Федеральное государственное образовательное бюджетное

учреждение высшего образования

**«Финансовый университет при Правительстве**

**Российской Федерации»**

Департамент анализа данных и машинного обучения

Пояснительная записка к курсовой работе по дисциплине

«Технологии анализа данных и машинное обучение»

на тему:

**«Машинное обучение в задачах распознавания объектов на фотографии»**

Выполнил:

студент ПИ19-3

Деменчук Г. М.

Научный руководитель:

доцент, канд. пед. наук

Норм мужик П. В.

**Москва**

**2022**

ОГЛАВЛЕНИЕ

[I. ВВЕДЕНИЕ 3](#_Toc101384996)

[II. ОСНОВНЫЕ ПОНЯТИЯ ПРИ КЛАССИФИКАЦИИ ИЗОБРАЖЕНИЙ 4](#_Toc101384997)

[Различия между глубоким и классическим машинным обучением 4](#_Toc101384998)

[Особенности обработки изображений 4](#_Toc101384999)

[Метрики качества 6](#_Toc101385000)

[III СБОР И ПРЕПРОЦЕССИНГ ДАННЫХ 8](#_Toc101385001)

[Сбор данных 8](#_Toc101385002)

[Фильтрация данных датасетов 10](#_Toc101385003)

[Предварительный анализ данных 12](#_Toc101385004)

[Анализ собственного датасета Drom.ru 12](#_Toc101385005)

[Анализ датасета Стенфордского университета 14](#_Toc101385006)

[Анализ датасета CIFAR-10 15](#_Toc101385007)

[Преобразование данных 16](#_Toc101385008)

[Представление изображений в матричном виде 17](#_Toc101385009)

[Уменьшение размеров изображения 18](#_Toc101385010)

[Перевод изображений в черно-белый формат 18](#_Toc101385011)

[Нормализация изображений 19](#_Toc101385012)

[IV. СОЗДАНИЕ АЛГОРИТМОВ И ОБУЧЕНИЕ МОДЕЛЕЙ 21](#_Toc101385013)

[Метод опорных векторов 21](#_Toc101385014)

[Метод Байеса 22](#_Toc101385015)

[Метод k-ближайших соседей 24](#_Toc101385016)

[Логистическая регрессия 25](#_Toc101385017)

[Метод «Cлучайного леса» 25](#_Toc101385018)

[Линейный классификатор с SGD-обучением 27](#_Toc101385019)

[V. СРАВНЕНИЕ МОДЕЛЕЙ 29](#_Toc101385020)

[V. ЗАКЛЮЧЕНИЕ 30](#_Toc101385021)

[X. СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ 31](#_Toc101385022)

[XI. ПРИЛОЖЕНИЯ 33](#_Toc101385023)

[Приложение A. Диаграмма классов разработанного приложения-сервера с учётом иерархии модулей 33](#_Toc101385024)

# I. ВВЕДЕНИЕ

На сегодняшний день методы машинного обучения используются во всех областях, где необходимо предсказать, классифицировать или обобщить данные на основе каких-либо критериев: распознавание речи, обработка естественных языков, медицинская диагностика, интеллектуальные игры, техническая диагностика, но большим и наиболее ресурсозатратным направлением является компьютерное зрение, которое в том числе включает в себя задачи многоклассовой классификации объектов на видеозаписях и фотографиях.

Целью данной работы является сравнение классических методов машинного обучения (машина опорных векторов SVM, наивный метод Байеса, метод k-ближайших соседей, логистическая регрессия, случайный лес, линейный классификатор с стохастическим градиентным спуском SGD) с глубоким машинным обучением (классические нейронные сети ANN, сверточные нейронные сети CNN) при работе с изображениями. Все классические алгоритмы анализируются сразу на нескольких датасетах (собственный, CIFAR-10, машины Стенфордского университета) в целях уменьшения влияния конкретных особенностей каждого из них на общий результат.

Описаны наиболее популярные и актуальные механизмы для сбора данных: с помощью классической эмуляции веб-сайта и мобильного приложения.

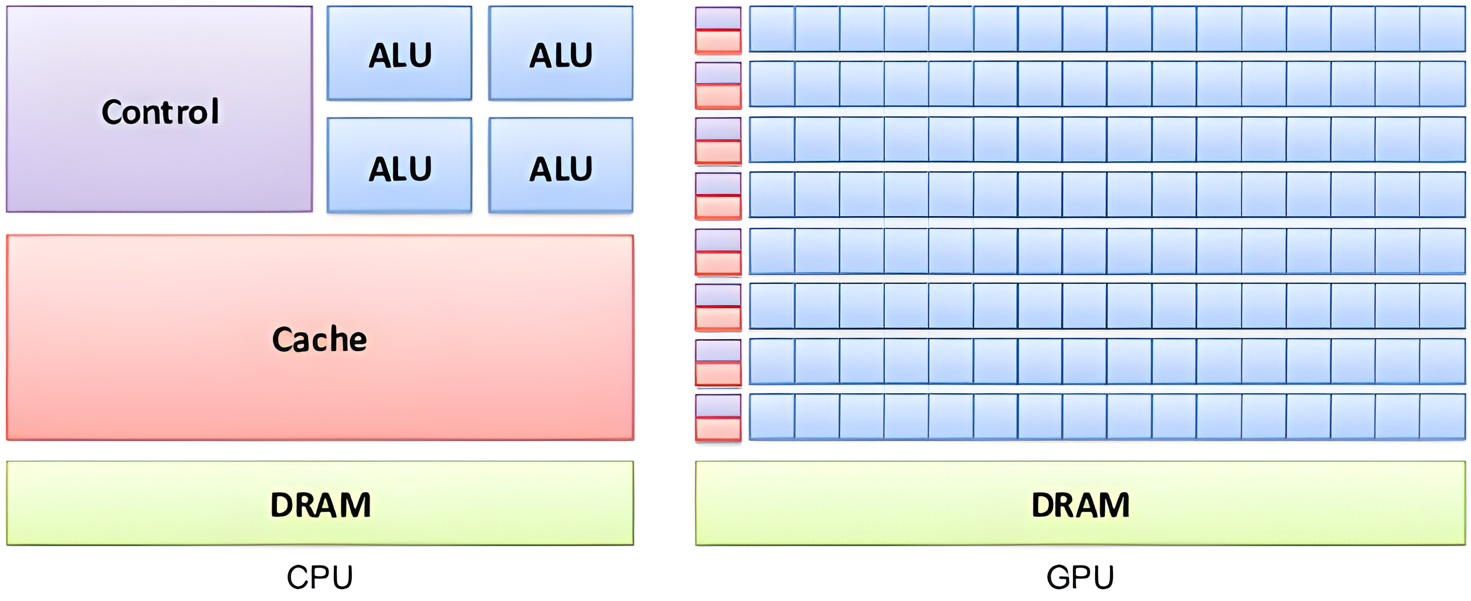
Большой акцент сделан на анализе и предобработке исходных изображений для их последующего использования при обучении в т.ч. решение проблемы разбалансировки количества элементов классов в исходных датасетах.

В качестве командного интерпертатора используется Python с интерактивными ноутбуками Jupyter Notebook. Для работы с классическими алгоритмами машинного обучения применяется библиотека scikit-learn, для глубоких – Keras.

# II. ОСНОВНЫЕ ПОНЯТИЯ ПРИ КЛАССИФИКАЦИИ ИЗОБРАЖЕНИЙ

## Особенности обработки изображений

Отличительной чертой при обработке изображений является многофакторность – для изображения пикселей в RGB формате существует входных аргумента. Именно из-за этой особенности все операции для машинного обучения изображений обычно осуществляются не на CPU, где акцент смещен на оперативность обработки данных, а на GPU, где сделан больший акцент на массовый параллелизм задач по обработке изображений ввиду архитектуры (изображение 1).



Для работы с GPU разных производителей есть несколько наиболее популярных программно-аппаратных архитектур параллельных вычислений:

* CUDA – решение от NVIDIA, позволяющая производить вычисления на только на видеокартах данного производителя. Проприетарная разработка, свой собственный компилятор, можно предкомпилировать под разные версии API. Существует несколько версий API для работы с разными поколениями видеокарт. На macOS поддержка прекращена.
* OpenCL – открытый фрейворк для параллельной обработки данных на видеоускорителях. Поддерживаются дискретные видеокарты от AMD, NVIDIA, а также встроенные в Intel, AMD и Apple. Возможности предкомпиляции нет
* Metal – низкоуровневое API от Apple для работы на системах собственного производства. Предоставляет функции, аналогичные связке OpenCL + OpenGL в одном API, при этом производительность ввиду хорошей оптимизации на устройствах выше.

Из особенностей мы получаем сразу несколько следствий:

1. Т.к. библиотека scikit-learn обычно используется либо как препроцессор для библиотек глубокого обучения (pytorch, keras), либо как набор классических алгоритмов и не поддерживает API для работы с GPU, то обучение на столь большом кол-ве факторов может длится от нескольких часов до нескольких дней, причем некоторые алгоритмы даже невозможно распараллелить по нескольким ядрам процессора.
2. Необходимо тщательно фильтровать исходные датасеты для получения хоть сколько-нибудь существенного процента распознавания данных на классических алгоритмах, причем использовать достаточно очевидные и разноплановые классы (например, дерево, птица, автомобиль). Например, модели автомобилей классифицировать практически невозможно т.к. существует достаточно большое кол-во общих элементов и их очертаний (колёса, решетка радиатора, фары и т.д.)

## Метрики качества

Нам нужны некоторые метрики, чтобы оценить модель и понять, насколько точно она определяет тон текста. Рассмотрим пример распределения данных на два класса - "Положительный" и "Отрицательный". Для простоты определим TP (True Positive) и TN (True Negative) как те данные, которые были правильно отнесены к своим классам, а FP (False Positive) и FN (False Negative) - наоборот, те, которые были отнесены к неправильному классу.

Начнём с доли ответов, которые были правильно определены алгоритмом, которое вычисляется как отношение суммы корректно классифицированных данных в каждом классе ко всем элементам выборки. Данную метрику называют accuracy, и она применяется только в задачах, где классы имеют равный размер, в противном случае она не несёт особой информативности.

(1)

Чаще всего чтобы оценить качество классификации текстовых данных, применяют метрику точности precision и полноты recall.

Точность представляет собой отношение количества элементов, правильно отнесённых к корректному классу, к общему количеству элементов, которые наша модель отнесла к этому классу.

Точность представляет собой отношение количества элементов, правильно отнесённых к корректному классу, к общему количеству элементов, которые наша модель отнесла к этому классу.

(2)

(3)

Тогда как полнота – отношение количества элементов, правильно отнесённых к корректному классу, к общему количеству элементов, относящихся к этому классу.

(4)

(5)

Ещё одна важная метрика – F-мера, являющаяся средним гармоническим между точностью и полнотой. Она даёт возможность сбалансировать и агрегировать эти метрики за счёт введения дополнительный параметр , который определяет вес точности в метрике.

(6)

Отметим, что F-мера будет близка к нулю в том случае, когда один или оба аргумента близки к нулю.

# III СБОР И ПРЕПРОЦЕССИНГ ДАННЫХ

## Сбор данных

Точность классификации моделей при использовании классических алгоритмов напрямую зависит от качества исходных датасетов изображений и классов. Для подтверждения данных суждений возникает необходимость создания нескольких датасетов разных классов и качества:

* Собственноручно собранный датасет изображений реальных автомобилей с сайта по продаже автомобилей Drom.ru. Содержит 34747 нефильтрованных изображений, от 910 × 411 до 1200 × 2667 пикселей, 90 классов фирм-производителей.
* Готовый датасет машин Стэнфордского университета. Содержит 16185 изображения 196 моделей машин. По умолчанию предлагается разделение данных 50% на 50% (8144 обучающих и 8041 тестовых изображений)
* Готовый датасет объектов CIFAR-10. Состоит из 60000 цветных изображений 32x32 в 10 классах, по 6000 изображений на класс. По умолчанию разделяется на 50000 обучающих и 10000 тестовых изображений. 10 классов абсолютно разного характера (самолет, автомобиль, птица, кот, олень, собака, лягушка, лошадь, корабль, грузовик)

Наиболее интересен способ сбора собственноручного датасета с Drom.ru. Обычно, если есть необходимость сбора каких-либо данных, то идут двумя самыми распространенными путями:

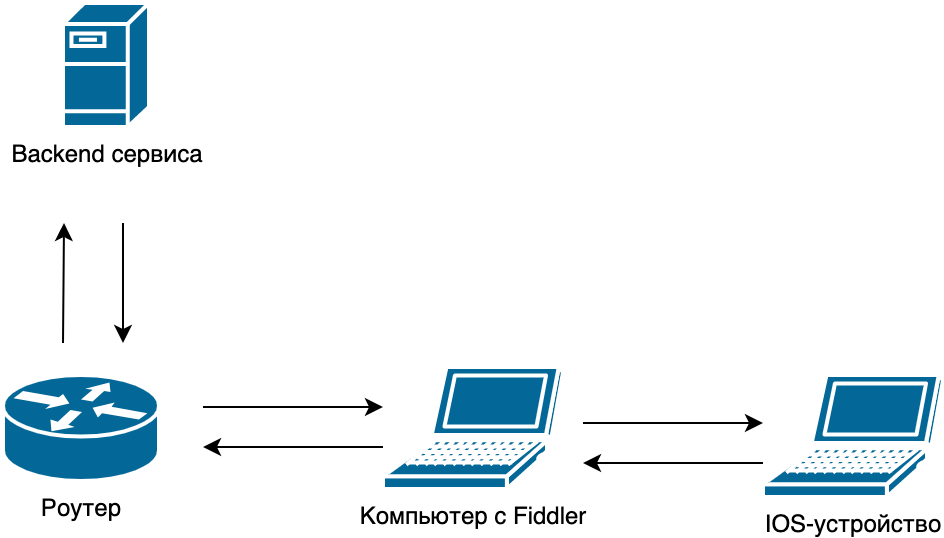
* Парсинг сайта. На стеке Python типично используется связка Chromedriver + Selentium + BeautifulSoup, если сайт генерирует HTML страницу на стороне клиента и Requests + BeautifulSoup, если страничка не использует javascript. Здесь есть пару проблем, из-за которых работа может очень сильно осложниться. Во-первых, это капча. После примерно первых 100 просмотренных объявлений начнет срабатывать механизм защиты от роботов и программ-парсеров. Конечно же, есть возможность использовать сервисы по для её обхода по типу antigate или OpenCV (при условии, что капча не очень сложная), но это существенно замедляет процесс разработки и искусственно расширяет исходную задачу по сбору данных. Во-вторых, при изменении дизайна страницы сайта возникнет необходимость в переписывании парсера, что требует постоянной поддержки и мониторинга со стороны разработчика.
* Парсинг мобильного приложения. Обычно используется MITM атака посредством самоподписанного HTTPS-сертификата приложения. В этом есть ряд существенных преимуществ: во-первых, современное мобильное приложение общается с бекендом сайта посредством REST или GraphQL запросов, в ответ присылается XML или JSON с данными, что очень удобно для последующей обработки через какой-либо язык программирования (в т.ч. Python). Во-вторых, API мобильного приложения меняется очень редко, а если и меняется, то старое, как правило, работает еще некоторое количество времени для того, чтоб пользователи на старых версиях операционных систем имели возможность использовать сервис. В-третьих, мобильное приложение реже вызывает механизмы проверки пользователя на программу-парсер т.к. возможно такой метод сбора данных все ещё не очевиден.

Очевидно, что был выбран способ сбора датасета через мобильное приложение. Возникает вопрос: на какой платформе орагнизовать сбор данных? IOS или Android?

* Android – присутствует возможность установки сторонних программ с помощью .apk файлов, их перекомпиляция и доработка. Возможна организация Root доступа для расширения функционала.
* IOS – по умолчанию закрытая ОС. При осуществлении джейлбрейка есть возможность установки .ipa и сторонних программ (твиков)

С первого взгляда кажется, что выбор очевиден в пользу Android, но все не так очевидно. Android-приложения в больших компаниях, как правило, больше тестируются на безопасность, в т.ч. на подмену сертификатов, а у меня уже был опыт перехвата трафика под IOS, так что исходное устройство будет под управлением IOS.

Для прокси, которое будет использоваться для перехвата трафика использовался Fiddler, а результате схема трафика выглядит так:

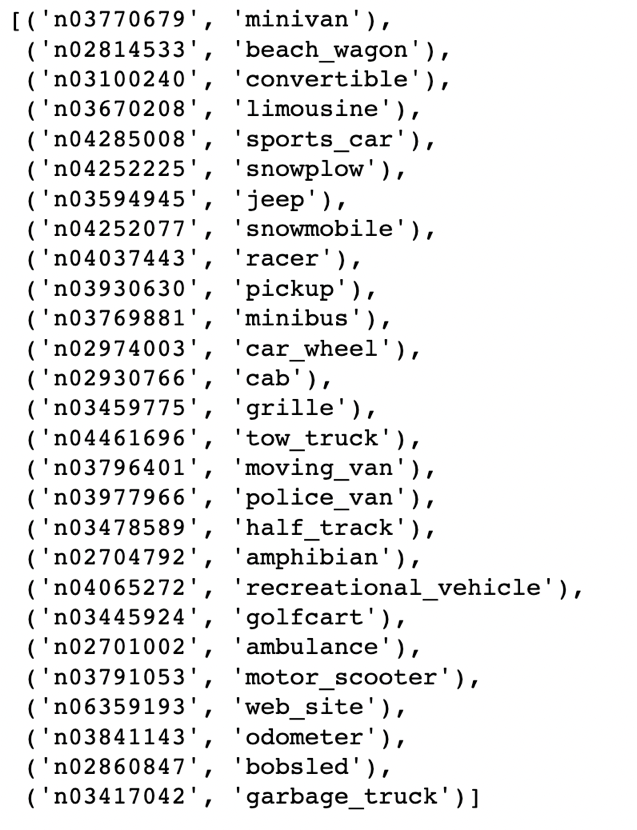


После нескольких часов работы созданного аппаратно-программный комплекса было получено 2241 файла формата JSON, содержащих большое кол-во информации о каждом объявлении, в т.ч. модель, марку автомобиля и url с фотографией автомобиля

## Фильтрация данных датасетов

Т.к. датасеты CIFAR-10 и датасет машин Стэнфорда уже были предварительно подготовлены, то речь в данным разделе пойдет исключительно о фильтрации собственноручно собранного датасета.

1. Все собранные JSON помещаются в один файл в одной папке. Размер собранного файла составляет 171,9 мегабайта и включет 35531 объявлений о продаже автомобиля.
2. Для каждого объявления выкачивается основное фото автомобиля. Дополнительные фото игнорируются, поле URL в JSON меняется на локальный путь к файлу
3. Определяем, что перед нами фото автомобиля. Иногда пользователи публикуют в качестве главного фото не фото самой машины, а приборной панели, вид из салона, документы. Для этого используется сторонняя сверточная сеть глубокого обучения VGG16 с огромным датасетом ImageNet, включающим 14 млн изображений и 1000 классов. Для нас только важно то, что ее точность очень высокая и её результатам можно доверять. Она используется из-за того, что перебирать вручную 35 тысяч изображений достаточно трудозатратная задача. Для всех изображений мы формируем классы, которые относятся к ним и считаем самые распространенные. Изображения, классы которого которые относятся к первым 27 популярным классам, изображают машину

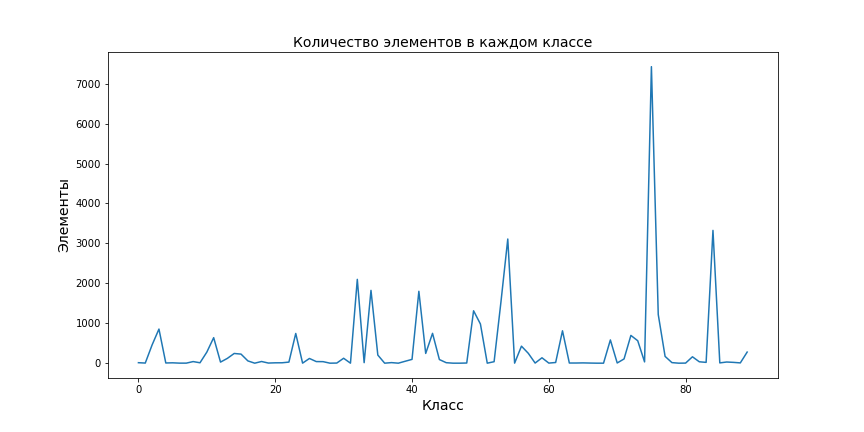


Таким образом было отфильтровано 784 записей и 34747 значений осталось. Использование моделей глубокого обучения для формирования новых датасетов вполне себе обычная практика на сегодняшний день.

## Предварительный анализ данных

### Анализ собственного датасета Drom.ru

Датасет содержит 34747 изображения автомобилей с сервиса объявлений Drom.ru. Изображения распределены по 90 классам. Для начала необходимо посмотреть сбалансированность классов между собой.



Как видно из изображения, на данном этапе данные совсем не подходят для обучения. Разброс огромный: от 1 фото для многих редких моделей (Диаграмма N) до нескольких тысяч для наиболее популярных автопроизводителей (Диаграмма N)

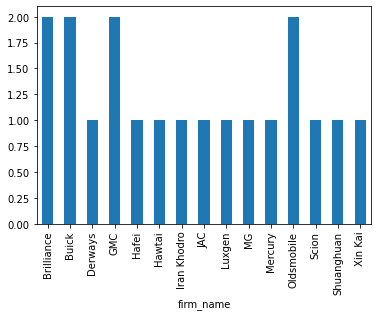


Диаграмма N. Наиболее редкие автомобили

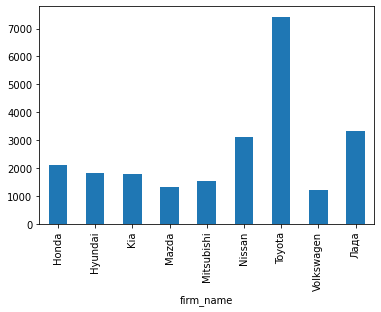


Диаграмма N. Наиболее популярные автомобили

Возьмем наиболее популярных автопроизводителей и сделаем так, чтоб в каждом классе было одинаковое кол-во изображений, как у самого малого класса (1220, как у Volkswagen), остальные просто исключаем из выборки.

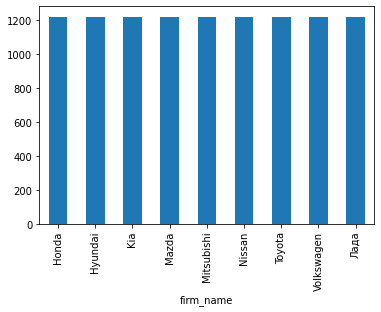
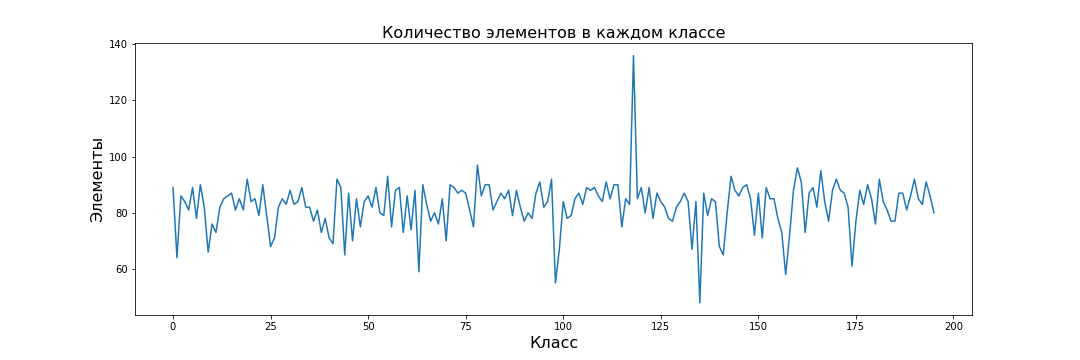


Диаграмма N. Наиболее популярные автомобили

В итоге получилось 9 классов, в каждом по 1220 изображений, всего 10980. Такая модель является сбалансированной и с ней возможно производить дальнейшие действия.

### Анализ датасета Стенфордского университета

Данный набор данных содержит 16185 изображений автомобилей в 196 классах. Аналогично предыдущей выборке, строим график по количеству элементов в каждом классе для разбалансировки классов.



С помощью диаграммы размаха ящик с усами определяем медиану, равную 84.



Диаграмма N. Размах баланса классов для датасета Стенфордского университета

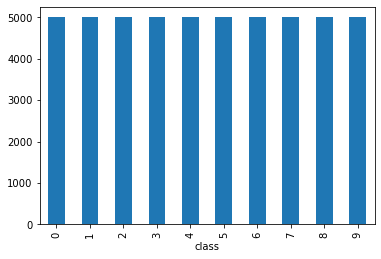
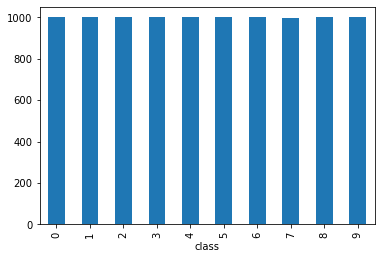
Для балансировки выборки оставим классы классы, в которых количество элементов не меньше медианы. Классы с большим количеством элементов уменьшаем путем отсечения лишних элементов до медианы.



После преобразований получилось N классов, по NNN в каждом. Всего 99999 изображений, что все щее достаточно хорошая выборка.

### Анализ датасета CIFAR-10

Все данные в датасете идеально сбалансированы и уже поделены на тестовую и обучающую выборку. В тестовой выборке 10 классов по 1000 изображений, всего 10000. В обучающей выборке 10 классов по 5000 изображений, всего 50000.

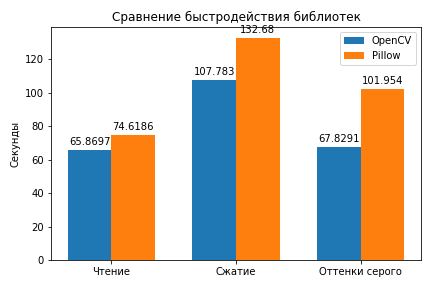


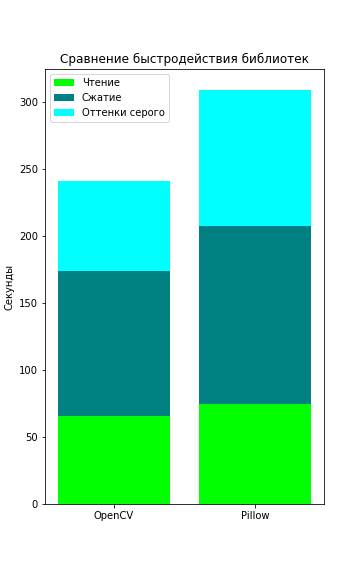
## Преобразование данных

Наиболее распространены две библиотеки для решения задач по предобработке изображений:

* Pillow – свободно распространяемая библиотека для работы с изображениями на Python с открытым исходным кодом, предоставляет ряд алгоритмов для работы, включая функции для загрузки изображений из файлов и для создания новых изображений. Разработана на Python и C
* OpenCV — библиотека компьютерного зрения c открытым исходным кодом, предназначена для анализа, классификации и обработки изображений. Написана на языке программирования C++, есть обёртки для Python и Java.

Сравним их по скорости в алгоритмах чтения, сжатия и перевода в оттенки серого на датасете автомобилей Стэнфордского университета.

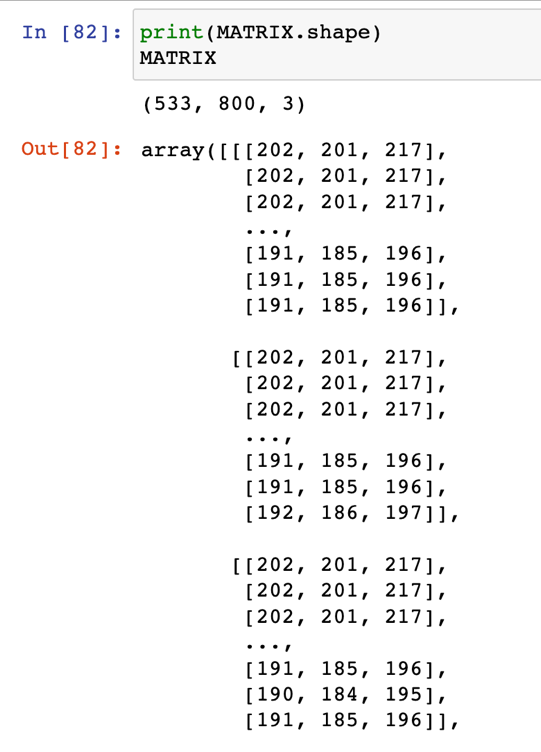




Как видим из графиков, библиотека OpenCV намного быстрее работает с необходимыми нам алгоритмами, в особенности с преобразованием изображения в черно-белый формат. Т.к. производительность при большом количестве данных является приоритетом, то в нашем случае выбор падает на OpenCV. Рассмотрим каждое действие над изображением подробнее.

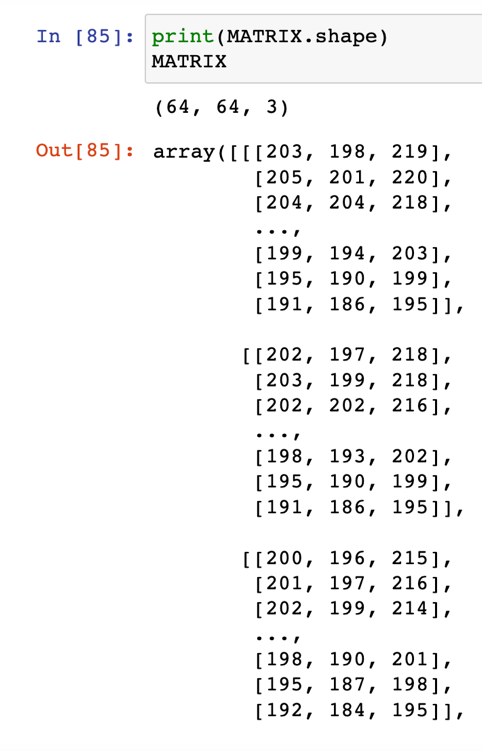
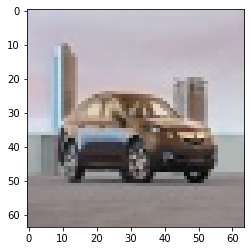
### Представление изображений в матричном виде

Для начала работы необходимо представить изображение RGB в виде матрицы размерностью , где - количество пикселей по вертикали, M – количество пикселей по горизонтали, 3 – количество цветовых каналов. Используем метод OpenCV cv2.imread для чтения изображения и его представления в виде матрицы.



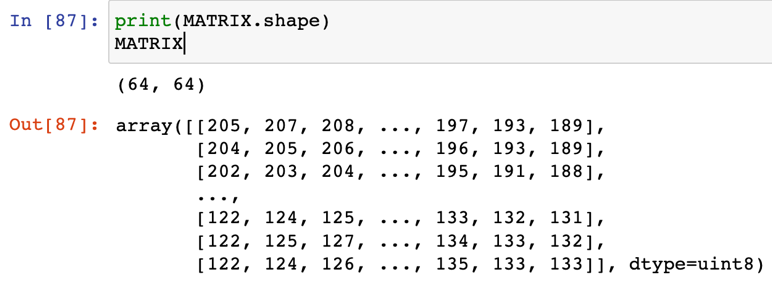
### Уменьшение размеров изображения

Т.к. все исходные изображения разного разрешения и соотношения сторон, то при помощи метода OpenCV cv2.resize происходит уменьшение изображения до 64 пикселя. Очевидно, что это также влияет на производительность решения в лучшую сторону.



### Перевод изображений в черно-белый формат

Для большинства алгоритмов обработки изображений требуется перевод исходных данных в монохромный режим т.е. в преобразование в черно-белый формат. Это необходимо в целях уменьшения каналов изображения с 3 (для RGB) до 1. Как следствие, каждый пиксель на изображении кодируется всего одним значением от 0 до 255.

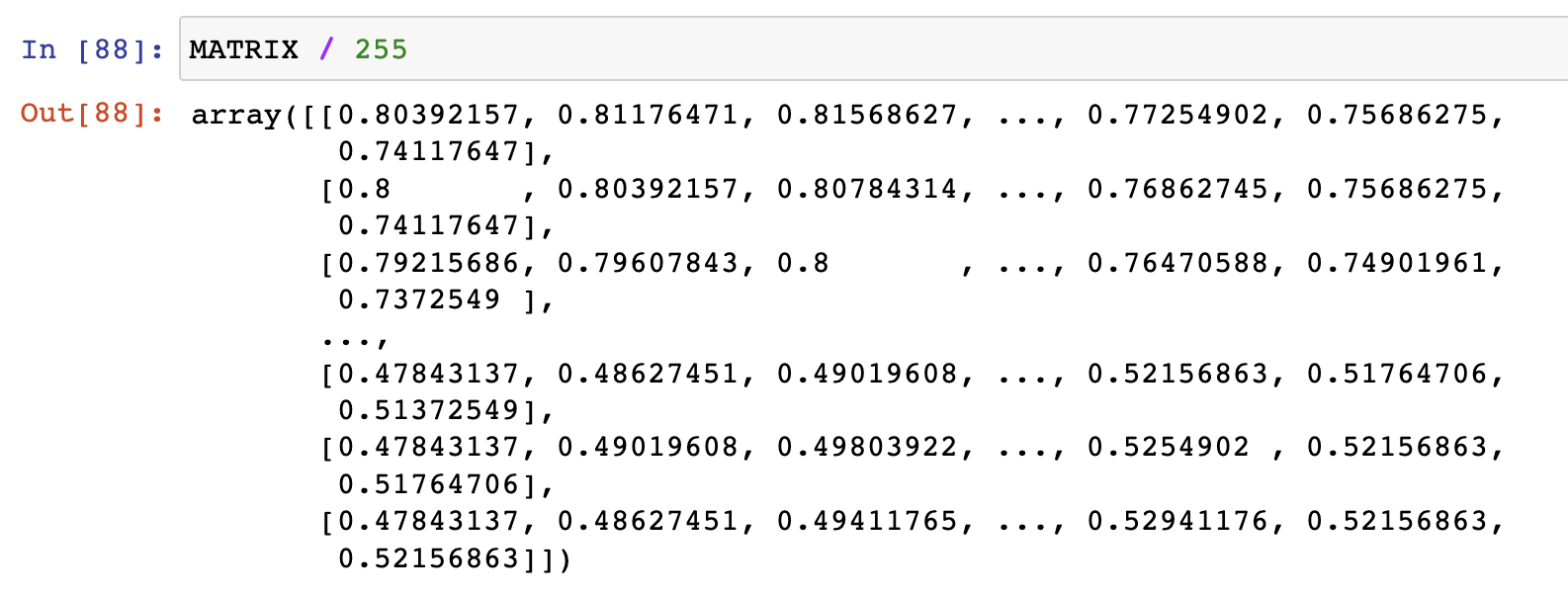


### Нормализация изображений

Для большинства методов машинного обучения требуются нормализованные значения, т.е. значения в диапазоне от 0 до 1. Наиболее распространенной нормализацией изображения является метод максимальной и минимальной нормализации:

где - значение пикселя изображения, и - минимальное и максимальное значение пикселя соответственно.

поэтому каждый элемент делим на 255.



### Разделение выборок

Для собственного датасета и машин Стенфордского университета необходимо разделить выборки на обучающую и проверочную. У каждого датасета достаточно изображений (10980 для собственного, N для Стенфордского), то выборки делим в соотношении 80 20 с помощью метода Sikitlearn. Чета\_там\_раздлеить

Далее с помощью sikitlearn split мы осуществляем разделение выборки на тестируемую и обучающую.

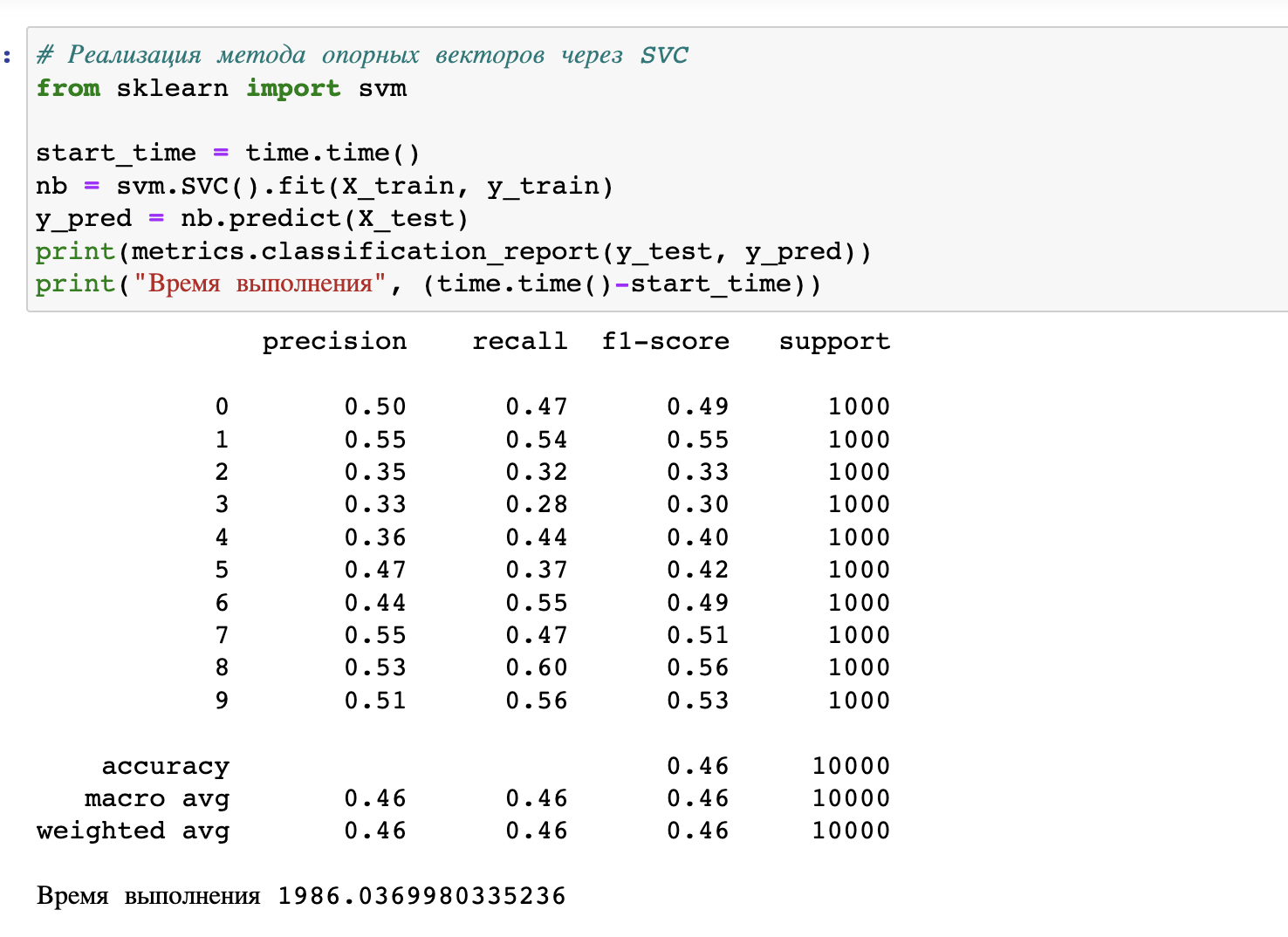
# IV. СОЗДАНИЕ АЛГОРИТМОВ И ОБУЧЕНИЕ МОДЕЛЕЙ

## Машина опорных векторов

Машина опорных векторов SVM является универсальным решением в задачах классификации, наравне с нейронной сетью и бустингом.

Метод опорных векторов обладает особым свойством, заключающимся в том, что он постоянно уменьшает эмпирическую ошибку классификации, увеличивая разрыв, поэтому его также называют методом классификатора с максимальным разрывом.

Ключевая идея заключается в переводе опорных векторов в пространство более высокой размерности и поиске разделяющей гиперплоскости с наибольшим разрывом в ней. По обе стороны от гиперплоскости, разделяющей классы, находятся две параллельные гиперплоскости. Разделяющей гиперплоскостью будет та гиперплоскость, которая создает наибольшее расстояние между параллельными гиперплоскостями. Согласно алгоритму, чем больше разница или расстояние между этими параллельными гиперплоскостями, тем меньше будет средняя ошибка. В sklearn реализация выглядит так:



Как видим из изображения выше, точность классификации составляет 46%, что является отличным показателем относительно других. Время вычисления составило 33 минуты 10 секунд.

## Метод Байеса

Помимо того, что наивный байесовский классификатор является одним из самых простых, он также использует удобную математическую модель и является быстродействующим методом, который часто используется в качестве базового. Кроме того, этот метод обычно дает хорошие результаты в задачах классификации текста.

Алгоритм нередко называют наивным байесовым классификатором из-за того, что в нём приято считать, что каждый фактор вносит свой независимый вклад в эмоциональный окрас текста, что с точки зрения естественного языка не совсем верно. Данный момент всё равно сохраняет высокую точность алгоритма, зато при независимых возможно преобразовать исходную формулу

(9)



Метод показал относительно невысокие результаты точности в районе 23%, зато время выполнения алгоритма составило менее секунды.

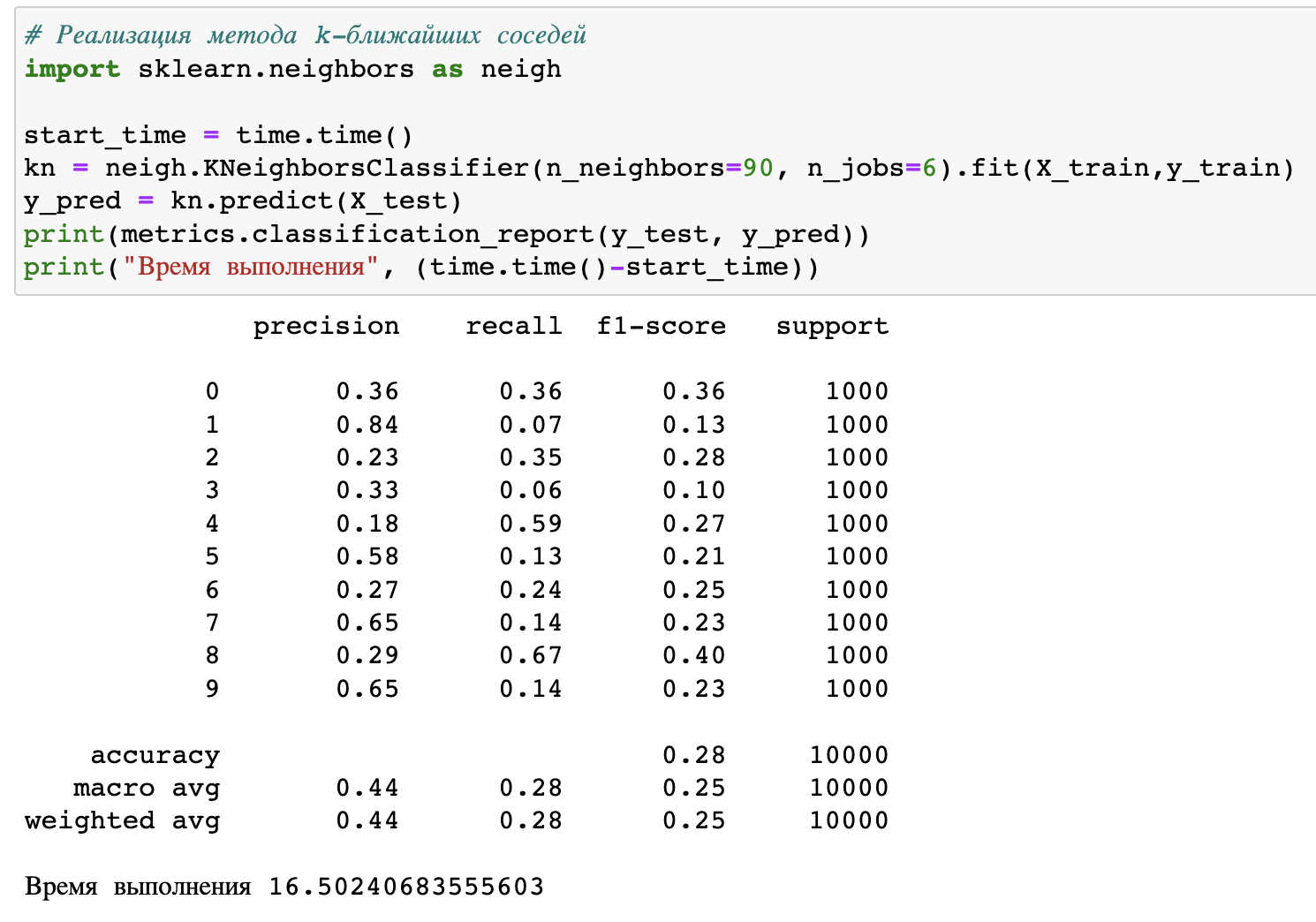
## Метод k-ближайших соседей

В методе ближайших соседей не требуется фаза обучения, так как он производит сравнение интересующего нас объекта со всеми элементами обучающей выборки в результате чего находится cos угла между векторами признаков или по-другому – расстояние:

(10)

Далее определяется параметр k (как правило от 1 до 100) и в обучающей выборке определяют k ближайших элементов к интересующему нас объекту b, вычисляется релевантность и объект относят к тому классу, у которого она выше заданного порога:

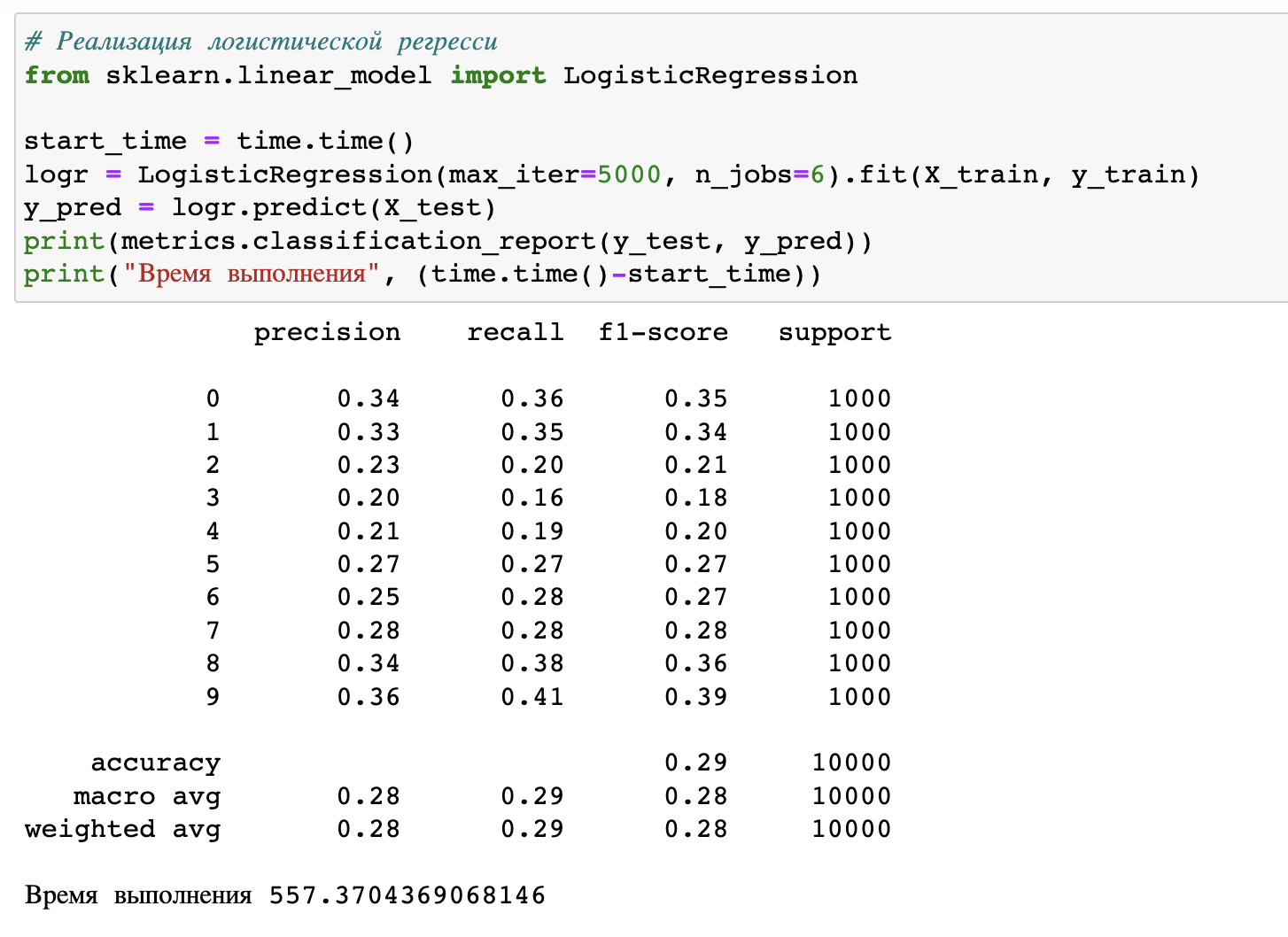
(11)



Метод выполнялся чуть дольше наивного байесовского классификатора: 16 секунд, получил точность чуть более высокую 28% вместо 23% у Байеса.

## Логистическая регрессия

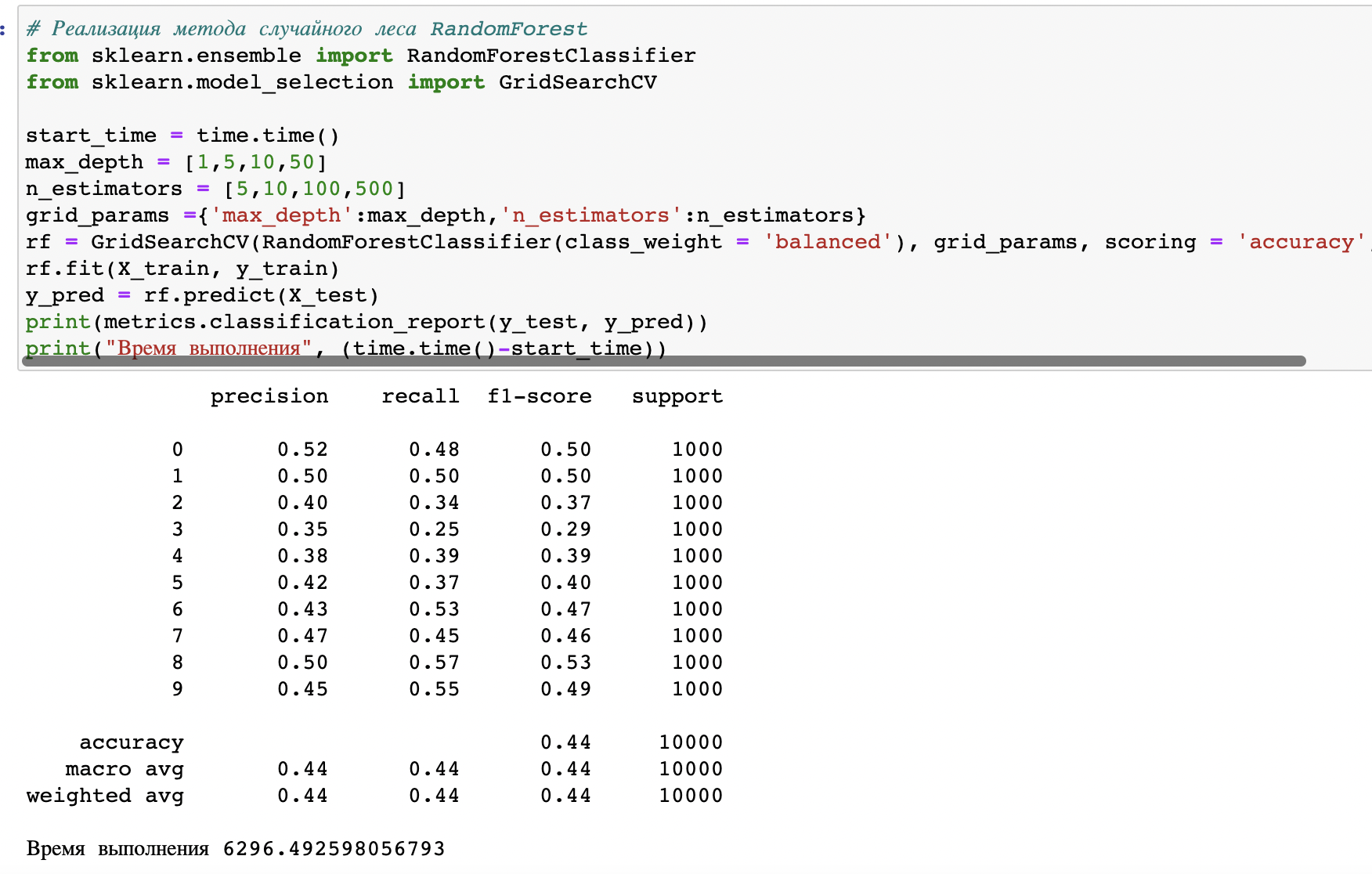
Логистическая регрессия получила своё название по историческим причинам, и данный метод не имеет отношения к регрессии, поскольку является подходом к проблемам классификации. Он основывается на вероятностях принадлежности данных к определённому классу, которые моделируются с помощью логистической функции, являющейся кумулятивной функцией распределения.



Классификатор логистической регрессии также показывает точность, сопоставимую с методом k-ближайших соседей – 29%, при этом вычисления производятся в 34 раза медленнее – 9 минут 17 секунд.

## Метод «Cлучайного леса»

«Random forest» - метод, который был придуман Лео Брейманом и Адель Катлер ещё в XX веке и отличается своей универсальностью как по качеству работы (средне значение из разных источников – около 70%), так и по спектру выполняемых задач (регрессия, классификация, кластеризация, селекция признаков поиск аномалий и так далее). Случайный лес – это множество решающих деревьев. Если мы работаем с задачей классификации, то решение будет принято голосованием по большинству в отличии от задач регрессии, где ответы усредняются. Деревья являются независимыми и строятся по следующему алгоритму. Из обучающей выборки выбирается специально для каждого дерева своя подвыборка, по которой оно и строится. Далее строится расщепление в дереве, для чего просматривается определённое количество случайный признаком, причём для каждого расщепления свои признаки. Затем, используя заранее заданный критерий, определяются наилучшие признак и расщепление по нему. Построение дерева продолжается то тех пор, пока в листьях не останутся представители только одного класса, то есть до исчерпания выборки, но отметим, что современные реализации алгоритма включают в себя ограничение на число объектов в листья, высоту дерева, а также размер подвыборки, при котором проводится расщепление. Реализуем данный метод:



Обработка матриц данным классификатором занимает наибольшее время из всех предложенных алгоритмов: 1 час, 44 минуты, при этом точность распознавания 44%, что на уровне метода опорных векторов, у которого 46%.

## Линейный классификатор с SGD-обучением

Линейный классификатор представляет собой алгоритм, который базируется на построении линейной разделяющей плоскости, являющейся гиперплоскостью в случае двух классов, а если их больше, то кусочно-линейной. Наши объекты описываются числовыми признаками в количестве n, и получаем пространство признаковых описаний . Обозначим множество номеров\меток\имён за Y. Допустим, Y = {-1, 1}. В таком случае линейный алгоритм классификации будет иметь вид

(15)

В данной формуле – признак, - порог принятия решений, – вес j-ого признака, которые образуют вектор весов , < > - скалярное произведение. Если мы работаем с классификацией для произвольного числа классов, то рассматриваем вектор весов для каждого класса , а формула примет следующий вид:

(16)

Обучать линейный классификатор можно различными способами. Ранее уже были рассмотрены логистическая регрессия и метод опорных векторов, но сейчас речь пойдёт о стохастическом градиентном спуске, который даёт возможность решить проблему требовательности к ресурсам обычного градиентного спуска, ведь при больших наборах обучающей выборки, вычислять функцию ошибки для каждого её элемента довольно затратно. Для этого мы разбиваем выборку на K частей размера M.

(17)

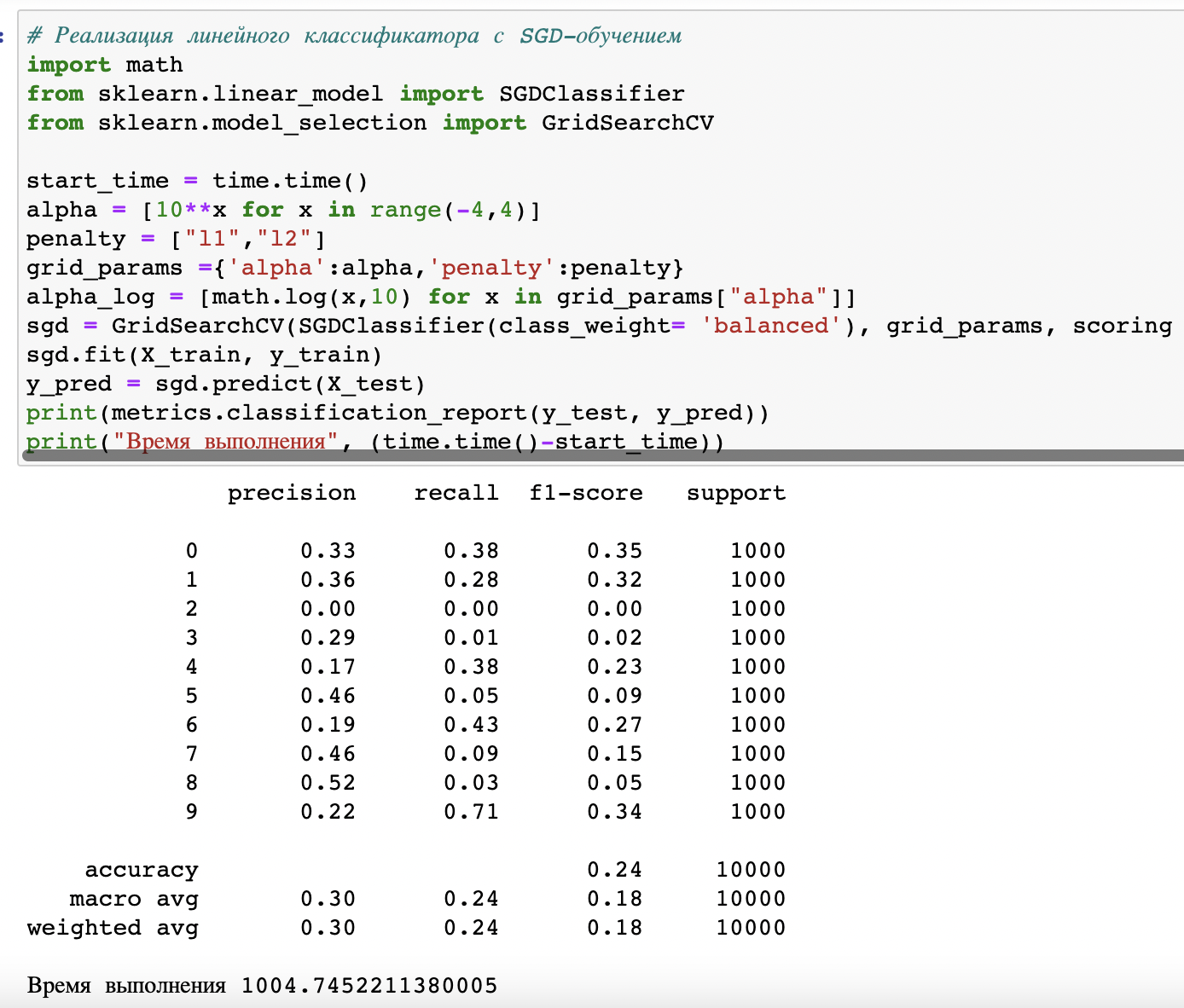
(18)

Получим функцию и набор из K мини итераций градиентного спуска вместо одной:

(19)

(20)

Назовём эпохой каждую большую итерацию p, между которыми обучающая выборка перемешивается для обеспечения попадания каждому элементу в разные мини выборки. Получаем, что наши штрафные функции суммируются по разным поднаборам X и Y.



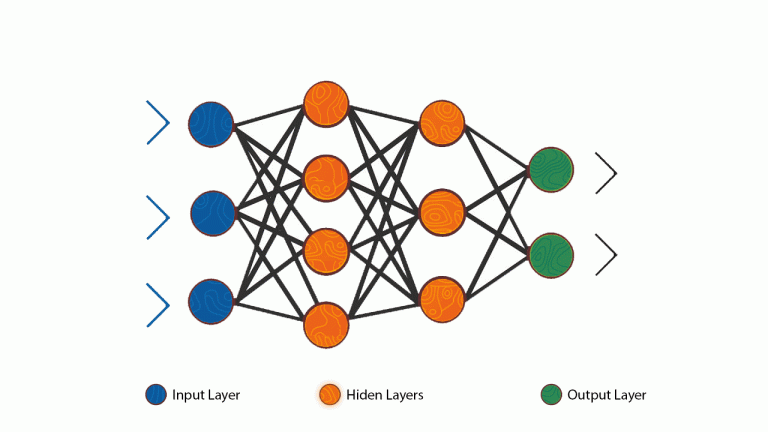
Линейный классификатор с SGD-обучением по точности аналогичен наивному методу Баеса (24% распознавания), но при этом работает 16 минут 44 секунды, что примерно в 1000 раз дольше.

## Нейронные сети с использованием библиотеки Keras

Если переходить к более пкатическому использованию решения, то в жизни использется GPU…

### Классическая нейронная сеть ANN

#### Один перцептрон (или нейрон) можно представить как логистическую регрессию. Искусственная нейронная сеть, или ANN, представляет собой группу из нескольких перцептронов/нейронов на каждом слое. ANN также известна как нейронная сеть с прямой передачей данных, поскольку входные данные обрабатываются только в прямом направлении



ANN состоит из 3 слоев - входного, скрытого и выходного. Входной слой принимает входные данные, скрытый слой обрабатывает входные данные, а выходной слой выдает результат. По сути, каждый слой пытается выучить определенные веса.

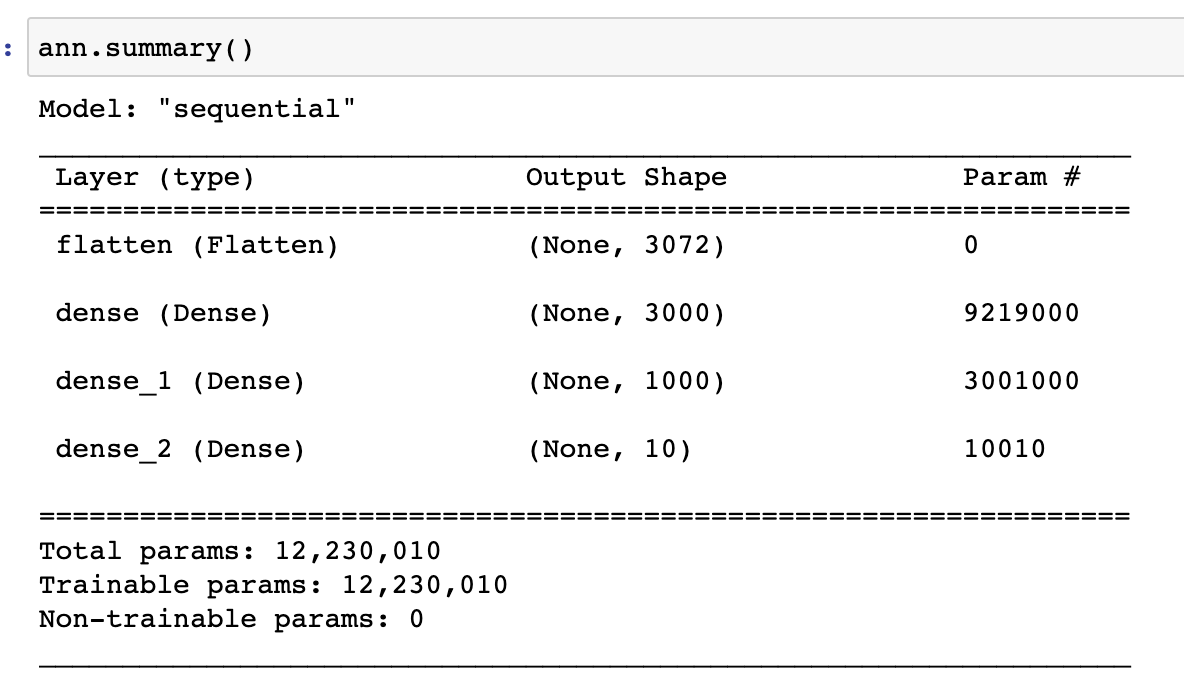
Искусственная нейронная сеть способна обучать любую нелинейную функцию. Поэтому эти сети известны как универсальные аппроксиматоры функций. ИНС обладают способностью к обучению весов, которые отображают любой вход на выход.

Одной из основных причин универсальной аппроксимации является функция активации. Функции активации придают сети нелинейные свойства. Это помогает сети изучать любые сложные отношения между входом и выходом.

С помощью библиотеки Keras была спроектирована сеть следующего вида:







Данная нейронная сеть имеет следующие типы слоёв:

Входной слой Flatten. Flatten используется для конвертации входящих данных в меньшую размерность.

Dense слой. Dense-слой получает информацию со всех узлов предыдущего слоя. Вычисляет вывод следующим образом: output=activation(dot(input,kernel)+bias), где activation — это активатор, а kernel — взвешенная матрица, применяемая к входящим тензорам. bias — это константа, помогающая настроить модель наилучшим образом.

Сами слои выстроены следующим образом:

Flatten. Входящий слой размерности (32\*32\*3) «выравнивается» для вывода размерности (3072).

Dense 3000 activation=relu

Dense 1000 activation relu

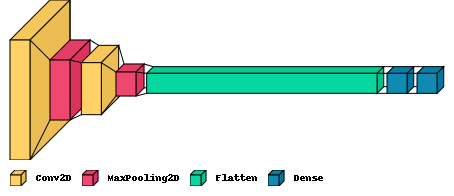
Dense 10 activation = softmax – 10 т.к. 10 выходных класов

#### Сверточная нейронная сеть CNN

Сверточная нейронная сеть CNN – это одна из видов искусственных нейронных сетей. Нацеленная на эффективное распознавание образов, входит в состав технологий глубокого обучения и наиболее предпочтительна на сегодняшний день. В отличии от ANN, не требует большого количества эпох и уже при 10 эпохах способна выдавать результаты, аналогичные 200 эпохам в ANN или даже выше.

CNN улавливает пространственные особенности изображения. Пространственные особенности относятся к расположению пикселей и взаимосвязи между ними на изображении. Они помогают точно идентифицировать объект, определить его местоположение, а также его связь с другими объектами на изображении. CNN также следует концепции разделения параметров. Один фильтр применяется к различным частям входного сигнала для создания карты характеристик.

В Keras была спроектирована сеть со следующими слоями:



# V. СРАВНЕНИЕ МОДЕЛЕЙ

В ходе испытаний были получены следующие данные по классическим алгоритмам на всех 3 наборах данных:

# V. ЗАКЛЮЧЕНИЕ

# X. СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ

Законодательные и нормативные акты

1. ГОСТ 2.316-2008. Правила нанесения надписей, технических требований и таблиц на графических документах.
2. ГОСТ 7.1-2003. Библиографическая запись. Библиографическое описание. Общие требования и правила составления. – М.: ИПК Издательство стандартов, 2004. – 169 с.
3. ГОСТ 7.32-2001. Система стандартов по информацию, библиотечному и издательскому делу. Отчет о научно-исследовательской работе. Структура и правила оформления. – М.: ИПК Издательство стандартов, 2001. – 21 с.

Ссылки

<https://ai.stanford.edu/~jkrause/cars/car_dataset.html> - датасет машин Стэнфордского униерситета

<https://www.cs.toronto.edu/~kriz/cifar.html> - датасет CIFAR-10

Учебная и научная литература

1. Володченкова Л.А., Козырев Д.В. Разработка серверной части программного приложения для удаленного хранения данных // МСиМ. 2020. №1 (53).
2. Байдыбеков А.А., Гильванов Р.Г., Молодкин И.А. СОВРЕМЕННЫЕ ФРЕЙМВОРКИ ДЛЯ РАЗРАБОТКИ WEB-ПРИЛОЖЕНИЙ // Интеллектуальные технологии на транспорте. 2020. №4 (24).
3. Гасанов Заурбек Зубаирович Анализ производительности многопоточных программ, написанных на языках Java и Go // Наука и образование сегодня. 2018. №6 (29).
4. Барабанов В.Ф., Донских А.К., Гребенникова Н.И., Кенин С.Л. ПОЛУЧЕНИЕ МЕТРИК JAVA-ПРИЛОЖЕНИЯ В КОНТЕЙНЕРАХ DOCKER // Вестник ВГТУ. 2020. №2.
5. О.А. Ляшенко, О.О. Конашков, Н.А. Солодкая СРАВНИТЕЛЬНЫЙ АНАЛИЗ ВЫПОЛНЕНИЯ ЗАПРОСОВ К СЕРВЕРАМ БАЗ ДАННЫХ MYSQL И MONGODB // Вестник Херсонского национального технического университета. 2019. №4 (71).

# XI. ПРИЛОЖЕНИЯ

## Приложение A. Диаграмма классов разработанного приложения-сервера с учётом иерархии модулей

Тут приложение хехех

Точность классификации сильно зависит от исходных данных, в то числа от методов разбалансировки классов