****

# **Реферат**

Пояснительная записка содержит 58 страниц, 10 рисунков, 3 таблицы.

Количество использованных источников 19.

Ключевые слова: пространство эмоций, базовые эмоции, PAD, FACS, action units, виртуальные ассистенты, линейная регрессия и Ridge/Lasso регрессия, Random Forest, KNN, нейронные сети, машинное обучение.

Целью данной работы является продолжение разработки алгоритмов и программных средств, предназначенных для отображения между пространствами эмоций различных размерностей, в виде обучения ML моделей и демонстрации их результатов предсказаний.

В первом разделе рассматриваются вопросы представления эмоций в виде формальных моделей, описана постановка задач работы.

Во втором разделе изучаются существующие решения поставленных задач, анализируются возможности применения различных моделей машинного обучения.

В третьем разделе проводится углубленный анализ данных, а также выбран метод поиска оптимальных значений ключевых параметров модели.

В четвёртом разделе происходит подбор инструментов для обучения моделей, обучение моделей, сравнение их работы с помощью метрик и отбор лучших моделей.

В заключении подводятся итоги проведённой работы с кратким описанием результатов.

**СОДЕРЖАНИЕ**

[**Реферат 2**](#_30j0zll)

[**Введение 5**](#_3dy6vkm)

[**Раздел 1. Аналитическая часть. 7**](#_4d34og8)

[1.1 Технологии и подходы для реализации виртуальных ассистентов. 7](#_2s8eyo1)

[1.2 Изучение и анализ литературы на тему представления эмоций в виде формальных моделей. 9](#_17dp8vu)

[1.2.1 Общие характеристики дискретных и многомерных моделей. 9](#_3rdcrjn)

[1.2.2 Модель эмоционального состояния PAD. 12](#_26in1rg)

[1.2.3 Система кодирования лицевых движений FACS. 13](#_lnxbz9)

[1.3 Постановка задач, исследовательские вопросы и гипотезы. 15](#_35nkun2)

[1.3.1 Постановка задач. 15](#_1ksv4uv)

[1.3.2 Общая информация об исходных данных. 16](#_44sinio)

[1.3.3 Гипотезы о данных и моделях. 16](#_2jxsxqh)

[**Раздел 2. Теоретическая часть 18**](#_1y810tw)

[2.1 Изучение существующих решений поставленных задач. 18](#_4i7ojhp)

[2.1.1 FaceReader. 18](#_2xcytpi)

[2.2 Математическая постановка задач. 19](#_1ci93xb)

[2.2.1 Углубленный анализ данных. 20](#_3whwml4)

[2.2.2 Обучение моделей машинного обучения. 21](#_3as4poj)

[2.3 Сравнение моделей машинного обучения. 22](#_1pxezwc)

[2.3.1 Линейная Регрессия. 23](#_49x2ik5)

[2.3.2 Полиномиальная Регрессия. 23](#_2p2csry)

[2.3.3 Ridge и Lasso Регрессии. 23](#_147n2zr)

[2.3.4 Случайный Лес (Random Forest). 24](#_3o7alnk)

[2.3.5 K-ближайших Соседей (KNN). 25](#_23ckvvd)

[2.3.6 Многослойный Перцептрон (MLPRegressor). 25](#_ihv636)

[**Раздел 3. Инженерная часть 26**](#_1hmsyys)

[3.1 Выбор датасета, анализ данных, формирование обучающей и тестовой выборок. 26](#_41mghml)

[3.1.1 Выбор датасета. 26](#_2grqrue)

[3.1.2 Углубленный анализ данных. 26](#_vx1227)

[3.1.3 Формирование обучающей и тестовой выборок. 33](#_19c6y18)

[3.2 Метод поиска оптимальных значений ключевых параметров моделей. 35](#_3tbugp1)

[3.3 Формирование метрик для оценки моделей. 36](#_28h4qwu)

[**Раздел 4. Технологическая и практическая часть 38**](#_37m2jsg)

[4.1 Выбор инструментов для анализа данных и обучения моделей. 38](#_1mrcu09)

[4.1.1 Инструменты для анализа и обработки данных. 38](#_46r0co2)

[4.1.2 Инструменты для обучения моделей. 40](#_2lwamvv)

[4.1.3 Инструменты для визуализации. 42](#_111kx3o)

[4.1.4 Инструменты для сохранения моделей. 44](#_3l18frh)

[4.2 Обучение моделей. 44](#_206ipza)

[4.3 Оценка точности работы моделей. 46](#_4k668n3)

[4.3.1 Визуализация работы обученных моделей. 46](#_2zbgiuw)

[4.3.2 Выводы по таблицам результатов обучения моделей. 49](#_1egqt2p)

[4.4 Сохранение лучших моделей в формате файла обученных моделей. 54](#_3ygebqi)

[**Заключение 56**](#_sqyw64)

[**СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ 57**](#_3cqmetx)

# 

# 

# 

# 

# **Введение**

Существуют различные подходы к описанию эмоций человека. Одни модели включают в себя фиксированное количество базисных (фундаментальных) эмоций, другие используют ограниченное число переменных (осей), которые задают эмоциональную точку в пространстве. Первые модели называются дискретными, а вторые – многомерными [1].

Исследователи, разделяющие дискретную теорию эмоций, расходятся во мнениях относительно количества и содержания фундаментальных эмоций. Сторонники этого подхода сосредоточены на изучении внутренней организации индивидуальных эмоций, а не на их взаимосвязи. Наиболее известным психологом в этой области считается Пол Экман, выделяющий 6 базовых эмоций — гнев, удивление, страх, радость, грусть и отвращение. В рамках данной работы мы будем считать, что базовых эмоций всего 6 [1].

Многомерные модели эмоциональных состояний различаются между собой количеством измерений (осей) и их смысловой нагрузкой. В данном исследовании мы будем рассматривать многомерную модель PAD [2], в которой всего 3 измерения: P – Pleasure (удовольствие), A – Arousal (возбуждение), D – Dominance (доминирование).

Полом Экманом в XX веке была разработана система кодирования лицевых движений (FACS) [3]. Эта система позволяет классифицировать выражения лица человека на основе его мимики. С использованием FACS можно закодировать практически любое, анатомически возможное выражение лица, деконструируя его на вызвавшие его конкретные двигательные единицы (action units) и их временные сегменты [4].

С помощью двигательных единиц из FACS Полом Экманом и Уоллесом Фризеном была создана эмоциональная система кодирования лицевых движений (EmFACS), которая рассматривает только лицевые движения, связанные с эмоциями. Стоит отметить, что в научном мире нет единого мнения на тему однозначного соответствия между мимикой и эмоциями для всех людей [5]. Таким образом нельзя с полной уверенностью сказать, что, например, выражение лица любого человека, который испытывает страх выглядит именно так, а никак иначе. Однако, большинство людей в современном обществе выражает эмоции приблизительно одинаково, что даёт возможность использовать труды Пола Экмана в данной работе.

С учётом вышесказанного, встаёт вопрос о необходимости иметь возможность отображать пространства эмоций различных размерностей друг в друга.

# **Раздел 1. Аналитическая часть.**

## **1.1 Технологии и подходы для реализации виртуальных ассистентов.**

Виртуальный ассистент [6] (англ. Virtual assistant) — программный агент, который может выполнять задачи (или сервисы) для пользователя на основе информации, введенной пользователем, данных о его местонахождении, а также информации, полученной из различных интернет-ресурсов (погода, уличное движение, новости, курсы валют и ценных бумаг, розничные цены в магазинах и т. д.). Примерами таких агентов являются программы Siri, Google Assistant (Google Now), Amazon Alexa, Microsoft Cortana, Bixby, Voice Mate, Алиса, Маруся, Дуся, Салют и другие.

Искусственный интеллект уже стал частью нашей повседневной жизни. В каждом мобильном телефоне есть умный голосовой помощник, который расскажет о погоде, подскажет, где ближайший банкомат, вызовет такси или закажет билеты в кино. Но это только один из примеров виртуального помощника, самый известный и знакомый. На самом деле у таких решений гораздо больше областей применения.

Так что же такое виртуальный помощник (ассистент)? В общем смысле это сервис (программа, мобильное приложение) — выполняющий действия в ответ на информацию, введенную пользователем и/или найденную в Сети. Кроме того, искусственный интеллект избавляет человека от необходимости самостоятельно выполнять некоторые операции – как правило, простые, то есть такие, которые под силу роботу.

Возможности виртуальных помощников используются во всех сферах жизни. В автомобили внедряются голосовые помощники, чтобы владелец мог без помощи рук открывать и закрывать двери, запускать двигатель, узнавать уровень топлива в бензобаке. Роботы управляют устройствами умного дома, их устанавливают в гостиницах, например, чтобы гости могли заказывать услуги в номер с помощью простых команд.

Основная функция голосовых виртуальных ассистентов – имитировать реальное общение. Уровень развития этой функции у цифровых помощников разный. Некоторые роботы способны поддерживать только самые простые диалоги, читая предварительно записанные звуковые сообщения. Другие неотличимы от человека: понимают смысл обращенной к ним речи, знают синонимы, отвечают без промедления, замолкают, когда их перебивают, и даже употребляют слова-паразиты. В результате собеседник даже не всегда может определить, что ведет диалог с роботом. Понятно, что такие возможности могут быть очень полезны для бизнеса.

Кроме коммуникационных функций, виртуальные ассистенты (голосовые боты) могут выполнять различные действия по заданным сценариям: планировать маршрутизацию звонков, отправлять SMS, записывать результаты вызовов в виде отчетов, использовать рабочие данные по информационным ресурсам компании и даже фиксировать изменения в них – при условии, что робот интегрирован с нужными системами.

Интернет-магазины с помощью голосовых ботов сообщают об акциях и специальных предложениях, регистрируют клиентов в программе лояльности, активируют корзину, увеличивают продажи, помогают клиентам с заказом.

Виртуальный помощник в салоне красоты или другой организации из сферы услуг напомнит клиенту о приеме, при необходимости перенесет время визита, в результате чего показатель «рентабельности» повысится.

Службы доставки посредством цифровых помощников информируют получателей о статусе заказа, согласовывают доставку.

Банки, страховые компании, операторы связи и интернет-провайдеры используют виртуальных помощников, чтобы оповещать о новых услугах и тарифах, напоминать клиентам о задолженности и облегчить совершение платежей.

И это не все области, где используются виртуальные помощники, и не все решаемые ими задачи. Практически в любой компании виртуальный ассистент может выполнять функции технической поддержки: отвечать на частые вопросы клиентов, консультировать по несложным задачам, перенаправлять звонок оператору. Кроме того, его можно использовать для проведения опросов и анкетирования, например, чтобы узнать мнение клиентов о новом продукте. Виртуальные помощники также помогают во внутрикорпоративных коммуникациях: решают текущие вопросы сотрудников, участвуют во взаимодействии отделов.

Ассистент может быть оптимизирован для своего рабочего канала. Как правило, виртуальные помощники используются в голосовых каналах (на телефоне) и в интернет-каналах (на веб-сайтах). Первый тип относится в основном к голосовым помощникам, ко второму – чат-ботам. У них разные сильные и слабые стороны, которые необходимо учитывать в реальных проектах. Хороший результат для голосового канала может не подойти для интернет-чата и наоборот.

Указанные выше ассистенты (голосовые помощники и чат-боты) смогли быть реализованы в связи с решением главных задач, лежащих в основе этих ассистентов. Например, внутри голосового помощника заложены алгоритмы распознавания и синтеза речи. Впервые эти задачи были реализованы в коммерческих программах распознавания речи в 90-х годах [7][8]. Чат-боты же основаны на заранее определенных бизнес-правилах, определяющих ход диалога между пользователем и ботом. Как правило, чат-боты избегают вопросов, требующих свободного ответа, а ставят их в виде «тестового» варианта ответа (с конечным количеством вариантов). Эта технология построения интеллектуальных систем хорошо изучена и уже нашла широкое применение на практике [9].

Также, перспективной ветвью развития виртуальных ассистентов могут стать помощники, использующие цифровых аватаров [10] – виртуальных персонажей, имеющих внешний облик, схожий с человеком. Данный вид ассистентов находится сейчас на ранних этапах своего развития, т.к. часть задач, лежащих в их основе не имеет достаточной фундаментальной базы. Так, задача распознавания лиц уже имеет коммерческие решения на данный момент [11], но она является лишь частью необходимой базы для создания виртуальных ассистентов с цифровыми аватарами. Продолжаются работы по синтезу выражений лиц для виртуальных персонажей, т.к. в качественном ассистенте, мимика и эмоции аватаров должны быть близки к реальным для людей, что на данный момент далеко от желаемого результата.

## **1.2 Изучение и анализ литературы на тему представления эмоций в виде формальных моделей.**

Много лет учёные всего мира работали над задачей распознавания мимики человека. В наши дни уже есть успехи в этом направлении. Если раньше для разложения выражения лица на составляющие в FACS требовался специально обученный человек, то теперь нейросети могут делать это автоматически.

### **1.2.1 Общие характеристики дискретных и многомерных моделей.**

Есть два принципиально разных подхода к изучению организации эмоций. С точки зрения дискретной модели эмоциональная сфера состоит из определенного количества первичных, базовых или фундаментальных эмоций. Комбинации первичных эмоций - это эмоции второго порядка. Статус первичных эмоций обычно приписывается гневу, страху, грусти, счастью и т.д. Сторонники этого подхода сосредоточены на изучении внутренней организации индивидуальных эмоций, не уделяется внимание взаимосвязям между эмоциями.

В многомерной модели акцент делается на многомерном пространстве эмоций, образованном ограниченным числом шкал (осей), например, негативная - позитивная, сильная - слабая, активная - пассивная. Такие модели ориентированы на изучение сходства и различий между эмоциями.

Исследователи, разделяющие дискретную теорию эмоций, расходятся во мнениях относительно количества и принципов фундаментальных эмоций. Так, П. Экман и его сотрудники на основе изучения выражения лица выделяют 6 основных эмоций: гнев, страх, отвращение, удивление, печаль и радость. Р. Плутчик рассматривает 8 основных эмоций: одобрение, гнев, ожидание, отвращение, радость, страх, печаль, удивление. Классификация К. Изарда с 10-ю основными эмоциями, стала очень популярной. Он отмечает гнев, презрение, отвращение, дистресс (горе - страдание), страх, вину, интерес, радость, стыд, удивление. Есть и более простые классификации. Например, Я. Панксеп различает всего 5 основных эмоций: страх - тревога, ярость - гнев, паника - горе, ожидание - исследование, радость - игра.

Дж. Грей исследует три основные эмоции: ярость - ужас, тревогу и радость. По мнению Р. Мауэра, вообще следует говорить только о двух первичных эмоциях - удовольствии и боли.

Для дискретной модели вопрос о составе фундаментальных эмоций является критическим. Вопрос в том, сколько всего эмоций. Ответ зависит от того, какие показатели, какие реакции использовались при исследовании эмоций (вегетативные, биохимические, ЭМГ мышц лица и т.д.). По выражению лица сложно выделить более 10 паттернов активации мимических мышц, а по вегетативным параметрам можно описать еще меньше комплексов, связанных с эмоциями.

Многомерные модели эмоциональных состояний различаются между собой количеством измерений (осей) и их смысловой нагрузкой.

Адекватным методами для многомерной модели являются методы многомерного шкалирования (ММШ) или факторный анализ.

ММШ впервые был протестирован для изучения распознавания эмоций другого человека по внешним экспрессивным реакциям лица. Первоначальная процедура масштабирования применяет оценку внешних проявлений эмоций только по внутренним, заранее определенным шкалам. С развитием методов многомерной статистики в качестве исходных данных стали использоваться субъективные оценки различия или сходства лиц, выражающих различные эмоциональные переживания. Это позволяет определить и интерпретировать минимально допустимое количество шкал (факторов), которые ориентированы на предмет при вынесении суждений о различии, и тем самым приблизиться к механизму декодирования эмоций. В многомерном пространственном представлении каждое выражение представляет собой точку со своими собственными координатами или вектором, компоненты которого соответствуют вкладам главных осей (масштабов) пространства.

К первым работам в этом направлении относятся исследования Р. Вудворта (Woodworth R.S.) и Г. Шлосберга (Schlosberg N.). Р. Вудворте предложил первую классификацию выражений эмоций на лице с использованием шестичленной линейной шкалы: любовь, счастье, веселье; удивление; страх, страдание; гнев, решимость; отвращение; презрение.

Г. Шлоссберг, применив классификацию Вудворта к анализу фотографий людей с разной мимикой, обратил внимание на частое смешение самых далеких групп внешних выражений эмоций. Позже, в рамках теории активации Д. Линдсли (D. Lindsley), Г. Шлоссберг добавил третье измерение: сон - напряжение, создаваемое, таким образом, тремя измерениями эмоций, предложенными В. Вундтом в 1896 году: удовольствие - неудовольствие, расслабление - напряжение, спокойствие – волнение.

Однако в работе П. Абельсона и В. Сермата (Abelson R., Sermat V.), а также Р. Шепарда (Chapard R.) с использованием ММШ, когда испытуемые оценивали степень схожести между всеми возможными комбинациями пар из 13 фотографий по 9-балльной системе, были выдвинуты аргументы в пользу двухмерного решения. Была выделена шкала знаков эмоций (удовольствие - неудовольствие) и шкала активации (сон - бодрствование). Эти два фактора составляют примерно 75% от общего веса всех факторов. Аналогичные результаты были получены с использованием других наборов фотографий. В более поздних работах некоторые исследователи выделили новые измерения, такие как контроль - импульсивность, внимание - невнимательность, уверенность - неуверенность, которые были предложены в качестве третьей шкалы или вместо шкалы активации.

В экспериментах Дж. Рассела и М. Буллока (Russal J., Billok M.) с ранжированием мимики по фотографиям принимали участие дети 2, 3, 4 и 5 лет, а также студенты. Для испытуемых каждой возрастной группы было получено двумерное пространство, образованное двумя ортогональными осями: знаками и уровнем активации. Эмоции, изображенные на фотографиях, представлены точками, образующими фигуру, близкую по форме к кругу. Сходство эмоциональных пространств и взрослых детей в возрасте, когда еще нет дифференцированных представлений о различных эмоциях, свидетельствует о том, что структура опосредована восприятием эмоций, а не смысловыми связями, образующимися в процессе усвоения языка. Третья ось со значительно меньшим весом как у детей 4 лет, так и у взрослых была интерпретирована как шкала уверенности - смущения.

Особо следует выделить серию работ Дж. Рассела и его коллег. Применяя ММШ к субъективным оценкам сходства эмоций, он получил двумерную круговую модель эмоционального опыта (A Circumplex model of affect [12]). Эта двухмерная модель является PA-частью модели PAD.

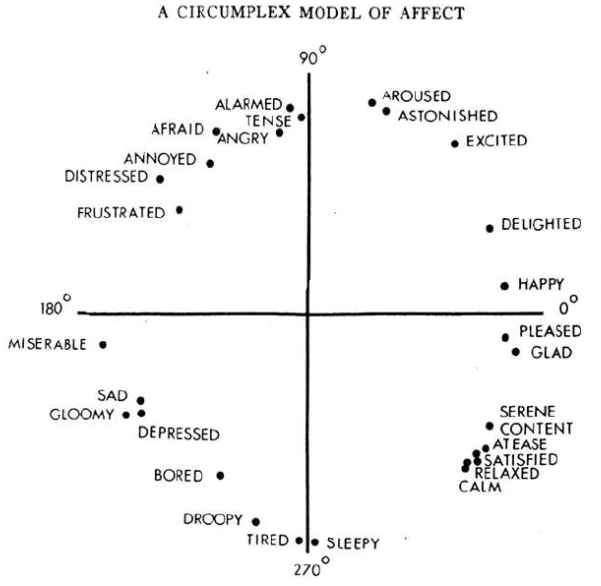


Рисунок 1.1 – A Circumplex model of affect

### **1.2.2 Модель эмоционального состояния PAD.**

В данном исследовании мы будем рассматривать многомерную модель PAD, в которой всего 3 измерения: P – Pleasure (удовольствие), A – Arousal (возбуждение), D – Dominance (доминирование) для всех представлений эмоций. Модель эмоционального состояния PAD - это психологическая модель, разработанная Альбертом Мехрабианом и Джеймсом А. Расселом (1974 г. и позже) для описания и измерения эмоциональных состояний. Первоначально использовавшаяся в теории психологии окружающей среды, основная идея заключалась в том, что физическая среда влияет на людей через их эмоциональное воздействие. Впечатление было использовано Питером Лэнгом и его коллегами, чтобы предложить физиологическую теорию эмоций. Его также использовал Джеймс А. Рассел для разработки теории эмоциональных эпизодов (относительно коротких эмоционально заряженных событий). PA-часть PAD была встроена в закрытую модель эмоционального опыта, и эти два измерения были названы «базовым аффектом». Часть D PAD была переосмыслена как часть процесса оценки эмоционального эпизода (холодная когнитивная оценка эмоциональной ситуации). Более подробная версия этого подхода называется психологической теорией построения эмоций. Модель PAD (Удовольствие, Возбуждение, Доминирование) использовалась для изучения невербального общения, такого как язык тела в психологии. Он также использует в потребительском маркетинге и создает анимированных персонажей, которые выражают эмоции в виртуальных мирах.

PAD использует трехмерные шкалы, которые теоретически могут иметь любое числовое значение. Размерная структура напоминает работы Вильгельма Вундта 19-го века, который также использовал трехмерную систему, а также работы Чарльза Э. Осгуда 20-го века.

Шкала удовольствия-неудовольствия измеряет, насколько приятно или неприятно человек чувствует что-то. Например, и гнев, и страх - неприятные эмоции, и оба они считаются недовольством. Однако радость - это приятное чувство.

Шкала возбуждения-бездействия измеряет, насколько человек чувствует себя возбужденным или гипнотическим. Дело не в интенсивности эмоций - горе и депрессия могут быть легким возбуждением и сильными чувствами. Хотя и гнев, и ярость являются неприятными эмоциями, ярость имеет более высокую интенсивность или более высокое состояние возбуждения. Однако скука, которая также является неприятным состоянием, имеет низкую ценность для возбуждения.

Шкала «Доминирование-покорность» представляет собой ощущение контроля и доминирования в сравнении с тем, что вы контролируете или подчиняетесь. Например, если страх и гнев - это неприятные эмоции, гнев - это доминирующая эмоция, а страх - это эмоция подчинения.

Более сокращенная версия модели использует всего 4 значения для каждого измерения, обеспечивая только 64 значения для возможных эмоций. Например, гнев - довольно неприятная, довольно возбужденная и умеренно доминирующая эмоция, а скука - слегка неприятная, совсем не возбужденная и в большинстве случаев не доминирующая эмоция.

### **1.2.3 Система кодирования лицевых движений FACS.**

Полом Экманом в XX веке была разработана система кодирования лицевых движений (FACS). Эта система позволяет классифицировать выражения лица человека на основе его мимики. С использованием FACS можно закодировать практически любое, анатомически возможное выражение лица, реконструируя его на вызвавшие его конкретные двигательные единицы (action units) и их временные сегменты.

FACS фактически не предусматривает любую биомеханическую информацию о степени активации мышц, но основные мышцы, участвующие в формировании выражений лица, были добавлены сюда для удобства читателя.

Двигательные единицы (action untis) представляют собой основные движения, совершаемые отдельными мышцами или группой мышц.

Двигательные дескрипторы (action descriptors) представляют собой движения, совершаемые группами мышц (например, выдвигание нижней челюсти вперед). Мышечная основа для этих движений не указана.

Для обозначения интенсивности задействования action units, в FACS к номеру AU добавляют латинские буквы с A по E, в зависимости от минимальной — максимальной интенсивности движения. Например, AU 1А является самым трудноразличимым движением AU 1, а AU 1Е является максимально возможной интенсивностью для определенного человека.

* A Слабо различимое
* B Незначительное
* C Заметное или ярко выраженное
* D Сильное или крайне заметное
* E Предельное

Двигательные единицы относительно воображаемой вертикальной оси лица могут быть:

* Двусторонние, симметричные (B — bilateral). Например, (B)1.
* Односторонние (U — unilateral). Например, U12, U14:
* левая (L — left). Например, L12.
* правая (R — right). Например, R14.

Примеры кодирования лицевых движений с помощью FACS можно изучить подробнее по ссылке [3].

С помощью двигательных единиц из FACS Полом Экманом и Уоллесом Фризеном была создана эмоциональная система кодирования лицевых движений (EmFACS), которая рассматривает только лицевые движения, связанные с эмоциями. Стоит отметить, что в научном мире нет единого мнения на тему однозначного соответствия между мимикой и эмоциями для всех людей [5]. Таким образом нельзя с полной уверенностью сказать, что, например, выражение лица любого человека, который испытывает страх выглядит именно так, а никак иначе. Однако, большинство людей в современном обществе выражает эмоции приблизительно одинаково, что даёт возможность использовать труды Пола Экмана в данной работе.

## **1.3 Постановка задач, исследовательские вопросы и гипотезы.**

### **1.3.1 Постановка задач.**

С учётом вышесказанного, встаёт вопрос о необходимости иметь возможность отображать пространства эмоций различных размерностей друг в друга. По сути, необходимо построить 6 функций:

* для отображения вектора модели эмоционального состояния PA (координатная плоскость pleasure, arousal из пространства PAD) в вектор 7-ми эмоций из модели Пола Экмана (Neutral – нейтральная, Happy – радость, Sad – печаль, Angry – гнев, Surprised – удивление, Scared – страх, Disgusted – отвращение) ( тип преобразования 2 -> 7)
* для отображения вектора 7-ми эмоций из модели Пола Экмана в вектор модели эмоционального состояния PA (тип преобразования 7 -> 2)
* для отображения вектора выражения лица человека, закодированного с помощью FACS, в вектор 7-ми эмоций из модели Пола Экмана (тип преобразования 42 -> 7)
* для отображения вектора 7-ми эмоций из модели Пола Экмана в вектор выражения лица человека, закодированного с помощью FACS (тип преобразования 7 -> 42)
* для отображения вектора модели эмоционального состояния PA в вектор выражения лица человека, закодированного с помощью FACS (тип преобразования 2 -> 42)
* для отображения вектора выражения лица человека, закодированного с помощью FACS, в вектор модели эмоционального состояния PA (тип преобразования 42 -> 2)

Ранее в НИРе 1-го семестра были реализованы нейросетевые модели автоэнкодеров, решающие задачу отображения из двухмерного пространства в семимерное (тип преобразования 2 -> 7).

В этом семестре планируется решить задачу обратного отображения из 7-ми мерного в 2-х мерное (тип преобразования 7 -> 2), а также представить модели отображения векторов выражений лиц людей, закодированных с помощью FACS, в вектора многомерной модели 7-ми эмоций Пола Экмана (тип преобразования 42 -> 7). Фактически, соединив 2 типа полученных моделей последовательно (тип преобразования 42 -> 7 и тип преобразования 7 -> 2) мы получим модель преобразования 42 -> 2.

В данной работе, помимо нейросетевых, будут использованы ещё и классические модели машинного обучения [13] (регрессионно-классификационные модели), такие как: регрессия (линейная, полиномиальная, Ridge/Lasso), Random Forest (случайный лес), KNN (K-Ближайших соседей). Классические модели обладают большей интерпретируемостью [14], в отличии от нейросетевых моделей, которые представляют из себя “модели чёрного ящика”. Подобное разнообразие из классических и нейросетевых моделей позволит сравнивать результаты между собой, а также оценивать их качество. 

Более подробно математическая постановка задач описана в подразделе 2.2.

### **1.3.2 Общая информация об исходных данных.**

В НИРе 1-го семестра, в качестве исходных данных для решения поставленной задачи представлены логи работы программы FaceReader. Ранее были записаны видеоматериалы, на которых испытуемые демонстрировали свои эмоции. Видеофайлы представляют из себя записи длительностью от нескольких секунд (10-20) до нескольких минут (1-2). Программа FaceReader проанализировала каждую видеозапись и выдала соответствующие результаты в виде логов.

В ходе работы предстоит проанализировать логи FaceReader’а и отобрать нужные данные для решения задачи отображения эмоций. Необходимо выделить показательную выборку, при этом постараться минимизировать влияние шумов.

Стоит отметить, что классическое измерение pleasure (удовольствие) из модели PAD в логах представлено как valence (валентность). При этом диапазон допустимых значений никак не меняется и остаётся прежним [-1; 1]. Но при этом координата Arousal находится в диапазоне [0; 1], что нетипично ввиду возможности наличия отрицательных значений Arousal (Рисунок 1.1). D – Dominance измерение опущено ввиду отсутствия необходимых данных.

Координаты семимерного вектора эмоций имеют диапазон [0; 1], хотя в сумме не дают 1. Т.е. возможны случаи, когда испытуемый одновременно демонстрирует несколько эмоций. Тогда сумма всех координат вектора превышает 1.

Все логи были объединены в 2 файла: data.csv, data\_clear.csv. Более подробное описание данных см. в подразделе 3.1.1 “Выбор датасета.”.

### **1.3.3 Гипотезы о данных и моделях.**

Задачи, поставленные в текущем исследовании напоминают задачу регрессионного анализа, суть которой заключается в установлении формы зависимости между переменными, оценка неизвестных значений (прогноз значений) зависимой переменной. Как следствие, существуют различные методы решения этой задачи как статистические, так и нейросетевые.

Более точно можно классифицировать задачу, как задачу повышения (снижения) размерности. По своему смыслу это преобразование данных, состоящее в уменьшении числа переменных путём получения главных переменных (в случае сжатия). Преобразование может быть разделено на отбор признаков и выделение признаков.

К статистическим методам решения задачи можно отнести метод главных компонент (МГК), линейный дискриминантный анализ (ЛДА) и обобщённый дискриминантный анализ (ОДА). К нейросетевым — применение автокодировщика (автоэнкодера).

В предыдущей работе Д.В.Тихомирова и А.В.Самсонович использовали метод главных компонент (PCA) для уменьшения размерностей данных из 42 признаков FACS в 7 компонент (аналог 'Neutral', 'Happy', 'Sad', 'Angry', 'Surprised', 'Scared', 'Disgusted'). Оказалось, что 7 главных компонент описывают около 80 % данных [15].

Чтобы обучение моделей (см. подраздел 1.3.1 “Постановка задачи.”) было более обоснованным, выдвинем аналогичную гипотезу о высоком % накопленной остаточной дисперсии PCA из 7 компонент ('Neutral', 'Happy', 'Sad', 'Angry', 'Surprised', 'Scared', 'Disgusted') в 2 (аналог 'Valence', 'Arousal'). Проверка гипотезы и её результаты описаны в подразделе 3.1.2 “Углубленный анализ данных.”.

Помимо регрессионных моделей, отдельный практический интерес в данной задаче представляют модели KNN (K-Ближайших соседей) и Random Forest (случайный лес). Более подробное обоснование выбора этих моделей описано в подразделах 2.3 “Сравнение классических моделей машинного обучения.” и 3.1.2 “Углубленный анализ данных.”.

# **Раздел 2. Теоретическая часть**

## **2.1 Изучение существующих решений поставленных задач.**

Из темы этого НИР следует, что эффективные модели машинного обучения, преобразующие вектора одного эмоционального пространства в другое, являются лишь частью от общего продукта, который требуется получить в итоговом результате.

Обученные модели отвечают за внутреннюю логику и решают основную задачу отображения пространств, но помимо моделей необходим удобный программный интерфейс для ввода/загрузки данных, для визуализации векторов в эмоциональных пространствах (VA, FACS, 7-ми мерное). Web-сервис, решающий эту задачу уже был разработан в НИРе 2-го семестра и доработан в ходе летней практики.

Конвертер, отображающий пространства эмоций различных размерностей может иметь множество практических применений. Его могут использовать различные виртуальные ассистенты или цифровые аватары (см. подраздел 1.1 “Технологии и подходы для реализации виртуальных ассистентов.”). Особенно это актуально в связи с активным развитием этого IT-направления и отсутствием единого стандарта для использования эмоциональных пространств. Иными словами, разные коммерческие решения могут использовать различные пространства эмоций различных размерностей. Для разрешения таких сложностей и может пригодиться разрабатываемый конвертер, в общем доступе.

Вдобавок к сказанному, анализ [16] рынка систем распознавания эмоций показал, что на данный момент нет бесплатных решений для отображения пространств эмоций различных размерностей. Есть лишь коммерческие решения и то, которые не выполняют преобразование напрямую (см. ниже 2.1.1 “FaceReader.” недостатки программы).

Таким образом, конечным результатом нескольких семестров НИРов будет готовый web-конвертер для решения задачи отображения пространств эмоций различных размерностей.

### **2.1.1 FaceReader.**

Много лет учёные всего мира работали над задачей распознавания мимики человека. В наши дни уже есть успехи в этом направлении. Если раньше для разложения выражения лица на составляющие в FACS требовался специально обученный человек, то теперь нейросети могут делать это автоматически.

Ярким примером является приложение FaceReader [11, 16]. Программа способна отличать различные эмоциональные выражения лица, такие как «счастливое», «грустное», «сердитое», «удивленное», «испуганное», «недовольное» и «нейтральное». Также FaceReader может отличать пол, возраст и этническую группу.

Для реализации используются технологии компьютерного зрения. С изображения лица человека считываются action units, которые преобразуются в точку на PA-модели.

Также, реализован метод Active Appearance Model, с помощью которого можно создавать искусственную модель лица с учетом контрольных точек и деталей поверхности, и сравнивать ее с образцами, заложенными в память.

Классификация происходит методами нейронных сетей с тренировочным набором в 2 000 фотографий.

Возможности программы:

* средний процент распознавания эмоций равен 89%. Для некоторых эмоций он выше, для некоторых ниже;
* наклон лица может быть любым в плоскости, его система обнаружит;
* программа работает с загружаемым видео в форматах с кодеками MPEG1, MPEG2, XviD, DivX4, DivX5, DivX6, DV-AVI и uncompressed AVI, причем определять эмоции можно пофреймно, либо полностью при просмотре всего видео. Также, FaceReader может работать со статичными изображениями, а также в реальном времени, если у пользователя подключена веб-камера;
* программа прекрасно визуализирована: всегда можно посмотреть гистограммы, диаграммы, процентаж выражаемых эмоций. А на таймлайне видны проявления микровыражений в определенный промежуток времени;
* FaceReader генерирует два текстовых файла, один – это лог проявления эмоций, а другой – статический, для сопряженной с данной программой уникальной системы управления визуальными данными The Observer XT, разработанной этой же компанией.

Недостатки программы:

FaceReader не натренирован для распознавания детей до 5-ти лет;

* Если человек в очках, то распознавание эмоций неточное, либо классификация не ведется;
* Люди с разным цветом кожи по-разному воспринимаются системой, программа не до конца адаптирована;
* Повернутое лицо не детектируется.

## **2.2 Математическая постановка задач.**

Первая задача этой работы - проведение углубленного анализа данных (см. подраздел 2.2.1 “Углубленный анализ данных.”) и проверка гипотез, выдвинутых в подразделе 1.3.3 “Гипотезы о данных и моделях.”.

Вторая задача, рассматриваемая в данном исследовании, является задачей многомерной (мульти-выходной) регрессии (см. подраздел 2.2.2 “Обучение моделей машинного обучения.”).

### **2.2.1 Углубленный анализ данных.**

В этом подразделе приведено описание теоретической составляющей углубленного анализа данных, включая метод главных компонент (PCA) и корреляционный анализ. Практические результаты оформлены в подразделе 3.1.2 “Углубленный анализ данных.”.

#### **2.2.2.1 Метод главных компонент (PCA).**

PCA - это метод для уменьшения размерности данных, который применяется для обнаружения структуры в данных путем нахождения нового набора переменных, называемых главными компонентами. Главная идея заключается в том, чтобы найти линейные комбинации исходных переменных (признаков), которые содержат максимальное количество информации.

Несколько основных шагов PCA включают в себя:

* **Стандартизация данных:** Важно стандартизировать данные перед применением PCA, чтобы убедиться, что все признаки имеют одинаковый масштаб.
* **Вычисление ковариационной матрицы:** PCA использует ковариационную матрицу для определения степени зависимости между признаками. Ковариационная матрица показывает, насколько два признака варьируются вместе.
* **Вычисление главных компонент:** Главные компоненты вычисляются как линейные комбинации исходных признаков таким образом, чтобы максимизировать объясненную дисперсию в данных. Главные компоненты упорядочиваются по убыванию объясненной дисперсии, что позволяет выбрать наиболее информативные компоненты.
* **Проекция данных на главные компоненты:** Затем исходные данные проецируются на пространство главных компонент, чтобы получить новый набор признаков с меньшей размерностью.

#### **2.2.2.2 Корреляционный анализ.**

Корреляционный анализ - это статистический метод, используемый для изучения связи между двумя или более переменными. Он позволяет определить, насколько тесно связаны переменные между собой и в каком направлении идет эта связь.

Корреляционная матрица - это таблица, в которой значениями являются коэффициенты корреляции между парами переменных. Она позволяет исследовать связи между всеми парами признаков в наборе данных. Значения коэффициента корреляции могут варьироваться от -1 до 1: значение ближе к 1 указывает на сильную положительную корреляцию (систематическую прямую пропорциональность), значение ближе к -1 - на сильную отрицательную корреляцию (систематическую обратную пропорциональность), а значение около 0 - на отсутствие корреляции.

### **2.2.2 Обучение моделей машинного обучения.**

Многомерная регрессия (мульти-выходная или мульти-таргет регрессия) — это тип задачи машинного обучения, где модель обучается предсказывать несколько непрерывных выходных переменных на основе одного или нескольких входных признаков.

Математически задачу можно сформулировать следующим образом:

* Дан набор входных данных *X*, где каждый элемент *xi* ∈ *X* представляет собой вектор входных признаков.
* Соответствующий набор выходных данных *Y*, где каждый элемент *yi* ∈ *Y* является вектором целевых переменных.
* Цель состоит в том, чтобы найти функцию *f*, которая наилучшим образом отображает входные данные на выходные: *f*(*xi*)≈*yi*

В контексте данной задачи размерность и вид входных и выходных данных зависит от типа преобразования (см. подраздел 1.3.1), а именно:

* тип преобразования 2 -> 7  
  Входные данные (*X*): двумерное пространство 'Valence' и 'Arousal'.

Выходные данные (*Y*): семь эмоциональных состояний ('Neutral', 'Happy', 'Sad', 'Angry', 'Surprised', 'Scared', 'Disgusted').

* тип преобразования 7 -> 2  
  Входные данные (*X*): семь эмоциональных состояний ('Neutral', 'Happy', 'Sad', 'Angry', 'Surprised', 'Scared', 'Disgusted').

Выходные данные (*Y*): двумерное пространство 'Valence' и 'Arousal'.

* тип преобразования 42 -> 7  
  Входные данные (*X*): вектор выражения лица человека, закодированного с помощью FACS (42 признака с именами, попадающих под шаблон *r'Action\_Unit\_\d{2}\_\_\_\w+'*).

Выходные данные (*Y*): семь эмоциональных состояний ('Neutral', 'Happy', 'Sad', 'Angry', 'Surprised', 'Scared', 'Disgusted').

* тип преобразования 7 -> 42  
  Входные данные (*X*): семь эмоциональных состояний ('Neutral', 'Happy', 'Sad', 'Angry', 'Surprised', 'Scared', 'Disgusted').

Выходные данные (*Y*): вектор выражения лица человека, закодированного с помощью FACS (42 признака с именами, попадающих под шаблон *r'Action\_Unit\_\d{2}\_\_\_\w+'*).

* тип преобразования 2 -> 42  
  Входные данные (*X*): двумерное пространство 'Valence' и 'Arousal'.

Выходные данные (*Y*): вектор выражения лица человека, закодированного с помощью FACS (42 признака с именами, попадающих под шаблон *r'Action\_Unit\_\d{2}\_\_\_\w+'*).

* тип преобразования 42 -> 2  
  Входные данные (*X*): вектор выражения лица человека, закодированного с помощью FACS (42 признака с именами, попадающих под шаблон *r'Action\_Unit\_\d{2}\_\_\_\w+'*).

Выходные данные (*Y*): двумерное пространство 'Valence' и 'Arousal'.

## **2.3 Сравнение моделей машинного обучения.**

В этом разделе мы проведем сравнительный анализ основных типов классических моделей машинного обучения, а также рассмотрим их особенности, указывая на типичные задачи, которые можно решить с помощью каждой из них.

* Регрессионные модели — предсказывают числовую характеристику объекта. Например, стоимость квартиры исходя из её площади или количество людей в аэропорту в зависимости от дня недели.
* Модели классификации — предсказывают категорию объекта по заранее заданным параметрам. Например, диагноз пациента на основе истории болезни, результатов анализов, симптомов.
* Регрессионно-классификационные (классические) модели — умеют решать оба типа задач.

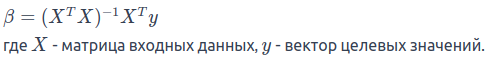
Применение классических моделей машинного обучения обусловлено тем, что использование автоэнкодера (см. НИР 1-го семестра) в задачах регрессии не является стандартным подходом. Автоэнкодеры обычно используются для уменьшения размерности или выявления скрытых признаков в данных, а не для прямого прогнозирования значений. По этой причине к задачам из подраздела 1.3.1 “Постановка задачи.” стоит добавить задачу типа преобразования 2 -> 7 для классических моделей.

В данном исследовании использовались, в основном, классические модели машинного обучения для решения задачи преобразования между пространствами эмоций различных размерностей. Ниже представлен краткий обзор каждой из используемых моделей и их обучение.

### **2.3.1 Линейная Регрессия.**

Линейная регрессия - это базовая модель машинного обучения, предназначенная для предсказания непрерывных выходных значений на основе линейной комбинации входных признаков.

Модель представляется уравнением: y=Xβ+ε, где  
y - выходные значения,   
X - матрица входных признаков,   
β - вектор весов,   
ε - ошибка модели. Обучение модели заключается в минимизации суммы квадратов ошибок между предсказанными и фактическими значениями.

Линейная регрессия стремится минимизировать сумму квадратов разностей между наблюдаемыми и предсказанными значениями. Аналитическое решение для оценки весов β может быть найдено с помощью метода наименьших квадратов:

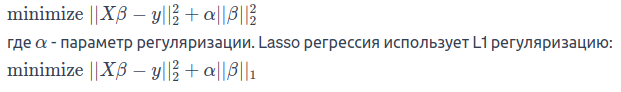
### **2.3.2 Полиномиальная Регрессия.**

Полиномиальная регрессия расширяет линейную регрессию, добавляя полиномиальные признаки (степени входных переменных). Это позволяет модели лучше аппроксимировать нелинейные зависимости.

Формула модели имеет вид: y=Xβ+ε, но X теперь включает полиномиальные признаки.

Аналитическое решение для полиномиальной регрессии аналогично линейной регрессии, но с учетом того, что входные данные трансформируются в полиномиальные признаки перед применением метода наименьших квадратов.

### **2.3.3 Ridge и Lasso Регрессии.**

Ridge (гребневая) и Lasso (лассо) регрессии - это варианты линейной регрессии, которые добавляют регуляризацию для предотвращения переобучения. Ridge регрессия минимизирует:  
  
Решения для Ridge и Lasso регрессий включают оптимизацию функций потерь с регуляризацией. Для Ridge:



аналитического решения из-за L1 регуляризации и решается с помощью численных методов оптимизации.

### **2.3.4 Случайный Лес (Random Forest).**

Случайный лес - это ансамблевая модель, состоящая из множества деревьев решений. Каждое дерево обучается на случайной подвыборке данных с использованием случайного подмножества признаков. Предсказание модели является усредненным результатом всех деревьев.

В случайном лесе каждое дерево решений стремится минимизировать функцию потерь, например, среднюю квадратичную ошибку (*MSE*) для задач регрессии. Особенностью случайного леса является использование ансамбля деревьев, где каждое дерево обучается на подвыборке данных с использованием случайного подмножества признаков. Финальное предсказание получается путем усреднения предсказаний отдельных деревьев.

Дерево решений - это модель машинного обучения, используемая как для классификации, так и для регрессии. Она строит модель принятия решений в виде древовидной структуры.

* Выбор Признака и Разбиения:   
  Для каждого узла дерева алгоритм выбирает наилучший признак для разбиения данных на подгруппы. Выбор основывается на метрике, такой как прирост информации (для классификации) или уменьшение среднеквадратичной ошибки (для регрессии).
* Метрики для Регрессионных Деревьев:  
  Среднеквадратичная ошибка (*MSE*) (см. подраздел 3.3 “Формирование метрик для оценки моделей.”)
* Уменьшение MSE:  
  Определяется как разность между MSE до и после разбиения. Узел с наибольшим уменьшением MSE выбирается как лучшее разбиение.
* Процесс Разбиения: Для каждого кандидата на разбиение (например, каждого значения признака) вычислить уменьшение MSE.   
  Выбрать точку разбиения, которая приводит к наибольшему уменьшению MSE.Разделить данные на две подгруппы на основе этого разбиения.
* Остановка Роста Дерева:   
  Рост дерева останавливается, когда достигнут один из критериев: достижение максимальной глубины, минимального числа образцов в узле или если улучшение MSE становится незначительным.

Деревья решений имеют несколько преимуществ, включая легкость интерпретации и способность обрабатывать как числовые, так и категориальные данные. Однако они могут быть склонны к переобучению, особенно в случае сложных данных. Для устранения этой проблемы часто используются ансамблевые методы, такие как случайный лес, который строит множество деревьев и усредняет их предсказания.

### **2.3.5 K-ближайших Соседей (KNN).**

Модель KNN работает на принципе нахождения "k" ближайших обучающих примеров в пространстве признаков и предсказывает значение на основе близости к этим соседям.

KNN не имеет явной тренировочной фазы. Вместо этого, он использует следующий псевдоалгоритм при предсказании:

* Для данного тестового примера *xtest*, вычислить расстояние до каждого обучающего примера.
* Выбрать *k* обучающих примеров, которые находятся ближе всего к *xtest*.
* Прогнозировать выходное значение как среднее (для регрессии) или как наиболее частое значение (для классификации) среди выбранных *k* ближайших соседей.

### **2.3.6 Многослойный Перцептрон (MLPRegressor).**

MLPRegressor - это тип нейронной сети, состоящий из нескольких слоев нейронов, которые могут моделировать сложные нелинейные отношения. Он обучается с использованием методов обратного распространения и градиентного спуска.

# **Раздел 3. Инженерная часть**

## **3.1 Выбор датасета, анализ данных, формирование обучающей и тестовой выборок.**

В данном разделе выбираются датасеты, данные в датасетах подвергаются тщательному анализу, формируются обучающие и тестовые выборки.

### **3.1.1 Выбор датасета.**

Для обучения и тестирования моделей взяты данные, полученные в ходе работы НИРа 1-го семестра. Это логи программы FaceReader, объединённые в 2 .csv файла: data.csv, data\_clear.csv.

* data.csv - все данные логов программы FaceReader (≈ 30 тыс. векторов), объединённые в 1 файл. Данные содержат все признаки эмоций в различных пространствах: (Valence; Arousal), (Neutral; Happy; Sad; Angry; Surprised; Scared; Disgusted), (42 признака с именами, попадающих под шаблон *r'Action\_Unit\_\d{2}\_\_\_\w+'*).   
  Также в файл data.csv были добавлены данные логов видео с участником Полом Экманом. Т.к. он является основателем отдельного эмоционального пространства, то данные, полученные из анализа его видео, считаются более точными.
* data\_clear.csv - в файле содержится подвыборка данных чистых эмоций (7-ми мерные вектора с значениями одной координаты близкой к 1, а остальных — к 0). Они были также отобраны ранее и сохранены в отдельный файл. Работа с чистыми данными происходит в *DataFram*'е *df\_clear*. Всего чистых эмоций 530 строк.

Более подробное описание данных см. в НИРе 1-го семестра в подразделе 3.1 “Выбор датасета, анализ данных, формирование обучающей и тестовой выборок.”

### **3.1.2 Углубленный анализ данных.**

#### **3.1.2.1 Диаграмма рассеяния с указанием доминирующей эмоции.**

Ниже представлена диаграмма рассеяния всех точек данных в координатах (Valence; Arousal) из data.csv (> 30к точек). Каждая точка имеет цвет доминирующей эмоции (Neutral; Happy; Sad; Angry; Surprised; Scared; Disgusted). Цветная визуализация данных сделана с использованием параметра hue (библиотека seaborn) - см. подраздел 4.1.

Как видно из диаграммы рассеяния рисунок 3.1, эмоции довольно хорошо разделимы и представляют из себя кластеры. Поэтому можно попробовать применить метрические методы.

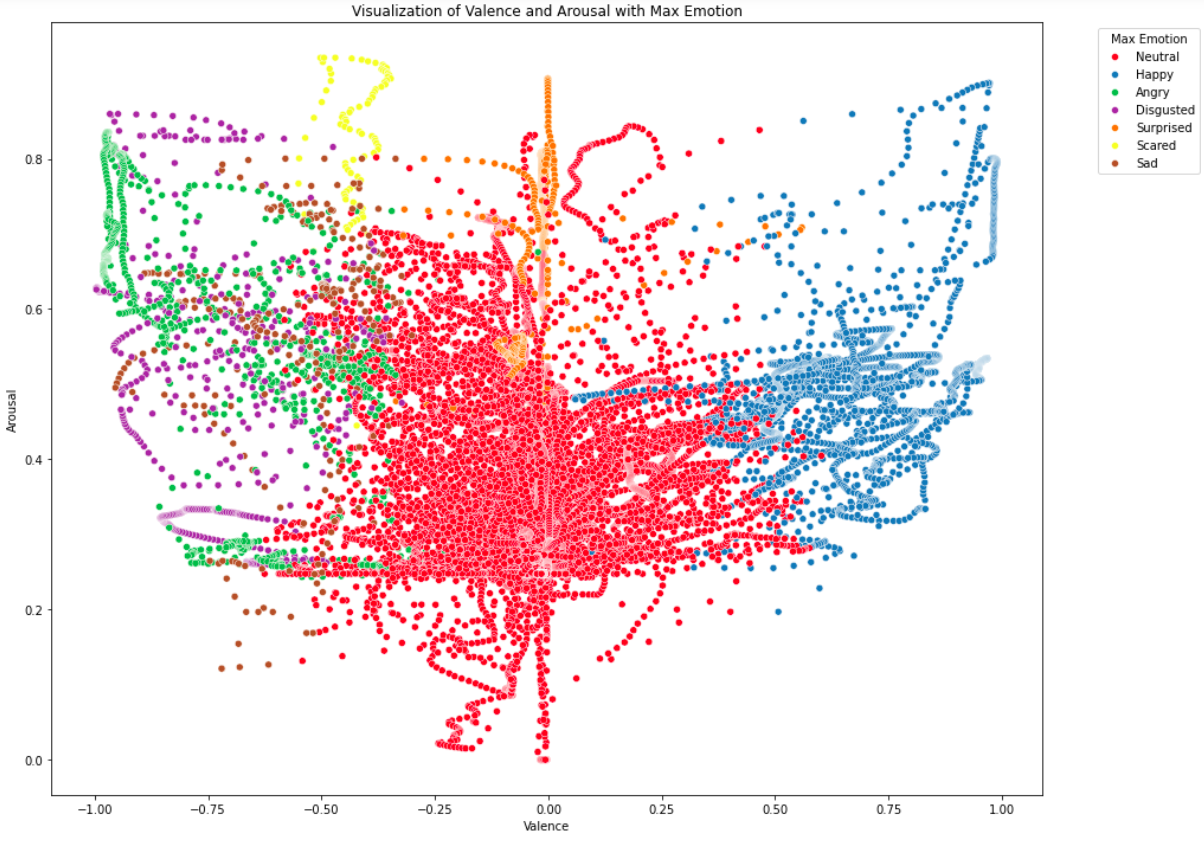


Рисунок 3.1 – Диаграмма рассеяния с указанием доминирующей эмоции.

Метрические методы по сути своей локальны: они исходят из допущения, что свойства объекта можно узнать, имея представление о его соседях. Один из самых известных метрических алгоритмов — метод k-ближайших соседей, или k-nearest neighbors (KNN) (см. подраздел 2.3 “Сравнение классических моделей машинного обучения.”). В ходе работы модель KNN применяется для всех типов преобразований: 2 -> 7, 42 -> 7, 7 -> 2. Подробнее обучение моделей и оценка их точностей описаны в подразделах 4.2 “Обучение моделей.”, 4.3 “Оценка точности работы моделей.”.

#### **3.1.2.2 Проверка гипотез с помощью МГК.**

В подразделе 1.3.3 “Гипотезы о данных и моделях.” мы выдвинули гипотезу о высоком % накопленной остаточной дисперсии PCA из 7 компонент ('Neutral', 'Happy', 'Sad', 'Angry', 'Surprised', 'Scared', 'Disgusted') в 2 (аналог 'Valence', 'Arousal' или 'Pleasure', 'Arousal' - см. “Введение.”). Подтвердив эту гипотезу, мы получим аналитическое обоснование возможности преобразований между пространствами эмоций 7 -> 2 с помощью МГК (PCA).

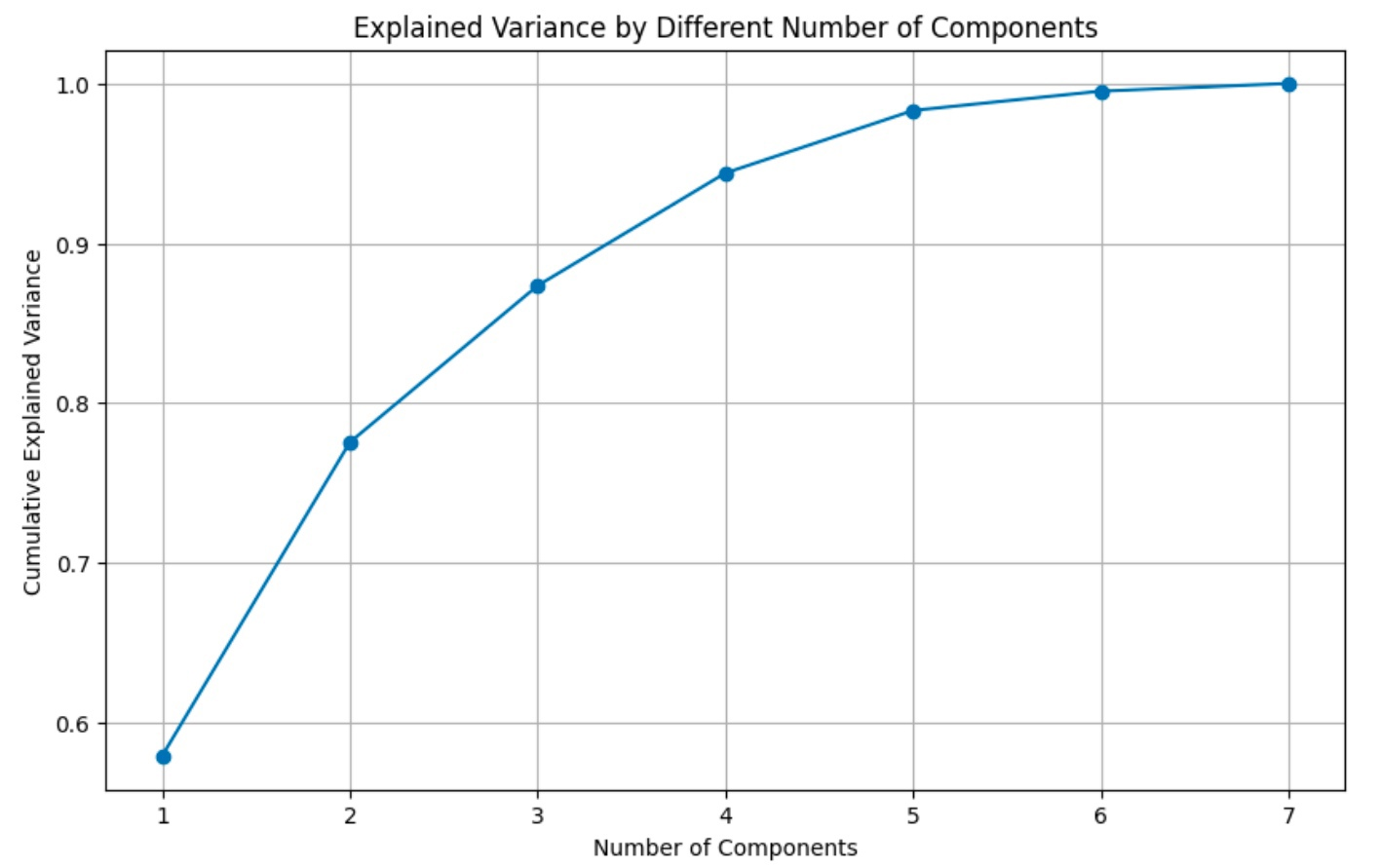
На графике рисунок 3.2 показано, как изменяется накопленная объясненная дисперсия в зависимости от количества компонент в методе главных компонент (PCA) для данных из подраздела 3.1.1. 

Рисунок 3.2 – Накопленная объясненная дисперсия в зависимости от количества компонент (PCA преобразование из 7 компонент в 2).

Из этого графика можно сделать вывод о том, сколько компонент необходимо, чтобы объяснить значительную часть вариативности данных. Так, 2 признака (аналог 'Pleasure', 'Arousal') имеют накопленную объясненную дисперсию == 0,7754936 из всех 7-ми эмоций. Это достаточный % вариативности данных, для преобразования 7 -> 2.

Также можно добавить 3-ий признак (аналог Dominance в пространстве PAD - см. “Введение.”), тогда накопленная объясненная дисперсия == 0,87347856. Такое высокое значение (> 85 %) объясняет почему существует и активно используется именно 3-х мерное пространство PAD, а не 2-х мерное пространство PA как у нас.

Аналогично, построим график рисунок 3.3 накопленной остаточной дисперсии в зависимости от количества компонент в методе главных компонент (PCA) для случая 42 компонент (action units в FACS) в 7 компонент ('Neutral', 'Happy', 'Sad', 'Angry', 'Surprised', 'Scared', 'Disgusted').

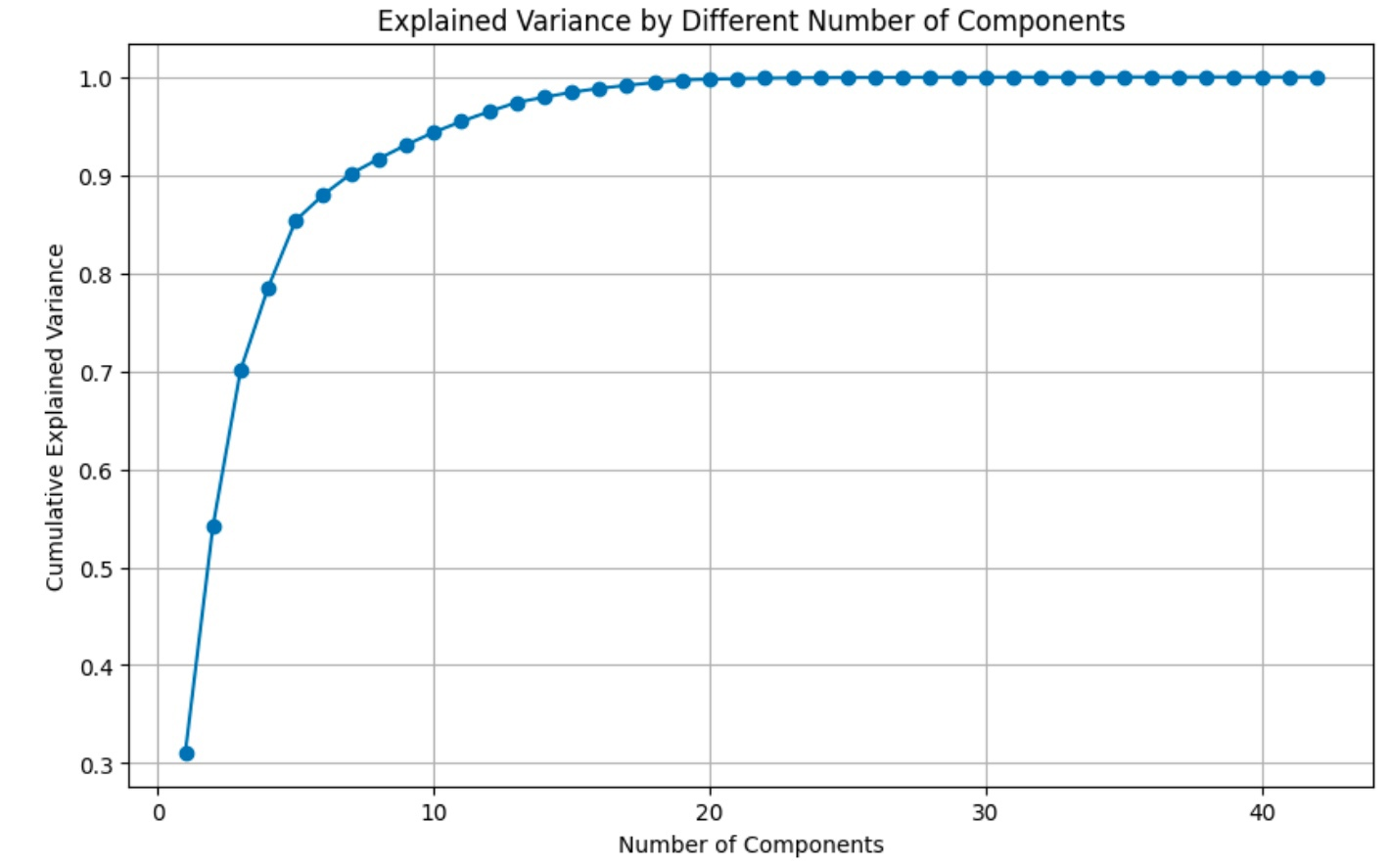


Рисунок 3.3 – Накопленная объясненная дисперсия в зависимости от количества компонент (PCA преобразование из 42 компонент в 7).

Из графика следует, что накопленная объясненная дисперсия > 0.9 при n\_components==7. Т.е. 7 признаков вполне достаточно, чтобы описать 42-мерное пространство.

Таким образом, мы получили аналитическое обоснование возможности преобразований между пространствами эмоций 7 -> 2 и 42 -> 7 с помощью МГК (PCA).

#### **3.1.2.3 Корреляционный анализ.**

Также в ходе углубленного анализа данных был проведён корреляционный анализ, который включал и расчёт корреляционной матрицы между 7 признаками ('Neutral', 'Happy', 'Sad', 'Angry', 'Surprised', 'Scared', 'Disgusted') и 2-мя ('Valence', 'Arousal').

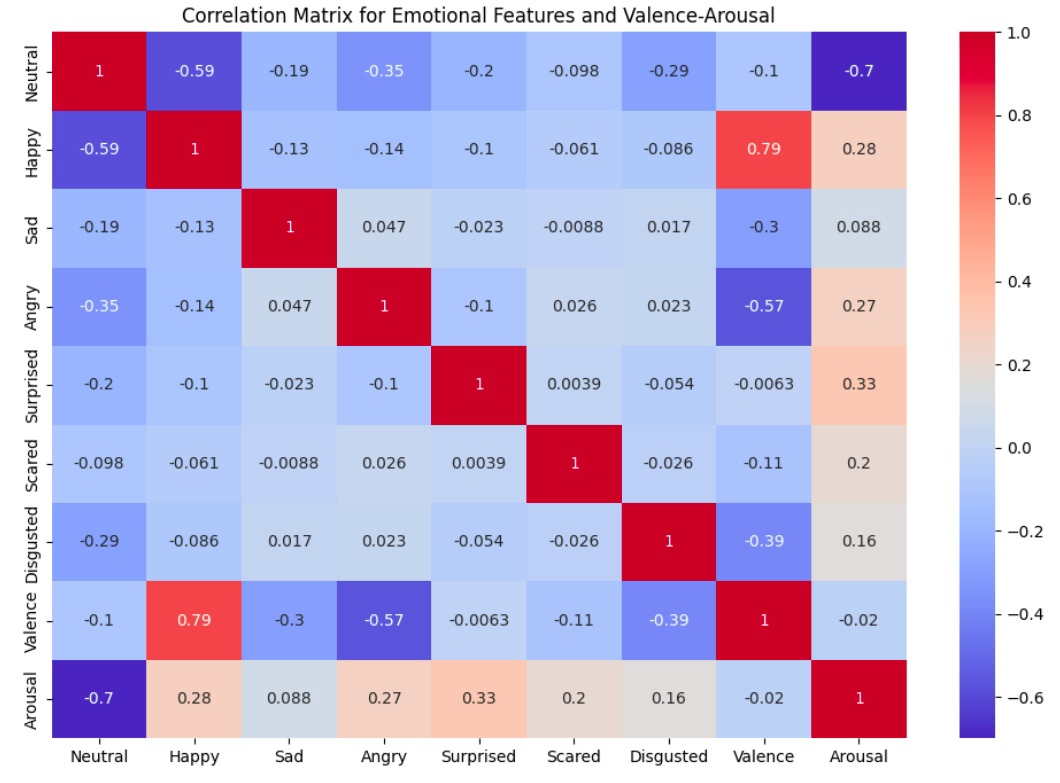


Рисунок 3.4 – матрица корреляции между 7 признаками ('Neutral', 'Happy', 'Sad', 'Angry', 'Surprised', 'Scared', 'Disgusted') и 2-мя ('Valence', 'Arousal')

Корреляционный анализ эмоциональных состояний и двухмерного пространства Valence-Arousal был выполнен для понимания взаимосвязей между различными переменными. В случае зависимостей, знак корреляции укажет на наличие систематической прямой пропорциональности (корреляция > 0) или систематической обратной пропорциональности (корреляция < 0).

Ниже выделены основные выводы, которые можно сделать из матрицы корреляции рисунок 3.4:

1. Корреляция признаков эмоциональных состояний ('Neutral', 'Happy', 'Sad', 'Angry', 'Surprised', 'Scared', 'Disgusted'):
   * Наблюдается значительная отрицательная корреляция между 'Neutral' и 'Happy' (-0.578), а также между 'Neutral' и 'Arousal' (-0.705), что указывает на то, что эти состояния имеют тенденцию быть противоположными. Так, с ростом значений признака 'Neutral', признак 'Arousal' (возбуждение) уменьшается.
   * Также обнаружена положительная корреляция между 'Happy' и 'Valence' (0.801), что логично, учитывая природу этих переменных.
2. Корреляция признаков Valence и Arousal:
   * 'Valence' имеет сильную положительную корреляцию с 'Happy' и отрицательную с 'Angry', 'Disgusted' и 'Sad'. Это согласуется с ожиданиями, так как 'Valence' отражает степень положительности или отрицательности эмоции.
   * 'Arousal', с другой стороны, имеет более слабые корреляции, но наибольшая из них наблюдается с 'Surprised', 'Happy' и 'Angry'. Корреляции положительны, что означает, что с ростом этих эмоциональных признаков ('Surprised', 'Happy' и 'Angry') будет увеличиваться и значение 'Arousal'.

Корреляционный анализ выявил ключевые взаимосвязи между различными эмоциональными состояниями и их проекциями на Valence и Arousal.

Эти результаты предоставляют важную основу для дальнейшего моделирования и анализа, позволяя лучше понять, как различные эмоциональные состояния взаимодействуют и проявляются в двумерном пространстве.

Убедиться в выводах корреляционного анализа позволяет цветная визуализация границ кластеров эмоциональных состояний (см. подраздел 3.1.2.4 “Визуализация границ кластеров эмоциональных состояний.”)).

#### **3.1.2.4 Визуализация границ кластеров эмоциональных состояний.**

Для большей наглядности взаимосвязей между различными переменными, можно визуализировать работу модели KNN (KNN показала довольно хорошие результаты при обучении) на плоскости двух входных признаков (Valence и Arousal) и отображать предсказания модели для каждого из семи целевых признаков (эмоциональных состояний). Однако, поскольку KNN - это метод, основанный на соседях, и он не даёт явной функции принятия решений (как, например, линейная регрессия), визуализация будет заключаться в отображении границ решений, созданных KNN на основе обучающих данных.

Для каждого из семи целевых признаков (например, Happy, Sad и т.д.) можно создать отдельную визуализацию.

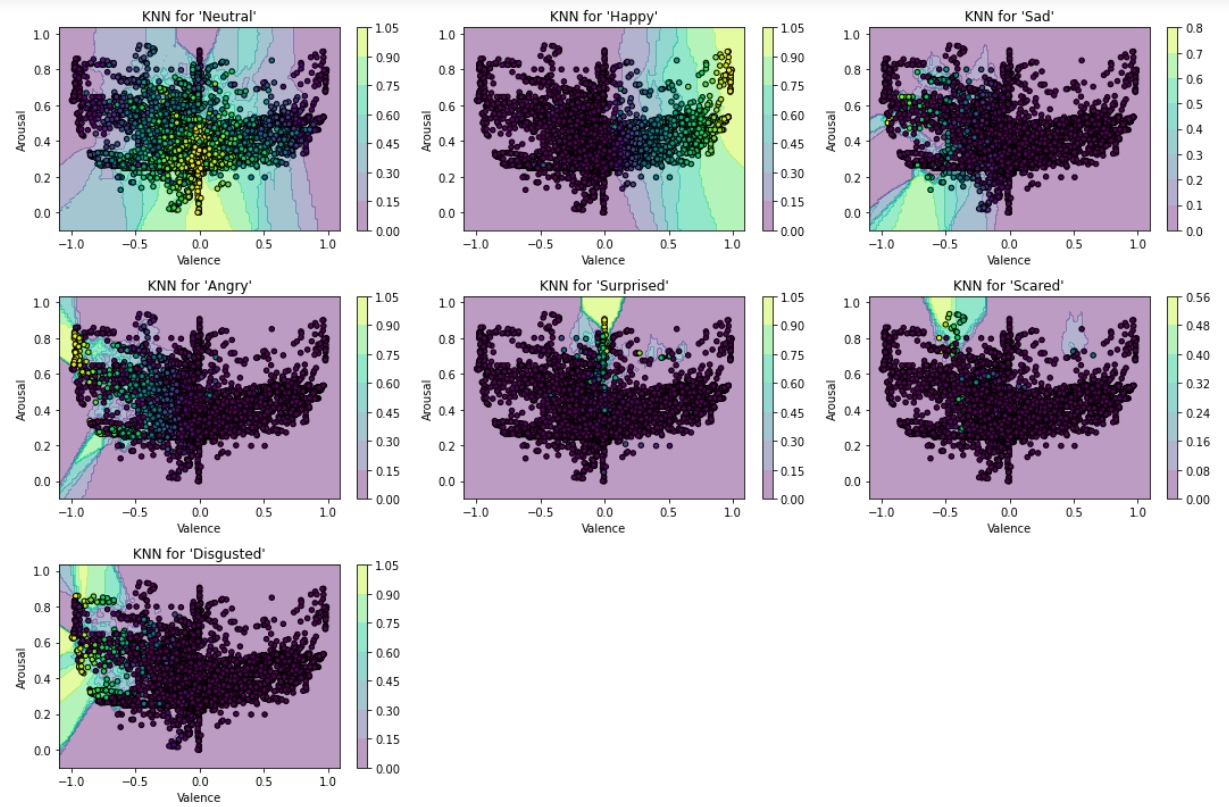


Рисунок 3.5 – Визуализация границ кластеров эмоциональных состояний для модели KNN

Выше на рисунке 3.5 представлена визуализация взаимного расположения кластеров эмоций 'Neutral', 'Happy', 'Sad', 'Angry', 'Surprised', 'Scared', 'Disgusted' в пространстве Valence-Arousal с использованием метода ближайших соседей (KNN). Каждый график представляет собой комбинацию контурных линий (plt.contourf), которые представляют различные уровни значений эмоций, и точек тестовых данных, с заданными значениями Valence и Arousal.

Цвета на графиках отображают значения эмоций для каждой точки данных в тестовом наборе. Визуализация позволяет увидеть, как эмоциональные состояния группируются в пространстве Valence-Arousal и как они различаются в зависимости от их положения в этом пространстве.

Цвета на графиках контуров представляют собой градацию значений высот (Z - выход модели, один из семи целевых признаков, для 7-ми обученных моделей), где плоскость графика аналогична географической карте высот. Каждый уровень высоты отображается разным цветом и представляет собой разные значения эмоций. Выделяются несколько (n) уровней (levels) высоты от Z.min, до Z.max. Все высоты, попадающие в диапазон i-го уровня, закрашиваются одним и тем же цветом, что позволяет визуально выделить области с схожими значениями эмоций.

Визуализация границ кластеров эмоциональных состояний соответствует выводам корреляционного анализа (см. подраздел 3.1.2.3 “Корреляционный анализ.”)). Также визуализация соответствует расположению чистых эмоций, продемонстрированных в НИРе 1-го семестра (см. рисунок 3.6).

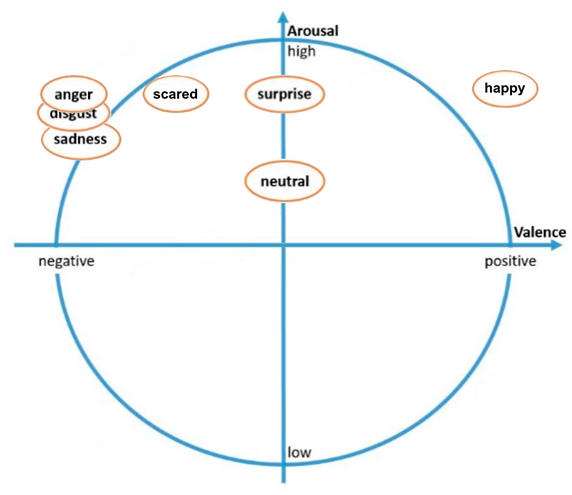


Рисунок 3.6 – Расположения семи чистых эмоций на плоскости PA (VA)

Реализация функции для визуализации моделей *plot\_model\_predictions* описана в подразделе 4.3.1 “Визуализация работы обученных моделей.”. Там же можно увидеть и визуализацию других обученных моделей типа преобразования 2 -> 7.

### **3.1.3 Формирование обучающей и тестовой выборок.**

Ранее в подразделе 1.3.1 “Постановка задач.” были указаны типы преобразований, реализуемые в этом семестре: 7 -> 2, 42 -> 7. Позднее в подразделе 2.3 “Сравнение моделей машинного обучения.” был добавлен тип преобразования 2 -> 7 для классических моделей.

Отберём данные по столбцам, формируя в списках (pa\_fields, seven\_fields, action\_unit\_features) названия столбцов, которые относятся к соответствующему N-мерному пространству:

pa\_fields = [ *# список содержит имена двух*

**'Valence'**, *# признаков: 'Valence' и 'Arousal'*

**'Arousal'** *# вектора модели PA (двухмерное*

] *# пространство)*

seven\_fields = [ *# список имён семи эмоциональных*

**'Neutral'**, *# состояний, модель Пола Экмана*

**'Happy'**, *# вектор 7-ми эмоций*

**'Sad'**,

**'Angry'**,

**'Surprised'**,

**'Scared'**,

**'Disgusted'**

]

action\_unit\_features = df.columns[df.columns.str.startswith(**'Action\_Unit\_'**)]

*# список имён 42-мерного пространства FACS (action units)*

*# вектора выражения лица человека, закодированного с помощью FACS*

Для создания удобного конвейера машинного обучения [17], будем поочередно комментировать все пары (*X*, *y*) кроме одной, соответствующей обучающему типу преобразования. Например, для типа преобразования 42 -> 7 оставим незакомментированной пару (*X*, *y*) со столбцами X = df[action\_unit\_features] и y = df[seven\_fields]:

# X = df[pa\_fields] # тип преобразования 2 -> 7

# y = df[emotion\_columns]

# X = df[emotion\_columns] # 7 -> 2

# y = df[pa\_fields]

X = df[action\_unit\_features] # 42 -> 7

y = df[seven\_fields]

# X = df[seven\_fields] # 7 -> 42

# y = df[action\_unit\_features]

# X = df[action\_unit\_features] # 42 -> 2

# y = df[pa\_fields]

# X = df[pa\_fields] # 2 -> 42

# y = df[action\_unit\_features]

Далее с помощью функции *train\_test\_split* из библиотеки scikit-learn разделим исходного набора данных на две части: обучающий набор данных и тестовый набор данных. Обычно обучающий набор данных используется для обучения модели, а тестовый набор данных - для оценки ее производительности.

* Параметр *test\_size=0.3* указывает, что 30% данных будут выделены для тестовой выборки, а оставшиеся 70% будут использованы для обучения модели.
* *random\_state=42* используется для установки начального состояния генератора случайных чисел, что обеспечивает воспроизводимость разделения данных.

Результатом выполнения кода являются четыре переменные:

*X\_train* - обучающая выборка признаков.

*X\_test* - тестовая выборка признаков.

*y\_train* - обучающая выборка целевых переменных.

*y\_test* - тестовая выборка целевых переменных.

Также используется кросс-валидация (с использованием GridSearchCV (см. подраздел 3.2 “Метод поиска оптимальных значений ключевых параметров моделей.”)). Кросс-валидация позволяет более точно оценить производительность модели путем разбиения данных на несколько фолдов (наборов обучающих и тестовых данных), обучения модели на каждом фолде и оценки ее производительности на оставшихся фолдах. Это помогает уменьшить вероятность переобучения модели и дает более обобщенную оценку ее производительности.

Таким образом, *train\_test\_split* используется для создания обучающего и тестового наборов данных для первоначального разделения данных, а затем кросс-валидация используется для более точной оценки производительности модели.

## **3.2 Метод поиска оптимальных значений ключевых параметров моделей.**

В НИРе 1-го семестра также обучалось большое количество однотипных моделей (более 300 автоэнкодеров см. НИР 1-го семестра подраздел 4.3 “Обучение нейросетей на обучающей и тестирование на тестовой выборках.”)), отличающихся друг от друга только кол-вом слоёв и нейронов в слое. Тогда для решения подзадачи выбора оптимальных значений слоёв и нейронов в слое (гиперпараметров моделей) писался собственный класс *NeuralNetwork* (см. НИР 1-го семестра подраздел 4.2 “Программная реализация модулей для обучения нейросетевых моделей.”).

В данной работе возникают схожие подзадачи выбора оптимальных значений гиперпараметров моделей. Однако на этот раз используется решётчатый поиск [18] (Grid Search) - эффективный метод, который предназначен для корректировки параметров в моделях контролируемого машинного обучения с целью получения наилучшей обобщающей способности.

Релизация метода решётчатого поиска уже есть в библиотеке scikit-learn. *GridSearchCV* - это класс из модуля *sklearn.model\_selection*, который предоставляет функциональность для выполнения поиска по сетке (Grid Search Cross-Validation) по заданному пространству параметров модели. Он используется для автоматического подбора оптимальных гиперпараметров модели, которые обеспечивают наилучшую производительность на основе выбранной метрики оценки.

Процесс работы *GridSearchCV* включает следующие шаги:

1. Определение модели, которую нужно настроить, и ее параметров, которые нужно оптимизировать.
2. Задание сетки параметров, которые должны быть протестированы. Это может быть сделано с помощью словаря, в котором ключи представляют имена параметров, а значения - списки значений этих параметров для тестирования.
3. Выбор метрики оценки качества модели (например, среднеквадратичная ошибка, средняя абсолютная ошибка и т. д.), которая будет использоваться для оценки каждой комбинации параметров.
4. Запуск *GridSearchCV*, который автоматически перебирает все комбинации параметров из заданной сетки и оценивает модель с помощью кросс-валидации.
5. Выбор наилучших параметров, которые обеспечивают наилучшую производительность модели.

В результате шагов 1-5 *GridSearchCV* возвращает наилучшую модель с оптимальными параметрами, которые были найдены в ходе поиска по сетке.

Примеры использования *GridSearchCV* представлены в подразделах 4.1 “Выбор инструментов для анализа данных и обучения моделей.” и 4.2 “Обучение моделей.”.

## **3.3 Формирование метрик для оценки моделей.**

При обучении большого числа моделей, встаёт вопрос о необходимости выбора метрики (метрик) для сравнения и оценки точностей моделей.

В отличии от НИРа 1-го семестра, где использовались введённые нами метрики *'mean', 'norm', ‘stat’, 'clear'*, здесь использованы общеизвестные метрики в среде, связанной с анализом данных и машинным обучением: *MSE*, *MAE*, *R2*

Метрики, использованные для оценки моделей:

* Средняя квадратичная ошибка (*MSE*): , где — истинное значение, — предсказанное значение, — количество наблюдений. *MSE* измеряет средний квадрат разницы между истинными и предсказанными значениями, являясь индикатором точности модели.
* Средняя абсолютная ошибка (*MAE*):  
    
  *MAE* аналогичен *MSE*, но измеряет среднюю абсолютную разницу, что делает её менее чувствительной к выбросам по сравнению с *MSE*.
* Коэффициент детерминации (*R2*):  
   , где — среднее значение . *R2* измеряет долю вариативности в зависимой переменной, которая объясняется независимыми переменными модели. Он показывает, насколько хорошо данные подходят для модели, относительно простого усреднения.

Эти метрики были выбраны для оценки точности и адекватности предсказаний моделей, а также их способности объяснять вариабельность данных.

# 

# **Раздел 4. Технологическая и практическая часть**

## **4.1 В**[**ыбор инструментов для анализа данных и**](#_1mrcu09) **обучения моделей.**

В данной работе есть прогресс по сравнению с НИРом 1-го семестра, где большая часть кода была написана вручную. После трех семестров обучения найдено множество готовых инструментов, которые значительно упрощают процесс обработки данных, обучения моделей и визуализации результатов. Теперь большая часть работы связана с анализом данных и применением этих инструментов для создания и оценки моделей. Все коды, реализованные в текущей работе, написаны на языке Python.

### **4.1.1 И**[**нструмент**](#_1mrcu09)**ы для анализа и обработки данных.**

Для анализа данных и работы с ними используется библиотека pandas в Python. Вот несколько ключевых инструментов из pandas для анализа данных:

* *pd.DataFrame: DataFrame* - это двумерная метка, состоящая из столбцов и строк. Он используется для представления данных в табличной форме.
* *pd.read\_csv(‘<filename>.csv’)*: Этот метод позволяет читать данные из CSV-файла и создавать *DataFrame*, который является основной структурой данных в pandas.
* *pd.DataFrame.corr():* Этот метод используется для вычисления корреляционной матрицы между признаками *DataFrame*. Корреляционная матрица показывает степень линейной зависимости между парами признаков. В подразделе 3.1.2.3 “Корреляционный анализ.” с помощью метода *pd.DataFrame.corr()* была вычислена корреляционная матрица для *DataFrame* с 9-ю столбцами: ['Neutral', 'Happy', 'Sad', 'Angry', 'Surprised', 'Scared', 'Disgusted', 'Valence', 'Arousal'].

Для предобработки данных используется библиотека numpy - это инструмент для работы с многомерными массивами и матрицами, а также для выполнения математических операций над ними. Функции в numpy очень полезны для многих задач анализа данных и обработки числовых массивов в Python. Вот краткое описание некоторых функций из этой библиотеки:

* *numpy.cumsum*: Функция *numpy.cumsum* возвращает накопленную сумму элементов вдоль указанной оси массива.

Например: *arr = np.array([1, 2, 3, 4])*

*print(np.cumsum(arr))* # Вывод: [1, 3, 6,10]

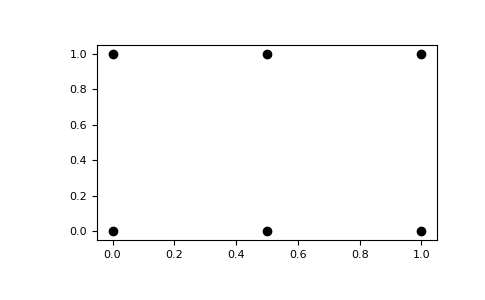
* *numpy.mean*: Эта функция вычисляет среднее арифметическое значение элементов в массиве. Например: *arr = np.array([1, 2, 3, 4, 5])*   
  *print(np.mean(arr))* # Вывод: 3.0
* *numpy.round*: Функция round округляет элементы массива до заданного числа десятичных знаков. Например: *arr = np.array([1.234, 2.567, 3.891])*  
  *print(np.round(arr, decimals=2))* # Вывод: [1.23, 2.57, 3.89]
* *numpy.linspace*: Функция linspace создает массив, содержащий равномерно распределенные числа в заданном интервале.   
  Например: *arr = np.linspace(0, 10, num=5)*  
  *print(arr)* # Вывод: [ 0., 2.5, 5., 7.5, 10. ]
* *numpy.meshgrid*: Эта функция используется для создания координатных сеток на основе одномерных массивов координат. Например:   
  *nx, ny = (3, 2)* # хотим равномерно расставить 3 точки по горизонтали и 2 по вертикали (3\*2 = 6 точек) - см. *#* Рисунок 4.1

*x = np.linspace(0, 1, nx) # np.array([0. , 0.5, 1. ])*

*y = np.linspace(0, 1, ny) # np.array([0., 1.])*

*xv, yv = np.meshgrid(x, y) # xv == np.array([[0. , 0.5, 1. ], [0. , 0.5, 1. ]])*

*# yv == np.array([[0., 0., 0.], [1., 1., 1.]])*

  
Рисунок 4.1 – Визуализация границ кластеров эмоциональных состояний для модели KNN

* *numpy.c\_*: Эта функция используется для объединения двух одномерных массивов вдоль второй оси, образуя двумерный массив (аналог *zip* в чистом Python). Например: *arr1 = np.array([1, 2, 3])*

*arr2 = np.array([4, 5, 6])*

*print(np.c\_[arr1, arr2])*  # Вывод: [[1, 4], [2, 5], [3, 6]]

В подразделе 1.3.3 “Гипотезы о данных и моделях.” мы выдвинули гипотезу, связанную с МГК (PCA - Principal Component Analysis). Теория по анализу данных представлена в подразделе 2.2.1 “Углубленный анализ данных.”. Результаты практической проверки описаны в подразделе 3.1.2.2 “Проверка гипотез с помощью МГК.”.

Scikit-learn предоставляет гибкий и удобный класс для применения метода *PCA*. Важные параметры включают количество компонент (n\_components) и способ вычисления (svd\_solver). Он также позволяет выполнить анализ данных с учителем при наличии целевой переменной.

Так, чтобы снизить размерность был создан объект класса *sklearn.decomposition.PCA(n\_components=2)*, с указанием двух главных компонент (аналог 'Pleasure', 'Arousal'). Когда мы вызываем метод *sklearn.decomposition.PCA.fit(df)*, мы производим процесс обучения модели PCA на наших данных df (*DataFram*e с 7 эмоциональных состояний). Этот процесс заключается в вычислении главных компонент, которые представляют собой новые направления в пространстве признаков. Цель состоит в том, чтобы эти новые компоненты максимально сохраняли дисперсию данных, чтобы мы могли сохранить как можно больше информации при снижении размерности. Атрибут *sklearn.decomposition.PCA.explained\_variance\_ratio\_* - возвращает относительную объясненную дисперсию каждой компоненты. Это показывает, какую долю общей дисперсии в данных объясняет каждая из главных компонент. С помощью массива *sklearn.decomposition.PCA.explained\_variance\_ratio\_* построены графики накопленной объясненной дисперсии в подразделе 3.1.2.2 “Проверка гипотез с помощью МГК.”

### **4.1.2 И**[**нструмент**](#_1mrcu09)**ы для обучения моделей.**

Одна из задач, рассматриваемых в данном исследовании, является задачей многомерной (мульти-выходной) регрессии (см. подраздел 2.2.2 “Обучение моделей машинного обучения.”). Используемые классические модели машинного обучения, для решения задачи преобразования между пространствами эмоций различных размерностей указаны в подразделе 2.3 “Сравнение моделей машинного обучения.”. Выбранный набор моделей уже разработан в Python библиотеке scikit-learn. В scikit-learn реализованы:

* линейная регрессия (класс *sklearn.linear\_model.LinearRegression*)
* полиномиальная регрессия (см. ниже)
* Ridge и Lasso Регрессии (классы *sklearn.linear\_model.Ridge* и *sklearn.linear\_model.Lasso*)
* Случайный Лес (Random Forest) (класс *sklearn.ensemble.RandomForestRegressor*)
* K-ближайших Соседей (KNN) (класс *sklearn.ensemble.KNeighborsRegressor*)
* Многослойный перцептрон (MLPRegressor) (класс *sklearn.neural\_network.MLPRegressor*)
* *sklearn.dummy.DummyRegressor* - простая модель, которая используется в качестве базовой линии для сравнения с другими моделями.

Благодаря этому разнообразию моделей, scikit-learn является мощным инструментом для исследования и применения различных методов машинного обучения.

Отдельного внимания заслуживают конвейеры. Конвейеры (*Pipelines*) в scikit-learn представляют собой удобный способ объединения последовательности этапов обработки данных и моделирования в единый объект. Они позволяют автоматизировать процесс предобработки данных и построения модели, обеспечивая более чистый и структурированный код.

Так, для реализации полиномиальной регрессии в конвейере используются два основных компонента:

1. *sklearn.preprocessing.PolynomialFeatures* - этот шаг преобразует исходные признаки в полиномиальные комбинации этих признаков. Например, если исходные признаки   
    и , то *PolynomialFeatures* создаст новые признаки вида , , и так далее в соответствии с заданной степенью полинома.
2. класс *sklearn.linear\_model.LinearRegression* - этот шаг строит линейную регрессию на преобразованных полиномиальных признаках. Поскольку *PolynomialFeatures* уже преобразовал исходные признаки в степени, *LinearRegression* работает с этими полиномиальными признаками как с линейными. Таким образом, комбинация *PolynomialFeatures* и *LinearRegression* позволяет реализовать полиномиальную регрессию.

Общий код для конвейера при этом имеет вид: *make\_pipeline(PolynomialFeatures(degree=n), LinearRegression())*

Учитывая многомерность регрессии, необходимо применить инструмент *sklearn.multioutput.MultiOutputRegressor*. Этот класс позволяет обернуть регрессор, который обучается на каждом выходе независимо, и делает его совместимым с API для одного выхода, таким как обучение, предсказание и оценка модели. Таким образом, *MultiOutputRegressor* применяется в случаях, когда нужно предсказывать несколько выходных переменных одновременно с использованием одного набора входных данных. Общий код для модели многомерной регрессии имеет вид: *MultiOutputRegressor(<класс модели>())*, например *MultiOutputRegressor(LinearRegression())* - реализует модель многомерной линейной регрессии.

Функция *sklearn.model\_selection.train\_test\_split* из библиотеки scikit-learn используется для разделения исходного набора данных на две части: обучающий набор данных и тестовый набор данных. Эта функция была описана в подразделе 3.1.3 “Формирование обучающей и тестовой выборок.”.

Класс *sklearn.model\_selection.GridSearchCV* - реализует метод решётчатого поиска, который был кратко описан в подразделе 3.2 “Метод поиска оптимальных значений ключевых параметров моделей.”. Решётчатый поиск автоматизирует процесс и применение нескольких моделей одновременно, предоставляя удобный инструментарий для настройки гиперпараметров моделей. Применение класса *GridSearchCV* рассмотрено в подразделе 4.2 “Обучение моделей.”.

После обучения модели, важно оценить её производительность, чтобы понять, насколько хорошо она работает на тестовых данных. Для этого используются различные метрики оценки качества модели. В частности, три широко используемые метрики (см. подраздел 3.3 “Формирование метрик для оценки моделей.”) для оценки качества регрессионных моделей, которые также были реализованы в библиотеке scikit-learn:

* *MSE* - *sklearn.metrics.mean\_squared\_error(y\_true, y\_pred)*. Здесь и далее предполагается, что у нас есть фактические значения *y\_true* и предсказанные значения *y\_pred*.
* *MAE* - *sklearn.metrics.mean\_absolute\_error(y\_true, y\_pred)*
* *R2* - *sklearn.metrics.r2\_score(y\_true, y\_pred)*

### **4.1.3 И**[**нструмент**](#_1mrcu09)**ы для визуализации.**

Библиотека Matplotlib предоставляет мощные инструменты для визуализации данных в Python. Функции модуля *matplotlib.pyplot* (*plt*) помогают создавать различные типы графиков и визуализировать данные для лучшего понимания информации. Вот краткое описание функций и методов, используемых для создания графиков:

* *plt.figure(figsize=(6, 4))*: Создает новую фигуру для графика с заданным размером. Например, (6, 4) задает размер фигуры шириной 6 дюймов и высотой 4 дюйма.
* *plt.title(title)*: Устанавливает заголовок для графика.
* *plt.plot(range(1, 8), my\_array, marker='o')*: Строит линию, принимая на вход данные по оси *x* (в данном случае *range(1, 8)*) и данные по оси *y* (в данном случае *my\_array*). Параметр *marker='o'* указывает на использование круглых маркеров на точках графика.
* *plt.subplot(3, 3, i + 1)*: Создает подграфик в сетке графиков размером 3x3 и выбирает текущий подграфик по индексу *i + 1*.
* *plt.xlabel('Ось x')* и *plt.ylabel('Ось y')*: Устанавливают подписи осей x и y соответственно.
* *plt.grid(True)*: Включает отображение сетки на графике.
* *plt.legend(title='Max Emotion', bbox\_to\_anchor=(1.05, 1), loc=2)*: Добавляет легенду к графику. Параметр *title* устанавливает заголовок легенды, *bbox\_to\_anchor=(1.05, 1)* устанавливает положение легенды вне графика, а l*oc=2* устанавливает положение легенды в верхнем левом углу.
* *plt.show()*: Отображает график.
* *plt.bar(range(coefficients.shape[0]), coefficients)*: Строит столбчатую диаграмму.
* *plt.xticks(rotation=90)*: Поворачивает подписи оси x на 90 градусов.
* *plt.contourf(xx, yy, Z, alpha=0.4)*: Создает контурный график (карту высот для кластеров эмоций в подразделе 3.1.2.4 “Визуализация границ кластеров эмоциональных состояний.”) с заданным массивом значений Z.
* *plt.scatter(X\_test['Valence'], X\_test['Arousal'], c=y\_test[target\_feature], s=20)*: Строит диаграмму рассеяния с данными из X\_test по осям Valence и Arousal. Цвет точек определяется значениями из y\_test[target\_feature], а размер точек - 20.
* *plt.tight\_layout()*: Автоматически регулирует расположение подграфиков для более четкого отображения.

Библиотека seaborn (sns) предоставляет высокоуровневые интерфейсы для создания красивых и информативных графиков:

* *sns.barplot(x=['PC1', 'PC2'], y=explained\_variance)*: Этот код создает столбчатую диаграмму, где по оси *x* отображаются названия главных компонент (PC1, PC2), а по оси *y* отображается объясненная дисперсия.
* *sns.heatmap(correlation\_matrix, annot=True, cmap='coolwarm')*: Здесь используется тепловая карта, которая визуализирует матрицу корреляции *correlation\_matrix*. Параметр *annot=True* указывает на отображение значений корреляции на карте, а *cmap='coolwarm'* задает цветовую палитру. Пример: см. подраздел 3.1.2.3 “Корреляционный анализ.”.
* *sns.scatterplot(x=features['Valence'], y=features['Arousal'], hue=max\_emotion)*: Этот код создает диаграмму рассеяния, где по оси X отображаются значения признака Valence, по оси Y - значения признака Arousal. Цвет точек определяется параметром *hue*, значением переменной *max\_emotion* (числом *float* от 0 до 1). Пример: см. подраздел 3.1.2.1 “Диаграмма рассеяния с указанием доминирующей эмоции.”.

### **4.1.4 И**[**нструмент**](#_1mrcu09)**ы для сохранения моделей.**

В этом семестре для сохранения обученных моделей в файлы используется модуль *sklearn.externals.joblib* из библиотеки scikit-learn. Модуль *joblib* подробно описан в подразделе 4.4 “Сохранение лучших моделей в формате файла обученных моделей.”

## **4.2 Обучение моделей.**

В данном разделе производится обучение моделей из подраздела 2.3 “Сравнение моделей машинного обучения.” с помощью программных инструментов из подраздела 4.1 “Выбор инструментов для анализа данных и обучения моделей.”.

Напомню, что ранее:

* был произведён углубленный анализ данных (подраздел 3.1.2 “Углубленный анализ данных.”),
* были подготовлены данные: выбран нужный тип преобразования (2 -> 7, 7 -> 2, 42 -> 7 и т.д.), созданы обучающий и тестовый наборы данных (подраздел 3.1.3 “Формирование обучающей и тестовой выборок.”),
* были выбраны модели для сравнения: использование различных типов классических моделей (подраздел 2.3 “Сравнение моделей машинного обучения.”)
* был описан способ (Grid Search) для настройки гиперпараметров модели (подраздел 3.2 “Метод поиска оптимальных значений ключевых параметров моделей.”).
* были выбраны метрики оценки: *MSE*, *MAE*, *R2* (подраздел 3.3 “Формирование метрик для оценки моделей.”)
* были выбраны инструменты для обучения моделей (подраздел 4.1 “Выбор инструментов для анализа данных и обучения моделей.”)

Теперь, когда всё готово для обучения моделей, можно приступить к этому этапу. Разобьём обучение моделей на шаги:

1. Определим список моделей (grid\_models) с соответствующими гиперпараметрами. Каждый элемент списка представляет кортеж-тройку вида: (<класс модели>, <гиперпараметры для обучения>, <короткое название модели>).  
   grid\_models = [

(MultiOutputRegressor(LinearRegression()), {}, **'Linear Regression'**),

(make\_pipeline(PolynomialFeatures(), LinearRegression()),

{**'polynomialfeatures\_\_degree'**: [2, 3, 4, 6, 8, 10]}, **'Polynomial Regression'**),

(MultiOutputRegressor(Ridge()), {**'estimator\_\_alpha'**: [0.1, 1, 10]}, **'Ridge'**),

(MultiOutputRegressor(Lasso()), {**'estimator\_\_alpha'**: [0.1, 1, 10]}, **'Lasso'**),

(MultiOutputRegressor(RandomForestRegressor(random\_state=42)),

{**'estimator\_\_n\_estimators'**: [10, 50, 100], **'estimator\_\_max\_depth'**: [**None**, 3, 5, 10, 20]}, **'Random Forest'**),

(MultiOutputRegressor(KNeighborsRegressor()),

{**'estimator\_\_n\_neighbors'**: [3, 5, 7, 20], **'estimator\_\_weights'**: [**'uniform'**, **'distance'**]}, **'KNN'**),

(MultiOutputRegressor(MLPRegressor(random\_state=42)),

{**'estimator\_\_hidden\_layer\_sizes'**: [

(3, 6), (4, 5, 6), (3, 5, 6), (3, 4, 5, 6),

(6, 3), (6, 5, 4), (6, 5, 3),(6, 5, 4, 3),

(50,50), (100,), (15, 5, 15),(50, 30, 10, 30, 50),

(100, 50, 30, 10, 5, 10, 30, 50, 100),

(50, 10, 5, 10, 50), (10, 5, 5, 2, 5, 5, 10)

], **'estimator\_\_alpha'**: [0.001, 0.01]}, **'MLPRegressor'**),

(MultiOutputRegressor(DummyRegressor(strategy=**"mean"**)), {}, **'Dummy Mean'**)

]

1. Создадим pd.DataFrame (grid\_results) со столбцами: **'Model'** (<короткое название модели>), **'MSE'**, **'MAE'**, **'R2'** (метрики лучшей модели) и **'Best Params'** (<лучшие гиперпараметры для обучения>) для сохранения результатов сравнения обученных моделей:  
   grid\_results = pd.DataFrame(columns=[**'Model'**, **'MSE'**, **'MAE'**, **'R2'**, **'Best Params'**])
2. Определим функцию perform\_grid\_search для поиска лучших гиперпараметров с помощью *GridSearchCV*, а также возврата результатов для записи в grid\_results:  
   **def** perform\_grid\_search(model, params, name):

grid\_search = GridSearchCV(model, params, cv=3, scoring=**'neg\_mean\_squared\_error'**, n\_jobs=-1)

grid\_search.fit(X\_train, y\_train)

best\_model = grid\_search.best\_estimator\_

best\_params = grid\_search.best\_params\_

y\_pred = best\_model.predict(X\_test)

mse = mean\_squared\_error(y\_test, y\_pred)

mae = mean\_absolute\_error(y\_test, y\_pred)

r2 = r2\_score(y\_test, y\_pred)

**del** grid\_search

**del** best\_model

**return** {**'Model'**: name, **'MSE'**: mse, **'MAE'**: mae, **'R2'**: r2, **'Best Params'**: best\_params}

1. Используем perform\_grid\_search для каждой модели отдельно, чтобы найти лучшие гиперпараметры:  
   **for** model, params, name **in** grid\_models:

result = perform\_grid\_search(model, params, name)

grid\_results = grid\_results.append(result, ignore\_index=**True**)

Стоит обратить внимание, что для некоторых моделей, таких как полиномиальная регрессия (**'Polynomial Regression'**), может потребоваться обширный поиск по сетке из-за большого количества комбинаций гиперпараметров. Также, обучение **'MLPRegressor'** может занять значительное время.

В итоге получены 3 таблицы с результатами обучения моделей. Описание таблиц представлено в подразделе 4.3.2 “Выводы по таблицам результатов обучения моделей.”.

## **4.3 Оценка точности работы моделей.**

### **4.3.1 Визуализация работы обученных моделей.**

В подразделе 3.1.2.4 “Визуализация границ кластеров эмоциональных состояний.” был продемонстрирован контурный график границ кластеров доминирующих эмоциональных состояний (Neutral; Happy; Sad; Angry; Surprised; Scared; Disgusted) для модели KNN. В этом разделе описана реализация функции, которая строит данные графики для любых обученных моделей (типа преобразования 2 -> 7).

Стоит заметить, что модель считается тем точнее, чем точнее расположены тестовые точки (по цвету) из диаграммы рассеяния на "географической карте высот".

Код функции *plot\_model\_predictions* создает визуализации для каждого эмоционального состояния с использованием предсказаний модели. Для каждого эмоционального признака будет сгенерирован отдельный график, показывающий как модель предсказывает этот признак в зависимости от значений Valence и Arousal. Можно использовать этот код для визуализации предсказаний любой модели (KNN, Random Forest и т.д.).

Модель model передается в функцию *plot\_model\_predictions*, а также данные для тестирования (X\_test), фактические значения целевых признаков (y\_test), целевые признаки (target\_features) - эмоциональные состояния (Neutral; Happy; Sad; Angry; Surprised; Scared; Disgusted).

**Алгоритм** функции *plot\_model\_predictions* состоит из шагов:

1. Создается сетка точек 100x100 в двумерном пространстве VA от x\_min до x\_max по оси x (**'Valence'**) и от y\_min до y\_max по оси y (**'Arousal'**).
2. Модель из аргумента model используется для предсказания значений на этой сетке (10000 предсказаний).
3. В цикле для каждой i-ой эмоции (Neutral; Happy; Sad; Angry; Surprised; Scared; Disgusted) строится отдельный график:
   1. Визуализация с использованием функции plt.contourf, нанесение "географической карты высот", где высота (Z) - i-ое значение предсказаний модели.
   2. Нанесение диаграммы рассеяния из тестовых точек данных

Код функции *plot\_model\_predictions* представлен ниже:

**def** plot\_model\_predictions(model, X\_test, y\_test, target\_features):

*# Определение размера графика*

plt.figure(figsize=(15, 15))

*# Создание сетки для визуализации*

x\_min, x\_max = -1.1, 1.1

y\_min, y\_max = -0.1, 1.1

xx, yy = np.meshgrid(np.linspace(x\_min, x\_max, 100), np.linspace(y\_min, y\_max, 100))

*# Предсказание для каждой точки на сетке*

Z\_ = model.predict(np.c\_[xx.ravel(), yy.ravel()])

*# Цикл по каждому эмоциональному состоянию для создания визуализации*

**for** i, target\_feature **in** enumerate(target\_features):

Z = Z\_[:, i].reshape(xx.shape)

*# Визуализация*

plt.subplot(3, 3, i + 1)

*# нанесение "географической карты высот"*

plt.contourf(xx, yy, Z, alpha=0.4)

*# нанесение тестовых точек в виде диаграммы рассеяния*

plt.scatter(X\_test[:, 0], X\_test[:, 1], c=y\_test[target\_feature], s=20, edgecolor=**'k'**)

plt.title(**f"{**model.\_\_class\_\_.\_\_name\_\_**} for '{**target\_feature**}'"**)

plt.xlabel(**'Valence'**)

plt.ylabel(**'Arousal'**)

plt.tight\_layout()

plt.show()

На рисунке 4.2 представлена визуализация для модели MLPRegressor, а на рисунке 4.3 для Random Forest.

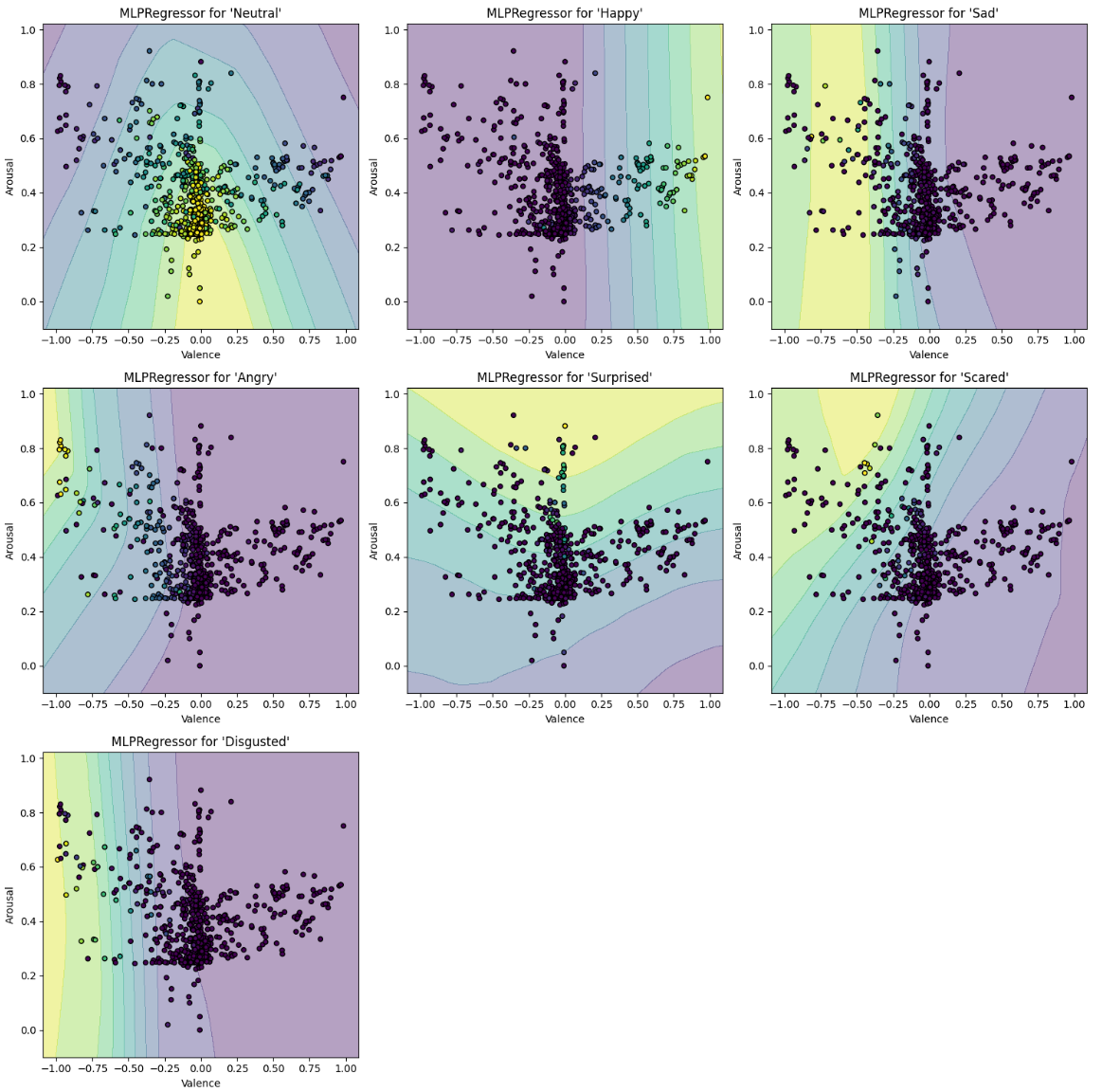


Рисунок 4.2 – Визуализация границ кластеров эмоциональных состояний для модели MLPRegressor

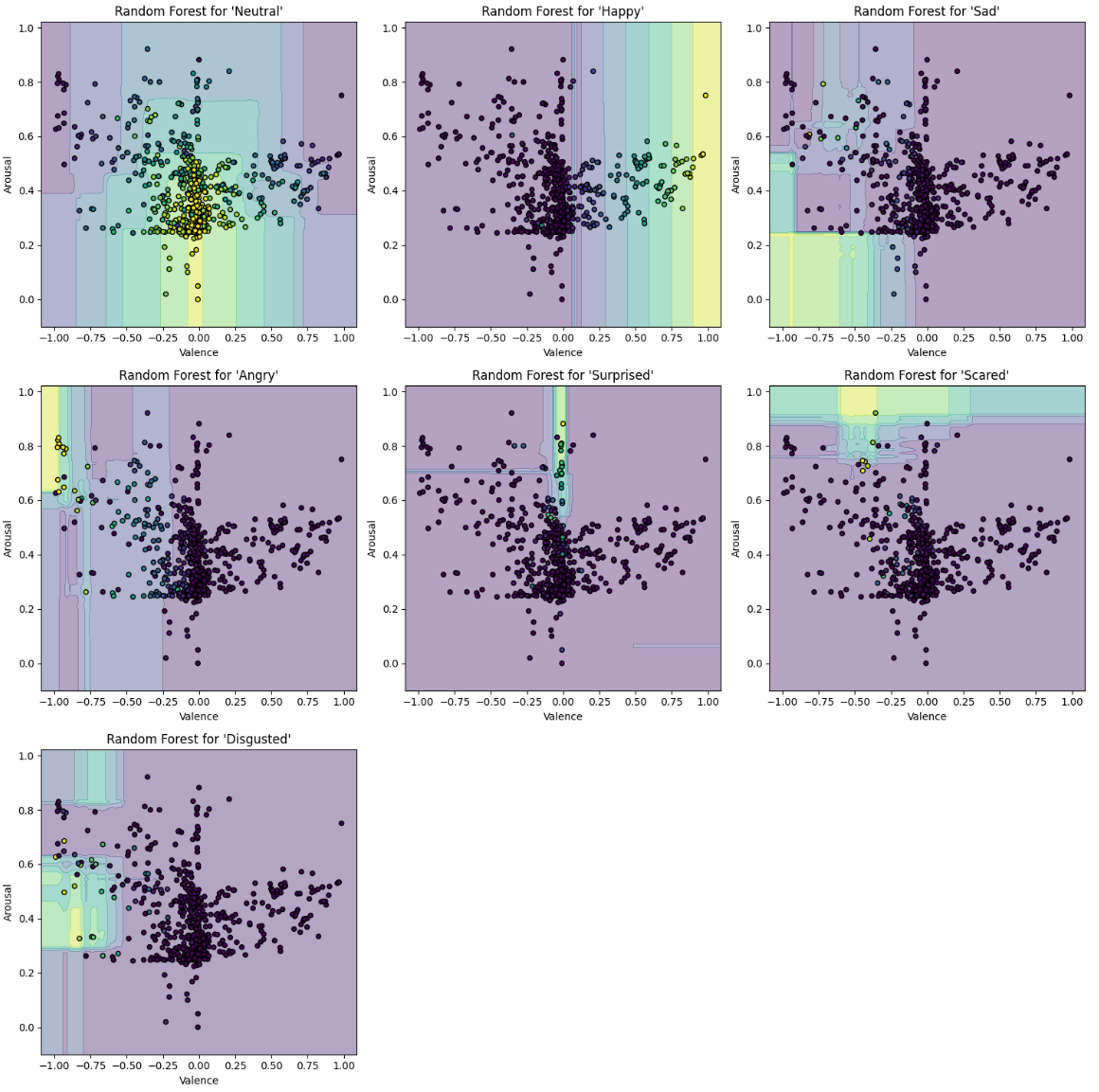


Рисунок 4.3 – Визуализация границ кластеров эмоциональных состояний для модели Random Forest

### **4.3.2 Выводы по таблицам результатов обучения моделей.**

Таблица 4.1 Таблица результатов обучения моделей (тип преобразования: 2 -> 7)

| **index** | **Model** | **MSE** | **MAE** | **R2** | **Best Params** |
| --- | --- | --- | --- | --- | --- |
| 0 | Linear Regression | 0.014231 | 0.066857 | 0.289288 | {} |
| 1 | Polynomial Regression | 0.005765 | 0.033479 | 0.604517 | {'polynomialfeatures\_\_degree': 10} |
| 2 | Ridge | 0.014231 | 0.066858 | 0.289289 | {'estimator\_\_alpha': 0.1} |
| 3 | Lasso | 0.026426 | 0.087998 | -0.000279 | {'estimator\_\_alpha': 0.1} |
| 4 | Random Forest | 0.004373 | 0.025275 | 0.696562 | {'estimator\_\_max\_depth': 10, 'estimator\_\_n\_estimators': 100} |
| 5 | KNN | 0.004509 | 0.024511 | 0.678621 | {'estimator\_\_n\_neighbors': 20, 'estimator\_\_weights': 'distance'} |
| 6 | MLPRegressor | 0.006113 | 0.032308 | 0.570686 | {'estimator\_\_alpha': 0.001, 'estimator\_\_hidden\_layer\_sizes': (50, 30, 10, 30, 50)} |
| 7 | Dummy Mean | 0.026426 | 0.087998 | -0.000279 | {} |

На основе предоставленной таблицы 4.1 с результатами моделей регрессии для восстановления семи эмоциональных состояний из двух признаков Valence и Arousal (тип преобразования: 2 -> 7), можно сделать следующие выводы:

* Random Forest (Случайный лес): Эта модель показала наилучшие результаты по большинству метрик (*MSE*, *R2*). С наименьшей средней квадратичной ошибкой (*MSE*) 0.004373, а также с самым высоким коэффициентом детерминации (*R2*) 0.696562, она значительно превосходит другие модели. Средняя абсолютная ошибка (*MAE*) 0.025275. Это указывает на высокие точность и надежность модели в предсказании эмоциональных состояний. Модель случайного леса обучена с параметрами: 'n\_estimators': 100 (ансамбль из 100 деревьев), 'max\_depth': 10 (максимальная глубина дерева).
* KNN (k-ближайших соседей): Эта модель занимает второе место по эффективности с *R2* 0.678621, что указывает на хорошую, но не наилучшую способность модели объяснить вариабельность данных. Важно отметить, что для KNN средняя абсолютная ошибка (*MAE*) 0.024511, что ниже чем для Random Forest, т.е. модель KNN самая лучшая по метрике *MAE* (для типа преобразования: 2 -> 7). Однако метрика *MAE* менее чувствительна к выбросам, чем *MSE*, поэтому можно сделать вывод, что модель KNN хуже чем Random Forest в некоторых эмоциональных состояниях, которые можно классифицировать как выбросы. Модель KNN обучена с параметрами 'n\_neighbors': 20 (k=20) и 'weights': 'distance' (веса соседей обратно пропорциональны расстоянию).
* Полиномиальная регрессия и MLPRegressor (автоэнкодер): эти модели показали средние результаты, заняв третье и четвёртое места соответственно. Метрики полиномиальная регрессии: *MSE* 0.005765, *R2* 0.604517 и *MAE* 0.033479, метрики MLPRegressor’а: *MSE* 0.006113, *R2* 0.570686 и *MAE* 0.032308. Это означает, что они менее точны, чем модели Random Forest и KNN, в предсказании эмоциональных состояний.
* Линейная регрессия и Ridge: Обе эти модели показали плохие результаты с похожими значениями *MSE* и *MAE* и слабыми значениями *R2* (0.289288 и 0.289289 соответственно). Их не стоит использовать для предсказаний.
* Lasso: Эта модель показала самые низкие результаты по всем метрикам. Особенно стоит отметить отрицательное значение *R2* (-0.000279), что указывает на то, что модель работает хуже, чем простое среднее значение.

По аналогии можно сделать выводы для моделей типов преобразования: 7 -> 2 (таблица 4.2) и 42 -> 7 (таблица 4.3).

Таблица 4.2 Таблица результатов обучения моделей (тип преобразования: 7 -> 2)

| **index** | **Model** | **MSE** | **MAE** | **R2** | **Best Params** |
| --- | --- | --- | --- | --- | --- |
| 0 | Linear Regression | 0.005053 | 0.042107 | 0.779691 | {} |
| 1 | Polynomial Regression | 0.003173 | 0.031580 | 0.847046 | {'polynomialfeatures\_\_degree': 3} |
| 2 | Ridge | 0.005053 | 0.042118 | 0.779693 | {'estimator\_\_alpha': 0.1} |
| 3 | Lasso | 0.057844 | 0.148904 | -0.000424 | {'estimator\_\_alpha': 0.1} |
| 4 | Random Forest | 0.000419 | 0.006948 | 0.980667 | {'estimator\_\_max\_depth': None, 'estimator\_\_n\_estimators': 100} |
| 5 | KNN | 0.001584 | 0.013407 | 0.924047 | {'estimator\_\_n\_neighbors': 3, 'estimator\_\_weights': 'distance'} |
| 6 | MLPRegressor | 0.003561 | 0.031623 | 0.827583 | {'estimator\_\_alpha': 0.001, 'estimator\_\_hidden\_layer\_sizes': (100, 50, 30, 10, 5, 10, 30, 50, 100)} |
| 7 | Dummy Mean | 0.057844 | 0.148904 | -0.000424 | {} |

На основе представленной таблицы 4.2 результатов обучения моделей для преобразования из семи эмоциональных состояний в два признака (тип преобразования: 7 -> 2), можно сделать следующие выводы:

* Random Forest (Случайный лес): Снова модель случайного леса демонстрирует наилучшие результаты, теперь уже по всем метрикам, с наименьшими значениями *MSE* (0.000419) и *MAE* (0.006948), а также с крайне высоким коэффициентом детерминации (*R2*) равным 0.980667. Это указывает на очень высокие точность и надежность модели в предсказании двух признаков из семи эмоциональных состояний. Модель случайного леса обучена с параметрами: 'n\_estimators': 100 (ансамбль из 100 деревьев), 'max\_depth': None (максимальная глубина дерева не указана).
* KNN (k-ближайших соседей): Эта модель занимает второе место по всем метрикам с очень высоким значением *R2* (0.924047) и низкими значениями *MSE* (0.001584) и *MAE* (0.013407). Это свидетельствует об отличной способности модели объяснить вариабельность данных. Модель KNN обучена с параметрами 'n\_neighbors': 3 (k=3) и 'weights': 'distance' (веса соседей обратно пропорциональны расстоянию).
* Полиномиальная регрессия и MLPRegressor: Модель полиномиальной регрессии и MLPRegressor показывают также хорошие результаты с высокими *R2* (0.847046 и 0.827583) и низкими значениями *MSE* (0.003173 и 0.003561) и *MAE* (0.031580 и 0.031623).
* Линейная регрессия и Ridge: Обе эти модели показывают средние результаты с неплохими значениями *R2* (0.779691 и 0.779693) и относительно высокими значениями *MSE* (0.005053 и 0.005053) и *MAE* (0.042107 и 0.042118). Это указывает на то, что они менее точны в предсказании двух признаков из семи эмоциональных состояний. Однако, их также можно использовать для предсказаний значений Valence, Arousal.
* Lasso: Эта модель также показывает низкие результаты с отрицательным значением *R2* и высокими значениями *MSE* и *MAE*.

Таблица 4.3 Таблица результатов обучения моделей (тип преобразования: 42 -> 7)

| **index** | **Model** | **MSE** | **MAE** | **R2** | **Best Params** |
| --- | --- | --- | --- | --- | --- |
| 0 | Linear Regression | 0.007074 | 0.041779 | 0.673049 | {} |
| 1 | Polynomial Regression | 36.044336 | 0.108675 | -1444.560669 | {'polynomialfeatures\_\_degree': 2} |
| 2 | Ridge | 0.007196 | 0.042227 | 0.665992 | {'estimator\_\_alpha': 0.1} |
| 3 | Lasso | 0.026426 | 0.087998 | -0.000279 | {'estimator\_\_alpha': 0.1} |
| 4 | Random Forest | 0.002452 | 0.017609 | 0.892180 | {'estimator\_\_max\_depth': 20, 'estimator\_\_n\_estimators': 100} |
| 5 | KNN | 0.002479 | 0.015884 | 0.890376 | {'estimator\_\_n\_neighbors': 3, 'estimator\_\_weights': 'distance'} |
| 6 | MLPRegressor | 0.003707 | 0.028134 | 0.815822 | {'estimator\_\_alpha': 0.001, 'estimator\_\_hidden\_layer\_sizes': (50, 30, 10, 30, 50)} |
| 7 | Dummy Mean | 0.026426 | 0.087998 | -0.000279 | {} |

Из результатов (таблица 4.3) оценки моделей для задачи преобразования данных типа 42 -> 7 вытекает следующее:

* Random Forest: Эта модель показала наилучшие результаты по всем метрикам (*MSE*: 0.002452, *MAE*: 0.017609, *R2*: 0.892180). Она обеспечивает наиболее точные и надежные предсказания среди всех рассмотренных моделей. Модель была обучена с параметрами 'max\_depth': 20 (максимальная глубина дерева) и 'n\_estimators': 100 (ансамбль из 100 деревьев).
* KNN (k-ближайших соседей): Вторая по точности модель после Random Forest. Она также обеспечивает хорошие предсказания, близкие к результатам Random Forest, и демонстрирует высокий уровень точности по метрикам *MSE*: 0.002479, *MAE*: 0.015884, *R2*: 0.890376. Модель обучена с параметрами 'n\_neighbors': 3 (k=3) и 'weights': 'distance' (веса соседей обратно пропорциональны расстоянию).
* MLPRegressor находится на среднем уровне с метриками *MSE*: 0.003707, *MAE*: 0.028134, *R2*: 0.815822. Это означает, что она может использоваться в некоторых случаях, но не является лучшим выбором для этой задачи. Архитектура автоэнкодера содержит 5 внутренних слоёв: (50, 30, 10, 30, 50), параметр 'alpha': 0.001.
* Линейная регрессия и Ridge: Обе модели демонстрируют худшие результаты по сравнению с Random Forest и KNN, но могут использоваться в редких случаях. Метрики моделей соответственно равны *MSE*: 0.007074, *MAE*: 0.041779, *R2*: 0.673049 - для линейной регрессии и *MSE*: 0.007196, *MAE*: 0.042227, *R2*: 0.665992 - для Ridge регрессии.
* Модель Lasso несостоятельна для этой задачи, что подтверждается ее высокими значениями *MSE* (0.026426) и *MAE* (0.087998), а также низким *R2* (-0.000279).
* Полиномиальная регрессия (со степенью 2) также показывает крайне плохие результаты с очень высоким значением *MSE* (36.044336) и отрицательным *R2* (-1444.560669), что указывает на переобучение.

## **4.4 Сохранение лучших моделей в формате файла обученных моделей.**

Чтобы упростить работу с моделями независимо от их типа (нейронная сеть или статистическая модель), в НИРе 2-го семестра были разработаны форматы для хранения файлов обученных моделей (подробнее см. НИРа 2-го семестра подраздел 2.3 “Разработка форматов файлов обученных моделей.”), было предложено использовать унифицированный формат файлов моделей. Унифицированный файл модели - это архив .tar.gz, внутри которого содержатся определенные папки и файлы:

1. type - файл, хранящий вид модели. Принимает одно из 8 допустимых значений: ['2->7 (Neural)', '2->7 (Stat)', '7->2 (Neural)', '7->2 (Stat)', '7->42 (Neural)', '42->7 (Stat)', '2->42', '42->2'].
2. .pkl файлы статистических моделей. Имена файлов должны начинаться с одинакового префикса ('model\_') и далее иметь название, означающее выход модели (model\_scared.pkl - статистическая модель, выдающая одну из эмоций (scared) 7-ми мерной модели Пола Экмана).
3. папки с нейросетевыми моделями. Имена папок также должны начинаться с одинакового префикса ('model\_') и далее иметь название, означающее выход модели (model\_surprised - нейросетвая модель, выдающая одну из эмоций (surprised) 7-ми мерной модели Пола Экмана).

Подробнее см. НИРа 2-го семестра подраздел 2.3 “Разработка форматов файлов обученных моделей.”.

В этом семестре обученные модели оказались более сложными в вычислительном плане (Random Forest, KNN, полиномиальная регрессия), чем в НИРе 1-го семестра (Keras автоэнкодеры с 1-3 внутренними слоями). Поэтому вместо использования Python модуля pickle, что может привести к долгим временам сохранения и загрузки из-за большого объема данных, предпочтительнее использовать модуль joblib.

Модуль *sklearn.externals.joblib* из библиотеки scikit-learn предназначен специально для сериализации объектов Python, таких как обученные модели машинного обучения. В отличие от модуля pickle, joblib оптимизирован для эффективного сохранения объектов, которые содержат большие массивы данных, особенно массивы numpy, типичные для моделей машинного обучения внутри scikit-learn моделей.

Таким образом, рекомендуется использовать joblib [19] для сохранения и загрузки моделей машинного обучения, особенно если модель содержит большие массивы данных, например, при использовании алгоритмов обучения на больших данных или при обучении моделей с использованием множества признаков.

С другой стороны, модуль pickle является общим инструментом сериализации Python и может быть использован для сохранения практически любого объекта Python. Однако при работе с моделями машинного обучения joblib обычно предпочтительнее из-за его эффективности и оптимизации для сохранения больших данных.

* **Сохранение модели**

Чтобы сохранить лучшую модель, полученную в результате Grid Search (см. подраздел 4.2 “Обучение моделей.”), на диск и затем считать её, воспользуемся функцией:

**from** sklearn.externals **import** joblib

joblib.dump(best\_model, **'best\_model.pkl'**)

* **Загрузка модели**

Когда нужно будет использовать модель в будущем, мы можем легко загрузить её с диска:

loaded\_model = joblib.load(**'best\_model.pkl'**)

* **Обертка модели в унифицированный формат файлов моделей**

Сохранение обученных моделей в унифицированный формат файлов моделей будет описан в НИРе следующего семестра.

# **Заключение**

Модели случайного леса и KNN выделяются как наиболее эффективные для прогнозирования эмоциональных состояний во всех трех типах преобразований.

Так, для задачи 2 -> 7 модели случайного леса ('n\_estimators': 100, 'max\_depth': 10) и KNN ('n\_neighbors': 20, 'weights': 'distance') являются наиболее предпочтительными. Они обеспечивают наиболее точные и надежные предсказания семи эмоциональных состояний на основе двух признаков Valence и Arousal.

На основании выводов, для задачи 7 -> 2 модели Random Forest ('n\_estimators': 100, 'max\_depth': None) и KNN ('n\_neighbors': 3, 'weights': 'distance') являются наиболее подходящими для решения задачи преобразования из семи эмоциональных состояний (Neutral; Happy; Sad; Angry; Surprised; Scared; Disgusted) в два признака. Эти модели обеспечивают наиболее точные и надежные прогнозы для двухмерного пространства PA.

Исходя из результатов, для задачи преобразования 42 -> 7 модели Random Forest и KNN являются наиболее предпочтительными. Модель Random Forest ('max\_depth': 20 и 'n\_estimators': 100) обеспечивает наиболее точные и надежные предсказания среди всех моделей по метрикам *MSE*, *MAE* и *R2*. Модель KNN ('n\_neighbors': 3 и 'weights': 'distance') также показывает хорошие результаты и является второй по точности предсказаний.

Для KNN моделей можно сделать вывод: чем больше входная размерность модели (42 или 7, в сравнении с 2), тем меньшее количество k-ближайших соседей необходимо, чтобы делать точные выходные предсказания. Так, для типов преобразования 7 -> 2, 42 -> 7 необходимо k=3 ('n\_neighbors') ближайших соседа, в то время как для 2 -> 7 моделей нужно значительно большее количество соседей k=20 ('n\_neighbors').

Переобучения в моделях быть не может, т.к. при обучении использовалась кросс-валидация (*GridSearchCV*), а метрики рассчитывались на тестовых, а не тренировочных данных.

Полиномиальная регрессия - достаточно хороша, однако она может быть неэффективной из-за возможного переобучения из-за сложности модели (в типе преобразования 42 -> 7: *R2* = -1444.560668 для полиномов степени 2).

MLPRegressor (в виде автоэнкодера) показывает средние результаты, которые хуже чем у моделей Random Forest и KNN, хотя и могут быть востребованы.

Линейные модели (а также Ridge регрессия), как правило, демонстрируют слабые результаты в сравнении с моделями на основе деревьев решений и k-ближайших соседей, но в некоторых случаях (типы преобразования 7 -> 2, 42->7) могут быть применимы.

# **СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ**

1. МНОГОМЕРНАЯ И ДИСКРЕТНАЯ МОДЕЛИ ЭМОЦИЙ [Электронный ресурс] / URL: https://ozlib.com/851130/psihologiya/mnogomernaya\_diskretnaya\_modeli\_emotsiy
2. Модель эмоционального состояния PAD [Электронный ресурс] / URL: https://ru.abcdef.wiki/wiki/PAD\_emotional\_state\_model
3. Система кодирования движений лица (FACS) - Визуальное руководство [Электронный ресурс] / URL: https://imotions.com/blog/facial-action-coding-system/
4. Система кодирования лицевых движений [Электронный ресурс] / URL: https://ru.wikipedia.org/wiki/Система\_кодирования\_лицевых\_движений
5. Экман, Пол - Критика [Электронный ресурс] / URL: https://ru.wikipedia.org/wiki/Экман,\_Пол#Критика
6. Виртуальный ассистент [Электронный ресурс] / URL: https://ru.wikipedia.org/wiki/Виртуальный\_ассистент
7. Распознавание речи - История [Электронный ресурс] / URL: https://ru.wikipedia.org/wiki/Распознавание\_речи#История
8. Обзор технологий синтеза речи [Электронный ресурс] / URL: https://habr.com/ru/company/tinkoff/blog/474782/
9. Основы построения интеллектуальных систем: учеб. пособ./ Г.В. Рыбина. – М.: Финансы и статистика; ИНФРА-М, 2010. – 432 с.: ил.
10. Цифровые аватары: как виртуальные помощники [Электронный ресурс] / URL: https://vc.ru/future/218083-cifrovye-avatary-kak-virtualnye-pomoshchniki-pereselyayutsya-iz-fantasticheskih-filmov-v-nashi-kvartiry-i-ofisy
11. EMOTION ANALYSIS FaceReader [Электронный ресурс] / URL: https://www.noldus.com/facereader
12. A Circumplex Model of Affect // Journal of Personality and Social Psychology 39(6), 1980, pp. 1161-1178
13. Какие бывают модели машинного обучения / Яндекс Практикум [Электронный ресурс] / URL: https://practicum.yandex.ru/blog/modeli-mashinnogo-obucheniya/#kakie-byvayut
14. Интерпретируемые модели / ИТМО вики-конспекты [Электронный ресурс] / URL: <https://neerc.ifmo.ru/wiki/index.php?title=%D0%98%D0%BD%D1%82%D0%B5%D1%80%D0%BF%D1%80%D0%B5%D1%82%D0%B8%D1%80%D1%83%D0%B5%D0%BC%D1%8B%D0%B5_%D0%BC%D0%BE%D0%B4%D0%B5%D0%BB%D0%B8>
15. Широкий В.Р. , Тихомирова Д.В., Владимиров Р. Д., Доленко С. А., Самсонович А.В The Loop of Nonverbal Communication Between Human and Virtual Actor: Mapping Between Spaces. Advances in Intelligent Systems and Computing, V.1310, pp.484-489 (год публикации - 2021) <https://doi.org/10.1007/978-3-030-65596-9_58>
16. Исследование рынка систем распознавания эмоций [Электронный ресурс] / URL: <https://habr.com/ru/post/133686/>
17. Архитектура платформы машинного обучения в продакшене [Электронный ресурс] / URL: https://habr.com/ru/articles/688406/
18. Введение в машинное обучение с помощью Python: руководство для специалистов по работе с данными. Андреас Мюллер, Сара Гвидо. - Москва [и др.] : Диалектика, 2017. - 472, [1] с. : ил., табл., цв. ил.;
19. Is it Better to Save Models Using Joblib or Pickle? [Электронный ресурс] / URL: https://medium.com/nlplanet/is-it-better-to-save-models-using-joblib-or-pickle-776722b5a095