Курсовая работа

по дисциплине «Машинное обучение»

на тему

«Машинное обучение в задачах классификации эмоций человека по аудиозаписи»

Вид исследуемых данных:

Наборы данных RAVDESS, SAVEE, TESS.

Выполнил:

студент гр. ПМ 3-3

Пьянков Георгий Игоревич

Научный руководитель:

Е.Ю. Щетинин.

**Москва – 2020**

Введение...…...……....…...……....…...……....…...……....…...……...3

Глава 1. Первичный анализ данных....…...……....…......…...……..4

Глава 2. Теоретическая часть....…...……....……...…...…………….

2.1. Классические алгоритмы машинного обучения....….......

2.1.1. Деревья решений....……....…...……....…...…………...5

2.1.2. Линейные модели....…...……....……....…...…………...7

2.1.3. Ансамблевые модели....…...……...……………............8

2.1.4. Бустинг....…….....…...……....…...……....…...…………..12

2.2. Алгоритмы глубокого обучения....……....…...……………..

2.2.1. ANN....…...…….....……....…...……....…...……………...14

2.2.2. CNN....…...…….....……....…...……....…...……………...16

2.2.3. RNN....…...…….....……....…...……....…...……………...18

2.3. Спектральный анализ звуковых волн....…...…..…...……19

Глава 3. Компьютерные эксперименты.......…....…...……………….

3.1. ML-алгоритмы...…...………...……....…...……....…...……..22

3.2. CNN-нейросеть...…...……......……....…...……....…...…….30

3.3. LSTM-нейросеть...…...……....…..…....…...……....…...…...32

Выводы...…...……....…...……....…..…....…...……....…...……....…..34

Коды программ…………………………………………………………..34

Литература...…...……....…...……......…………………………………35

**Введение**

Эмоции человека - удивительная вещь, ведь они напрямую отражают внутреннее состояние конкретной личности в данную секунду. Поэтому, одна из самых интересных задач в речевой технологии - определение эмоции с помощью компьютерных алгоритмов, основываясь либо на аудиозаписи, либо на текстовых признаках. Важность данной темы в индустрии сложно переоценить, учитывая огромное распространение голосовых ассистентов, таких как Google Assistant, Amazon Alexa, Яндекс.Алиса, Microsoft Cortana, и др. Цель подобных продуктов - выполнять ряд задач по типу управления умным домом, искать информацию в интернете, давать голосовые подсказки, и в целом облегчать обиход пользователя. Однако, поскольку психология человека - крайне сложная вещь, для голосовых ассистентов очень важно учитывать учитывать эмоции человека во время использования, для повышения точности работы алгоритмов, и, в конце концов, увеличения уровня пользовательского удовлетворения. В данной работе будут использоваться несколько подходов к решению данной задачи, а именно: использование классических алгоритмов машинного обучения на данных, состоящих из мел-кепстральных коэффициентов, а также использование двух типов нейронных сетей: сверточной, и рекуррентной.

**Глава 1. Первичный анализ данных....…...……....…......…...……..**

**Описание наборов данных**

Датасет, используемый в данной работе состоит из трех наборов данных: TESS, SAVEE, и RAVDESS.

Ссылки:

RAVDESS: <https://www.kaggle.com/uwrfkaggler/ravdess-emotional-speech-audio>

TESS:

<https://www.kaggle.com/ejlok1/toronto-emotional-speech-set-tess>

SAVEE:

<https://www.kaggle.com/ejlok1/surrey-audiovisual-expressed-emotion-savee>

TESS - датасет, состоящий из аудиозаписей эмоций двух человек: пожилой и молодой женщины. У набора семь классов эмоций: злость, отвращение, страх, страх, нейтральное настроение, грусть, удивление.

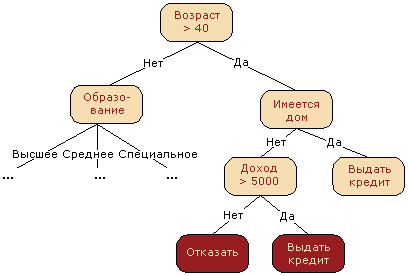
SAVEE - в наборе большое количество записываемых людей, однако, стоит отметить, что он содержит только мужские голоса, однако, это не должно быть проблемой, ибо они будут балансироваться женскими аудиозаписями из TESS.

RAVDESS - один из самых популярных наборов данных для решения задач классификации эмоций. В наборе содержатся очень качественные аудиозаписи эмоций 24-х человек, а также видеозаписи выражений лица во время произнесения той или иной эмоции. В отличие от предыдущих двух, в этой датасете 8 классов, и для балансировки, один из них я исключу, и будет проводиться классификация только по семи видам эмоций.

**Глава 2. Классические ML-алгоритмы**

**2.1Деревья решений**

У дерева очень много аналогий в реальной жизни, потому подобная модель нашла свое применение и в машинном обучении для решения как задач регрессии, так и задач классификации. В анализе решений, дерево может использоваться для визуальной интерпретации процесса принятия того или иного решения. Каким образом алгоритм может быть выражен в образе дерева? В качестве простого примера, приведу задачу из области кредитного скоринга, где дерево решений позволит принять решение о выдаче кредита клиенту банка.



Обобщая дерево решения как алгоритм, можно с легкостью сравнить его с набором простейших логических правил типа: если A > B, то C, иначе D. Большое преимущество данного алгоритма заключается в том, что результаты его работы очень легко интерпретировать, в отличие от ряда сложных моделей, которые скорее могут рассматриваться как черный ящик. Из-за этого преимущества, деревья решения обрели огромную популярность, и заняли свое место в списке 10 лучших data-mining алгоритмов.

В примере задачи кредитного скоринга, решение принималось на основе одного из множества признаков. Но какой признак выбрать первым? Для этого требуется выбрать такое разделение, которое оставит наименьшее количество вариантов. Интуитивно это разделение объясняется примером из игры “Акинатор”, где необходимо угадать знаменитость. Например вопрос “Это Джонни Депп?”, в случае ответа “нет”, оставит после себя примерно n вариантов ответа (где n - количество знаменитостей), когда вопрос “Это женщина?” уже отсечет примерно половину всех возможных вариантов. Это соответствует понятию энтропийного прироста информации. Энтропия - понятие из физики, чисто от 0 до 1, которое позволяет оценить степень хаоса в системе. В случайной выборке она равна единице.

Формула: где - вероятность нахождения системы в i-м состоянии.

В свою очередь, уменьшение энтропии называют приростом информации, который определяется как ,

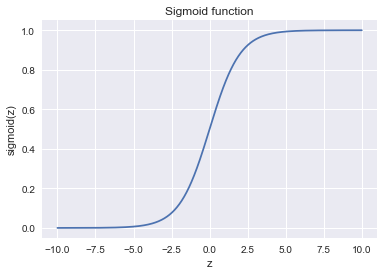
где - чисто групп после разделения, - число элементов выборки, у которых признак имеет i-е значение.

Соответственно, алгоритм выбора лучшего разделения состоит в том, чтобы выбрать признак, дающий наибольший прирост информации. Однако, существуют и другие критерии качества построения дерева. Например, “Неопределенность Джини (Gini Impurity)”: , максимизацию которой можно понимать под максимизацией количества пар объектов одного класса, находящихся в одном поддереве. Существует также “Ошибка классификации”: . Однако, в реальных задачах “ошибка классификации” практически не используется, а критерий Джини работает почти идентично Энтропии Шеннона, поэтому, чаще всего используется последняя.

**2.2 Линейные модели**

Основополагающий принцип работы линейных моделей заключается в возможности разделения пространства признаков плоскостью. Если это получается сделать, то пространство называется линейно разделимым. Если в качестве примера рассмотреть задачу бинарной классификации, обозначив признаки как “+1” и “-1”, то простейшей моделью классификации в этом случае будет: , где x - вектор значений признака, а w - вектор весов. Частным случаем линейного классификатора является ***логистическая регрессия***, главным преимуществом которой является возможность предсказывать вероятность отнесения примера к определенному классу.

. Прогнозирование не только ответа, но и вероятности, является важнейшим требованием во многих задачах из бизнес-области. К примеру, в уже рассмотренной выше задаче кредитного скоринга зачастую прогнозируют вероятность невозврата клиентом кредита. Далее клиентов ранжируют по убыванию. В дальнейшем, кредит получат те клиенты, вероятность невозврата которых меньше заранее определенного граничного значения. Главное отличие линейной регрессии от логистической состоит в том, что последняя на выходе выдает вероятность. Для преобразования числа в вероятность требуется функция . В логистической регрессии эту задачу выполняет сигмоида: . График выглядит следующим образом:



Итак, алгоритм прогноза модели логистической регрессии:

1. Вычислить Уравнение позволит задать плоскость, разделяющую выборку на 2 класса.
2. Вычислить логарифм отношения шансов:
3. Вычислить - значение:

Далее, при помощи метода максимального правдоподобия, данная модель сводится к задаче минимизации логистической функции потерь, а именно:

**2.3 Ансамблевые модели.**

Основная идея моделей данного типа - объединение простейших моделей, которые сами по себе на конкретной выборке выдают достаточно слабые результаты, с целью получить хорошую точность. Существует несколько видов такого объединения: Стэкинг, Бэггинг, и случайный лес.

**Стекинг**

Основная суть стекинга - объединить некое множество случайных моделей, обучить полученную этим объединением модель, для вывода прогнозов, основанных на множественных предсказаниях, которые и возвращают эти слабые модели. В качестве примера, рассмотрим модель, построенную из Логистической регрессии и Метода опорных векторов. Нейросеть, полученная на их основе, будет принимать результаты двух более слабых моделей на вход, и выдавать предсказания, полученные на их основе. Алгоритм обучения такой модели следующий:

1.Тренировочная выборка делится на две части;

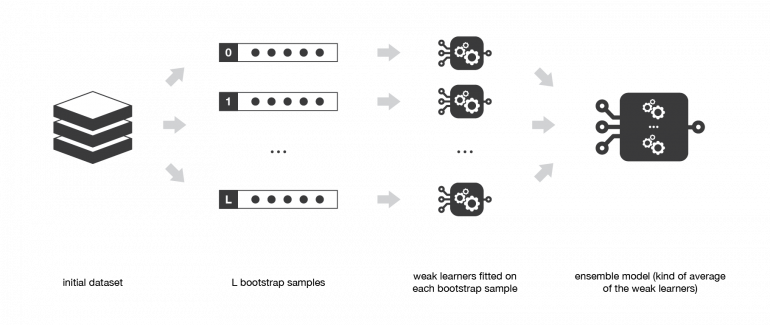
2.Выбираются L слабых моделей, и обучаются на первой части;

3.Для каждой и слабых моделей делаются прогнозы для объектов из второй части тренировочной выборки;

4.Модель обучается второй раз, используя в качестве входов предсказания, сделанные слабыми моделями.

Основной недостаток стекинга заключается в том, что набор данных разбивается на две части, следовательно, для обучения базовых моделей доступна только половина данных, и половина для обучения метамодели.

**Бэггинг**

Основная цель данного алгоритма состоит в создании более надежной ансамблевой модели, чем те, из которой она составляется. Идея достаточно проста: подбирается несколько независимых моделей, и их результаты усредняются для получения модели с меньшей ошибкой. Существует несколько способов объединения моделей для параллельного обучения. В случае задачи регрессии прогнозные данные в буквальном смысле усредняются, а в задачи классификации используется метод мажоритарного голосования (самое частое предсказание среди множества всех прогнозов). Основным преимуществом Бэггинга является параллелизм. На иллюстрации можно увидеть, как для увеличения производительности каждой отдельной модели используется распараллеливание. 

**Случайный лес (Random Forest)**

Из-за своей простоты и эффективности, деревья решения являются одним из самых популярный алгоритмов машинного обучения. Метод случайных подпространств позволяет снизить корреляцию между деревьями и избежать переобучения. Деревья решений, из которых и состоит случайный лес, обучаются индивидуально на разных подмножествах одной выборки, выделяющихся случайно. Алгоритм случайного леса из N деревьев заключается в следующем:

1. Сгенерировать выборку
2. Построить дерево по выборке . Выбирается наилучший признак и проводится разбиение до конца выборки. Дерево строится, пока в каждом листе не более объектов, или пока не достигается заданная высота дерева. При каждом разбиении оптимальное разделение ищется только из заранее отобранных случайных признаков.
3. Повторить от

В итоге получится классификатор: , в котором при решении задачи классификации выбирается решение голосованием по большинству, а в случае регрессионной задачи - усреднение.

В итоге, обобщая все вышеперечисленное о данном алгоритме, можно сказать, что Random Forest - это частный случай Бэггинга, в котором используются только решающие деревья, обучая которые, признаки выбираются из некоторого случайного подмножества признаков.

**Бустинг**

В продакшене данный алгоритм применяется в задачах ранжирования выдачи поисковых систем в интернете. Первой компанией, которая внедрила бустинг на прод, была AltaVista, а за ней пошли и большие поисковики. По своей сути алгоритм бустинга схож с предыдущими двумя, однако есть ряд отличий. Если, в случае Бэггинга, алгоритм работает на уменьшение разброса, то Бустинг - это метод, позволяющий адаптировать последовательно ряд слабых моделей: каждая из них подбирается так, чтобы придавать большее значение тем сэмплам в наборе данных, на которых больше всего ошибалась предыдущая модель. То есть фактически, суть бустинга заключается в том, что каждая следующая модель направлена на исправление ошибок предыдущей. Базовые модели, которые зачастую используются в бустинге - модели с маленьким разбросом, но большим смещением. Первопричина этому - низкие вычислительные затраты на обучение подобных моделей как самих по себе, так и последовательно. После выбора атомарных моделей, необходимо определить, как они будут последовательно подгоняться и агрегироваться.

**AdaBoost (Адаптивный бустинг)**

В данном алгоритме ансамблевая модель определяется как взвешенная сумма N слабых моделей, из которых она состоит.

, где - коэффициента, а - слабые ученики.

Поскольку в данном случае, поиск лучшей ансамблевой модели является сложной оптимизационной задачей, лучший способ найти ее - итеративный процесс. Фактически, слабые ученики добавляются поочередно, просматривая каждую итерацию, для поиска наилучшей возможной пары (коэффициент - ученик) для добавления к ансамблевой модели. Говоря языком математики, итеративно определяется , где и выбраны так, чтобы модель наилучшим образом соответствует обучающим данным, и это наилучшее возможное улучшение, в сравнении с предыдущей итерацией.

Далее можно определить

где - ошибка подгонки конкретной модели, а - функция потерь. В итоге, вместо глобальной оптимизации по всем моделям в сумме, мы приближаем оптимальное значение локальной оптимизацией через построение и добавление слабых моделей к ансамблю по одной.

Приведу пример работы данного алгоритмы при решении задачи бинарной классификации с N объектами. Изначально, все объекты имеют одинаковые веса . Далее, раз повторяются следующие шаги:

1. Обучается наилучшая слабая модель.
2. Вычисляется коэффициент обновления, являющийся, фактически, метрикой оценки слабой модели, показывающая, насколько вклад этой модели должен быть учтен в ансамбле.
3. Обновляется сильная модель с добавлением нового слабого ученика, умноженного на значение коэффициента его обновления.
4. Посчитать новые веса объектов, показывающие наблюдения, на которых необходимо сосредоточиться на следующей итерации.

С повторением всех этих шагов последовательно строятся L моделей, и объединяются в линейную комбинацию, взвешенную по коэффициентам, которые выражают эффективность каждого ученика.

**Градиентный бустинг (Gradient Boosting)**

В данном алгоритме, модель ансамбля, ровно как и в адаптивном бустинге, представляет собой взвешенную сумму слабых моделей. Здесь так же, как и в AdaBoost, используется итеративный оптимизационный подход, поскольку задача сводится к градиентному спуску: на каждой итерации слабый ученик подгоняется к антиградиенту ошибки подбора по отношению к текущей модели ансамбля.

На языке формул, процесс оптимизации методом градиентного спуска, в данном случае может быть записан следующим образом:

Приведу простой пример использования модели градиентного бустинга:

1. Обучается наилучший ученик через псевдо-остатки (приближается антиградиент по отношению к текущему сильному ученику)
2. Вычисляется оптимальный размер шага градиентного спуска
3. Обновляется ансамблевая модель, учитывая добавление нового слабого ученика, умноженного на размер шага.
4. Вычисляются новые псевдо-остатки, показывающие для каждого наблюдения, в каком направлении необходимо обновить прогнозы.

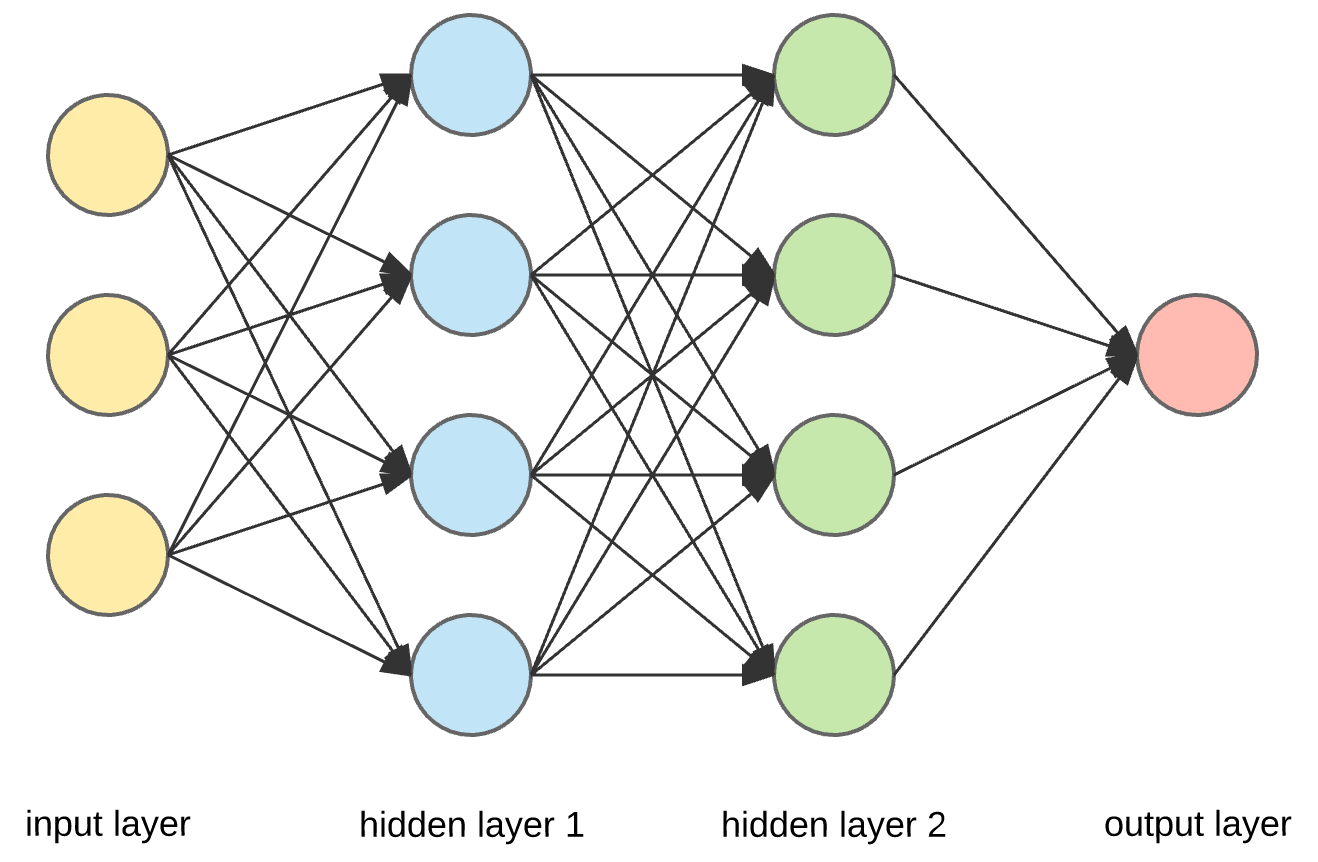
Повторяя этот алгоритм, строятся L моделей, и агрегируются в соответствии с подходом градиентного спуска. По итогу, AdaBoost для каждой итерации решает задачу локальной оптимизации (ищет лучшую слабую модель с ее коэффициентом для последующего добавления к ансамблю), а градиентный бустинг использует градиентный спуск, и легче адаптируется в большому количеству функций. Gradient Boosting можно с легкостью назвать обобщением AdaBoost для рандомных дифференцируемых функций потерь.

**Глава 3. Алгоритмы глубокого обучения**

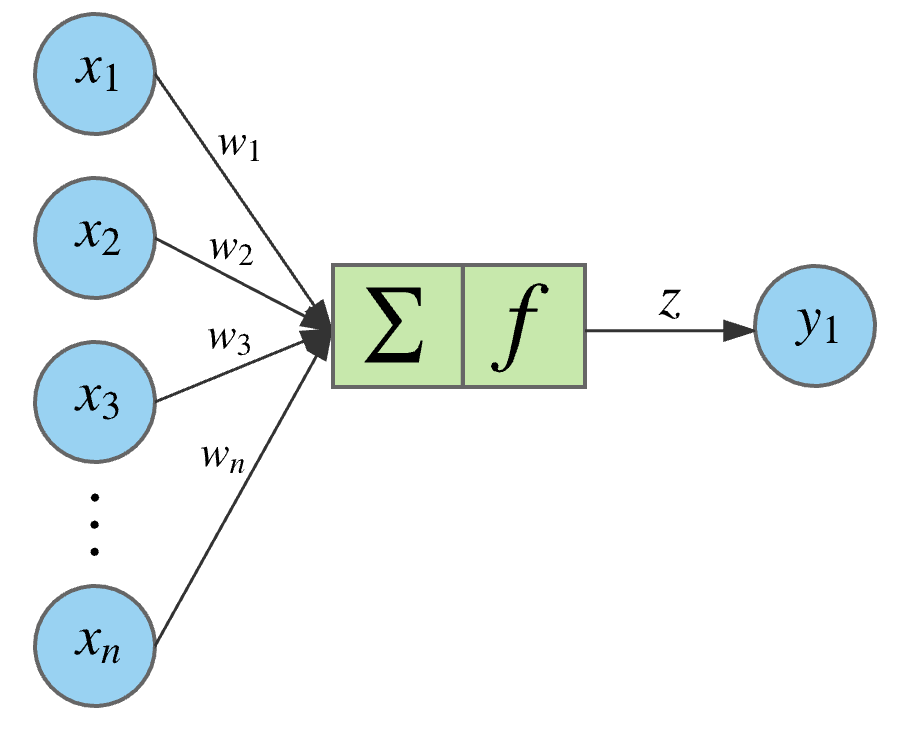
В данной работе для решения поставленной задачи будут рассмотрены три типа нейросетей, а именно: ANN (Искусственная Нейронная сеть), CNN (Сверточная Нейронная сеть), RNN (Рекуррентная Нейронная сеть).

**ANN/MLP. Многослойный перцептрон.**

Данный тип нейросетей представляет собой несколько полносвязных слоев, которые отражены на иллюстрации.



На данном рисунке слой желтых ячеек - входной слой, синие и зеленые ячейки - скрытые слои, и красная - выходной. Ячейка в каждом слое соединена с каждой ячейкой в следующем. Каждая ячейка скрытого слоя представляет собой следующую картину:



Ячейка получает на вход взвешенную сумму входов, и пропускает ее через активационную функцию. То, что получилось на выходе активационной функции, затем подается на вход ячейки следующего слоя. Таким образом сигнал идет через всю сеть слева направо, и по итогу, для получения результата, необходимо провести операции для каждой ячейки. Функция, заложенная в каждую ячейку, представляет собой формулу: , где b - смещение, x - входное значение, а w - вес.

Главная задача смещения - обеспечить тренировку модели даже в случае подачи нулей на вход, поскольку b = 1. Нейронная сеть обучается при помощи алгоритма обратного распространения ошибки.

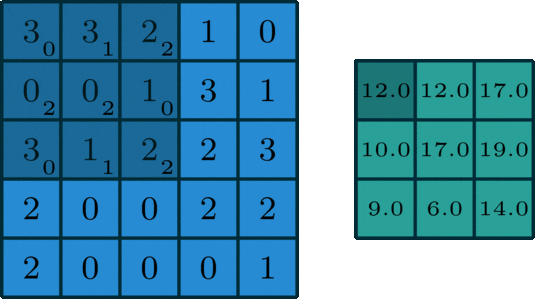
Общая схема обучения состоит из четырех шагов:

1. Случайным образом составить веса для каждой ячейки.
2. Для каждого сэмпла из тренировочной выборки, рассчитать прогнозные значения с использованием весов из первого шага.
3. Сравнить прогнозные и реальные значения, посчитать ошибку, используя лосс-функцию.
4. Используя метод обратного распространения ошибки и метод стохастического градиентного спуска, оптимизировать веса, двигаясь в направлении антиградиента.

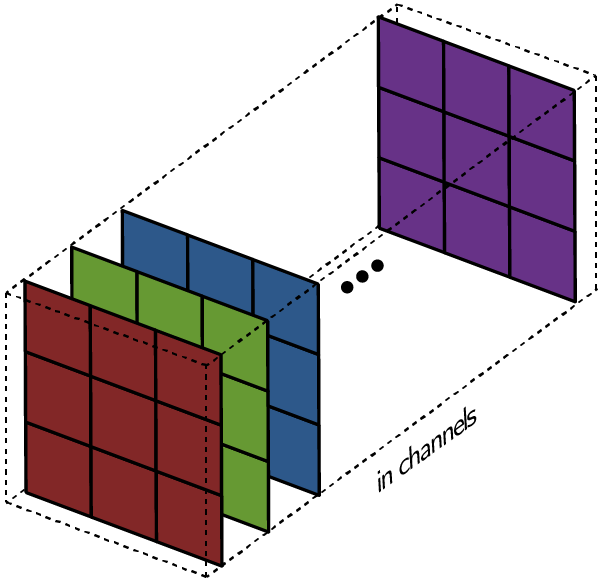
**CNN/Сверточная нейронная сеть.**

Данный тип нейросети основан на математической операции под названием “свертка”. Свертка в данном случае - операция поэлементного перемножения элемента в матрице наблюдения на элемент в матрице ядра.

Свертки делятся на два типа: двумерная и одномерная.



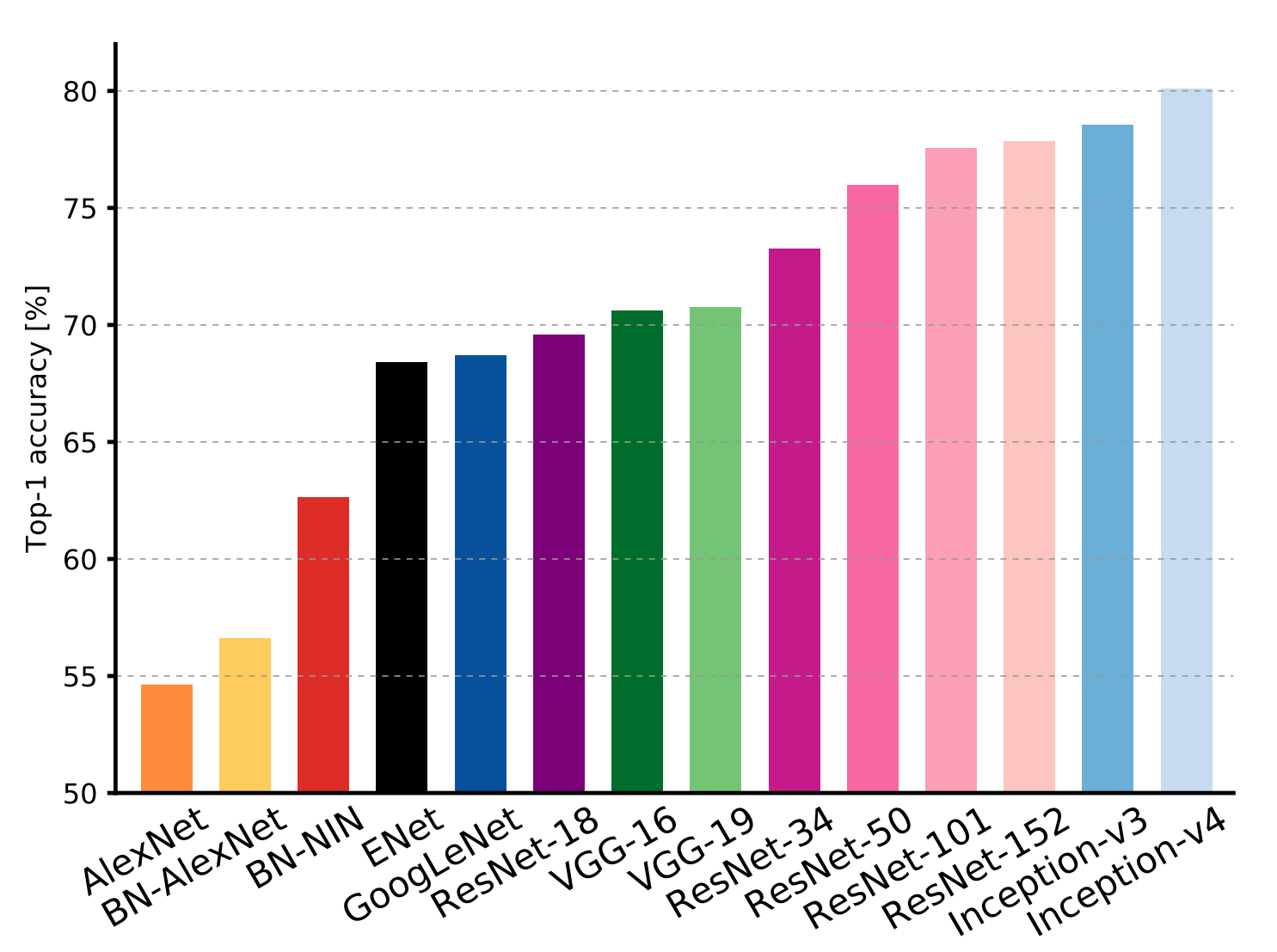
На данной иллюстрации показано, как ядро, скользя по матрице признаков, образует новую матрицу признаков. Поскольку сверточные нейросети используются, в основном, для решения задач классификации изображений, то у этого изображения, зачастую, три канала: R - red, G - green, B - blue. В таком случае, свертка происходит по тензору:



Пулинг. Слой пулинга нужен для снижения размерности изображения. Исходное изображение делится на блоки размером

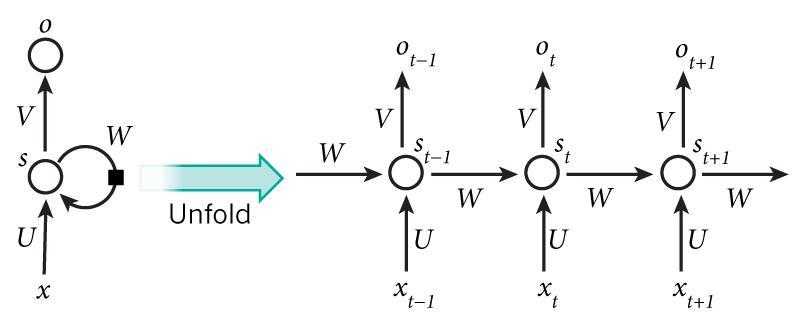
w×h и для каждого блока вычисляется некоторая функция. Чаще всего используется функция максимума (англ. *max pooling*) или (взвешенного) среднего (англ. *(weighted) average pooling*). Обучаемых параметров у этого слоя нет.

Известные модели сверточных нейронных сетей это: GoogLeNet, LeNet, VGG, ResNet. Ниже приведен рейтинг по точности классификации.



**RNN/Рекуррентные нейронные сети.**

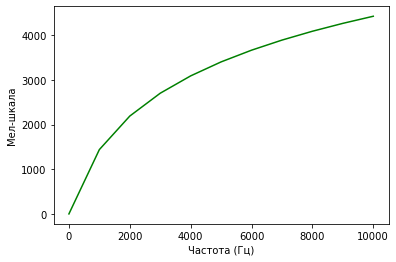
Основная идея данного типа нейронных сетей заключается в последовательном использовании информации. Первое отличие от многослойных перцептронов в том, что здесь используются зависимые входы и выходы. Во вторых, в нейроне рекуррентных нейросетей существует так называемая память, учитывающая предыдущую информацию из потока данных.



На данном рисунке показывается, как RNN разворачивается в полную сеть, где - вход на временном шаге , - память сети, зависящая от предыдущих состояний и текущего входа, - выход на шаге , который вычисляется исключительно на основе памяти. Поскольку RNN демонстрируют большой успех в задачах машинного перевода и обработки естественного языка, то их используют преимущественно в задачах NLP. Наиболее часто используемый тип RNN - LSTM (Long-Short-Time-Memory).

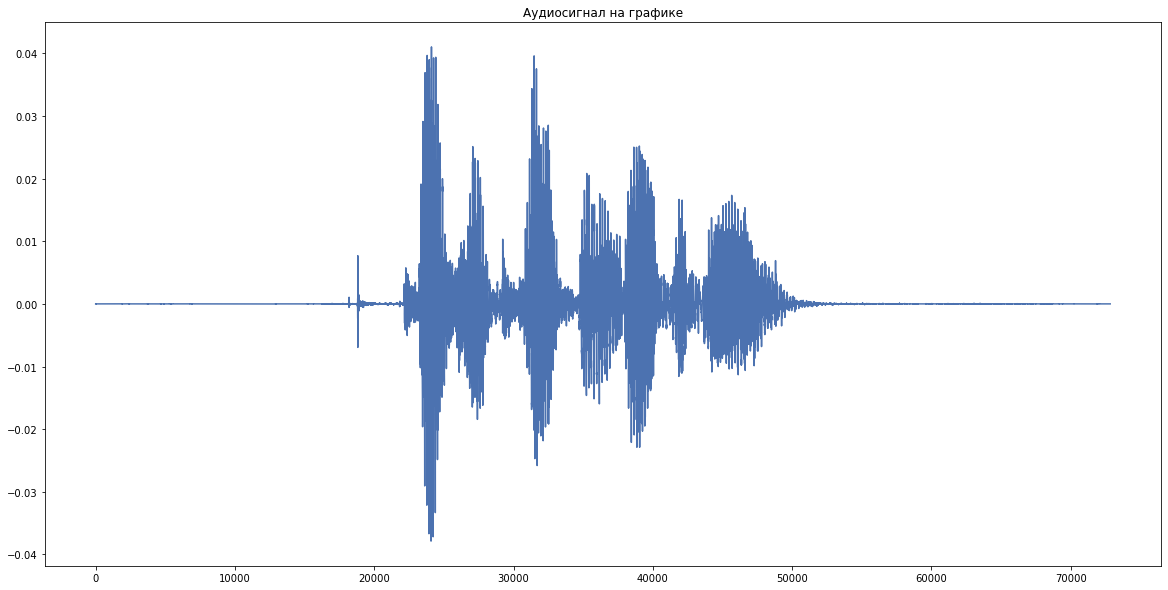
**Спектральный анализ звуковых волн**

В качестве основных признаков для решения задачи классификации эмоций будут использоваться мел-кепстральные коэффициенты. Начну с определения: ***мел*** - единица высоты звука, основанная на его восприятия органами слуха. Высота звука нелинейно зависит от его частоты, и эта зависимость может быть визуализирована на графике:

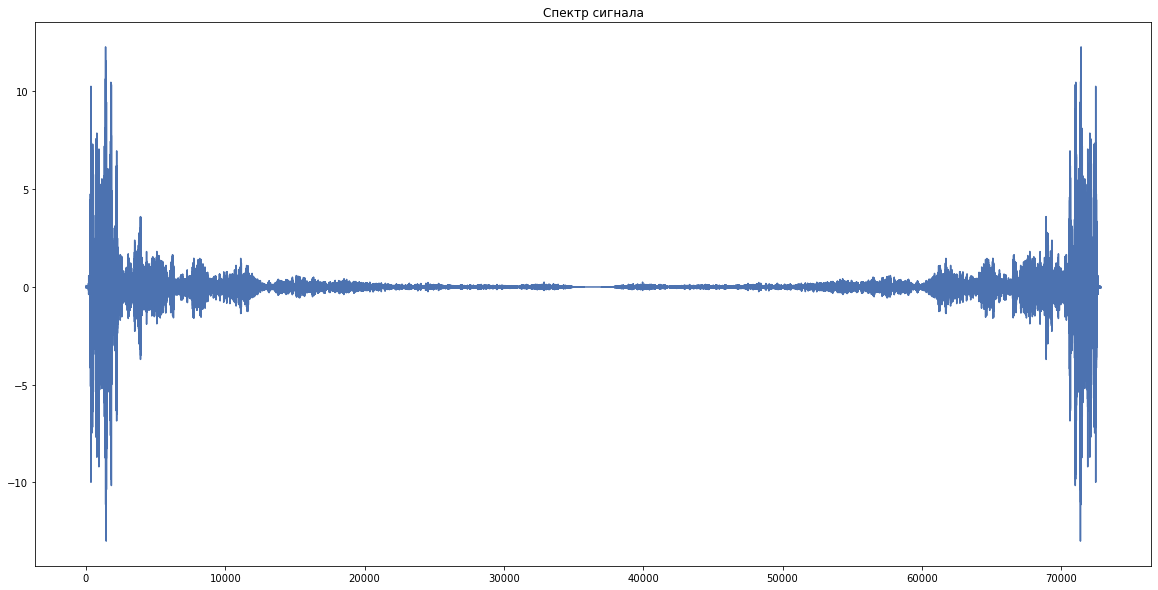


Формула данной зависимости выведена эмпирически: . В свою очередь ***кепстр*** - акустическая волна, которую излучает человек голосовыми связками. Для получения мел-кепстральных коэффициентов необходимо применить преобразование на аудиосигнал.

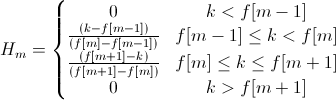
Для примера, возьмем графическую визуализацию сигнала одной из аудиозаписей:



Спектр этого сигнала:



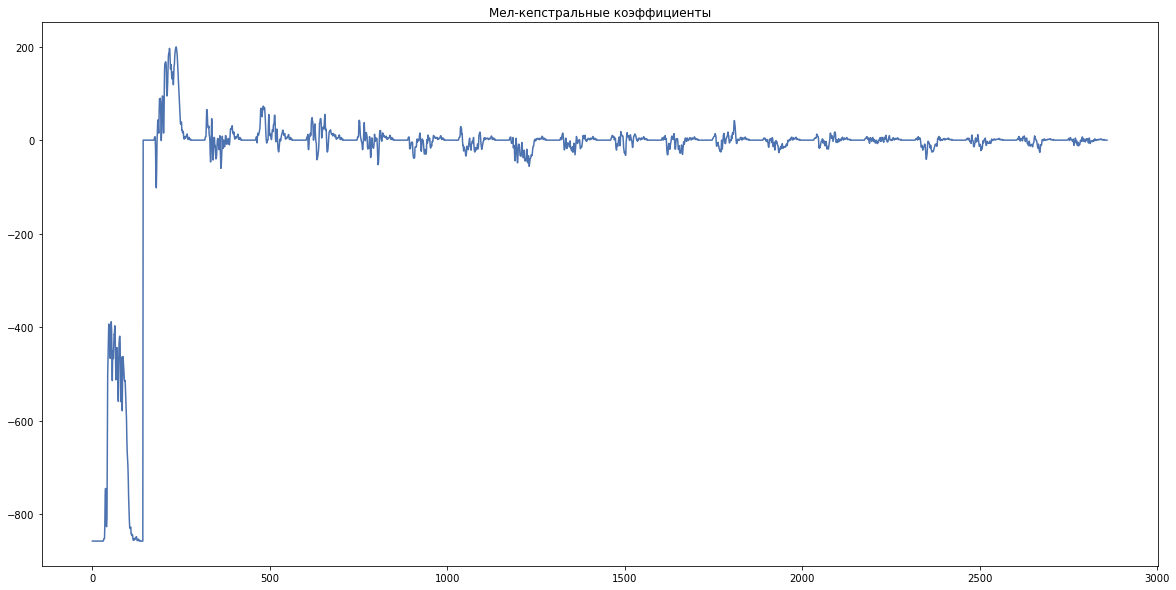
Через перемножение векторов спектра сигнала, и оконной функции



мел-шкалы, можно найти энергию сигнала, которая попадает в каждое из окон. В результате данной операции получится коэффициентов, которые называются мел-спектральными, однако нам необходимы мел-кепстральные коэффициенты, которые легко получаются из спектральных путем дискретного косинусного преобразования логарифма квадрата мел-спектральных коэффициентов



Отобразим их на графике:



В результате всех этих операций получается набор значений, который в десятки раз уменьшает вычислительную сложность для решения моей задачи, по сравнению с использованием полноценного спектра аудиосигнала.

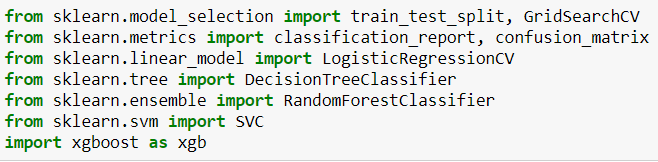
Спектрограмма:



**Глава 4. Компьютерные эксперименты**

Применим информацию, полученную из теоретической части на практике. Начну с классических классификационных ML-алгоритмов: Решающее дерево, Случайный лес, Логистическая регрессия.

Импорт библиотек:

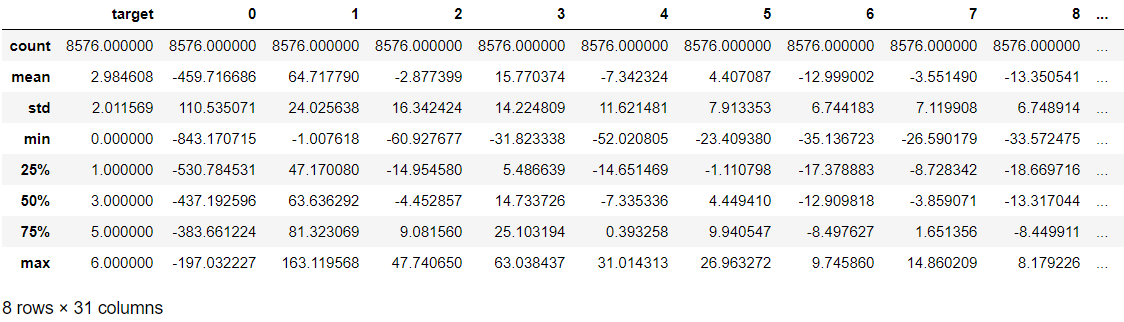


***train\_test\_split*** необходим для деления выборки на тренировочную и тестовую, а ***classification\_report*** и ***confusion\_matrix*** - для оценки результатов тренировки моделей.

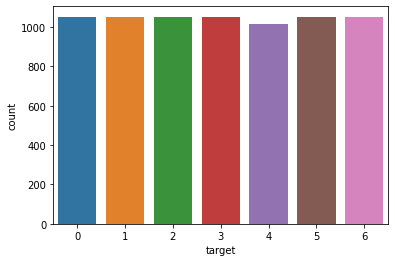
Прежде чем тренировать что-либо, необходимо сделать краткий анализ данных. Что мы имеем:



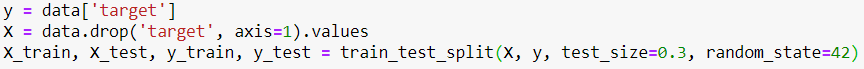
Описание набора данных:



Суды по этим сводкам, набор данных имеет 31 столбец: первый - целевая переменная, а остальные 30 - мел-кепстральные коэффициенты. Количество наблюдений - 7328, классы - числа от 0 до 6, что отражает 7 эмоций.

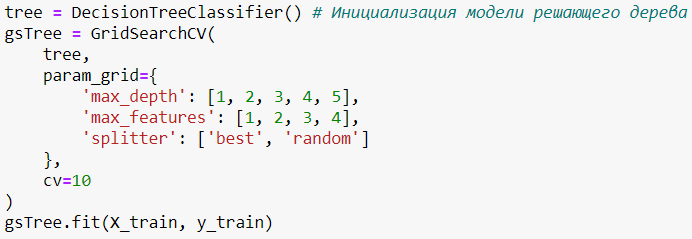


Классы практически идеально сбалансированы, что позволяет запустить процесс тренировки практически без потерь точности на классификации пятой эмоции. Поместим целевую переменную и регрессоры в разные ячейки памяти, и разделим выборку на тренировочную и тестовую:



**Решающее дерево**

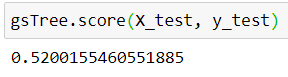
Тренировка решающего дерева с использованием оптимизации гиперпараметров через ***GridSearch*** с последующей кросс-валидацией на десяти фолдах:



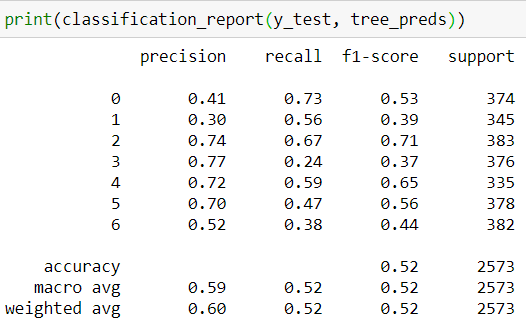
Поиск параметров нашел наилучшие:



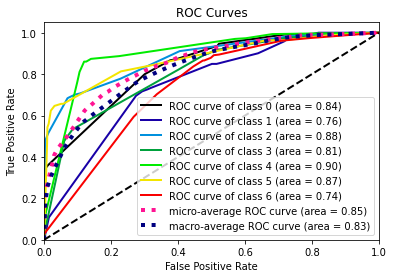
После всех оптимизаций, решающее дерево выдает следующий результат на тестовой выборке:



Для настолько простой модели это достаточно неплохо, посмотрим на метрики классификации:



ROC-AUC:



Первая эмоция - слабое место этой модели, ибо на ней она выдает лишь 0.41 точности.

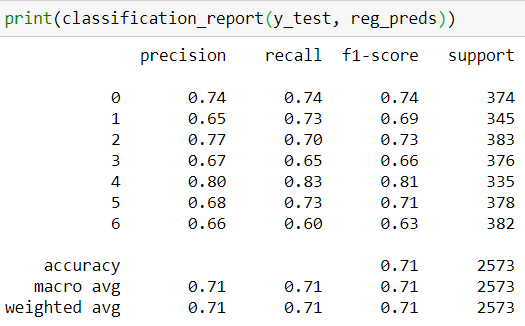
**Многоклассовая логистическая регрессия**

Используется именно этот тип логрегрессии, ибо изначально эта модель предназначена для бинарной классификации.

В данном случае я пришел к тому, что оптимизировать гиперпараметры не имеет смысла ввиду простоты модели, но кросс-валидация имеет место абсолютно всегда. Инициализирую модели со встроенной кросс-валидацией на пяти фолдах:



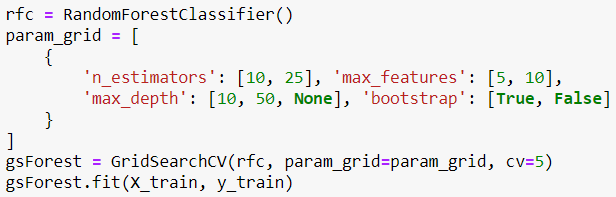
Получились следующие результаты:



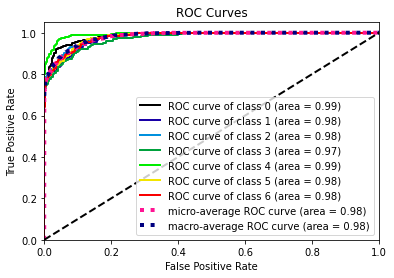
Неплохо, но что может предложить модель случайного леса?

**Случайный лес**

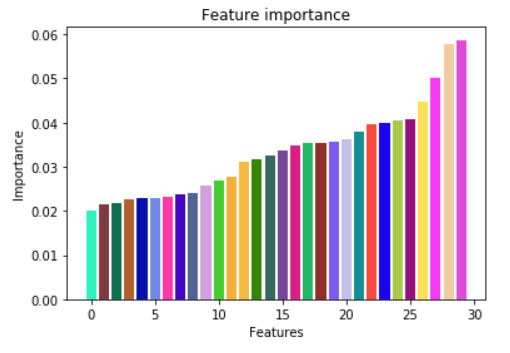
Будет использоваться модель случайного леса с подбором гиперпараметров из заданного словаря, и пятифолдовой кросс-валидацией.

****

ROC-кривая:

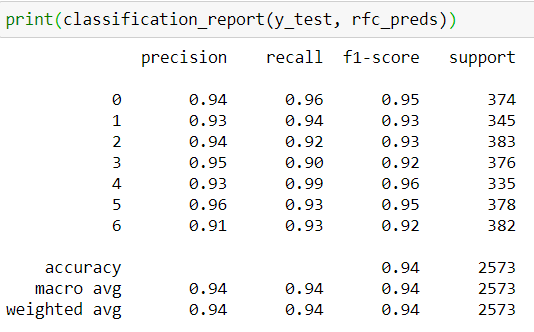


Одно из огромных преимуществ ансамблевых алгоритмов - возможность получить “силы” факторов. Изображу их на графике:



Как можно видеть, чем ближе коэффициент к номеру 30, тем сильнее он влияет на результат вычислений.

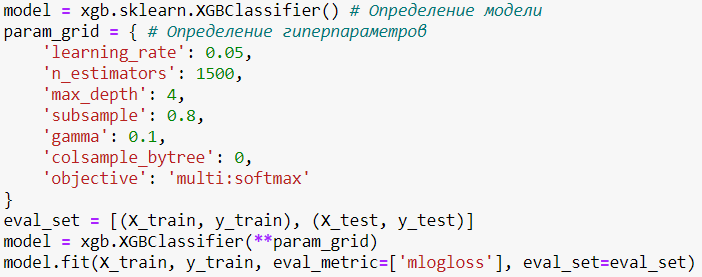
Результат работы модели случайного леса:



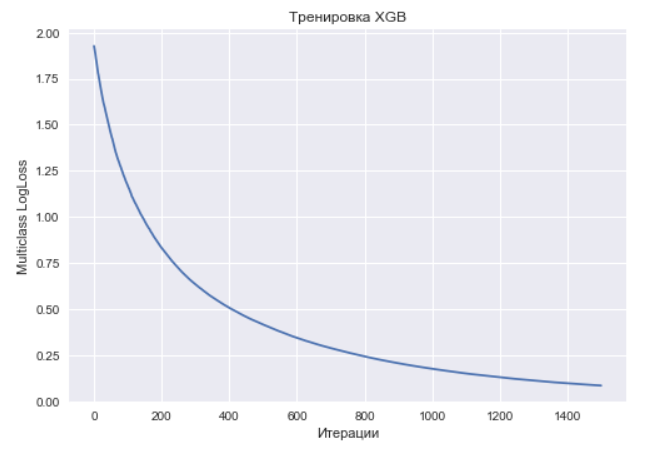
Из чего можно сделать вывод, что модель очень хорошо себя показывает на данном наборе данных, выдавая среднюю точность в 0.94.

**Градиентный бустинг**

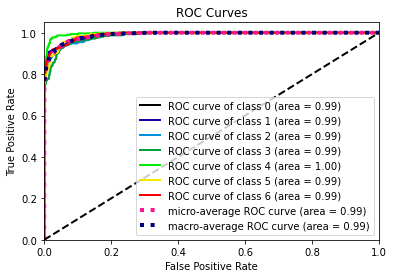
Последний классический алгоритм машинного обучения, использующийся в моей работе - градиентный бустинг с использованием библиотеки XGBoost, главными преимуществами которой является информативность и возможность тонкой настройки модели.



Гиперпарамеры подобраны опытным путем. Коэффициент регуляризации gamma - наилучший средний показатель для данного датасета. Для обучения я выбрал 1500 итераций. Метрика: LogLoss.

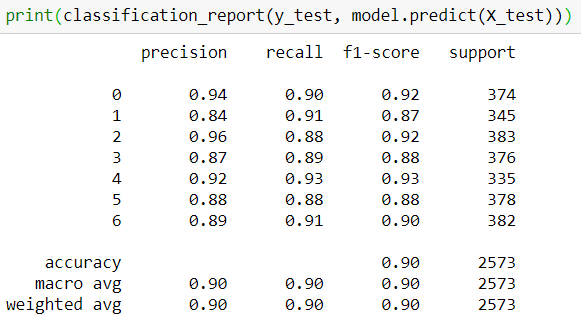


ROC-кривая:



Судя по графику, на протяжении всех итераций, значение функции потерь стремительно уменьшалось.

Посмотрим на результаты:

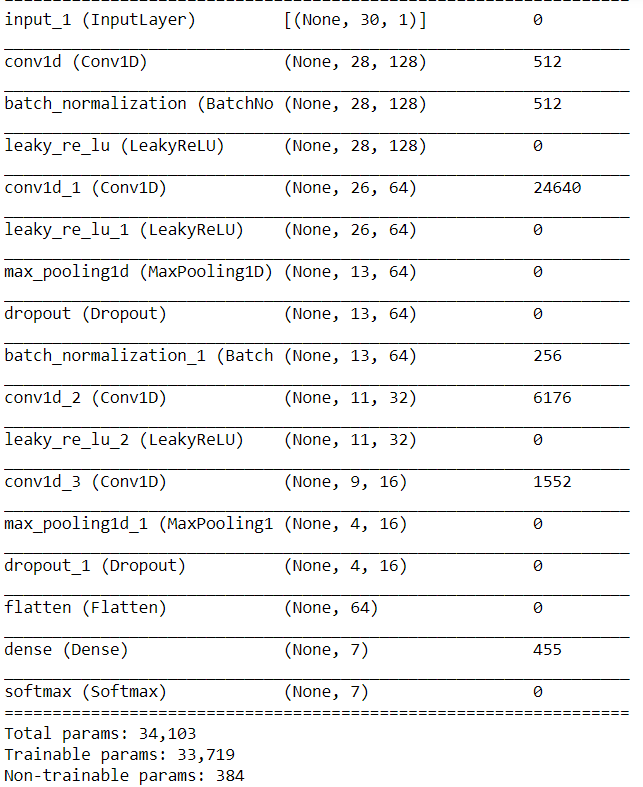


Результаты близки к модели случайного леса. Посмотрим, что получим от нейросетей.

**CNN-нейросеть**

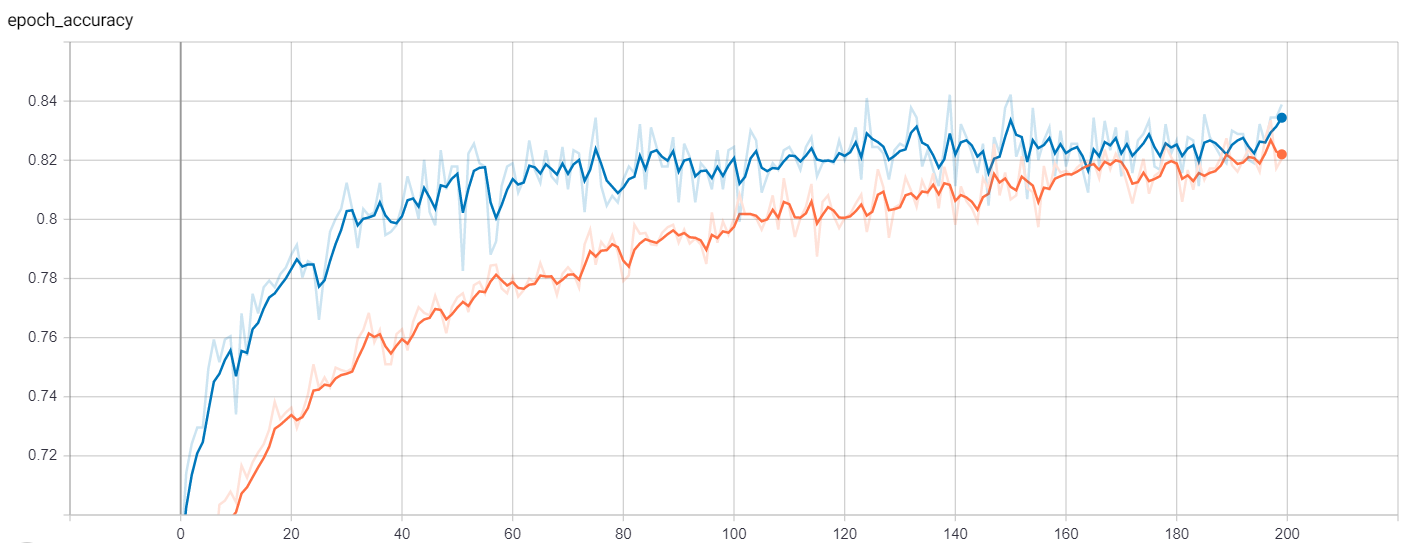
Сверточные нейросети всегда были хороши для классификации звуков и изображений.

Нейросеть, используемая в данной работе имеет следующую архитектуру:

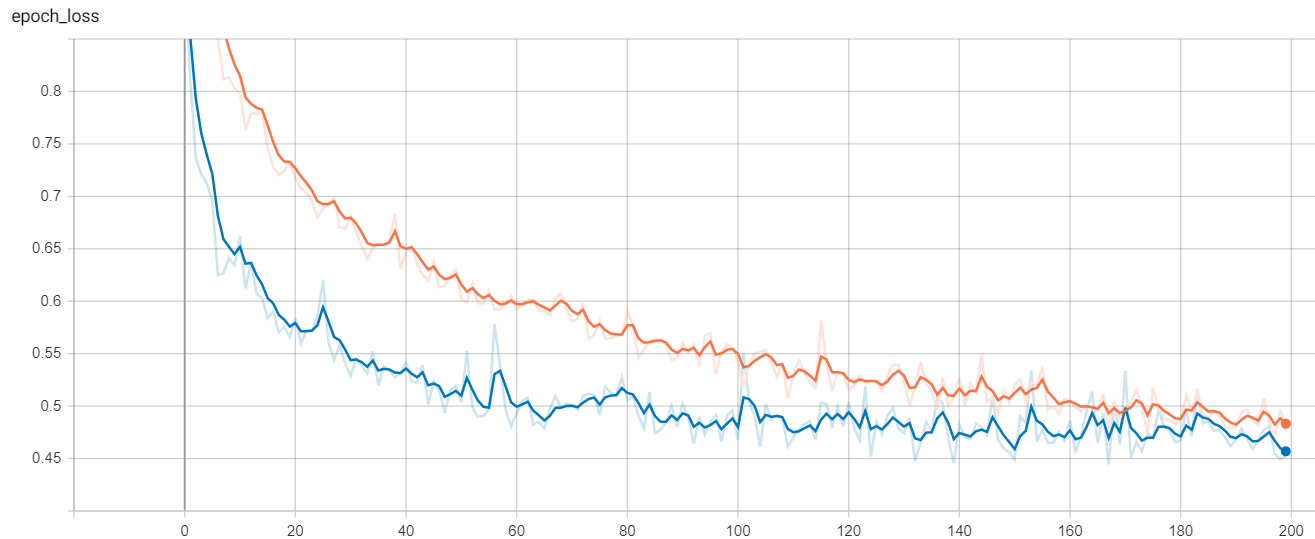


Графики из TensorBoard:

Accuracy-метрика:

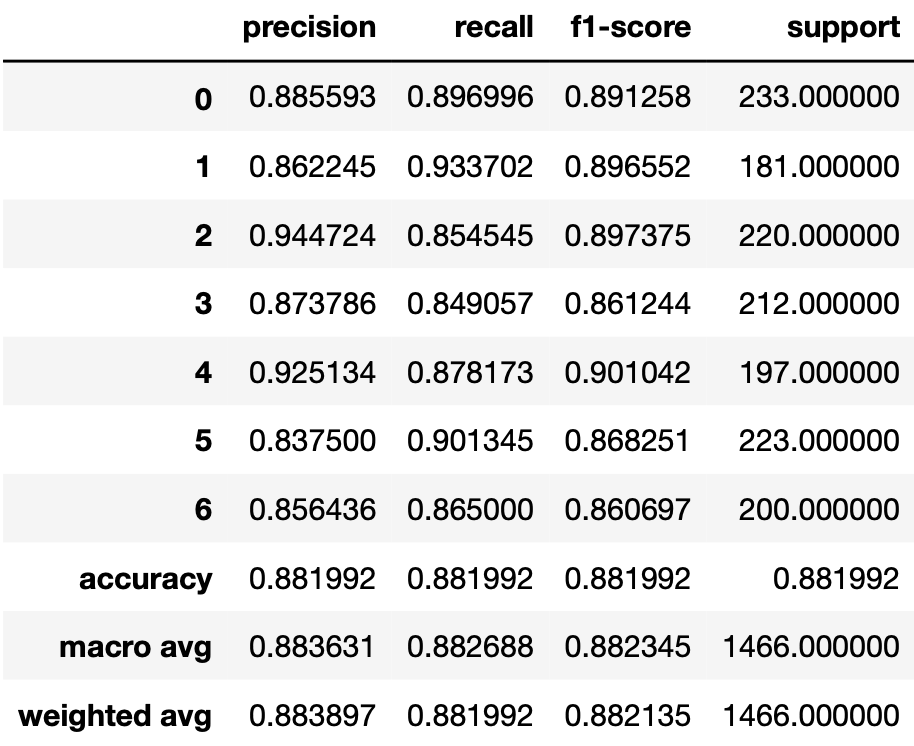


Ошибка сети:

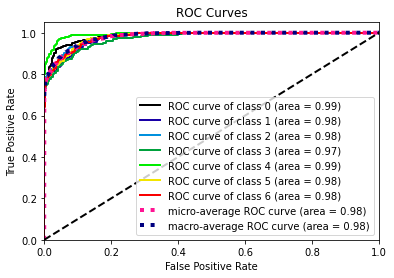


Точность сети составляет 0.89 на тестовой выборке, что является очень хорошим показателем.

Результаты классификации:



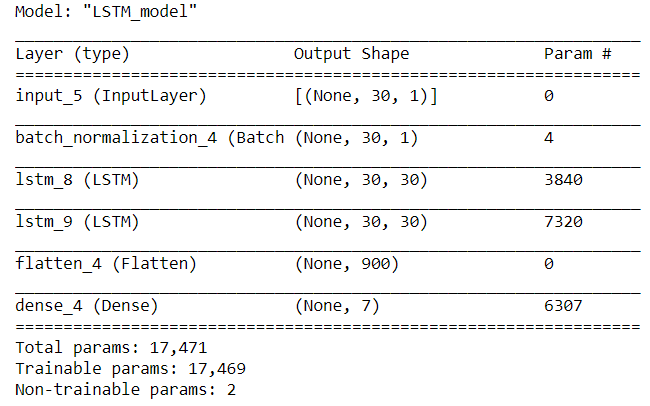
ROC-AUC:



**LSTM-нейросеть**

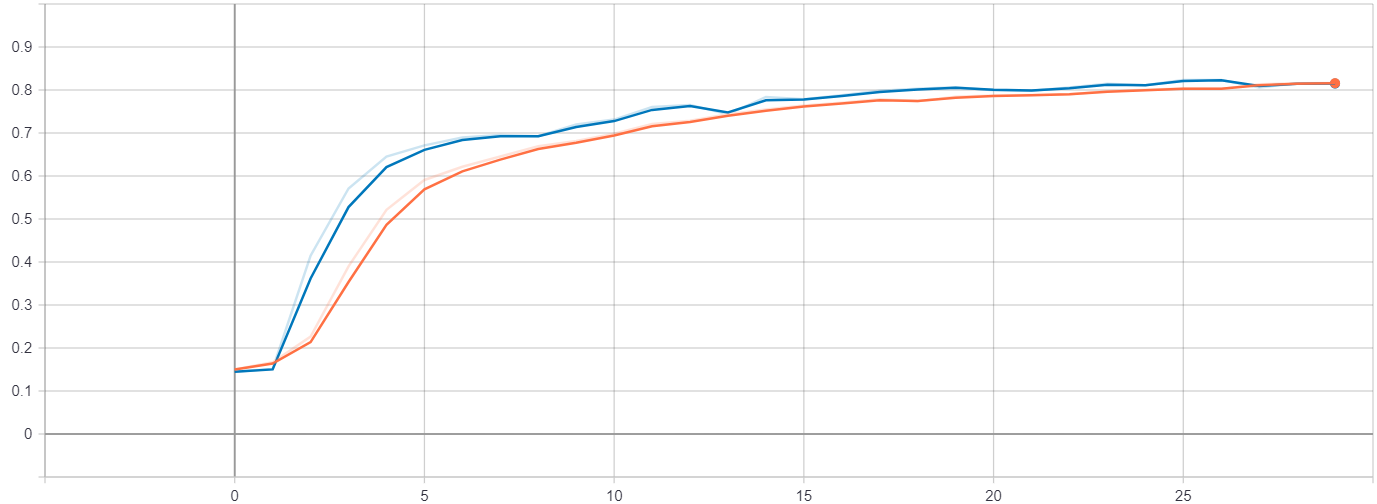
Экспериментальный тип нейросети. Посмотрим на результаты.

Архитектура сети:

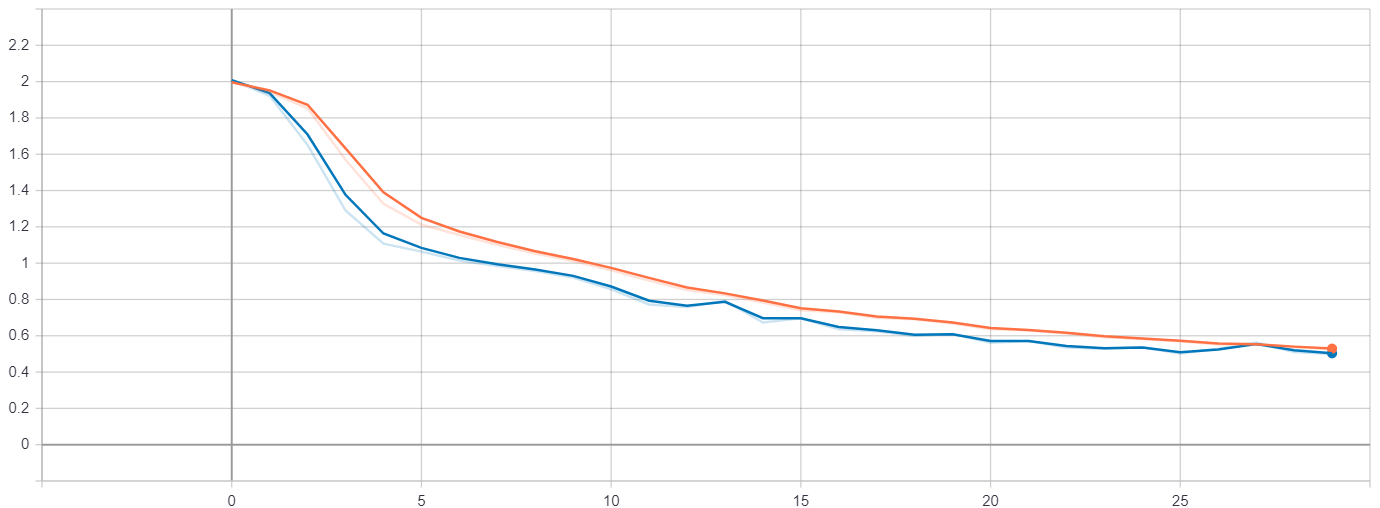


Графики из TensorBoard:

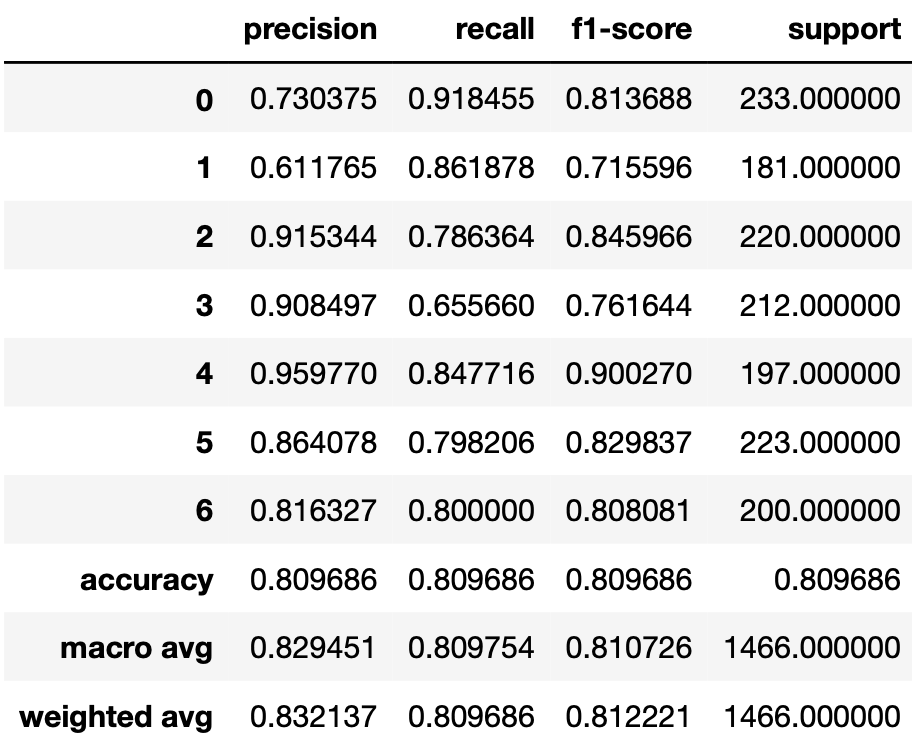
Accuracy-метрика:



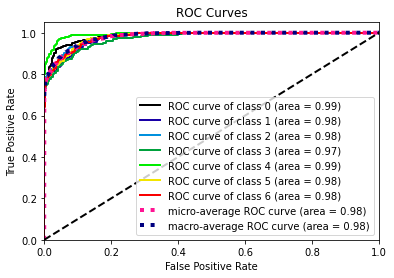
Ошибка сети:



Результаты классификации:



ROC-AUC:



**Выводы**

В данной работе были рассмотрены и применены различные алгоритмы машинного обучения. Деревья решений, как и следовало ожидать для задачи такой сложности, не показали впечатляющих результатов. Случайный лес и градиентный бустинг показали очень хорошие результаты с процентом точных предсказаний на тестовой выборке, близким к 0.9. Сверточная нейросеть показала отличные результаты со средней точностью в 0.84. Экспериментальный вариант простой рекуррентной Long-Short-Term-Memory нейросети показал среднюю точность в 0.81.

**Коды программ**

Для курсовой работы создан GitHub репозиторий:

<https://github.com/H3OX/nn_lab>

В папке data хранятся наборы данных

В папке nn хранятся все модели нейросетей

В папке utils хранятся коды для предобработки данных

Также, для быстрого ознакомления с моделями, в архиве прилагается Jupyter Notebook.

**Литература**

1. Рабинер Л., Шафер Р. Цифровая обработка речевых сигналов. — М.: Радио и связь, 1981. — 489 с.
2. 5. Т.В. Шарий, О проблеме параметризации речевого сигнала в современных системах распознавания речи
3. Тампель И.Б., Карпов А.А. Автоматическое распознавание речи. Учебное пособие. − СПб: Университет ИТМО, 2016. – 138 с.
4. Добрушкин Г. О., Данилов В. Я. Сопоставление качества Мел- и Барк-частотных кепстральных коэффициентов для параметризации речевого сигнала // Научные работы. — 2011. — Т. 160. — С. 167–171.
5. H. Kun, Yu. Dong, and I. Tashev, Speech emotion recognition using DNN and extreme learning machine, proceedings of INTERSPEECH, ISCA, Singapore, pp. 223227, 2014.
6. <https://www.machinelearning.ru>

# [Kaggle Tensorflow Speech Recognition Challenge](https://towardsdatascience.com/kaggle-tensorflow-speech-recognition-challenge-b46a3bca2501)

1. [Emotion Recognition: an overview](https://www.sciencedirect.com/topics/computer-science/emotion-recognition)