учебно-методическую литературу для внеаудиторной работы обучающихся;

в) материально-технические базы, обеспечивающие организацию всех видов дисциплинарной подготовки:

- учебные аудитории, оснащенные материалами и оборудованием для проведения учебного процесса;
- базы практик в медицинских и научных организациях;

в) кадровое обеспечение реализации Программы соответствует требованиям штатного расписания Передовой медицинской инженерной школы, реализующей дополнительные профессиональные программы;

г) законодательство Российской Федерации.

II. ПЛАНИРУЕМЫЕ РЕЗУЛЬТАТЫ ОБУЧЕНИЯ СЛУШАТЕЛЕЙ, УСПЕШНО ОСВОИВШИХ ПРОГРАММУ

Результаты обучения по Программе направлены на совершенствование компетенций, приобретенных в рамках полученного ранее профессионального образования на основе Федеральных государственных образовательных стандартов высшего образования по инженерным специальностям и на формирование профессиональных компетенций в рамках имеющейся квалификации, качественное изменение которых осуществляется в результате обучения.

Характеристика профессиональных компетенций подлежащих совершенствованию в результате освоения Программы

В результате освоения программы слушатель совершенствует следующую профессиональную компетенцию:

- Способен руководить проектированием специализированного программного обеспечения на основе технологий искусственного интеллекта в рамках цифровой трансформации сферы здравоохранения (ПК-1).

Трудовые действия (функции) руководителя разработки программного обеспечения

Трудовая функция (профессиональная компетенция)	Трудовые действия	Необходимые умения	Необходимые знания
Обобщенная трудовая функция: организация процессов разработки компьютерного программного обеспечения			
B/01.7 Управление проектированием компьютерного программного обеспечения	Анализ архитектуры компьютерного программного обеспечения и ее согласование с заинтересованными сторонами Распределение заданий на проектирование компьютерного программного обеспечения, структуры базы данных, программных интерфейсов Оценка качества проектирования компьютерного программного обеспечения, структуры базы данных, программных интерфейсов Принятие управленческих решений по результатам проектирования компьютерного программного обеспечения, структуры базы данных, программных интерфейсов	Применять принципы построения архитектуры компьютерного программного обеспечения и виды архитектуры программного обеспечения Применять методологии и средства проектирования компьютерного программного обеспечения Применять методы и средства проектирования баз данных Применять методы и средства проектирования программных интерфейсов Применять основные принципы и методы управления персоналом Применять методологию функциональной стандартизации для открытых систем Применять методы принятия управленческих решений Применять нормативно-технические документы (стандарты и регламенты) по процессу разработки архитектуры компьютерного программного обеспечения Осуществлять рабочие коммуникации с подразделениями организации и заинтересованными сторонами в рамках процесса проектирования компьютерного программного обеспечения, структуры	Принципы построения архитектуры компьютерного программного обеспечения и виды архитектуры программного обеспечения Методологии и средства проектирования компьютерного программного обеспечения Методы и средства проектирования баз данных Методы и средства проектирования программных интерфейсов Методы принятия управленческих решений Основные принципы и методы управления персоналом Методология функциональной стандартизации для открытых систем Технологии межличностной и групповой коммуникации в деловом взаимодействии, основы конфликтологии

		базы данных, программных интерфейсов	

Перечень знаний, умений и навыков, обеспечивающих формирование профессиональных компетенций

По окончании обучения слушатель должен знать:

- принципы построения архитектуры компьютерного программного обеспечения и виды архитектуры программного обеспечения
- методологии и средства проектирования компьютерного программного обеспечения
- методы и средства проектирования баз данных
- методы и средства проектирования программных интерфейсов
- основные модели искусственного интеллекта;
- принципы функционирования нейрокомпьютерных интерфейсов

По окончании обучения слушатель должен уметь:

- применять принципы построения архитектуры компьютерного программного обеспечения и виды архитектуры программного обеспечения
- применять методологии и средства проектирования компьютерного программного обеспечения
- применять методы и средства проектирования баз данных
- применять методы и средства проектирования программных интерфейсов
- применять нормативно-технические документы (стандарты и регламенты) по процессу разработки архитектуры компьютерного программного обеспечения
- формировать сборку интегральной модели из элементов искусственного интеллекта для решения задач аугментации и нейрореабилитации

По окончании обучения слушатель должен владеть:

- навыками анализа архитектуры компьютерного программного обеспечения;
- навыками распределения заданий на проектирование компьютерного программного обеспечения, структуры базы данных, программных интерфейсов;
- навыками оценки качества проектирования компьютерного программного обеспечения, структуры базы данных, программных интерфейсов;
- навыками пересборки интегральной модели искусственного интеллекта для повышения продуктивности решения задач аугментации и нейрореабилитации

Ш. РАБОЧАЯ ПРОГРАММА

Код	Наименования тем, элементов и подэлементов
1.	Искусственный интеллект в медицине
1.1.	Внедрение искусственного интеллекта в медицинскую сферу.
1.2.	Методы обучения ИИ на основе разнородной информации.
2.	Обучение с учителем и без учителя.
2.1.	Модель линейной регрессии. Модель логистической регрессии. Метод опорных векторов.

2.2.	Деревья принятия решений и случайный лес.
2.3.	Методы K-Means & PCA
3	Классификация снимков КТ с помощью сверточных нейронных сетей.
3.1.	Отладка модели сверточной нейронной сети.
3.2.	Классификация медицинских изображений сверточной нейронной сетью
4.	Построение представлений из многоканальных временных рядов ЭЭГ.
4.1	Классификация умственной нагрузки
4.2	Повышение точности классификации умственной нагрузки по ЭЭГ
5.	Проектно-технологическая практика
5.1	Работа в группе по проведению исследования в рамках индивидуального задания.
5.2	Разработка инструмента для анализа медицинских данных и интерпретации их результатов.

IV. УЧЕБНЫЙ ПЛАН ПРОГРАММЫ

Цель: подготовка инженерных кадров, способных решать задачи разработки новых моделей нейронных сетей, адаптации и внедрения элементов искусственного интеллекта в прикладных решениях; задачи разработки систем диагностики на основе анализа больших данных и искусственного интеллекта.

Категория обучающихся: получившие высшее образование по очной (очно-заочной) форме, лица, освоившие основную образовательную программу бакалавриата, специалитета, а также магистратуры по инженерным специальностям.

Трудоемкость обучения: 144 академических часов

Форма обучения: очно-заочная, с применением ДОТ

№ п/п	Наименование раздела (модуля)	Всего часов	В том числе			Форма контроля
			Л	ПЗ	СР	
1.	Искусственный интеллект в медицине	34	8	10	16	Написание фрагмента кода
2.	Обучение с учителем и без учителя	36	6	12	18	Написание фрагмента кода
3.	Классификация снимков КТ с помощью сверточных нейронных сетей	18	8	10	-	Написание фрагмента кода
4.	Построение представлений из многоканальных временных рядов ЭЭГ	18	8	10	-	Написание фрагмента кода

5.	Проектно-технологическая практика		-	-	36	практическое задание
	Итоговая аттестация	2				зачет
	Всего:	**144**				

* Л-лекции, ПЗ - практические занятия, СР – самостоятельная работа,

ДОТ и ЭО осуществляются на платформе Электронно-информационной образовательной среды СамГМУ https://samsmu.ru/edu/.

V. КАЛЕНДАРНЫЙ УЧЕБНЫЙ ГРАФИК

Сроки обучения: образовательный процесс по программе может осуществляться в течение всего года.

Трудоёмкость освоения: 144 (ак.ч.)

Режим занятий: 6 недель / 6 дней в неделю /6 ч. в день

№ пп	Наименование раздела(модуля)	Учебные недели					
		1	2	3	4	5	6
1.	**Искусственный интеллект в медицине**	Л(8) ПЗ(10) СР(16)					
2.	**Обучение с учителем и без учителя**		Л(6) ПЗ(12) СР(18)				
3.	**Классификация снимков КТ с помощью сверточных нейронных сетей**			Л(8) ПЗ(10)			
4.	**Построение представлений из многоканальных временных рядов ЭЭГ**				Л(8) ПЗ(10)		
5.	**Проектно-технологическая практика**					СР(36)	
	Итоговая аттестация						ИА (2)

Условные обозначения

Л Лекция
ПЗ Практические занятия
СР Самостоятельная работа
ПП Проектно-технологическая практика
ИА Итоговая аттестация

VI. ОРГАНИЗАЦИОННО-ПЕДАГОГИЧЕСКИЕ УСЛОВИЯ РЕАЛИЗАЦИИ ПРОГРАММЫ

Материально-техническое обеспечение, необходимое для организации всех видов дисциплинарной подготовки:

- учебно-методическую документацию и материалы по всем разделам (модулям) программы, в том числе в ЭИОС университета;
- учебно-методическую литературу для внеаудиторной работы обучающихся;
- материально-технические базы, обеспечивающие организацию всех видов дисциплинарной подготовки:
- учебные аудитории, оснащенные оборудованием для проведения учебного процесса;
- дистанционные и электронные ресурсы для самостоятельной подготовки обучающихся, в частности Электронно-информационная образовательная среда СамГМУ https://samsmu.ru/edu/.
- кадровое обеспечение: реализация Программы обеспечивается научно-педагогическими кадрами Университета систематически занимающихся научной и научно-методической деятельностью со стажем работы в системе высшего и/или дополнительного профессионального образования не менее 5 лет, допустимо привлечение к образовательному процессу высококвалифицированных специалистов ИТ-сферы и/или дополнительного профессионального образования в части, касающейся профессиональных компетенций в области руководства проектированием специализированного программного обеспечения в рамках цифровой трансформации сферы здравоохранения, с обязательным участием представителей профильных организаций-работодателей. Возможно привлечение региональных руководителей цифровой трансформации (отраслевых ведомственных и/или корпоративных) к проведению итоговой аттестации, привлечение работников организаций реального сектора экономики субъектов Российской Федерации.

Учебно-методическое и информационное обучение:

Основная литература:

1. Шарден, Б. Крупномасштабное машинное обучение вместе с Python / Шарден Б. , Массарон Л. , Боскетти А. , пер. с анг. А. В. Логунова. - Москва : ДМК Пресс, 2018. - 358 с. - ISBN 978-5-97060-506-6. - Текст : электронный // ЭБС "Консультант студента" : [сайт]. - URL : https://www.studentlibrary.ru/book/ISBN9785970605066.html
2. Замятин, А.В. Интеллектуальный анализ данных [Электронный ресурс] : учебное пособие / А. В. Замятин. - Томск : Издательский Дом Томского государственного университета, 2020. Режим доступа: https://www.studentlibrary.ru/book/ISBN9785946218986.html
3. Python и машинное обучение / С. Рашка - Москва : ДМК Пресс, 2017. - ISBN 978-5-97060-409-0. - Текст : электронный // ЭБС "Консультант студента" : [сайт]. - URL : https://www.studentlibrary.ru/book/ISBN9785970604090.html

Дополнительная литература:

1. Флах, П. Машинное обучение. Наука и искусство построения алгоритмов, которые извлекают знания из данных / Флах П. - Москва : ДМК Пресс, 2015. - 400 с. - ISBN 978-5-97060-273-7. - Текст : электронный // ЭБС "Консультант студента" : [сайт]. - URL : https://www.studentlibrary.ru/book/ISBN9785970602737.html
2. Келлехер, Дж. Наука о данных : Базовый курс / Дж. Келлехер, Б. Тирни. - Москва : Альпина Паблишер, 2020. - 222 с. - ISBN 978-5-9614-3170-4. - Текст : электронный // ЭБС

"Консультант студента" : [сайт]. - URL : https://www.studentlibrary.ru/book/ISBN9785961431704.html
3. Замятин, А. В. Введение в интеллектуальный анализ данных : учеб. Пособие / Замятин А. В. – Томск : Издательский Дом Томского государственного университета, 2016. – 120 с. – ISBN 978-5-94621-531-2. – Текст : электронный // ЭБС «Консультант студента» : [сайт]. – URL : https://www.studentlibrary.ru/book/ISBN9785946215312.html
4. Паттерсон, Дж. , Гибсон А. Глубокое обучение с точки зрения практика / Паттерсон Дж. , Гибсон А. , пер. с анг. А. А. Слинкина. - Москва : ДМК Пресс, 2018. - 418 с. - ISBN 978-5-97060-481-6. - Текст : электронный // ЭБС "Консультант студента" : [сайт]. - URL : https://www.studentlibrary.ru/book/ISBN9785970604816.html

Программное обеспечение:

При проведении занятий по дисциплине используется следующее общесистемное и прикладное программное обеспечение:

1. MS Windows Версия 10 pro, Open License № V6731190, бессрочная.
2. MS Office Standard, Версия 2016, Open License № V6731190, бессрочная.
3. Антивирусная программа DoktorWeb (Контракт № ДС645 от 23.08.2022 г.). Приобретение неисключительного права на использование ПО Dr.Web Desktop Security Suite на 250 рабочих мест до 28.06.2023 г.
4. Электронная информационно-образовательная среда (построена на основе системы управления обучением Moodle (Moodle - свободное программное обеспечение, распространяемое на условиях лицензии GNU GPL (https://docs.moodle. org/dev/Li cense).
5. Программное обеспечение «Anaconda» – платформа для анализа данных на языке Python (свободное программное обеспечение, распространяемое на условиях модифицированной лицензии BSD (https://www.anaconda.com/)

Базы данных, информационно-справочные системы:

1. СПС (справочно-правовая система) «Гарант» - https://www.garant.ru/
2. Информационная система "Единое окно доступа к образовательным ресурсам" – URL: http://window.edu.ru/
3. Официальный сайт Министерства образования и науки Российской Федерации https://minobrnauki.gov.ru/
4. Библиотека для глубокого обучения Keras – URL: https://keras.io/
5. Библиотека для глубокого обучения PyTorch – URL: https://pytorch.org/
6. ЭБС «Консультант студента» - https://www.studentlibrary.ru/
7. «DATALIB – библиотека цифровой экономики»: https://datalib.ru/
8. Национальная электронная библиотека - https://rusneb.ru/

VII.ТРЕБОВАНИЯ К ИТОГОВОЙ АТТЕСТАЦИИ

Итоговая аттестация проходит в форме зачета. Зачет проводится в форме собеседования с написанием фрагмента кода.

Перечень вопросов для подготовки к зачету

1. Построить модель линейной регрессии для анализа заданного набора данных, имея ограничения по времени и допустимой ошибки прогнозирования.
2. Построить модель логистической регрессии для анализа заданного набора данных, имея ограничения по времени и допустимой точности классификации.
3. Построить модель метода опорных векторов для анализа заданного набора данных, имея ограничения по времени и допустимой точности классификации.
4. Построить модель дерева принятия решения для анализа заданного набора данных, имея ограничения по времени и допустимой точности классификации.

5. Провести редукцию модели и исходного набора данных с использованием метода K-means и PCA.
6. Провести отладку сверточной нейронной сети модели для анализа медицинских изображений. Повысить эффективность, модифицируя конфигурацию модели.
7. Построить модель машинного обучения для анализа ЭЭГ, ЭМГ и МРТ сигналов.
8. Определение предмета машинного обучения.
9. Примеры задач и областей приложения.
10. Образы и признаки.
11. Типы задач предсказания.
12. Регрессия. Таксономия. Классификация. Типы ошибок классификации.
13. Обобщающая способность классификатора.
14. Этапы классификации.
15. Алгоритмы обучения классификаторов с учителем и без учителя.
16. Дискриминантный анализ.
17. Геометрическая интерпретация задачи классификации.
18. Проективный подход.
19. Байесовская классификация.
20. Условная вероятность.
21. Формула полной вероятности.
22. Формула Байеса.
23. Статистическое распознавание образов.
24. Наивный байесовский классификатор.
25. Задача классификации спама.
26. Критерий отношения правдоподобия.
27. Деревья решений. Основные понятия.
28. Классы решаемых задач: описание данных, классификация, регрессия.
29. Общий алгоритм построения дерева решений.
30. Критерии выбора наилучшего атрибута: прирост информации, относительный прирост информации, индекс Гини.
31. Корреляционные и причинно-следственные связи.
32. Корреляция признаков и структура данных.
33. Латентные структуры в данных.
34. Формальная и эффективная размерность данных.
35. Структура и шум в данных.
36. Понижение размерности данных.
37. Регрессия. Метод наименьших квадратов.
38. Теорема Гаусса-Маркова.
39. Обобщенный метод наименьших квадратов.
40. Рекурсивный метод наименьших квадратов.
41. Анализ регрессионных остатков.
42. Комитетные методы распознавания образов.
43. Теоретические предпосылки комитетных методов.
44. Одиночные модели и ансамбли моделей.
45. Нейронные сети.
46. Предпосылки возникновения нейросетей.
47. Перцептрон Розенблатта.
48. Многослойный перцептрон.
49. Карты Кохонена. Сети Хопфилда.
50. Методы обучения нейросетей.
51. Метод опорных векторов

Система оценивания итоговой аттестации

При проведении итоговой аттестации в форме зачета используется система оценивания: «зачтено» и «не зачтено».

Шкала и критерии оценивания результатов обучения

Шкала оценивания	
«не зачтено»	**«зачтено»**
знать	
Обучающийся не знает основные модели искусственного интеллекта и принципы функционирования нейрокомпьютерных интерфейсов; не знает составляющие жизненного цикла процесса разработки программного продукта	Обучающийся знает основные модели искусственного интеллекта и принципы функционирования нейрокомпьютерных интерфейсов; знает составляющие жизненного цикла процесса разработки программного продукта
уметь	
Обучающийся не умеет формировать сборку интегральной модели из элементов искусственного интеллекта для решения задач аугментации и нейрореабилитации и составлять жизненный цикл для разработки интеллектуальных модулей и их интеграции в специализированное медицинское ПО	Обучающийся умеет формировать сборку интегральной модели из элементов искусственного интеллекта для решения задач аугментации и нейрореабилитации и составлять жизненный цикл для разработки интеллектуальных модулей и их интеграции в специализированное медицинское ПО
владеть	
Обучающийся не владеет навыками пересборки интегральной модели искусственного интеллекта для повышения продуктивности решения задач аугментации и нейрореабилитации; не способен разрабатывать интеллектуальные модули специализированного медицинского ПО	Обучающийся владеет навыками пересборки интегральной модели искусственного интеллекта для повышения продуктивности решения задач аугментации и нейрореабилитации; способен разрабатывать интеллектуальные модули специализированного медицинского ПО

Лицам, успешно освоившим соответствующую ДПП ПК (в области руководства проектированием специализированного программного обеспечения в рамках цифровой трансформации сферы здравоохранения) и прошедшим итоговую аттестацию, выдается документ о квалификации: удостоверение о повышении квалификации.

Лицам, не прошедшим итоговую аттестацию или получившим на итоговой аттестации неудовлетворительные результаты, а также лицам, освоившим часть Программы, выдается справка об обучении или о периоде обучения по образцу, самостоятельно устанавливаемому Университетом.

VIII. ОЦЕНОЧНЫЕ СРЕДСТВА

Контроль знаний, полученных слушателями при освоении разделов (модулей) Программы, осуществляется в следующих формах:

- текущий контроль успеваемости – обеспечивает оценивание хода освоения разделов Программы, проводится в форме тестового контроля и практических заданий;
- итоговая аттестация – завершает изучение всей программы, проводится в форме зачета.

В ходе освоения Программы каждый слушатель выполняет следующие отчетные работы:

№ п/п	Наименование раздела (модуля)	Задание	Критерии оценки
1.	Искусственный интеллект в медицине	Написание фрагмента кода	**«зачтено»** – ставится за работу, если обучающийся правильно выполнил не менее 2/3 всей работы или допустил не более одной грубой ошибки и двух недочетов, не более одной грубой и одной негрубой ошибки, не более трех негрубых ошибок, одной негрубой ошибки и двух недочетов. **«не зачтено»** – ставится за работу, если правильно выполнено менее 2/3 всей работы.
2	Обучение с учителем и без учителя	Написание фрагмента кода	**«зачтено»** – ставится за работу, если обучающийся правильно выполнил не менее 2/3 всей работы или допустил не более одной грубой ошибки и двух недочетов, не более одной грубой и одной негрубой ошибки, не более трех негрубых ошибок, одной негрубой ошибки и двух недочетов. **«не зачтено»** – ставится за работу, если правильно выполнено менее 2/3 всей работы.
3	Классификация снимков КТ с помощью сверточных нейронных сетей	Написание фрагмента кода	**«зачтено»** – ставится за работу, если обучающийся правильно выполнил не менее 2/3 всей работы или допустил не более одной грубой ошибки и двух недочетов, не более одной грубой и одной негрубой ошибки, не более трех негрубых ошибок, одной негрубой ошибки и двух недочетов. **«не зачтено»** – ставится за работу, если правильно выполнено менее 2/3 всей работы.
4	Построение представлений из многоканальных временных рядов ЭЭГ	Написание фрагмента кода	**«зачтено»** – ставится за работу, если обучающийся правильно выполнил не менее 2/3 всей работы или допустил не более одной грубой ошибки и двух недочетов, не более одной грубой и одной негрубой ошибки, не более трех негрубых ошибок, одной негрубой ошибки и двух недочетов. **«не зачтено»** – ставится за работу, если правильно выполнено менее 2/3 всей работы.
5	Проектно-технологическая практика	Разработка инструмента для анализа медицинских данных и интерпретации их	**«зачтено»** – ставится за работу, если обучающийся правильно выполнил не менее 2/3 всей работы или допустил не более одной грубой ошибки и двух недочетов, не более одной грубой и

		результатов	одной негрубой ошибки, не более трех негрубых ошибок, одной негрубой ошибки и двух недочетов. **«не зачтено»** – ставится за работу, если правильно выполнено менее 2/3 всей работы.
	Итоговая аттестация	Собеседование с написанием фрагмента кода	**«Зачтено»** - обучающийся демонстрирует знание основных разделов программы изучаемого курса: его базовых понятий и фундаментальных проблем; приобрел необходимые умения и навыки, освоил вопросы практического применения полученных знаний, не допустил фактических ошибок при ответе, достаточно последовательно и логично излагает теоретический материал, допуская лишь незначительные нарушения последовательности изложения и некоторые неточности. **«Не зачтено»** - выставляется в том случае, когда обучающийся демонстрирует фрагментарные знания основных разделов программы изучаемого курса: его базовых понятий и фундаментальных проблем. У экзаменуемого слабо выражена способность к самостоятельному аналитическому мышлению, имеются затруднения в изложении материала, отсутствуют необходимые умения и навыки, допущены грубые ошибки и незнание терминологии, отказ отвечать на дополнительные вопросы, знание которых необходимо для получения положительной оценки.

Примеры оценочных средств для текущего контроля успеваемости: написание фрагмента кода

Текущий контроль успеваемости проводится в пределах аудиторного времени, отведённого на программу. Результаты самостоятельной работы оцениваются в ходе текущего контроля и учитываются в процессе итоговой аттестации. Текущий контроль успеваемости проводится по результатам выполнения практических работ.

Написание фрагмента кода

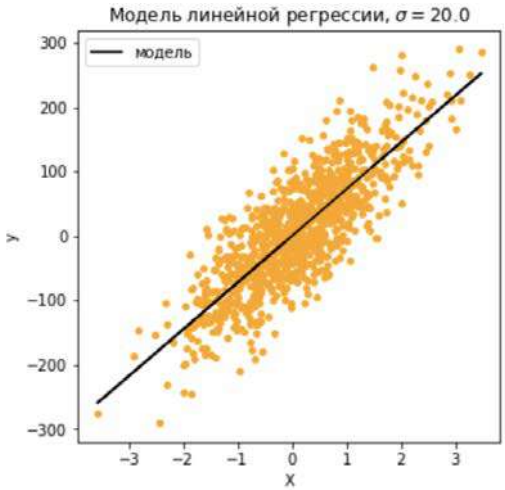
Задание 1.
Проанализировать параметры, загруженного набора данных diabetes:
https://archive.ics.uci.edu/ml/datasets/diabetes

Редактируя и запуская на исполнение ячейки в Jupiter Notebook:
https://colab.research.google.com/drive/1IkaJ1oZFztAuIXnh1Vmxod0SkIHR15sU?usp=sharing

подобрать параметры линейной модели и набор исходных признаков так, что значение метрики R2 повысилось на 0.25.
Интерпретировать полученные результаты.

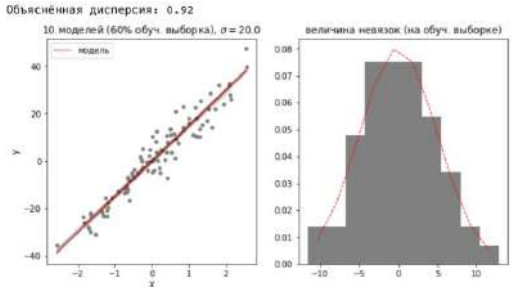
Пример выполнения:

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

from sklearn import datasets
from sklearn.linear_model import LinearRegression
from sklearn.model_selection import train_test_split
from sklearn.datasets import make_regression
from sklearn.pipeline import Pipeline
from sklearn.metrics import mean_squared_error, r2_score, explained_variance_score

import matplotlib.mlab as mlab
from scipy.stats import norm

X, y, coef =
 make_regression(n_samples=1000,      n_features=2,      noise=50.20,      coef=True,
random_state=38)

fig, ax = plt.subplots(1, 1, figsize=(5, 5))

ax.scatter(X[:,0], y, c='orange', s=15) #, s=10, cmap=plt.cm.Set1)
ax.plot(X[:,0],np.dot(coef[0], X[:,0]), '-', color='black', label='модель')
ax.legend(loc='upper left')
ax.set(xlabel='X',ylabel='y',title='Модель линейной регрессии, $\sigma=20.0$')
plt.show();
```




```
X
array([[-0.13704026, -1.64354232],
       [-1.31144302, -0.371127  ],
       [ 1.33978064, -0.82179757],
       ...,
       [ 0.87482733, -1.18834488],
       [-1.26105775, -0.25739925],
       [ 1.49654781,  0.18412844]])

y.shape
(1000,)

coef
array([72.59456432,  0.47026977])

X, y, coef =
 make_regression(n_samples=100, n_features=1, noise=5.0, coef=True, random_state=38)

fig, ax =
```

```
 plt.subplots(1, 2, figsize=(10, 5))

ax[0].scatter(X, y, c='gray', s=15)

for temp in range(10):

    X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.4) # ,
random_state=42)

    model = Pipeline([('linear', LinearRegression())])
    model.fit(X_train, y_train)

    ax[0].plot(X_train,model.predict(X_train), '-', color='black', alpha=0.1) #,
label='linearni model')

ax[0].plot(X,np.dot(coef,X), '-', color='red', alpha=0.5, label='модель')
ax[0].set(xlabel='X',ylabel='y',title='10 моделей (60% обуч. выборка), $\sigma=20.0$')
ax[0].legend(loc='upper left')

from numpy import *
def normpdf(x, mu, sigma):
    u = (x-mu)/abs(sigma)
    y =
(1/(sqrt(2*pi)*abs(sigma)))*exp(-u*u/2)
    return y

# Невязки
residuals =
 model.predict(X_train)-y_train
n, bins, patches =
 ax[1].hist(residuals, bins=10, density = True, color='gray')

bincenters =
 0.5*(bins[1:]+bins[:-1])

y_norm = normpdf(bincenters, np.mean(residuals), np.std(residuals))
ax[1].plot(bincenters, y_norm, 'r--', linewidth=1, label='gaussian')

ax[1].set(title='величина невязок (на обуч. выборке)')

print('Объяснённая                    дисперсия:                    '                    +
'{0:.2f}'.format(explained_variance_score(y_train,model.predict(X_train))))

plt.show();
```



Загрузим реальный набор данных и построим модель линейной регрессии.

```
diabetes = datasets.load_boston()
diabetes =
pd.DataFrame(data=np.c_[diabetes['data'], diabetes['target']],
                                columns=
np.append(['X'+str(i) for i in range(len(diabetes['data'][0]))],'target'))
diabetes.head()
```

	X0	X1	X2	X3	X4	X5	X6	X7	X8	X9	X10	X11	X12	target
0	0.00632	18.0	2.31	0.0	0.538	6.575	65.2	4.0900	1.0	296.0	15.3	396.90	4.98	24.0
1	0.02731	0.0	7.07	0.0	0.469	6.421	78.9	4.9671	2.0	242.0	17.8	396.90	9.14	21.6
2	0.02729	0.0	7.07	0.0	0.469	7.185	61.1	4.9671	2.0	242.0	17.8	392.83	4.03	34.7
3	0.03237	0.0	2.18	0.0	0.458	6.998	45.8	6.0622	3.0	222.0	18.7	394.63	2.94	33.4
4	0.06905	0.0	2.18	0.0	0.458	7.147	54.2	6.0622	3.0	222.0	18.7	396.90	5.33	36.2

```
diabetes.info()
```

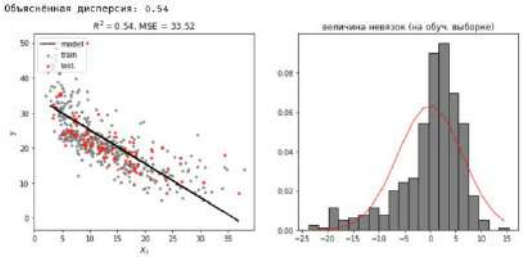
```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 506 entries, 0 to 505
Data columns (total 14 columns):
 #   Column  Non-Null Count  Dtype  
---  ------  --------------  -----  
 0   X0      506 non-null    float64
 1   X1      506 non-null    float64
 2   X2      506 non-null    float64
 3   X3      506 non-null    float64
 4   X4      506 non-null    float64
 5   X5      506 non-null    float64
 6   X6      506 non-null    float64
 7   X7      506 non-null    float64
 8   X8      506 non-null    float64
 9   X9      506 non-null    float64
 10  X10     506 non-null    float64
 11  X11     506 non-null    float64
 12  X12     506 non-null    float64
 13  target  506 non-null    float64
dtypes: float64(14)
memory usage: 55.5 KB
```

```
diabetes.drop('target',axis=1).corrwith(diabetes['target'])
```

```
X0    -0.388305
X1     0.360445
X2    -0.483725
X3     0.175260
X4    -0.427321
X5     0.695360
X6    -0.376955
X7     0.249929
X8    -0.381626
X9    -0.468536
X10   -0.507787
X11    0.333461
X12   -0.737663
dtype: float64
```

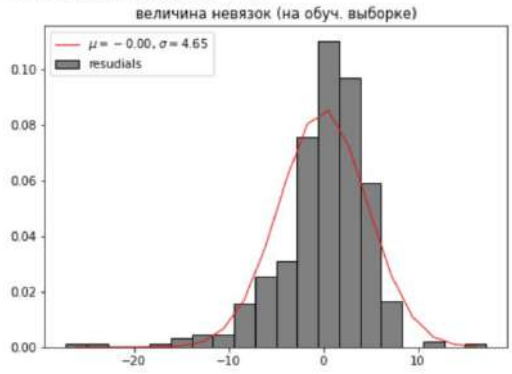
```
diabetes_train, diabetes_test =
train_test_split(diabetes[['X12','target']], test_size=0.20, random_state=42)

model = Pipeline([('linear',
 LinearRegression())])
model =
model.fit(diabetes_train.drop('target',axis=1),diabetes_train['target'])

diabetes_target_pred =
model.predict(diabetes_test.drop('target',axis=1))

fig, ax =
 plt.subplots(1, 2, figsize=(12, 5))

ax[0].scatter(diabetes_train.drop('target',axis=1).values,    diabetes_train['target'],
color='gray', marker='.', label='train')
ax[0].scatter(diabetes_test.drop('target',axis=1).values,     diabetes_test['target'],
color='red', marker='.', label='test')

ax[0].plot(diabetes_test.drop('target',axis=1).values,           diabetes_target_pred,
color='black', label='model')

R2 =
 r2_score(diabetes_test['target'], diabetes_target_pred)

MSE =
mean_squared_error(diabetes_test['target'], diabetes_target_pred)

ax[0].set(title='$R^2 =
 $'+'{0:.2f}'.format(R2)+', MSE =
 '+'{0:.2f}'.format(MSE))
ax[0].set(xlabel='$X_2$',ylabel='y')
ax[0].legend(loc='upper left')

# Величина невязок
residuals =
model.predict(diabetes_train.drop('target',axis=1))-diabetes_train['target']
n, bins, patches =
 ax[1].hist(residuals,
bins=20,density=True, color='gray',edgecolor='black')

bincenters =
 0.5*(bins[1:]+bins[:-1])
y_norm = norm.pdf( bincenters, np.mean(residuals), np.std(residuals))
ax[1].plot(bincenters, y_norm, 'r-', linewidth=1, label='gaussian')
```

```
ax[1].set(title='величина невязок (на обуч. выборке)')

explained_variance =
explained_variance_score(diabetes_train['target'],model.predict(diabetes_train.drop('t
arget',axis=1)))
print('Объяснённая дисперсия: ' + '{0:.2f}'.format(explained_variance))
plt.show();
```




```
np.std(residuals)
6.3055837818297436

# коэффициент модели линейной регрессии для выбранного признака X2
print(model.named_steps['linear'].coef_)
[-0.9665309]

diabetes_train, diabetes_test =
 train_test_split(diabetes, test_size=0.20, random_state=42)

model =
Pipeline([('linear', LinearRegression())])
model =
 model.fit(diabetes_train.drop('target',axis=1),diabetes_train['target'])

diabetes_target_pred =
model.predict(diabetes_test.drop('target',axis=1))

R2 =
 r2_score(diabetes_test['target'], diabetes_target_pred)
MSE =
mean_squared_error(diabetes_test['target'], diabetes_target_pred)

print('R2 =
 '+'{0:.2f}'.format(R2)+', MSE =
 '+'{0:.2f}'.format(MSE))
R2 = 0.67, MSE = 24.29

fig, ax = plt.subplots(1, 1, figsize=(7, 5))

residuals =
model.predict(diabetes_train.drop('target',axis=1))-diabetes_train['target']
n, bins, patches =
 ax.hist(residuals,   bins=20,   density   =   True,   color='gray',   edgecolor='black',
label='resudials')

bincenters =
 0.5*(bins[1:]+bins[:-1])
residuals_mu = np.mean(residuals)
residuals_std = np.std(residuals)
y_norm = norm.pdf( bincenters, residuals_mu, residuals_std )
l = ax.plot(bincenters, y_norm, 'r-', linewidth=1,
         label='$\mu='+'{0:.2f}'.format(residuals_mu)+'$,
$\sigma='+'{0:.2f}'.format(residuals_std)+'$')

ax.set(title='величина невязок (на обуч. выборке)')
ax.legend(loc='upper left')

explained_variance =
explained_variance_score(diabetes_train['target'],model.predict(diabetes_train.drop('t
arget',axis=1)))
print('Объяснённая дисперсия: ' + '{0:.2f}'.format(explained_variance))
plt.show();
```

```
Объяснённая дисперсия: 0.75
```




```
# коэффициенты модели линейной регрессии для всех признаков, включая X2
for i in zip(diabetes.drop('target',
axis=1).columns,model.named_steps['linear'].coef_):
    print(i)
```

```
('X0', -0.388305)
('X1', 0.360445)
('X2', -0.483725)
('X3', 0.175260)
('X4', -0.427321)
('X5', 0.695360)
('X6', -0.376955)
('X7', 0.249929)
('X8', -0.381626)
('X9', -0.468536)
('X10', -0.507787)
('X11', 0.333461)
('X12', -0.737663)
dtype: float64
```

Задание 2.

Редактируя и запуская на исполнение ячейки в Jupiter Notebook:
https://colab.research.google.com/drive/1n1QeaiGatd3Y3VWs0avxnXg9uZO68I32?usp=sharing
,
подобрать параметры логистической модели и набор исходных признаков так, что значение метрики F1 повысилось на 5%.

Пояснить как количество наблюдений n_samples и разброс значений cluster_std влияют на качество бинарной классификации (centers=2).

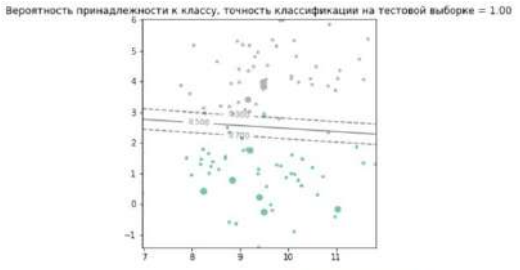
Пример выполнения:

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

from sklearn.linear_model import LogisticRegression
from sklearn.model_selection import train_test_split
from sklearn.datasets import make_blobs
from sklearn.pipeline import Pipeline
from sklearn.metrics import confusion_matrix, classification_report
```

```
Создадим функцию для построения разделяющей гиперплоскости и визуализации результатов
классификации calculate_decision_function().
```

```
def calculate_decision_function(X,model):
    '''
    Create a 2D grid to evaluate model.
    '''
    xx = np.linspace(X[:, 0].min(), X[:, 0].max(), 30)
    yy = np.linspace(X[:, 1].min(), X[:, 1].max(), 30)
    YY, XX = np.meshgrid(yy, xx)
    xy = np.vstack([XX.ravel(), YY.ravel()]).T
    Z = model(xy).reshape(XX.shape)
    return (XX,YY,Z)
```

```
# генерируем выборку из 100 линейно разделимых наблюдений
# с количеством классов "centers" и разбросом их значений "cluster_std"
X, y = make_blobs(n_samples=100, centers=2, cluster_std=1.0, random_state=4)

# разбиваем выборку на обучающую и тестовую
X_train, X_test, y_train, y_test =
 train_test_split(X, y, test_size=0.1, random_state=5)

# строим модель логистической регрессии
model = Pipeline([('logistic', LogisticRegression())])
model.fit(X_train, y_train)

# строим метрику для оценки качества модели
accuracy_test =
 model.score(X_test,y_test)
fig, ax = plt.subplots(1, 1, figsize=(5, 5))

ax.scatter(X_train[:, 0], X_train[:, 1], c=y_train, s=10, cmap=plt.cm.Set2)
ax.scatter(X_test[:, 0], X_test[:, 1], c=y_test, s=50, cmap=plt.cm.Set2)

(XX,YY,Z) =
 calculate_decision_function(X, lambda x: model.predict_proba(x)[:,0])
CS = ax.contour(XX, YY, Z, colors='k', levels=[0.3, 0.5, 0.7], alpha=0.5,
linestyles=['--', '-', '--'])
ax.clabel(CS, fontsize=9, inline=1)
ax.set(title='Вероятность принадлежности к классу, точность классификации на тестовой
выборке = ' + '{0:.2f}'.format(accuracy_test))
plt.show();
```



Качество классификации оценивается с помощью матрицы неточностей (confusion matrix)

		Прогноз класса	
		positives	negatives
Истинный класс	positives	*TP*	*FN*
	negatives	*FP*	*TN*

```
print(confusion_matrix(y_test, model.predict(X_test)))
[[11  0]
 [ 0 22]]

pd.crosstab(pd.Series(y_test,name='Истинный класс'),
 pd.Series(model.predict(X_test),name='Прогноз класса'),
            margins=True)
```

Прогноз класса	0	1	All
Истинный класс			
0	11	0	11
1	0	22	22
All	11	22	33

```
print(classification_report(y_test, model.predict(X_test)))
```

$$\text{precision} = \frac{TP}{TP + FP}$$

$$\text{recall} = \frac{TP}{TP + FN}$$

$$F_1 = 2\frac{\text{precision} \times \text{recall}}{\text{precision} + \text{recall}}$$

Задание 3.

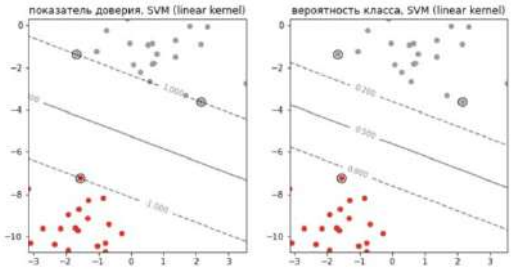
Построить модель классификации методом опорных векторов на медицинском наборе данных, загруженном из UCI Machine Learning Repository:
https://archive.ics.uci.edu/ml/index.php
Объяснить, каким образом параметры C, gamma и kernel влияют на результат классификации.

Интерпретировать полученный результат.

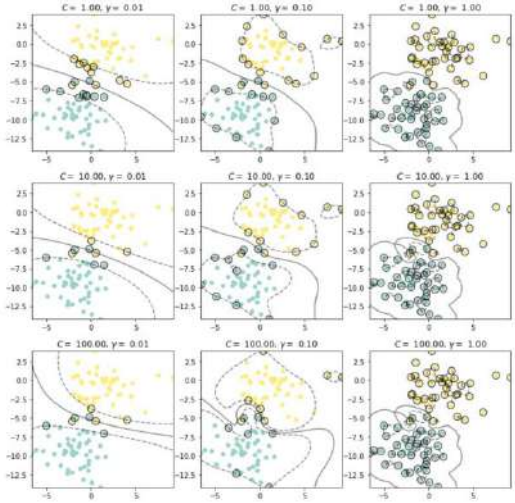
Пример выполнения:

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

from sklearn.preprocessing import PolynomialFeatures
from sklearn.linear_model import LinearRegression, LogisticRegression
from sklearn.pipeline import Pipeline
from sklearn.model_selection import train_test_split
from sklearn.datasets import make_blobs
from sklearn.svm import SVC
from sklearn import svm

%matplotlib inline
import matplotlib.pyplot as plt
import numpy as np
import pandas as pd

def calculate_decision_function(X,model):
    '''
    Create a 2D grid to evaluate model.
    '''

    xx = np.linspace(X[:, 0].min(), X[:, 0].max(), 30)
    yy = np.linspace(X[:, 1].min(), X[:, 1].max(), 30)
    YY, XX = np.meshgrid(yy, xx)
    xy = np.vstack([XX.ravel(), YY.ravel()]).T
    Z = model(xy).reshape(XX.shape)

    return (XX,YY,Z)

# генерируем 60 примеров
X, y = make_blobs(n_samples=60, centers=2, cluster_std=1.0, random_state=2)
X_train, X_test, y_train, y_test =
 train_test_split(X, y, test_size=0.33, random_state=42)

# обучаем модель без регуляризации параметров
kernel = 'linear' # 'linear', 'poly', 'rbf', 'sigmoid', 'precomputed'
model = Pipeline([('SVM', svm.SVC(kernel=kernel, probability=True))])
model.fit(X_train, y_train)
accuracy_test =
 model.score(X_test,y_test)

fig, ax = plt.subplots(1, 2, figsize=(10, 5))

for i in range(2):
    ax[i].scatter(X_train[:,0], X_train[:,1], c=y_train, s=30, cmap=plt.cm.Set1)
# строим опорные вектора
 ax[i].scatter(model.named_steps['SVM'].support_vectors_[:,0],
model.named_steps['SVM'].support_vectors_[:,1],
                  s=100, linewidths=1, edgecolors='black', facecolors='none')

# строим линии уровня показателя доверия
(XX,YY,Z) =
calculate_decision_function(X_train, model.decision_function)
CS = ax[0].contour(XX, YY, Z, colors='k', levels=[-1, 0, 1], alpha=0.5, linestyles=['-
-', '-', '--'])
ax[0].clabel(CS, fontsize=9, inline=1)
ax[0].set(title='показатель доверия, SVM (' + kernel + ' kernel)')
```

```
# строим линии уровня вероятности принадлежности к классу
(XX,YY,Z) =
calculate_decision_function(X_train, lambda x: model.predict_proba(x)[:,0])
CS = ax[1].contour(XX, YY, Z, colors='k', levels=[0.2, 0.5, 0.8], alpha=0.5,
linestyles=['--', '-', '--'])
ax[1].clabel(CS, fontsize=9, inline=1)
ax[1].set(title='вероятность класса, SVM (' + kernel + ' kernel)')
plt.show();
```




```
# Регуляризация модели с учетом параметров модели C и γ

X, y = make_blobs(n_samples=100, centers=2, cluster_std=2.0, random_state=2)

fig, ax = plt.subplots(3, 3, figsize=(10, 10))

for i,C in enumerate([1.0,10.0,
100.0]):
    for j,gamma in enumerate([0.01,0.1,1.0]):
        model = Pipeline([('SVM', svm.SVC(kernel='rbf', C=C, gamma=gamma))])
        model.fit(X, y)
        ax[i][j].scatter(X[:, 0], X[:, 1], c=y, s=30, cmap=plt.cm.Set3)
        # Построение разделяющей гиперплоскости
        (XX,YY,Z) = calculate_decision_function(X, model.decision_function)
        CS = ax[i][j].contour(XX, YY, Z, colors='k', levels=[-1, 0, 1], alpha=0.5,
linestyles=['--', '-', '--'])

        # Построение опорных векторов
ax[i][j].scatter(model.named_steps['SVM'].support_vectors_[:, 0],
model.named_steps['SVM'].support_vectors_[:, 1],
                         s=100, linewidths=1, edgecolors='black', facecolors='none')
        ax[i][j].set(title='$C=$'+'{0:.2f}'.format(C)+', $\gamma =$
'+'{0:.2f}'.format(gamma))
fig.tight_layout();
```



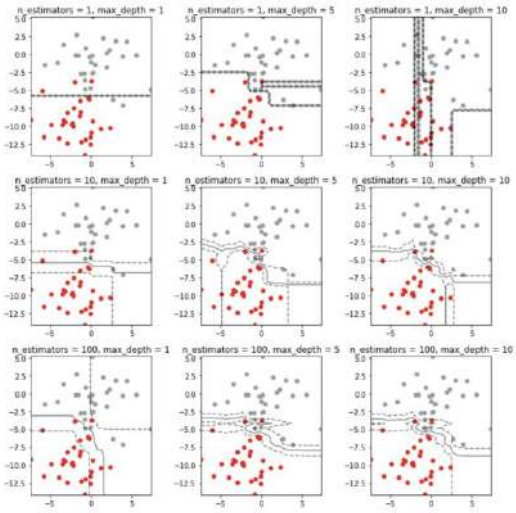
Задание 4.

Построить модель набора деревьев принятия решений на медицинском наборе данных, загруженном из UCI Machine Learning Repository:
https://archive.ics.uci.edu/ml/index.php
Объяснить, каким образом параметры n_estimators и max_depth влияют на результат классификации.

Интерпретировать полученный результат.

Пример выполнения:

```
from sklearn.datasets import make_blobs
from sklearn.pipeline import Pipeline
from sklearn.model_selection import train_test_split
from sklearn.ensemble import RandomForestClassifier

%matplotlib inline
import matplotlib.pyplot as plt
import numpy as np
import pandas as pd

def calculate_decision_function(X,model):
    '''
    Create a 2D grid to evaluate model.
    '''
    xx = np.linspace(X[:, 0].min(), X[:, 0].max(), 30)
    yy = np.linspace(X[:, 1].min(), X[:, 1].max(), 30)
    YY, XX = np.meshgrid(yy, xx)
    xy = np.vstack([XX.ravel(), YY.ravel()]).T
    Z = model(xy).reshape(XX.shape)
    return (XX,YY,Z)

# генерируем 60 примеров
X, y = make_blobs(n_samples=60, centers=2, cluster_std=2.5, random_state=2)

fig, ax = plt.subplots(3, 3, figsize=(10, 10))

for i,n_estimators in enumerate([1,10,100]):
    for j,max_depth in enumerate([1,5,10]):
        model                                =                                Pipeline([('RF',
RandomForestClassifier(n_estimators=n_estimators,max_depth=max_depth))])
        model.fit(X, y)
        ax[i][j].scatter(X[:, 0], X[:, 1], c=y, s=30, cmap=plt.cm.Set1)
       (XX,YY,Z) =
 calculate_decision_function(X, lambda x: model.predict_proba(x)[:,0])
        CS  =  ax[i][j].contour(XX,  YY,  Z,  colors='k',  levels=[0.3,  0.5,  0.7],
alpha=0.5, linestyles=['--', '-', '--'])
    ax[i][j].set(title='n_estimators = ' + str(n_estimators) + ', max_depth = ' +
str(max_depth))
fig.tight_layout();
```



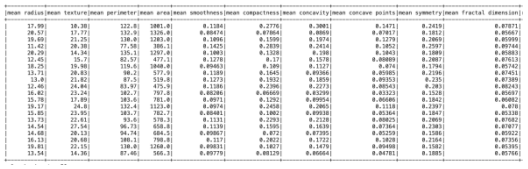
Задание 5.

Повысить точность исходной и улучшенной моделей классификации методом K-means, изменяя параметры метода PCA.

Интерпретировать полученный результат.

Пример выполнения:

```
# установка PySpark & Spark
!pip install pyspark
!pip install -U -q PyDrive
!apt install openjdk-8-jdk-headless -qq
import os
os.environ["JAVA_HOME"] = "/usr/lib/jvm/java-8-openjdk-amd64"

# импорт библиотек
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
%matplotlib inline

import pyspark
from pyspark.sql import *
from pyspark.sql.types import *
from pyspark.sql.functions import *
from pyspark import SparkContext, SparkConf

# инициализация SparkContext
sc = pyspark.SparkContext()
spark =
 SparkSession.builder.getOrCreate()


# датасет
Дан набор данных, содержащих информацию о диагностике рака молочной железы .
Выходные данные - характер опухоли (доброкачественная, злокачественная).
Каждая опухоль описывается при помощи 30 вещественных атрибутов.

# загрузка датасета
from sklearn.datasets import load_breast_cancer
breast_cancer = load_breast_cancer()

len(breast_cancer.data)
569

# преобразование данных
pd_df =
 pd.DataFrame(breast_cancer.data, columns=breast_cancer.feature_names)
df = spark.createDataFrame(pd_df)
df.show()
```



```
def map_to_nullable(spark, df, column_list, nullable=False):
    for struct_field in df.schema:
        if struct_field.name in column_list:
            struct_field.nullable =
 nullable
    df_mod =
 spark.createDataFrame(df.rdd, df.schema)
    return df_mod

df = map_to_nullable(spark, df, df.columns)
df = df.withColumn('features', array(df.columns))
vectors = df.rdd.map(lambda row: Vectors.dense(row.features))
df.printSchema()
```

```
root
 |-- mean radius: double (nullable = false)
 |-- mean texture: double (nullable = false)
 |-- mean perimeter: double (nullable = false)
 |-- mean area: double (nullable = false)
 |-- mean smoothness: double (nullable = false)
 |-- mean compactness: double (nullable = false)
 |-- mean concavity: double (nullable = false)
 |-- mean concave points: double (nullable = false)
 |-- mean symmetry: double (nullable = false)
 |-- mean fractal dimension: double (nullable = false)
 |-- radius error: double (nullable = false)
 |-- texture error: double (nullable = false)
 |-- perimeter error: double (nullable = false)
 |-- area error: double (nullable = false)
 |-- smoothness error: double (nullable = false)
 |-- compactness error: double (nullable = false)
 |-- concavity error: double (nullable = false)
 |-- concave points error: double (nullable = false)
 |-- symmetry error: double (nullable = false)
 |-- fractal dimension error: double (nullable = false)
 |-- worst radius: double (nullable = false)
 |-- worst texture: double (nullable = false)
 |-- worst perimeter: double (nullable = false)
 |-- worst area: double (nullable = false)
 |-- worst smoothness: double (nullable = false)
 |-- worst compactness: double (nullable = false)
 |-- worst concavity: double (nullable = false)
 |-- worst concave points: double (nullable = false)
 |-- worst symmetry: double (nullable = false)
 |-- worst fractal dimension: double (nullable = false)
 |-- features: array (nullable = false)
 |    |-- element: double (containsNull = false)
```

Построим 2 датафрейма:
features - датафрейм, содержащий информацию о признаках опухоли;
labels - бинарная серия с информацией о том, принадлежит ли соответствующий набор признаков злокачественной опухоли или нет.

```
from pyspark.ml.linalg import Vectors
features =
spark.createDataFrame(vectors.map(Row), ["features"])
labels =
 pd.Series(breast_cancer.target)
```

```
labels.values
```

```
array([0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 1, 1,
       0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0,
       0, 0, 1, 0, 1, 1, 1, 1, 1, 0, 0, 1, 0, 0, 1, 1, 1, 1, 0, 1, 0, 0,
       1, 1, 1, 1, 0, 1, 0, 0, 1, 0, 1, 0, 0, 1, 1, 1, 0, 0, 1, 0, 0, 0,
       1, 1, 1, 0, 1, 1, 0, 0, 1, 1, 1, 0, 0, 1, 1, 1, 1, 0, 1, 1, 0, 1,
       1, 1, 1, 1, 1, 1, 1, 0, 0, 0, 1, 0, 0, 1, 1, 1, 0, 0, 1, 0, 1, 0,
       0, 1, 0, 0, 1, 1, 0, 1, 1, 0, 1, 1, 1, 1, 0, 1, 1, 1, 1, 1, 1, 1,
       1, 1, 0, 1, 1, 1, 1, 0, 0, 1, 0, 1, 1, 0, 0, 1, 1, 0, 0, 1, 1, 1,
       1, 0, 1, 1, 0, 0, 0, 1, 0, 1, 0, 1, 1, 1, 0, 1, 1, 0, 0, 1, 0, 0,
       0, 0, 1, 0, 0, 0, 1, 0, 1, 0, 1, 1, 0, 1, 0, 0, 0, 0, 1, 1, 0, 0,
       1, 1, 1, 0, 1, 1, 1, 1, 1, 0, 0, 1, 1, 0, 1, 1, 0, 0, 1, 0, 1, 1,
       1, 1, 0, 1, 1, 1, 1, 1, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
       0, 0, 1, 1, 1, 1, 1, 1, 1, 0, 1, 0, 1, 1, 0, 1, 1, 0, 1, 0, 0, 1, 1,
       1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 0, 1, 1, 1, 0, 1, 1, 1, 1, 1, 1,
       1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 0, 1, 1, 1, 1, 1, 1, 1, 1, 0, 0,
       0, 1, 1, 1, 1, 0, 1, 0, 1, 0, 1, 1, 1, 0, 1, 1, 1, 1, 1, 1, 1, 0,
       0, 0, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 0, 0, 1, 0, 0, 0, 1, 0, 0,
       1, 1, 1, 1, 1, 0, 1, 1, 1, 1, 1, 0, 1, 1, 1, 0, 1, 1, 0, 0, 1, 1,
       1, 1, 1, 1, 0, 1, 1, 1, 1, 1, 1, 1, 0, 1, 1, 1, 1, 1, 1, 0, 1, 1, 0,
       1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 0, 1, 0, 0, 1, 0, 1, 1, 1, 1,
       1, 0, 1, 1, 0, 1, 0, 1, 1, 0, 1, 0, 1, 1, 1, 1, 1, 1, 1, 1, 0, 0,
       1, 1, 1, 1, 1, 1, 1, 0, 1, 1, 1, 1, 1, 1, 1, 1, 1, 0, 1, 1, 1, 1,
       1, 1, 1, 0, 1, 0, 1, 1, 0, 1, 1, 1, 1, 1, 1, 0, 0, 1, 0, 1, 0, 1, 1,
       1, 1, 1, 0, 1, 1, 0, 1, 0, 1, 0, 0, 1, 1, 1, 0, 1, 1, 1, 1, 1, 1,
       1, 1, 1, 1, 1, 1, 0, 1, 0, 0, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1,
       1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 0, 0, 0, 0, 0, 0, 1])
```

```
features.show()
```

```
+--------------------+
|            features|
+--------------------+
|[17.99,10.38,122....|
|[20.57,17.77,132....|
|[19.69,21.25,130....|
|[11.42,20.38,77.5...|
|[20.29,14.34,135....|
|[12.45,15.7,82.57...|
|[18.25,19.98,119....|
|[13.71,20.83,90.2...|
|[13.0,21.82,87.5,...|
|[12.46,24.04,83.9...|
|[16.02,23.24,102....|
|[15.78,17.89,103....|
|[19.17,24.8,132.4...|
|[15.85,23.95,103....|
|[13.73,22.61,93.6...|
|[14.54,27.54,96.7...|
|[14.68,20.13,94.7...|
|[16.13,20.68,108....|
|[19.81,22.15,130....|
|[13.54,14.36,87.4...|
+--------------------+
only showing top 20 rows
```

```
# Бинарная классификация методом K-means
```

Для решения задачи воспользуемся методом K-means из пакета SparkML. Так как количество классов равно двум, установите параметр k=2.
Вычислим точность полученной K-means модели. Сравним выходные данные модели с выходными даннными исходного датасета.
Изначально K-Means является алгоритмом кластеризации.
Это означает, что он возвращает номера кластеров. Данные номера необязательно равны выходной переменной.
Задача - понять, в какой кластер попали те или иные опухоли. И уже на основе этого оценивать accuracy метода.
$accuracy = S/N$,
где S - количество опухолей, отнесённых к правильному классу;
N - общий размер датасета.

```
# базовая модель
from pyspark.ml.clustering import KMeans
from pyspark.ml.evaluation import ClusteringEvaluator

kmeans = KMeans(k=2, seed=1)
model = kmeans.fit(features)

predictions =
 model.transform(features)
predictions.show()
```

```
+--------------------+----------+
|            features|prediction|
+--------------------+----------+
|[17.99,10.38,122....|         1|
|[20.57,17.77,132....|         1|
|[19.69,21.25,130....|         1|
|[11.42,20.38,77.5...|         0|
|[20.29,14.34,135....|         1|
|[12.45,15.7,82.57...|         0|
|[18.25,19.98,119....|         1|
|[13.71,20.83,90.2...|         0|
|[13.0,21.82,87.5,...|         0|
|[12.46,24.04,83.9...|         0|
|[16.02,23.24,102....|         0|
|[15.78,17.89,103....|         1|
|[19.17,24.8,132.4...|         1|
|[15.85,23.95,103....|         0|
|[13.73,22.61,93.6...|         0|
|[14.54,27.54,96.7...|         0|
|[14.68,20.13,94.7...|         0|
|[16.13,20.68,108....|         1|
|[19.81,22.15,130....|         1|
|[13.54,14.36,87.4...|         0|
+--------------------+----------+
only showing top 20 rows
```

```
Кластер 1 -> признак 0
Кластер 0 -> признак 1

predictions =
 np.array([rows[i].prediction for i in range(len(rows))])

# функция подсчета точности модели
def getAccuracy(modelRows, datasetLabels):
  coincidencesCount = 0;
  for i, label in enumerate(datasetLabels):
      if label !=
 modelRows[i].prediction:
          coincidencesCount =
 coincidencesCount + 1;
  labelsLength = len(datasetLabels);
  print(coincidencesCount, '/', labelsLength)

  return coincidencesCount/labelsLength;

# подсчет точности базовой модели
rows =
predictions.select('prediction').collect();
baseAccuracy = getAccuracy(rows, labels)
print('Точность базовой модели: ', baseAccuracy * 100, '%')

83 / 569
Точность базовой модели:  14.586994727592268 %
```

```
# модель с улучшенной точностью

# seed: 1 -> 100
# distanceMeasure: 'euclidean' (default) -> 'cosine'

kmeans = KMeans(k=2, seed=100, distanceMeasure='cosine')
model = kmeans.fit(features)

updPredictions =
 model.transform(features)
updPredictions.show()
```

```
+------------------+----------+
|          features|prediction|
+------------------+----------+
|[17.99,10.38,122....|         1|
|[20.57,17.77,132....|         1|
|[19.69,21.25,130....|         1|
|[11.42,20.38,77.5...|         1|
|[20.29,14.34,135....|         0|
|[12.45,15.7,82.57...|         1|
|[18.25,19.98,119....|         1|
|[13.71,20.83,90.2...|         1|
|[13.0,21.82,87.5,...|         1|
|[12.46,24.04,83.9...|         1|
|[16.02,23.24,102....|         1|
|[15.78,17.89,103....|         1|
|[19.17,24.8,132.4...|         0|
|[15.85,23.95,103....|         0|
|[13.73,22.61,93.6...|         0|
|[14.54,27.54,96.7...|         1|
|[14.68,20.13,94.7...|         1|
|[16.13,20.68,108....|         1|
|[19.81,22.15,130....|         1|
|[13.54,14.36,87.4...|         0|
+------------------+----------+
only showing top 20 rows
```

```
# подсчет точности улучшенной модели
updRows =
updPredictions.select('prediction').collect();
updAccuracy = getAccuracy(updRows, labels)
print('Точность улучшенной модели: ', updAccuracy * 100, '%')
509 / 569
Точность улучшенной модели:  89.45518453427064 %

# изменение точности
print(baseAccuracy * 100, '% ->', updAccuracy * 100, '%')
85.41300527240774 % ->
 89.45518453427064 %

# Понижение размерности методом PCA
Проведем понижение размерности при помощи метода главных компонент.
Более подробное изложение можно найти по ссылке. Данный метод также доступен в пакете
MLlib. Установим параметр k = 2, понизив размерность входного пространства в 15 раз.

from pyspark.ml.feature import PCA
from pyspark.ml.linalg import Vectors
pca = PCA(k=2, inputCol="features", outputCol="pcaFeatures")
model = pca.fit(features)
pcaFeatures =
model.transform(features).select("pcaFeatures")
pcaFeatures.show(truncate=False)
```

```
+----------------------------------------+
|pcaFeatures                             |
+----------------------------------------+
|[-2260.0138862925405,-187.96030122263687]|
|[-2368.9937557820535,121.58742425815493] |
|[-2095.66520154786,145.11398565870115]   |
|[-692.6905100570506,38.57692259208172]   |
|[-2030.2124927427058,295.29798399279287] |
|[-888.2800535760758,26.079796157025683]  |
|[-1921.0822124748443,58.80757247309935]  |
|[-1074.7813350047963,31.771227808469586] |
|[-908.5784781618829,63.83075279044626]   |
|[-861.578449407568,40.57073549705317]    |
|[-1404.5591306499468,88.23218257736238]  |
|[-1524.2324408687816,-3.2630573167779446]|
|[-1734.3856477464153,273.1626781511456]  |
|[-1162.9140032230355,217.63481808344625] |
|[-903.4301030498832,135.61517666084788]  |
|[-1155.8759954206846,76.80889383742178]  |
|[-1335.7294321308066,-2.4684005450354807]|
|[-1547.2640922523085,3.8056759725745044] |
|[-2714.964765181215,-164.37610864258824] |
|[-908.2502671870876,118.21642008223104]  |
+----------------------------------------+
only showing top 20 rows
```

```
# бинарная классификация методом K-means на данных меньшей размерности
```

Обучим модель KMeans также как в задании 2, но уже на признаках меньшей размерности. Оценим результат классификации. Выведем точечную диаграмму, отображающую зависимость первого признака от второго. Точки на диаграмме должны быть окрашены в зависимости от принадлежности тому или иному классу.

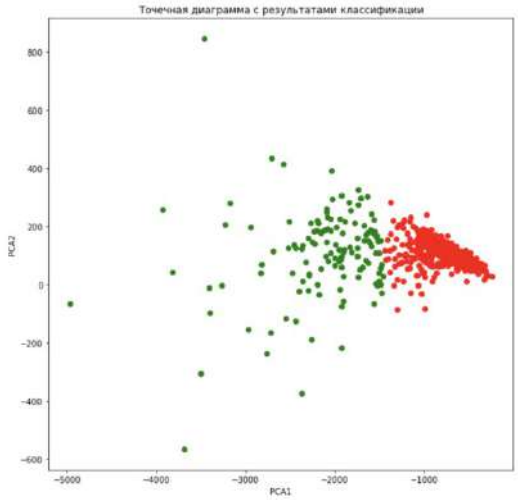
```
# базовая PCA модель
kmeans =
 KMeans(featuresCol="pcaFeatures", k=2, seed=1)
model = kmeans.fit(pcaFeatures)
pcaPredictions =
 model.transform(pcaFeatures)
pcaPredictions.show()
```

```
+--------------------+----------+
|         pcaFeatures|prediction|
+--------------------+----------+
|[-2260.0138862925...|         1|
|[-2368.9937557820...|         1|
|[-2095.6652015478...|         1|
|[-692.69051005705...|         0|
|[-2030.2124927427...|         1|
|[-888.28005357607...|         0|
|[-1921.0822124748...|         1|
|[-1074.7813350047...|         0|
|[-908.57847816188...|         0|
|[-861.57844940756...|         0|
|[-1404.5591306499...|         0|
|[-1524.2324408687...|         1|
|[-1734.3856477464...|         1|
|[-1162.9140032230...|         0|
|[-903.43010304988...|         0|
|[-1155.8759954206...|         0|
|[-1335.7294321308...|         0|
|[-1547.2640922523...|         1|
|[-2714.9647651812...|         1|
|[-908.25026718708...|         0|
+--------------------+----------+
only showing top 20 rows
```

```
# подсчет точности базовой PCA модели
pcaRows =
pcaPredictions.select('prediction').collect();
basePcaAccuracy =
 getAccuracy(pcaRows, labels)
print('Точность PCA модели: ', basePcaAccuracy * 100, '%')
```

```
486 / 569
Точность PCA модели:
  85.41300527240774 %
```

```
from sklearn.manifold import TSNE
rows = pcaPredictions.collect();
plt.figure(figsize=(10,10))
for row in rows:
  color = 'green' if row.prediction == 1 else 'red';
  plt.plot(row.pcaFeatures[0], row.pcaFeatures[1], 'o',
 color=color);
plt.xlabel('PCA1')
plt.ylabel('PCA2')
plt.title('Точечная диаграмма с результатами классификации')
plt.show()
```




```
# PCA модель с улучшенной точностью

# seed: 1 -> 10
# distanceMeasure: 'euclidean' (default) -> 'cosine'
# maxIter: 20 (default) -> 10
# initSteps: 2 (default) -> 100
kmeans =
 KMeans(featuresCol="pcaFeatures", k=2, seed=10, distanceMeasure='cosine', maxIter=10,
initSteps=100)
model = kmeans.fit(pcaFeatures)
updPcaPredictions =
 model.transform(pcaFeatures)
updPcaPredictions.show()
```

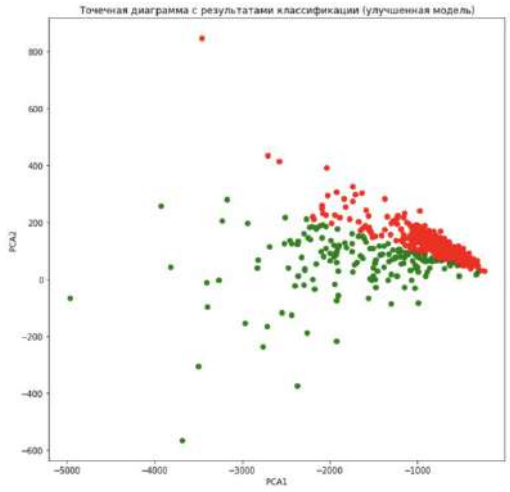
```
+------------------+----------+
|        pcaFeatures|prediction|
+------------------+----------+
|[-2260.0138862925...|         1|
|[-2368.9937557820...|         1|
|[-2095.6652015478...|         1|
|[-692.69051005705...|         1|
|[-2030.2124927427...|         0|
|[-888.28005357607...|         1|
|[-1921.0822124748...|         1|
|[-1074.7813350047...|         1|
|[-908.57847816188...|         1|
|[-861.57844940756...|         1|
|[-1404.5591306499...|         1|
|[-1524.2324408687...|         1|
|[-1734.3856477464...|         0|
|[-1162.9140032230...|         0|
|[-903.43010304988...|         0|
|[-1155.8759954206...|         1|
|[-1335.7294321308...|         1|
|[-1547.2640922523...|         1|
|[-2714.9647651812...|         1|
|[-908.25026718708...|         0|
+------------------+----------+
only showing top 20 rows
```

```
# подсчет точности улучшенной PCA модели
updPcaRows =
updPcaPredictions.select('prediction').collect();
updPcaAccuracy =
 getAccuracy(updPcaRows, labels)
print('Точность улучшенной PCA модели: ', updPcaAccuracy * 100, '%')
490 / 569
```

```
Точность улучшенной PCA модели:  86.11599297012302 %
```

```
# изменение точности
print(basePcaAccuracy * 100, '% ->', updPcaAccuracy * 100, '%')
85.41300527240774 % ->
 86.11599297012302 %


rows = updPcaPredictions.collect();
plt.figure(figsize=(10,10))

for row in rows:
  color = 'green' if row.prediction == 1 else 'red';
  plt.plot(row.pcaFeatures[0], row.pcaFeatures[1], 'o', color=color);
```

```
plt.xlabel('PCA1')
plt.ylabel('PCA2')
plt.title('Точечная диаграмма с результатами классификации (улучшенная модель)')
plt.show()
```



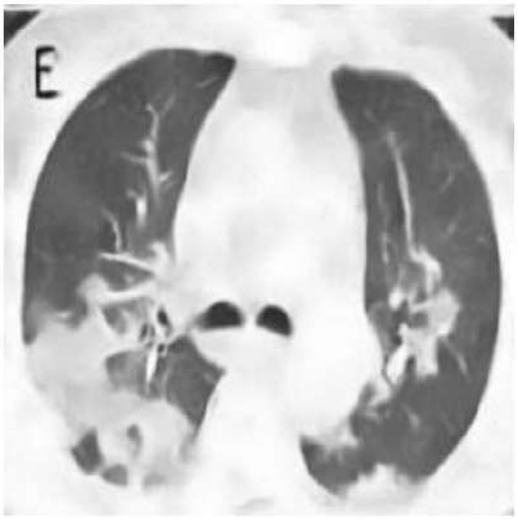
Задание 6.

Повысить точность классификации снимков, изменяя значения параметров в пунктах 6, 7 и 9 Jupiter Notebook: https://colab.research.google.com/drive/1ezLyEhg-hISuLfyjFk96pB-ftOwlbNLL?usp=sharing ; Интерпретировать полученный результат.

Пример выполнения:

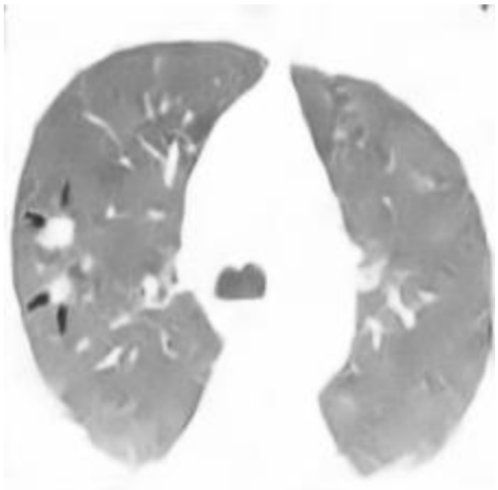
```
# импорт TensorFlow и других библиотек
import matplotlib.pyplot as plt
import numpy as np
import os
import PIL
import tensorflow as tf

from tensorflow import keras
from tensorflow.keras import layers
from tensorflow.keras.models import Sequential
print(tf.__version__)
2.4.0

# загрузка и просмотр датасета
В этом уроке используется датасет из 2560 фотографий снимков КТ. Датасет состоит из 2-ти директорий:
data/
  CT_Covid-19/
  CT_Non_Covid-19/

import pathlib
from google.colab import drive
drive.mount('/content/drive')
data_dir =
pathlib.Path('/content/drive/MyDrive/data')
print(data_dir)
Drive already mounted at /content/drive; to attempt to forcibly remount, call drive.mount("/content/drive", force_remount=True).
/content/drive/MyDrive/data

После загрузки доступна копия датасета. Всего 2560 изображений:
image_count = len(list(data_dir.glob('*/*.jpg')))
print(image_count)
2560

Посмотрим изображения из директории Not_Infected:
infected =
```

```
list(data_dir.glob('CT_Covid-19/*'))
PIL.Image.open(str(infected[2]))
```

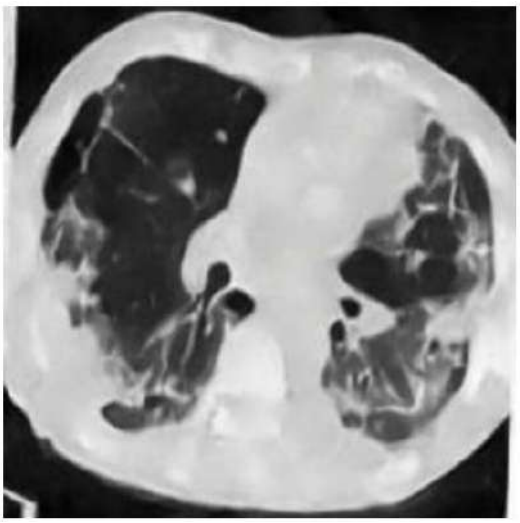



```
PIL.Image.open(str(infected[5]))
```

```
not_infected = list(data_dir.glob('CT_Non_Covid-19/*'))
PIL.Image.open(str(not_infected[0]))
```



```
PIL.Image.open(str(not_infected[1]))
```



загрузка с диска с использованием keras.preprocessing
Загрузим изображения с диска с помощью утилиты image_dataset_from_directory. Этот помощник создаст tf.data.Dataset всего-лишь несколькими строчками кода. Или, если вам нравится, вы можете создать загрузчик изображений с нуля, используя load images.

```
# создание датасета
batch_size = 32
img_height = 180
img_width = 180

train_ds =
tf.keras.preprocessing.image_dataset_from_directory(
  data_dir,
  validation_split=0.2,
  subset="training",
  seed=123,
  image_size=(img_height,
 img_width),
  batch_size=batch_size)
```

```
Found 2560 files belonging to 2 classes.
Using 2048 files for training.
```

```
val_ds =
tf.keras.preprocessing.image_dataset_from_directory(
  data_dir,
  validation_split=0.2,
  subset="validation",
  seed=123,
  image_size=(img_height,
 img_width),
  batch_size=batch_size)
```

```
Found 2560 files belonging to 2 classes.
Using 512 files for validation.
```

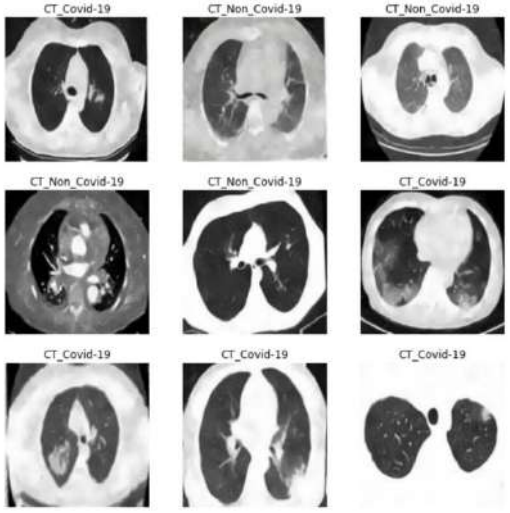
классы датасета в атрибуте `class_names`. Они соответствуют именам директорий и отсортированы в алфавитном порядке.

```
class_names = train_ds.class_names
print(class_names)
['CT_Covid-19', 'CT_Non_Covid-19']
```

визуализация данных
Первые 9 изображений из тренировочного датасета:

```
import matplotlib.pyplot as plt

plt.figure(figsize=(10, 10))
for images, labels in train_ds.take(1):
  for i in range(9):
    ax = plt.subplot(3, 3, i + 1)
```

```
    plt.imshow(images[i].numpy().
astype("uint8"))
   plt.title(class_names[labels[i]])
    plt.axis("off")
```




```
for image_batch, labels_batch in train_ds:
  print(image_batch.shape)
  print(labels_batch.shape)
  break
```

image_batch это тензор размерности (32, 180, 180, 3). В данном примере: 32 - размер пакета, 180x180 - размер изображения, 3 - количество цветовых каналов(RGB). label_batch - тензор размерности (32,), в данном примере содержит 32 метки для изображений.
Метод .numpy() на image_batch и labels_batch тензорах, чтобы преобразовать их в numpy.ndarray.

```
# конфигурирование датасета для улучшения производительности
AUTOTUNE =
 tf.data.experimental.AUTOTUNE
train_ds =
train_ds.cache().shuffle(1000).prefetch(buffer_size=AUTOTUNE)
val_ds =
val_ds.cache().prefetch(buffer_size=AUTOTUNE)

# нормирование данных
normalization_layer =
layers.experimental.preprocessing.Rescaling(1./255)

normalized_ds = train_ds.map(lambda x, y: (normalization_layer(x), y))
image_batch, labels_batch =
 next(iter(normalized_ds))
first_image = image_batch[0]
# значения пикселей теперь находятся в `[0,1]`.
print(np.min(first_image), np.max(first_image))

# создание модели
num_classes = 5

model = Sequential([
  layers.experimental.preprocessing.Rescaling(1./255,          input_shape=(img_height,
img_width, 3)),
  layers.Conv2D(16, 3, padding='same', activation='relu'),
  layers.MaxPooling2D(),
  layers.Conv2D(32, 3, padding='same', activation='relu'),
  layers.MaxPooling2D(),
  layers.Conv2D(64, 3, padding='same', activation='relu'),
  layers.MaxPooling2D(),
  layers.Flatten(),
  layers.Dense(128, activation='relu'),
  layers.Dense(num_classes)
])
```

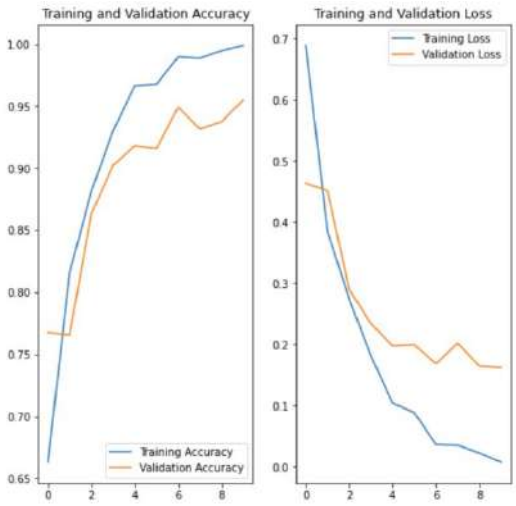
```
# отладка модели
model.compile(optimizer='adam',
              loss=tf.keras.losses.SparseCategoricalCrossentropy(from_logits=True),
              metrics=['accuracy'])

# структура модели
model.summary()
```

```
Model: "sequential_20"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
rescaling_20 (Rescaling)     (None, 180, 180, 3)       0
_________________________________________________________________
conv2d_51 (Conv2D)           (None, 180, 180, 16)      448
_________________________________________________________________
max_pooling2d_51 (MaxPooling (None, 90, 90, 16)        0
_________________________________________________________________
conv2d_52 (Conv2D)           (None, 90, 90, 32)        4640
_________________________________________________________________
max_pooling2d_52 (MaxPooling (None, 45, 45, 32)        0
_________________________________________________________________
conv2d_53 (Conv2D)           (None, 45, 45, 64)        18496
_________________________________________________________________
max_pooling2d_53 (MaxPooling (None, 22, 22, 64)        0
_________________________________________________________________
flatten_17 (Flatten)         (None, 30976)             0
_________________________________________________________________
dense_34 (Dense)             (None, 128)               3965056
_________________________________________________________________
dense_35 (Dense)             (None, 5)                 645
=================================================================
Total params: 3,989,285
Trainable params: 3,989,285
Non-trainable params: 0
_________________________________________________________________
```

```
# обучение модели
epochs=10
history = model.fit(
  train_ds,
  validation_data=val_ds,
  epochs=epochs
)
```

```
Epoch 1/10
64/64 [==============================] - 2s 23ms/step - loss: 0.9675 - accuracy: 0.5803 - val_loss: 0.4635 - val_accuracy: 0.7676
Epoch 2/10
64/64 [==============================] - 1s 21ms/step - loss: 0.3871 - accuracy: 0.8114 - val_loss: 0.4516 - val_accuracy: 0.7656
Epoch 3/10
64/64 [==============================] - 1s 21ms/step - loss: 0.2915 - accuracy: 0.8703 - val_loss: 0.2902 - val_accuracy: 0.8633
Epoch 4/10
64/64 [==============================] - 1s 20ms/step - loss: 0.1923 - accuracy: 0.9308 - val_loss: 0.2342 - val_accuracy: 0.9023
Epoch 5/10
64/64 [==============================] - 1s 23ms/step - loss: 0.1069 - accuracy: 0.9648 - val_loss: 0.1977 - val_accuracy: 0.9180
Epoch 6/10
64/64 [==============================] - 1s 20ms/step - loss: 0.0649 - accuracy: 0.9800 - val_loss: 0.1995 - val_accuracy: 0.9160
Epoch 7/10
64/64 [==============================] - 1s 20ms/step - loss: 0.0404 - accuracy: 0.9893 - val_loss: 0.1686 - val_accuracy: 0.9492
Epoch 8/10
64/64 [==============================] - 1s 21ms/step - loss: 0.0365 - accuracy: 0.9885 - val_loss: 0.2019 - val_accuracy: 0.9316
Epoch 9/10
64/64 [==============================] - 1s 21ms/step - loss: 0.0246 - accuracy: 0.9945 - val_loss: 0.1649 - val_accuracy: 0.9375
Epoch 10/10
64/64 [==============================] - 1s 20ms/step - loss: 0.0113 - accuracy: 0.9982 - val_loss: 0.1621 - val_accuracy: 0.9551
```

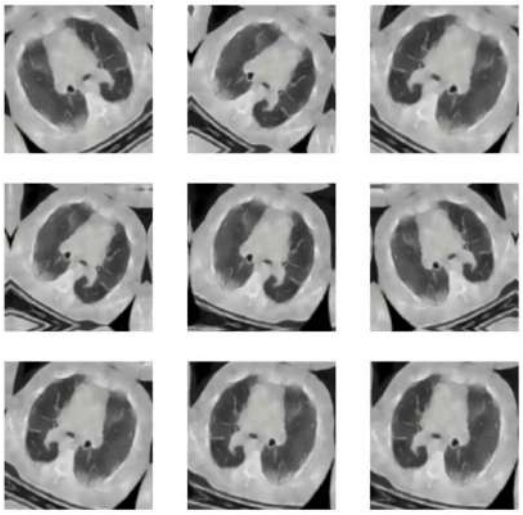
```
# визуализация результатов
acc = history.history['accuracy']
val_acc =
 history.history['val_accuracy']

loss=history.history['loss']
val_loss=history.history['val_loss']

epochs_range = range(epochs)

plt.figure(figsize=(8, 8))
plt.subplot(1, 2, 1)
plt.plot(epochs_range, acc, label='Training Accuracy')
plt.plot(epochs_range, val_acc, label='Validation Accuracy')
plt.legend(loc='lower right')
plt.title('Training and Validation Accuracy')

plt.subplot(1, 2, 2)
plt.plot(epochs_range, loss, label='Training Loss')
plt.plot(epochs_range, val_loss, label='Validation Loss')
plt.legend(loc='upper right')
plt.title('Training and Validation Loss')
plt.show()
```



Переобучение (Overfitting)
На графиках выше точность обучения линейно увеличивается со временем, тогда как точность проверки в процессе обучения составляет 95%. Разница между точностью обучения и точностью проверки - признак переобучения.
Когда модель обучается на небольшом количестве данных, она начинает учиться на шумах или нежелательных деталях обучающих примеров. До такой степени, что это отрицательно влияет на точность модели на новых примерах. Это явление известно как переобучение. Это значит, что модели будет сложно обобщить новый набор данных.
Есть несколько способов борьбы с переобучением в тренировочном процессе. В этом руководстве вы будете использовать увеличение данных и добавление Dropout в свою модель.

Расширение набора данных (Data augmentation)
Переобучение обычно происходит при небольшом количестве обучающих примеров. Увеличение данных использует подход, основанный на создании дополнительных обучающих данных из существующих примеров, путем использования случайных преобразований, которые дают правдоподобные изображения. Это дает возможность модели потренироваться на большем количестве данных и лучше обобщить.
Для аугментации данных мы будем использовать Keras Preprocessing Layers. Слои предварительной обработки(Keras Preprocessing Layers) могут быть включены в модель, как и другие слои, и запускаться на графическом процессоре.

```
data_augmentation =
 keras.Sequential(
  [
    layers.experimental.preprocessing.RandomFlip("horizontal",
input_shape=(img_height,
img_width,                                                    3)),
  layers.experimental.preprocessing.RandomRotation(0.1),
  layers.experimental.preprocessing.RandomZoom(0.1),
  ]
)

plt.figure(figsize=(10, 10))
for images, _ in train_ds.take(1):
  for i in range(9):
    augmented_images =
 data_augmentation(images)
    ax = plt.subplot(3, 3, i + 1)
    plt.imshow(augmented_images[0].numpy().astype("uint8"))
    plt.axis("off")
```



```
# исключение из обучения (Dropout)
model = Sequential([
  data_augmentation,
  layers.experimental.preprocessing.Rescaling(1./255),
  layers.Conv2D(16, 3, padding='same', activation='relu'),
  layers.MaxPooling2D(),
  layers.Conv2D(32, 3, padding='same', activation='relu'),
  layers.MaxPooling2D(),
  layers.Conv2D(64, 3, padding='same', activation='relu'),
  layers.MaxPooling2D(),
  layers.Dropout(0.2),
  layers.Flatten(),
  layers.Dense(128, activation='relu'),
  layers.Dense(num_classes)
])

# отладка и обучение модели
model.compile(optimizer='adam',
              loss=tf.keras.losses.SparseCategoricalCrossentropy(from_logits=True),
              metrics=['accuracy'])
model.summary()
```

```
Model: "sequential_24"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
sequential_21 (Sequential)   (None, 180, 180, 3)       0
_________________________________________________________________
rescaling_23 (Rescaling)     (None, 180, 180, 3)       0
_________________________________________________________________
conv2d_60 (Conv2D)           (None, 180, 180, 16)      448
_________________________________________________________________
max_pooling2d_60 (MaxPooling (None, 90, 90, 16)        0
_________________________________________________________________
conv2d_61 (Conv2D)           (None, 90, 90, 32)        4640
_________________________________________________________________
max_pooling2d_61 (MaxPooling (None, 45, 45, 32)        0
_________________________________________________________________
conv2d_62 (Conv2D)           (None, 45, 45, 64)        18496
_________________________________________________________________
max_pooling2d_62 (MaxPooling (None, 22, 22, 64)        0
_________________________________________________________________
dropout_17 (Dropout)         (None, 22, 22, 64)        0
_________________________________________________________________
flatten_20 (Flatten)         (None, 30976)             0
_________________________________________________________________
dense_40 (Dense)             (None, 128)               3965056
_________________________________________________________________
dense_41 (Dense)             (None, 5)                 645
=================================================================
Total params: 3,989,285
Trainable params: 3,989,285
Non-trainable params: 0
_________________________________________________________________
```

```
epochs = 10
history = model.fit(
  train_ds,
  validation_data=val_ds,
  epochs=epochs
)
```

```
Epoch 1/10
64/64 [==============================] - 3s 29ms/step - loss: 0.9465 - accuracy: 0.4875 - val_loss: 0.6303 - val_accuracy: 0.6348
Epoch 2/10
64/64 [==============================] - 2s 27ms/step - loss: 0.6486 - accuracy: 0.6820 - val_loss: 0.5785 - val_accuracy: 0.7031
Epoch 3/10
64/64 [==============================] - 2s 27ms/step - loss: 0.6136 - accuracy: 0.6546 - val_loss: 0.5478 - val_accuracy: 0.7168
Epoch 4/10
64/64 [==============================] - 2s 27ms/step - loss: 0.5907 - accuracy: 0.6888 - val_loss: 0.5363 - val_accuracy: 0.7051
Epoch 5/10
64/64 [==============================] - 2s 27ms/step - loss: 0.5874 - accuracy: 0.7258 - val_loss: 0.4992 - val_accuracy: 0.7598
Epoch 6/10
64/64 [==============================] - 2s 27ms/step - loss: 0.5536 - accuracy: 0.7220 - val_loss: 0.5289 - val_accuracy: 0.7305
Epoch 7/10
64/64 [==============================] - 2s 27ms/step - loss: 0.5374 - accuracy: 0.7046 - val_loss: 0.5074 - val_accuracy: 0.7363
Epoch 8/10
64/64 [==============================] - 2s 27ms/step - loss: 0.5434 - accuracy: 0.7176 - val_loss: 0.5416 - val_accuracy: 0.7090
Epoch 9/10
64/64 [==============================] - 2s 27ms/step - loss: 0.4989 - accuracy: 0.7461 - val_loss: 0.4751 - val_accuracy: 0.7559
Epoch 10/10
64/64 [==============================] - 2s 27ms/step - loss: 0.5171 - accuracy: 0.7375 - val_loss: 0.4900 - val_accuracy: 0.7480
```

визуализация результатов
После применения аугментации данных и слоя `Dropout` переобучение уменьшилось, а точность на обучении и на контроле стали гораздо ближе.

```
acc = history.history['accuracy']
val_acc =
 history.history['val_accuracy']

loss = history.history['loss']
val_loss =
 history.history['val_loss']

epochs_range = range(epochs)

plt.figure(figsize=(8, 8))
plt.subplot(1, 2, 1)
plt.plot(epochs_range, acc, label='Training Accuracy')
plt.plot(epochs_range, val_acc, label='Validation Accuracy')
plt.legend(loc='lower right')
plt.title('Training and Validation Accuracy')

plt.subplot(1, 2, 2)
plt.plot(epochs_range, loss, label='Training Loss')
plt.plot(epochs_range, val_loss, label='Validation Loss')
plt.legend(loc='upper right')
plt.title('Training and Validation Loss')
plt.show()
```

предсказание на новых данных
Проверим как модель классифицирует изображение, которого не было ни в тренировочных, ни в проверочных данных.

Аугментация данных и Dropout неактивны во время получения предсказаний (метод predict).

```
data_dir =
 '/content/drive/MyDrive/covid.jpg'
data_dir = pathlib.Path(data_dir)

img =
 keras.preprocessing.image.load_img(
    data_dir, target_size=(img_height, img_width)
)
img_array =
 keras.preprocessing.image.img_to_array(img)
```

```
img_array =
 tf.expand_dims(img_array, 0)

predictions =
 model.predict(img_array)
score =
 tf.nn.softmax(predictions[0])

print(
    "Снимок относится к {} с {:.2f} % вероятности."
    .format(class_names[np.argmax(score)], 100 * np.max(score))
)
```

```
This image most likely belongs to CT_Non_Covid-19 with a 96.89 percent confidence.
```

Задание 7.

Запустите код, написанный для анализа сигналов ЭЭГ и их классификации по умственной нагрузке:
https://colab.research.google.com/drive/1NzgkDcT1vnNYhg3_lJh0_lFkdEF7uSFp?usp=sharing
Собрать датафрейм для обучения из набора многоканальных временных рядов ЭЭГ.
Выбрать модель и построить классификатор умственной нагрузке.
Повысить точность классификации умственной нагрузки, изменяя значения параметров модели сверточной нейронной сети.

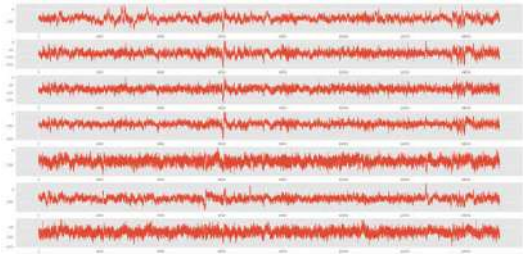
Пример выполнения:

```
from eeg_learn_functions import *
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
import scipy.stats as scs
import re
from numpy import genfromtxt

%matplotlib inline
plt.style.use('ggplot')
from IPython.core.display import display, HTML
display(HTML("<style>.container { width:100% !important; }</style>"))
plt.rcParams["figure.figsize"] =
 (12,12)
pd.options.display.max_columns =
 None
pd.options.display.precision = 4
```

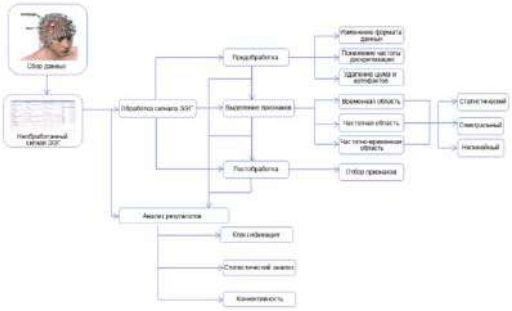
```
from numpy import genfromtxt
data =
 genfromtxt('data/ML101_KS.csv', delimiter=',').T
data
```

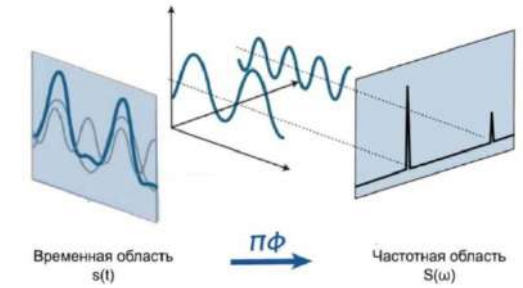
```
array([[ -78.03,  -75.99,  -88.74, ...,  -69.87, -102.  , -116.79],
       [ -79.56,  -77.01,  -92.82, ...,  -59.67,  -97.41, -121.38],
       [ -77.52,  -77.01,  -92.31, ...,  -71.91, -102.  , -139.23],
       ...,
       [ -81.09,  -94.35,  -92.31, ...,  -92.82,  -67.83,  -75.48],
       [ -83.64,  -94.86,  -94.86, ...,  -92.31,  -71.91,  -84.15],
       [ -94.35,  -95.37,  -99.45, ..., -103.53,  -71.4 ,  -98.94]])
```

```
fig, axs = plt.subplots(14, 1, figsize=(30,30))
for i, ax in enumerate(fig.axes):
    ax.plot(df.iloc[:,i])
plt.show()
```



этапы обработки ЭЭГ сигналов





ЭЭГ характеристики

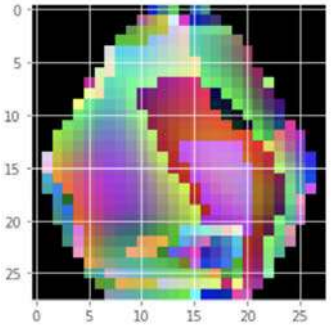
Частоты мозговых волн:
Gamma, 30 to 50 Hz.
Beta, 14 to 30 Hz.
Alpha, 8 to 14 Hz.
Theta, 4 to 8 Hz.
Delta, 0.1 to 4 Hz.

```
theta = (4,8)
alpha = (8,12)
beta = (12,40)

def get_fft(snippet):
    # частота сэмплирования
    Fs = 128.0;
    snippet_time = len(snippet)/Fs
    Ts = 1.0/Fs; # интервал
    # вектор времени
```

```
    t = np.arange(0,snippet_time,Ts)
    y = snippet
    n = len(y)
    k = np.arange(n)
    T = n/Fs
    frq = k/T
    frq = frq[range(n//2)]
    Y = np.fft.fft(y)/n
    Y = Y[range(n//2)]
    return frq, abs(Y)

def theta_alpha_beta_averages(f,Y):
    theta_range = (4,8)
    alpha_range = (8,12)
    beta_range = (12,40)
    theta = Y[(f>theta_range[0]) & (f<=theta_range[1])].mean()
    alpha = Y[(f>alpha_range[0]) & (f<=alpha_range[1])].mean()
    beta = Y[(f>beta_range[0]) & (f<=beta_range[1])].mean()
    return theta, alpha, beta

def make_steps(samples, frame_duration, overlap):
    '''
    in:
    samples - количество сэмплов в одной сессии
    frame_duration - длительность фрейма в секундах
    overlap - часть накладываемого фрейма в диапазоне (0,1)

    out: list of tuple ranges
    '''
    Fs = 128
    i = 0
    intervals = []
    samples_per_frame = Fs *
 frame_duration
    while i+samples_per_frame <=
 samples:
        intervals.append((i,i+samples_per_frame))
        i = i + samples_per_frame -
 int(samples_per_frame*overlap)
    return intervals

def make_frames(df,frame_duration):
    '''
    in: датафрейм или массив со всеми каналами, длительностью фрейма в секундах
    out: массив средних значений theta, alpha, beta для каждой пробы в каждый момоент
времени
        размерность: (n-frames,m-probes,k-brainwave bands)
    '''
    Fs = 128.0
    frame_length = Fs*frame_duration
    frames = []
    steps =
make_steps(len(df),frame_duration,overlap)
    for i,_ in enumerate(steps):
        frame = []
        if i == 0:
            continue
        else:
            for channel in df.columns:
                snippet =
 np.array(df.loc[steps[i][0]:steps[i][1],int(channel)])
                f,Y =
  get_fft(snippet)
                theta, alpha, beta =
 theta_alpha_beta_averages(f,Y)
                frame.append([theta, alpha, beta])

        frames.append(frame)
    return np.array(frames)

# расположение электродов
```

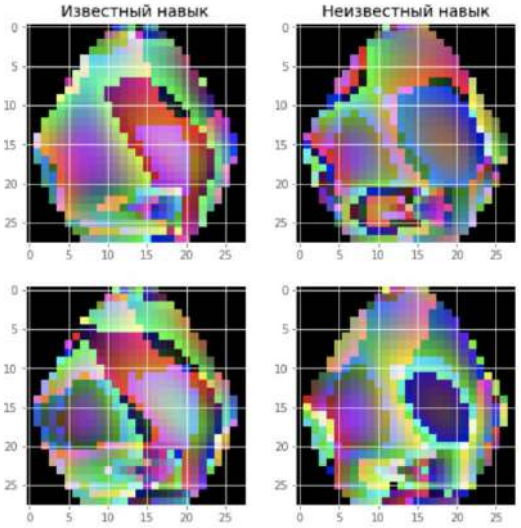
```
locs_2d = [(-2.0,4.0),
           (2.0,4.0),
           (-1.0,3.0),
           (1.0,3.0),
           (-3.0,3.0),
           (3.0,3.0),
           (-2.0,2.0),
           (2.0,2.0),
           (-2.0,-2.0),
           (2.0,-2.0),
           (-4.0,1.0),
           (4.0,1.0),
           (-1.0,-3.0),
           (1.0,-3.0)]
def make_data_pipeline(file_names,labels,image_size,frame_duration,overlap):
    '''
    in:
    file_names - список строк для каждого входного файла (один на каждого индивида)
    labels - лист меток
    image_size - размер выходного изображения в формате (x, x)
    frame_duration - длительность каждого фрейма (секунды)
    overlap - часть накладываемого фрейма в диапазоне (0,1)
    out:
    X: np array фреймов (unshuffled)
    y: np array меток для каждого фрейма (1 или 0)
    '''
        Fs = 128.0   #частота сэмлирования
    frame_length = Fs * frame_duration

    print('Генерирование обучающей выборки ...')
    for i, file in enumerate(file_names):
        print ('Сессия обработки: ', file, '. (', i+1, ' of ', len(file_names), ')')
        data = genfromtxt(file, delimiter=',').T
        df = pd.DataFrame(data)
        X_0 = make_frames(df, frame_duration)
        X_1 = X_0.reshape(len(X_0), 14*3)

        images =
 gen_images(np.array(locs_2d), X_1, image_size, normalize=False)
        images = np.swapaxes(images, 1, 3)
        print(len(images), ' фреймов, сгенерированных с меткой ', labels[i], '.')
        print('\n')
        if i == 0:
            X = images
            y =
 np.ones(len(images))*labels[0]
            print(y[0])
            print(y.shape)
        else:
            X = np.concatenate((X, images),axis = 0)
            print(labels[i])
            y = np.concatenate((y, np.ones(len(images))*labels[i]), axis = 0)
            print(y)
            print(y.shape)
    return X, np.array(y)

file_names = ['data/ML101_KS.csv',
              'data/ML101_US.csv',
              'data/ML102_KS.csv',
              'data/ML102_US.csv',
              'data/ML103_KS.csv',
              'data/ML103_US.csv',
              'data/ML104_KS.csv',
              'data/ML104_US.csv',
              'data/ML105_KS.csv',
              'data/ML105_US.csv',
              'data/ML106_KS.csv',
              'data/ML106_US.csv',
              'data/ML107_KS.csv',
              'data/ML107_US.csv',
              'data/ML108_KS.csv',
```

```
             'data/ML108_US.csv']
labels = [1, 0, 1, 0, 1, 0, 1, 0, 1, 0, 1, 0, 1, 0, 1, 0]
image_size = 28
frame_duration = 1.0
overlap = 0.5
X, y =
 make_data_pipeline(file_names, labels, image_size, frame_duration, overlap)

Генерирование обучающей выборки ...
Сессия обработки:  data/ML101_KS.csv . ( 1  of  16 )
234  фреймов, сгенерированных с меткой  1 .
Сессия обработки:  data/ML101_US.csv . ( 2  of  16 )
224  фреймов, сгенерированных с меткой  0 .
Сессия обработки:  data/ML102_KS.csv . ( 3  of  16 )
222  фреймов, сгенерированных с меткой  1 .
Сессия обработки:  data/ML102_US.csv . ( 4  of  16 )
218  фреймов, сгенерированных с меткой  0 .
Сессия обработки:  data/ML103_KS.csv . ( 5  of  16 )
226  фреймов, сгенерированных с меткой  1 .
Сессия обработки:  data/ML103_US.csv . ( 6  of  16 )
208  фреймов, сгенерированных с меткой  0 .
Сессия обработки:  data/ML104_KS.csv . ( 7  of  16 )
202  фреймов, сгенерированных с меткой  1 .
Сессия обработки:  data/ML104_US.csv . ( 8  of  16 )
204  фреймов, сгенерированных с меткой  0 .
Сессия обработки:  data/ML105_KS.csv . ( 9  of  16 )
214  фреймов, сгенерированных с меткой  1 .
Сессия обработки:  data/ML105_US.csv . ( 10  of  16 )
226  фреймов, сгенерированных с меткой  0 .
Сессия обработки:  data/ML106_KS.csv . ( 11  of  16 )
230  фреймов, сгенерированных с меткой  1 .
Сессия обработки:  data/ML106_US.csv . ( 12  of  16 )
278  фреймов, сгенерированных с меткой  0 .
Сессия обработки:  data/ML107_KS.csv . ( 13  of  16 )
246  фреймов, сгенерированных с меткой  1 .
Сессия обработки:  data/ML107_US.csv . ( 14  of  16 )
236  фреймов, сгенерированных с меткой  0 .
Сессия обработки:  data/ML108_KS.csv . ( 15  of  16 )
240  фреймов, сгенерированных с меткой  1 .
Сессия обработки:  data/ML108_US.csv . ( 16  of  16 )
234  фреймов, сгенерированных с меткой  0 .

X.shape
(3642, 28, 28, 3)
y.shape
(3642,)

import matplotlib.pyplot as plt
%matplotlib inline
plt.imshow((X[0] *
 255).astype(np.uint8));
```




```
f, axarr = plt.subplots(2,2, figsize = (8,8))
axarr[0][0].set_title('Известный навык')
axarr[0][0].imshow((X[0]*
 255).astype(np.uint8))
axarr[1][0].imshow((X[1]*
 255).astype(np.uint8))
axarr[0][1].set_title('Неизвестный навык')
axarr[0][1].imshow((X[20]*
```

```
 255).astype(np.uint8))
axarr[1][1].imshow((X[21]*
 255).astype(np.uint8))
```




```
from sklearn.model_selection import train_test_split
x_train, x_test, y_train, y_test =
 train_test_split(X, y,
 test_size=0.20,shuffle=True)

# input image dimensions
img_rows, img_cols = 28, 28

x_train = x_train.astype('float32')
x_test = x_test.astype('float32')
print('x_train размер:',
 x_train.shape)
print(x_train.shape[0], 'обучающих
 примеров')
print(x_test.shape[0], 'контрольных
 примеров')
input_shape = (img_rows, img_cols,
 3)

x_train размер: (2913, 28, 28, 3)
2913 обучающих примеров
729 контрольных примеров

#  построение модели сверточной нейронной сети
from tensorflow import keras
from keras.models import Sequential
from keras.layers import Dense, Dropout, Activation, Flatten
from keras.layers import Conv2D, MaxPooling2D

batch_size = 128
num_classes = 2
epochs = 400

# преобразование вектора меток в бинарные матрицы
y_train = keras.utils.to_categorical(y_train, num_classes)
y_test = keras.utils.to_categorical(y_test, num_classes)

model = Sequential()
model.add(Conv2D(32, (3, 3), padding='same',input_shape=input_shape))
model.add(Activation('relu'))
model.add(Conv2D(32, (3, 3)))
model.add(Activation('relu'))
model.add(MaxPooling2D(pool_size=(2, 2)))
model.add(Flatten())
model.add(Dense(10))
model.add(Activation('relu'))
model.add(Dense(num_classes))
model.add(Activation('softmax'))
```

```
# инициализация алгоритма оптимизации
opt =
 keras.optimizers.RMSprop(lr=0.0001, decay=1e-6)

# обучение модели
model.compile(loss='categorical_crossentropy', optimizer=opt,
              metrics=['accuracy'])

x_train = x_train.astype('float32')
x_test = x_test.astype('float32')

model.fit(x_train, y_train,
          batch_size=batch_size,
          epochs=epochs,
          validation_data=(x_test, y_test), shuffle=True)
```

```
23/23 [==============================] - 4s 172ms/step - loss: 0.3511 - accuracy: 0.8373 - val_loss: 0.4984 - val_accuracy: 0.7764
Epoch 386/400
23/23 [==============================] - 4s 172ms/step - loss: 0.3552 - accuracy: 0.8400 - val_loss: 0.4961 - val_accuracy: 0.7819
Epoch 387/400
23/23 [==============================] - 4s 172ms/step - loss: 0.3488 - accuracy: 0.8507 - val_loss: 0.5078 - val_accuracy: 0.7764
Epoch 388/400
23/23 [==============================] - 4s 171ms/step - loss: 0.3482 - accuracy: 0.8435 - val_loss: 0.4997 - val_accuracy: 0.7805
Epoch 389/400
23/23 [==============================] - 4s 171ms/step - loss: 0.3535 - accuracy: 0.8445 - val_loss: 0.5016 - val_accuracy: 0.7833
Epoch 390/400
23/23 [==============================] - 4s 171ms/step - loss: 0.3488 - accuracy: 0.8479 - val_loss: 0.6072 - val_accuracy: 0.7449
Epoch 391/400
23/23 [==============================] - 4s 172ms/step - loss: 0.3450 - accuracy: 0.8520 - val_loss: 0.5055 - val_accuracy: 0.7723
Epoch 392/400
23/23 [==============================] - 4s 172ms/step - loss: 0.3497 - accuracy: 0.8462 - val_loss: 0.4964 - val_accuracy: 0.7833
Epoch 393/400
23/23 [==============================] - 4s 172ms/step - loss: 0.3453 - accuracy: 0.8435 - val_loss: 0.5039 - val_accuracy: 0.7929
Epoch 394/400
23/23 [==============================] - 4s 171ms/step - loss: 0.3499 - accuracy: 0.8520 - val_loss: 0.4979 - val_accuracy: 0.7901
Epoch 395/400
23/23 [==============================] - 4s 172ms/step - loss: 0.3463 - accuracy: 0.8448 - val_loss: 0.5081 - val_accuracy: 0.7723
Epoch 396/400
23/23 [==============================] - 4s 172ms/step - loss: 0.3465 - accuracy: 0.8428 - val_loss: 0.4968 - val_accuracy: 0.7833
Epoch 397/400
23/23 [==============================] - 4s 171ms/step - loss: 0.3472 - accuracy: 0.8465 - val_loss: 0.4923 - val_accuracy: 0.7846
Epoch 398/400
23/23 [==============================] - 4s 172ms/step - loss: 0.3344 - accuracy: 0.8562 - val_loss: 0.5123 - val_accuracy: 0.7737
Epoch 399/400
23/23 [==============================] - 4s 170ms/step - loss: 0.3404 - accuracy: 0.8452 - val_loss: 0.4955 - val_accuracy: 0.7860
Epoch 400/400
23/23 [==============================] - 5s 203ms/step - loss: 0.3341 - accuracy: 0.8582 - val_loss: 0.5066 - val_accuracy: 0.7750
<keras.callbacks.History at 0x7fc9d83b8ad0>
```

Задание 8.

Построить модель CNN для классификации ЭЭГ паттернов воображаемых моторных команд, используя датасет Physionet MI/ME:
https://www.physionet.org/pn4/eegmmidb/

Пример выполнения:

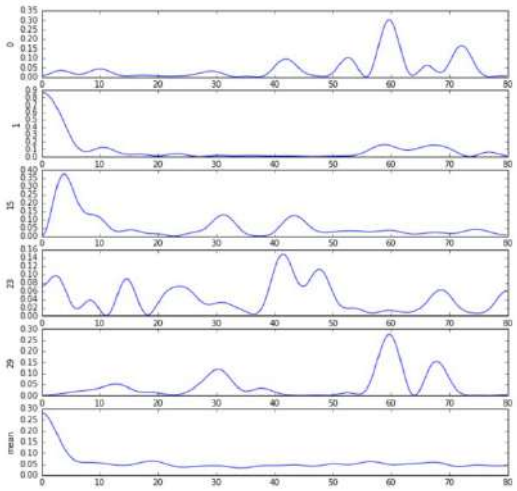
```
from keras.callbacks import ModelCheckpoint

SPLITS = 5
input_length = 3 * 160 # = 3s
electrodes = range(64)
epochs = 3
epoch_steps = 5 # record performance 5 times per epoch
batch = 16
#nclasses = [2, 3, 4]
nclasses = [3]
#splits = range(5)
splits = [0]

results = np.zeros((len(nclasses), len(splits), 4, epochs*epoch_steps))
for j,nclasses in enumerate(nclasses):
    try:
        del X,y
    except:
        pass
    X,y =
 nndata.load_raw_data(electrodes=electrodes, num_classes=nclasses)

    steps_per_epoch =
 np.prod(X.shape[:2]) / batch * (1-1./SPLITS) / epoch_steps
    for ii,i in enumerate(splits):
        print "%d CLASS, SPLIT %d" % (nclasses, i)
        idx = range(len(X))
        train_idx, test_idx =
 nndata.split_idx(i, 5, idx)

        model =
 models.create_raw_model(
            nchan=len(electrodes),
            nclasses=nclasses,
            trial_length=input_length
        )
```

```
        weights_path =
 "weights-%dcl-%d.hdf5" % (nclasses,i)
        checkpoint =
[ModelCheckpoint(filepath=weights_path, save_best_only=True)]

        h = models.fit_model(
            model, X, y, train_idx, test_idx, input_length=input_length,
            batch_size=batch,                          steps_per_epoch=steps_per_epoch,
epochs=epochs*epoch_steps,
            callbacks=checkpoint
        )

        results[j, ii, :, :] = [
            h.history["acc"],
            h.history["loss"],
            h.history["val_acc"],
            h.history["val_loss"]
        ]

SPLITS = 5

input_length = 3*160
electrodes = range(64)
classes = 2

X, y =
nndata.load_raw_data(electrodes=electrodes, num_classes=classes)

weights_file = "weights-2cl-%d.hdf5"
model = models.create_raw_model(
    nchan=len(electrodes),
    nclasses=classes,
    trial_length=input_length
)

results = np.zeros((len(X), 2))

for split in range(5):
    idx = range(len(X))
    train_idx, test_idx =
 nndata.split_idx(split, SPLITS, idx)
    Xsub, ysub =
 nndata.crossval_test(X, y,
 test_idx, input_length,
 flatten=False)

    # for each subject in the training set
    for i, subject_X in enumerate(Xsub):
        current_subject =
 test_idx[i]
        subject_y = ysub[i,:]
        print current_subject,
                model.load_weights(weights_file % (split))
        results[current_subject, 0] = model.evaluate(
            subject_X.reshape((-1,) + (input_length, len(electrodes), 1)),
            subject_y.reshape((-1, classes)), verbose=0
        )[1]

        epochs = 5
        temp_results = []
        for subject_split in
 range(4):
            trial_idx =
 range(len(subject_X))
            train_trials, test_trials =
 nndata.split_idx(subject_split, 4,
 trial_idx)

            model.load_weights(weights_file %
 (split))
            h = model.fit(
```

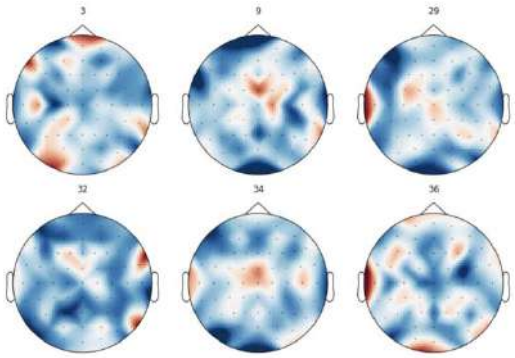
```
                subject_X[train_trials,:], subject_y[train_trials,:],
                validation_data=(subject_X[test_trials,:], subject_y[test_trials,:]),
                epochs=5, batch_size=2, verbose=0
            )
                        temp_results.append(np.max(h.history["val_acc"]))
        results[current_subject, 1] = np.mean(temp_results)

BATCH = 16
SPLITS = 5
electrodes = range(64)
classes = 3
X,y =
 nndata.load_raw_data(electrodes, num_classes=classes)

epochs = 3
splits = range(5)
steps_per_epoch =
 np.prod(X.shape[:2]) / BATCH * (1-1./SPLITS)

#lengths = np.arange(30, 160, 10) #
 short, less spacing
lengths = np.arange(160, 960, 40) #
 long, more spacing

results = np.zeros((len(lengths), len(splits), 4, epochs))
for ii,i in enumerate(splits):
    for j, length in enumerate(lengths):
        print "Length %d, SPLIT %d" % (length, i)
        idx = range(len(X))
        train_idx, test_idx =
 nndata.split_idx(i, 5, idx)
        model =
 models.create_raw_model(
            nchan=len(electrodes),
            nclasses=classes,
            trial_length=length
        )

        h = models.fit_model(
            model, X, y, train_idx, test_idx,
            input_length=length, batch_size=BATCH,
            steps_per_epoch=steps_per_epoch, epochs=epochs
        )
        results[j, ii, :, :] = [
            h.history["acc"],
            h.history["loss"],
            h.history["val_acc"],
            h.history["val_loss"]
        ]

classes = 3
split = 0

step = 20
trial_length = 6*160
channels =  range(64)
# load model
model =
models.create_raw_model(nchan=len(channels),                          nclasses=classes,
trial_length=input_length)
model.load_weights("weights-%dcl-%d.hdf5" % (classes, split))

idx = range(len(X))
train_idx, test_idx =
 nndata.split_idx(split, 5, idx)
acc =
np.zeros((trial_length-input_length)/ step + 1)
cert =
 np.zeros((trial_length-input_length)/ step + 1)
falsenull =
np.zeros((trial_length-input_length)/ step + 1)
```

```
for i, off in enumerate(np.arange(0, trial_length-input_length, step)):
    print off,
    xt,yt = nndata.crossval_test(X, y, test_idx, input_length, fix_offset=off)
    yp = model.predict(xt)
    cert[i] = yp[yt>0].mean()
    acc[i] =
 sum(yp.argmax(1)==yt.argmax(1)) /
 float(len(yt))
    if classes > 2:
        falsenull[i] =
 sum(yp[yt[:,2]==0].argmax(1)==2) /
 float(len(yt))

np.save("online-certainty.npy")
np.save("online-accuracy.npy")
if classes > 2:
    np.save("online-falsenull.npy")

from keras import backend as K
import matplotlib.pyplot as plt
import numpy as np
from sklearn.decomposition import PCA
import nndata
import models

input_length = 960
electrodes = range(64)
classes = [2,3,4]
split = 0
weights_path = "weights-%dcl-0.hdf5"

for cls in classes:
    X,y =
nndata.load_raw_data(electrodes=electrodes, num_classes=cls)
    model =
models.create_raw_model(nchan=len(electrodes),                                nclasses=cls,
trial_length=input_length, l1=0)
    model.load_weights(weights_path % cls)

    get_layer_output =
 K.function([model.layers[0].input], [model.layers[4].output])

    idx = range(len(X))
    train_idx, test_idx =
 nndata.split_idx(split, 5, idx)
    Xtest,ytest =
 nndata.crossval_test(X, y, test_idx, input_length)
    results =
 np.zeros((Xtest.shape[0], model.layers[4].output_shape[1]))
    for i, xtest in enumerate(Xtest):
        results[i,:] =
 get_layer_output([[xtest]])[0].reshape(-1)
    savemap = dict()
    savemap["output"] = results
    savemap["classes"] =
 ytest.argmax(1)
    savemap["classification"] =
 model.predict(Xtest).argmax(1)
    np.save("pca/%dclass.npy" % cls, savemap)

fnames = ["pca/2class.npy",
 "pca/3class.npy", "pca/4class.npy"]
markers = ["xr", "xb", "xg", "xk"]
plt.figure(figsize=(16,8))
for i, name in enumerate(fnames):
    m = np.load(name).item()
    results = m["output"]
    labels = m["classes"]

    pca = PCA(n_components=4)
    pca.fit(results)
```

```
    Xtrans = pca.transform(results)

    plt.subplot(1,3,i+1)
    for j in [0,1,2,3]:
        plt.plot(Xtrans[labels==j, 0], Xtrans[labels==j, 1], markers[j])

    plt.xlabel("PC 0")
    plt.ylabel("PC 1")
    print "File %s" % name,
    print          "variance          explained:          %.2f          %%"          %
(np.sum(pca.explained_variance_ratio_[:2])*100)

import numpy as np
from scipy.signal import freqz
import math
import models
from matplotlib import pyplot as plt
from matplotlib2tikz import save as tikz_save

electrodes = range(64)
nclasses=3
input_length = 3 * 160

model = models.create_raw_model(
    nchan=len(electrodes),
    nclasses=nclasses,
    trial_length=input_length
)

model.load_weights("weights-3cl-0.hdf5")

filters =
model.get_weights()[0].reshape((30,40))

results = np.zeros((40,512))
for i in range(40):
    f = filters[:,i]
    freq, h = freqz(f,[1])
    results[i,:] = abs(h) ** 2

def mid_freq(row, freq):
    n = len(row)
    nrow = row/row.mean()
    return sum(np.multiply(
nrow[:n/2+1],freq[:n/2+1]))


order = map(lambda r: mid_freq(r, freq), results)
idx_ordered = sorted(range(40), key=lambda v: order[v])

plt.figure(figsize=(10,10))
#selection = range(40)
selection = [0,1,15,23,29]
for i,s in enumerate(selection):
    plt.subplot(len(selection)+1,1,i+1)
    plt.plot(freq / math.pi * 80, results[idx_ordered[s]])
    #plt.ylabel("|H(f)_%d|^2" % s)
    plt.ylabel(s)
    #plt.ylim(0,0.6)
plt.subplot(len(selection)+1,1,len(selection)+1)
plt.plot(freq / math.pi * 80, results.mean(0))
plt.ylabel("mean")
pass
```



```
import numpy as np
import models
from scipy.signal import freqz
import math
from matplotlib import pyplot as plt
from matplotlib2tikz import save as tikz_save
import util

electrodes = range(64)
nclasses=3
input_length = 3 * 160

model = models.create_raw_model(
    nchan=len(electrodes),
    nclasses=nclasses,
    trial_length=input_length
)

model.load_weights("weights-3cl-0.hdf5")

idx = range(42) + range(44,63)

channels =
model.get_weights()[2][0,idx,:,:].mean(2)
positions =
util.projection_2d(util.get_physionet_electrode_positions())[idx,:]

import mne
from matplotlib2tikz import save as save_tikz

selection=[3,9,29,32,34,36]
fig, axes = plt.subplots(nrows=2, ncols=3, figsize=(10,7))
for i,idx in enumerate(selection):
    ax = axes[i/3,i%3]
    ax.set_title(idx)
    mne.viz.plot_topomap(channels[:,idx], positions, show=False, axes=ax, contours=0,
image_interp="nearest")

fig.tight_layout()
fig.savefig("foo.pdf", bbox_inches='tight')
```



Позиции электродов:

```
Fp1 -0.308829 0.950477 -0.034899
Fpz 0.000000 0.999391 -0.034899
Fp2 0.308829 0.950477 -0.034899
AF7 -0.615286 0.787531 -0.034899
AF3 -0.375595 0.884846 0.275637
AFz 0.000000 0.933580 0.358368
AF4 0.375595 0.884846 0.275637
AF8 0.615286 0.787531 -0.034899
F7 -0.808524 0.587427 -0.034899
F5 -0.728993 0.633704 0.258819
F3 -0.545007 0.673028 0.500000
F1 -0.286965 0.710264 0.642788
Fz 0.000000 0.719340 0.694658
F2 0.286965 0.710264 0.642788
F4 0.545007 0.673028 0.500000
F6 0.728993 0.633704 0.258819
F8 0.808524 0.587427 -0.034899
FT7 -0.950477 0.308829 -0.034899
FC5 -0.882718 0.338844 0.325568
FC3 -0.676377 0.359636 0.642788
FC1 -0.374710 0.374710 0.848048
FCz 0.000000 0.390731 0.920505
FC2 0.374710 0.374710 0.848048
FC4 0.676377 0.359636 0.642788
FC6 0.882718 0.338844 0.325568
FT8 0.950477 0.308829 -0.034899
T9 -0.906308 0.000000 -0.422618
T7 -0.999391 0.000000 -0.034899
C5 -0.933580 0.000000 0.358368
C3 -0.719340 0.000000 0.694658
C1 -0.390731 0.000000 0.920505
Cz 0.000000 0.000000 1.000000
C2 0.390731 0.000000 0.920505
C4 0.719340 0.000000 0.694658
C6 0.933580 0.000000 0.358368
T8 0.999391 0.000000 -0.034899
T10 0.906308 0.000000 -0.422618
TP7 -0.950477 -0.308829 -0.034899
CP5 -0.882718 -0.338844 0.325568
CP3 -0.676377 -0.359636 0.642788
CP1 -0.374710 -0.374710 0.848048
CPz 0.000000 -0.390731 0.920505
CP2 0.374710 -0.374710 0.848048
CP4 0.676377 -0.359636 0.642788
CP6 0.882718 -0.338844 0.325568
TP8 0.950477 -0.308829 -0.034899
P7 -0.808524 -0.587427 -0.034899
P5 -0.728993 -0.633704 0.258819
P3 -0.545007 -0.673028 0.500000
P1 -0.286965 -0.710264 0.642788
Pz 0.000000 -0.719340 0.694658
P2 0.286965 -0.710264 0.642788
P4 0.545007 -0.673028 0.500000
P6 0.728993 -0.633704 0.258819
P8 0.808524 -0.587427 -0.034899
```

```
PO7 -0.587427 -0.808524 -0.034899
PO3 -0.375595 -0.884846 0.275637
POz 0.000000 -0.933580 0.358368
PO4 0.375595 -0.884846 0.275637
PO8 0.587427 -0.808524 -0.034899
O1 -0.308829 -0.950477 -0.034899
Oz 0.000000 -0.999391 -0.034899
O2 0.308829 -0.950477 -0.034899
Iz 0.000000 -0.906308 -0.422618

# создание модели
import gc
import nndata
from keras.models import Sequential
from keras.layers import Dense, Flatten
from keras.layers.wrappers import TimeDistributed
from keras.layers.convolutional import Conv2D
from keras.layers.pooling import AveragePooling2D
from keras.layers.recurrent import LSTM
from keras import regularizers
from keras.callbacks import ModelCheckpoint

def create_raw_model(nchan,
 nclasses, trial_length=960, l1=0):
    input_shape = (trial_length, nchan, 1)
    model = Sequential()
    model.add(Conv2D(40,            (30,            1),            activation="relu",
kernel_regularizer=regularizers.l1(l1), padding="same", input_shape=input_shape))
    model.add(Conv2D(40,           (1,           nchan),           activation="relu",
kernel_regularizer=regularizers.l1(l1), padding="valid"))
    model.add(AveragePooling2D((30, 1), strides=(15, 1)))
    model.add(Flatten())
    model.add(Dense(80, activation="relu"))
    model.add(Dense(nclasses, activation="softmax"))
    model.compile(loss="categorical_crossentropy", optimizer="adam", metrics=["acc"])
    return model

def create_raw_model2(nchan,
 nclasses, trial_length=960, l1=0, full_output=False):
    input_shape = (trial_length, nchan, 1)
    model = Sequential()
    model.add(Conv2D(40,            (30,            1),            activation="relu",
kernel_regularizer=regularizers.l1(l1), padding="same", input_shape=input_shape))
    model.add(Conv2D(40,           (1,           nchan),           activation="relu",
kernel_regularizer=regularizers.l1(l1), padding="valid"))
    model.add(AveragePooling2D((5, 1), strides=(5, 1)))
    model.add(TimeDistributed(Flatten()))
    model.add(LSTM(40,               activation="sigmoid",               dropout=0.25,
return_sequences=full_output))
    model.add(Dense(nclasses, activation="softmax"))
    model.compile(loss="categorical_crossentropy",               optimizer="rmsprop",
metrics=["acc"])
    return model

def fit_model(model, X, y,
 train_idx, test_idx, input_length=50, batch_size=32, epochs=30, steps_per_epoch=1000,
callbacks=None):
    gc.collect()
    return model.fit_generator(
        nndata.crossval_gen(X,y, train_idx, input_length, batch_size),
        validation_data=nndata.crossval_test(X, y, test_idx, input_length),
        steps_per_epoch=steps_per_epoch, epochs=epochs, callbacks=callbacks
    )
```

IX. НОРМАТИВНЫЕ ПРАВОВЫЕ АКТЫ

Дополнительная профессиональная программа (программа повышения квалификации) ИТ-профиля «Инженерия искусственного интеллекта в медицине» разработана в соответствии с нормами Федерального закона РФ от 29 декабря 2012 года № 273-ФЗ «Об образовании в Российской Федерации», с учетом требований приказа Минобрнауки России от 1 июля 2013 г. № 499 «Об утверждении Порядка организации и осуществления образовательной деятельности по дополнительным профессиональным программам», с изменениями, внесенными приказом Минобрнауки России от 15 ноября 2013 г. № 1244 «О внесении изменений в Порядок организации и осуществления образовательной деятельности по дополнительным профессиональным программам, утвержденный приказом Министерства образования и науки Российской Федерации от 1 июля 2013 г. № 499», приказа Министерства образования и науки РФ от 23 августа 2017 г. N 816 «Об утверждении Порядка применения организациями, осуществляющими образовательную деятельность, электронного обучения, дистанционных образовательных технологий при реализации образовательных программ»; ФГОС ВО по направлению подготовки 09.04.01 Информатика и вычислительная техника (уровень магистратуры), утвержденного приказом Минобрнауки России от 19 сентября 2017 г. № 918, а также профессионального стандарта «Руководитель разработки программного обеспечения» утвержденного приказом Министерства труда и социальной защиты РФ от 22 июля 2022 г. № 423н.