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 4](#_Toc101719202)

[II. ОСНОВНЫЕ ПОНЯТИЯ ПРИ КЛАССИФИКАЦИИ ИЗОБРАЖЕНИЙ 5](#_Toc101719203)

[Понятие классического алгоритма машинного обучения 5](#_Toc101719204)

[Особенности обработки изображений 6](#_Toc101719205)

[Метрики качества 8](#_Toc101719206)

[III СБОР И ПРЕПРОЦЕССИНГ ДАННЫХ 11](#_Toc101719207)

[Сбор данных 11](#_Toc101719208)

[Фильтрация данных датасетов 13](#_Toc101719209)

[Предварительный анализ данных 15](#_Toc101719210)

[Анализ собственного датасета Drom.ru 15](#_Toc101719211)

[Анализ датасета Стэнфордского университета 17](#_Toc101719212)

[Анализ датасета CIFAR-10 19](#_Toc101719213)

[Преобразование данных 20](#_Toc101719214)

[Представление изображений в матричном виде 21](#_Toc101719215)

[Уменьшение размеров изображения 22](#_Toc101719216)

[Перевод изображений в черно-белый формат 22](#_Toc101719217)

[Нормализация изображений 23](#_Toc101719218)

[Разделение выборок 24](#_Toc101719219)

[IV. СОЗДАНИЕ АЛГОРИТМОВ И ОБУЧЕНИЕ МОДЕЛЕЙ 25](#_Toc101719220)

[Алгоритмы машинного обучения 25](#_Toc101719221)

[Машина опорных векторов 25](#_Toc101719222)

[Метод Байеса 26](#_Toc101719223)

[Метод k-ближайших соседей 27](#_Toc101719224)

[Логистическая регрессия 29](#_Toc101719225)

[Метод «Cлучайного леса» 30](#_Toc101719226)

[Линейный классификатор с SGD-обучением 32](#_Toc101719227)

[Алгоритмы глубокого обучения 34](#_Toc101719228)

[Нейронная сеть прямого распространения ANN 34](#_Toc101719229)

[Сверточная нейронная сеть CNN 38](#_Toc101719230)

[V. СРАВНЕНИЕ МОДЕЛЕЙ 41](#_Toc101719231)

[V. ЗАКЛЮЧЕНИЕ 44](#_Toc101719232)

[X. СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ 45](#_Toc101719233)

[XI. ПРИЛОЖЕНИЯ 46](#_Toc101719234)

[Приложение A. Диаграмма классов разработанного приложения-сервера с учётом иерархии модулей 46](#_Toc101719235)

# 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 (ANN, CNN) при работе с изображениями. Все классические алгоритмы анализируются сразу на нескольких датасетах (собственный, CIFAR-10, машины Стэнфордского университета) в целях уменьшения влияния конкретных особенностей каждого из них на общий результат.

Описаны наиболее популярные и актуальные механизмы для сбора данных: с помощью парсинга веб-сайта и анализа трафика мобильного приложения.

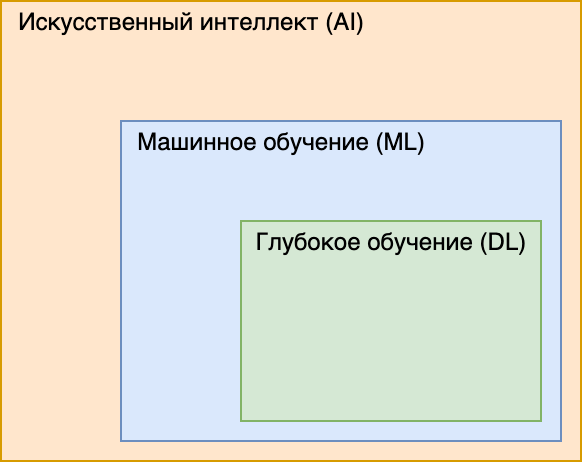
Большой акцент сделан на анализе и предобработке исходных изображений для их последующего использования при обучении в т.ч. решение проблемы разбалансировки количества элементов классов в исходных датасетах.

В качестве командного интерпертатора используется Python с интерактивными ноутбуками Jupyter Notebook. Для препроцессинга задействуется библиотека компьютерного зрения OpenCV. При работе с классическими алгоритмами машинного обучения применяется библиотека scikit-learn, с глубокими – Keras.

# II. ОСНОВНЫЕ ПОНЯТИЯ ПРИ КЛАССИФИКАЦИИ ИЗОБРАЖЕНИЙ

## Понятие классического алгоритма машинного обучения

Существует некоторая путаница между машинным и глубоким обучением.



Машинное обучение – это подмножество искусственного интеллекта, которое связано с созданием алгоритмов, у которых есть возможность изменения своих свойств без ручного регулирования человеком для получения желаемого результата - путем подачи себя через структурированные данные.

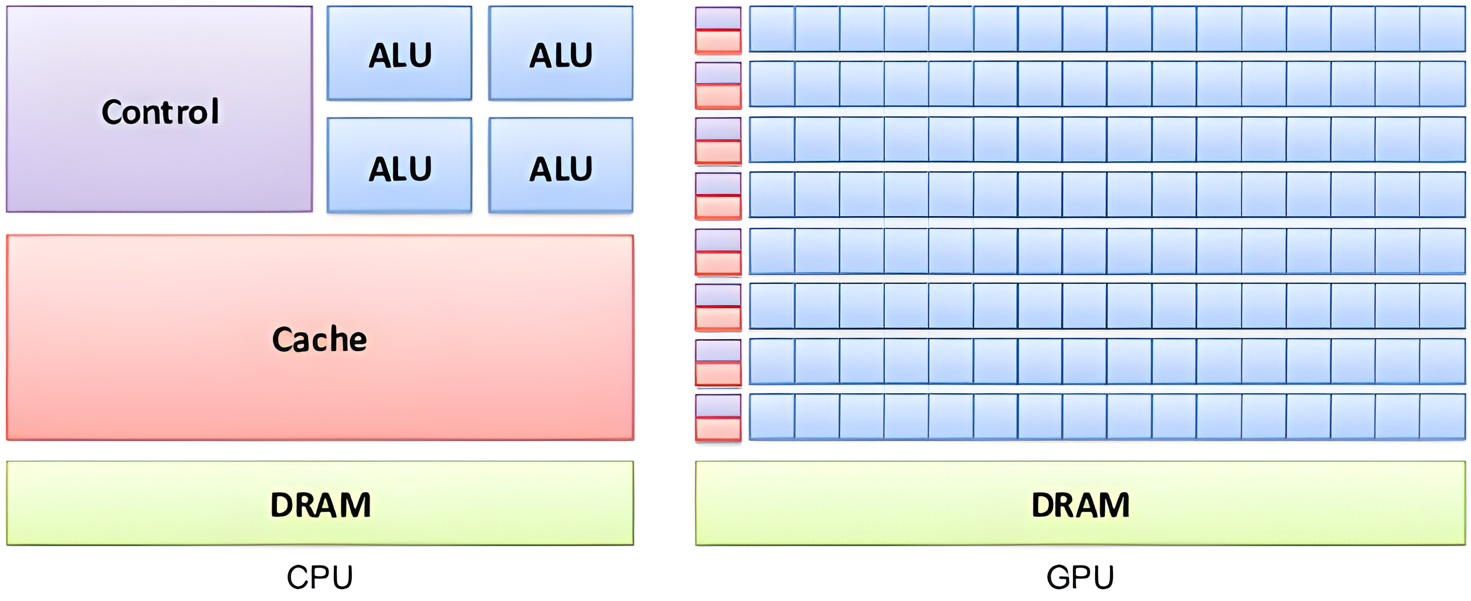
Глубокое обучение – подмножество машинного обучения, где есть несколько уровней этих алгоритмов, каждый из которых обеспечивает различную интерпретацию данных, которые он передает. Такая модель алгоритмов называется искусственными нейронными сетями (ИНС).

Под понятием классического алгоритма машинного обучения понимаются такие, которые не связаны с глубоким обучением, но используются в машинном обучении. Их использование будет осуществляться с помощью библиотеки scikit-learn с использованием только ресурсов процессора (CPU)

Под понятием алгоритма глубокого обучения понимаются искусственные нейронные сети, которые входят только в глубокое обучение. Их создание будет происходить при помощи библиотеки Keras и использовать ресурсы видеокарты (GPU), что существенно увеличит скорость обучения моделей.

## 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:

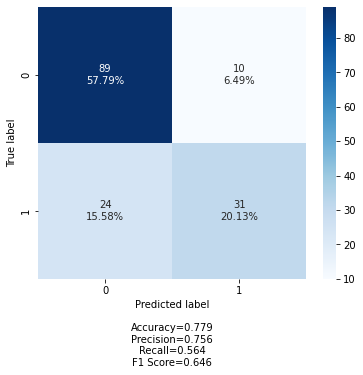
* CUDA – решение от NVIDIA, позволяющая производить вычисления на только на видеокартах данного производителя. Проприетарная разработка, свой собственный компилятор, можно предкомпилировать под разные версии API. Существует несколько версий API для работы с разными поколениями видеокарт. На macOS поддержка 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 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 не-глубоких алгоритмах машинного обучения, причем использовать достаточно очевидные и разноплановые классы (например, дерево, птица, автомобиль). Например, модели автомобилей классифицировать с высокой точностью (85% и более) практически невозможно так как существует достаточно большое кол-во общих элементов и их очертаний (колёса, решетка радиатора, фары и т.д.)

## Метрики качества

Для подсчета точности распределения распознанных объектов на фотографии в нашем классификаторе необходимо ввести ряд метрик. Все метрики будут завесить от значений в матрице путаницы. Для упрощения рассмотрим матрицу путаницы для задачи бинарной классификации, где есть только два класса, в таком случае матрица путаницы представляет собой матрицу размерностью с целыми числами.



Рассмотрим каждый элемент подробнее:

TN (True Negative) - классификатор верно утверждает, что объект не принадлежит к рассматриваемому классу.

TP (True Positive) - классификатор верно отнёс объект к рассматриваемому классу.

FN (False Negative) - классификатор неверно утверждает, что объект не принадлежит к рассматриваемому классу.

FP (False Positive) - классификатор неверно отнёс объект к рассматриваемому классу.

Наиболее популярной метрикой является accuracy, которая выражает долю ответов, которые были правильно определены системой. Вычисляется как отношение суммы правильно классифицированных объектов (TP и TN) ко всем объектам исходной выборки

(1)

Однако ориентироваться на данную метку можно только в случае, когда классы сбалансированы и имеют равный размер. Если же классы не сбалансированы, то применяется метрика точности precision.

Точность вычисляется как отношение количества объектов, правильно отнесённых к корректному классу на сумму элементов, которое модель отнесла к этому классу.

Т.е. для положительной точности

Для отрицательной точности

Альтернативой метрике precision можно считать метрику полноты recall. Полнота – это отношение количества объектов, правильно отнесённых к корректному классу к сумме объектов, относящихся к этому классу.

Т.е. для положительной полноты

(4)

Для отрицательной полноты

(5)

Для обобщения результатов полноты и точности обычно используется метрика F-меры. Данная метрика вводит коэффициент β, который определяет вес точности в метрике в целях агрегации и балансировки значений точности и полноты.

(6)

Исходя из формулы, если один из коэффициентов близок к нулю, то общее значение F-меры также будет приближено 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:

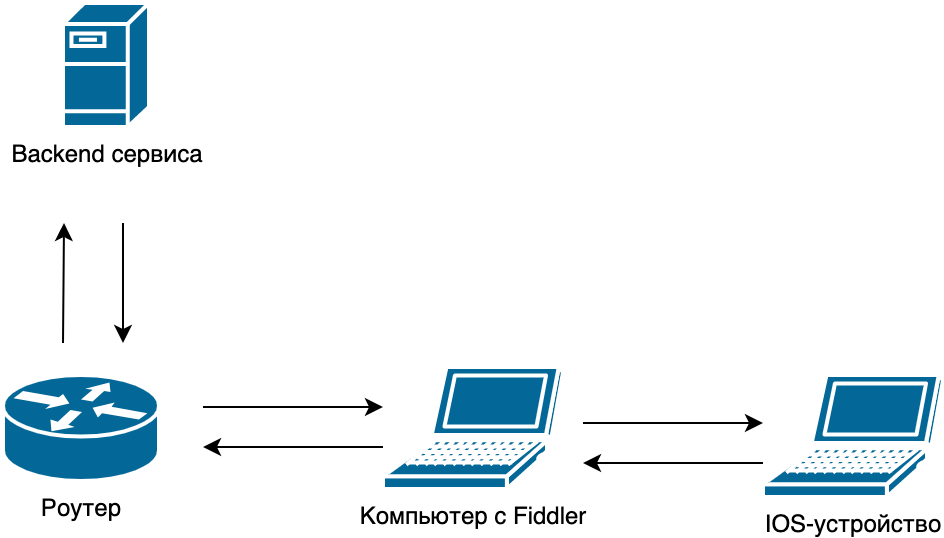
* Парсинг сайта. На стеке Python типично используется связка Chromedriver + Selentium + BeautifulSoup, если сайт генерирует HTML страницу на стороне клиента и Requests + BeautifulSoup, если страничка не использует javascript. Здесь есть пару проблем, из-за которых работа может очень сильно осложниться. 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:

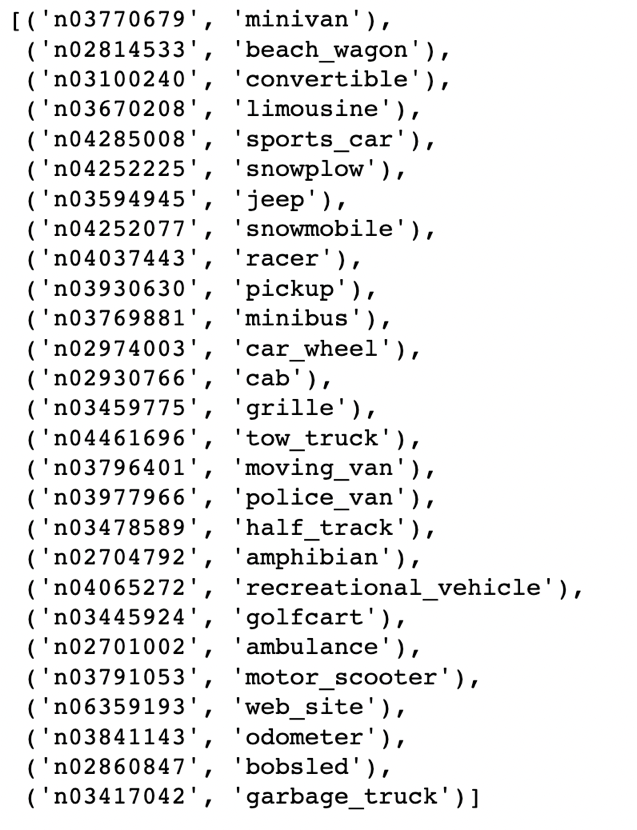


После нескольких часов работы созданного аппаратно-программный комплекса было получено 2241 файла формата JSON, содержащих большое кол-во информации о каждом объявлении, в т.ч. модель, марку автомобиля и url с фотографией автомобиля

## 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

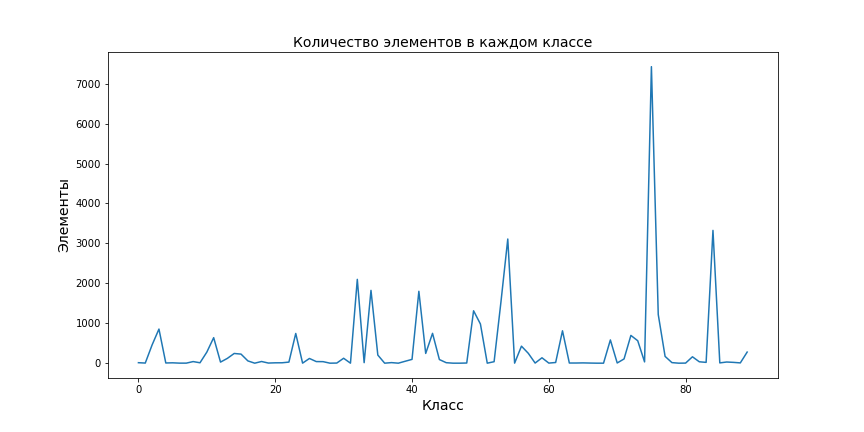


Таким образом было отфильтровано 784 записей и 34747 значений осталось. Использование моделей глубокого обучения для формирования новых датасетов вполне себе обычная практика на сегодняшний день.

## 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)

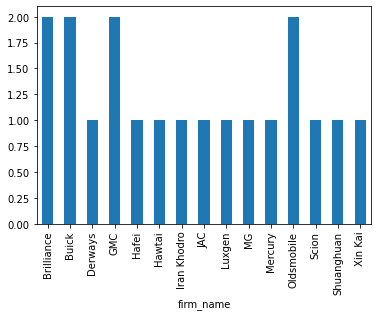


Диаграмма N. Наиболее редкие автомобили

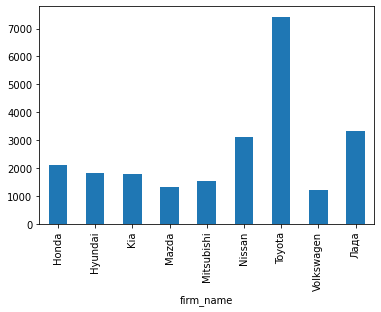


Диаграмма N. Наиболее популярные автомобили

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.

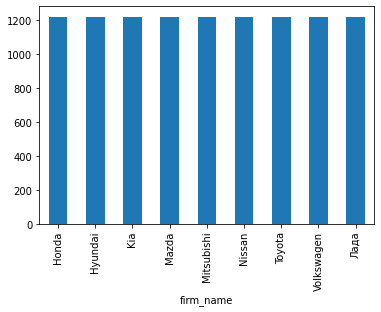
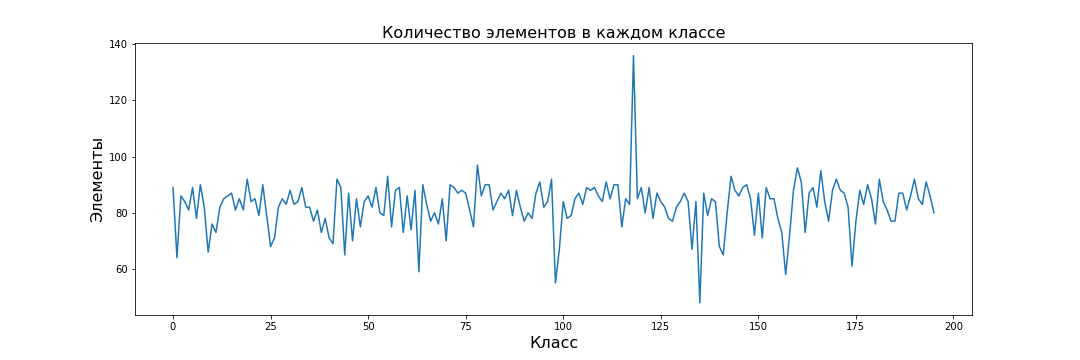


Диаграмма N. Наиболее популярные автомобили

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 Стэнфордского 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.

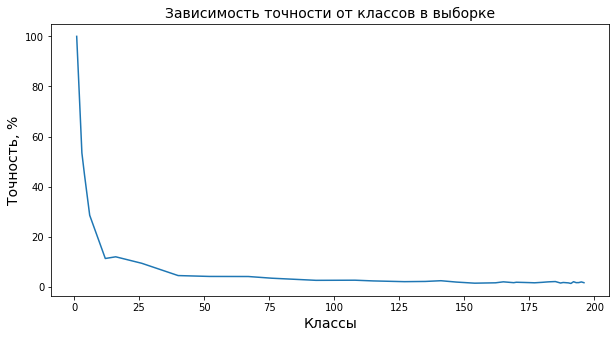


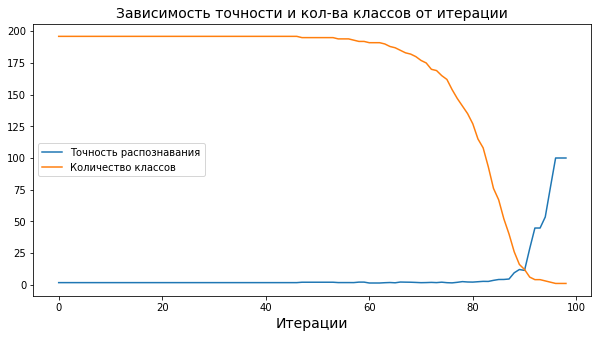
С помощью диаграммы размаха ящик с усами определяем медиану, равную 84.



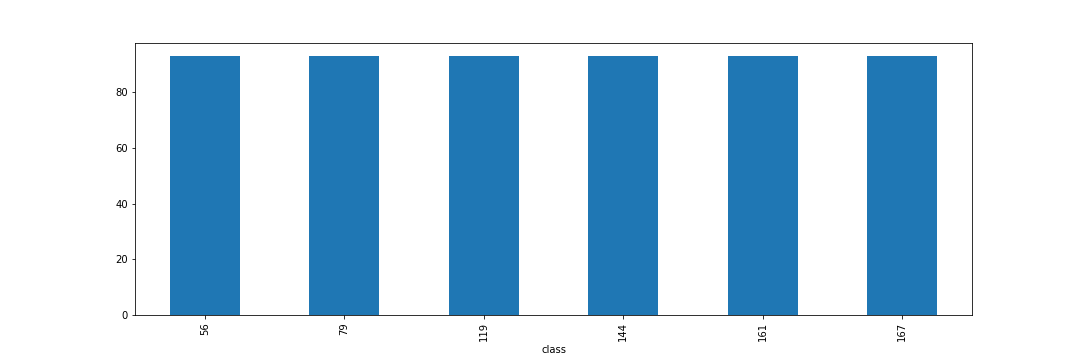
Диаграмма N. Размах баланса классов для датасета Стэнфордского университета

Для балансировки выборки мы оставляем только 6 наиболее объемных класса и делаем срез по минимальному количеству элементов в данной выборке. Мы намерено уменьшаем количество классов т.к при увеличении количества классов точность начинает значительно снижаться. Для измерения точности использовался наивный байесовский классификатор т.к. он является наиболее быстрым среди всех моделей.



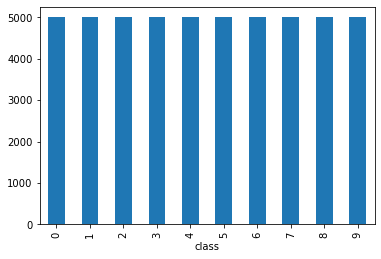
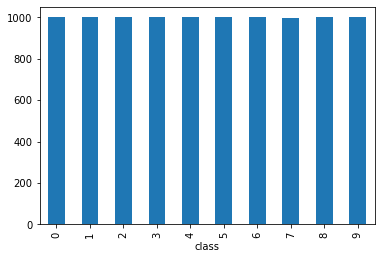


После сокращения и перебалансировки получаем 93 элемента в 6 классах, всего 558 изображений.



### 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.

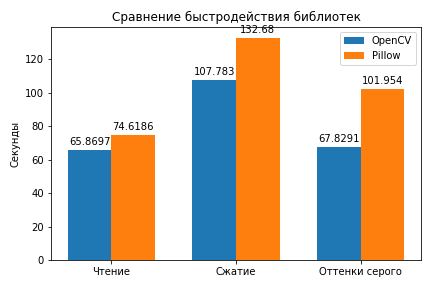


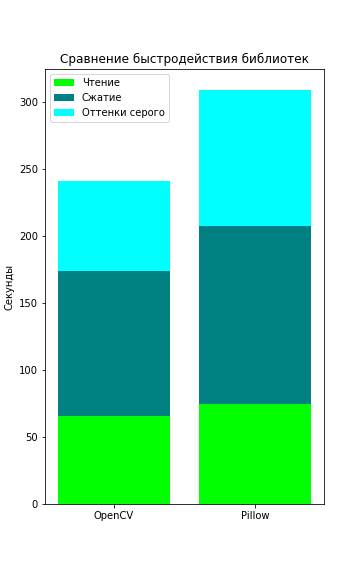
## Преобразование данных

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.

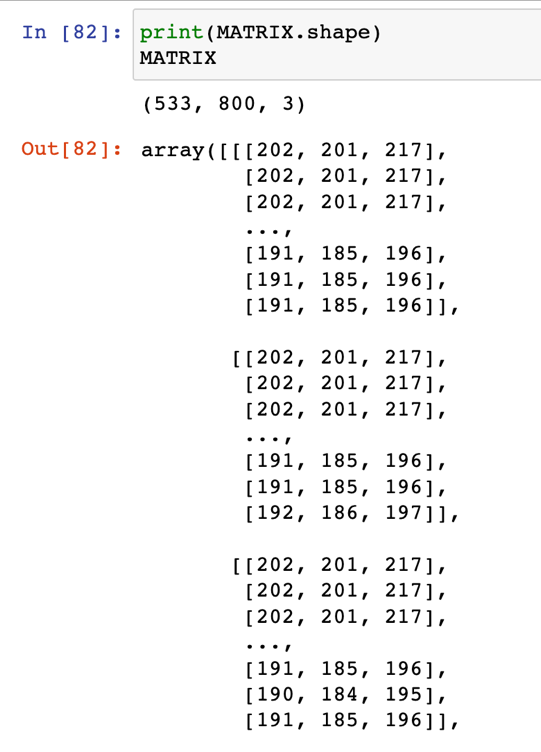




Как видим из графиков, библиотека OpenCV намного быстрее работает с необходимыми нам алгоритмами, в особенности с преобразованием изображения в черно-белый формат. Т.к. производительность при большом количестве данных является приоритетом, то в нашем случае выбор падает на OpenCV. Рассмотрим каждое действие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

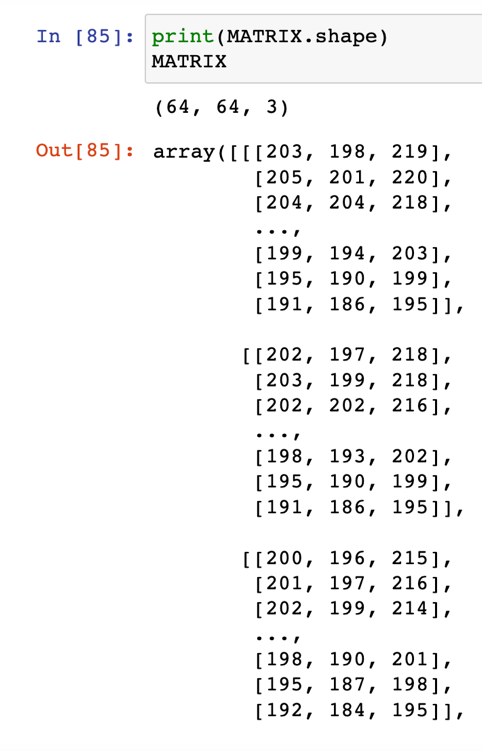
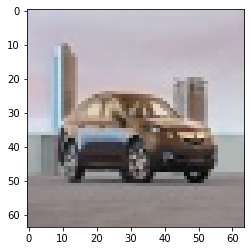
### Представление изображений в матричном виде

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.



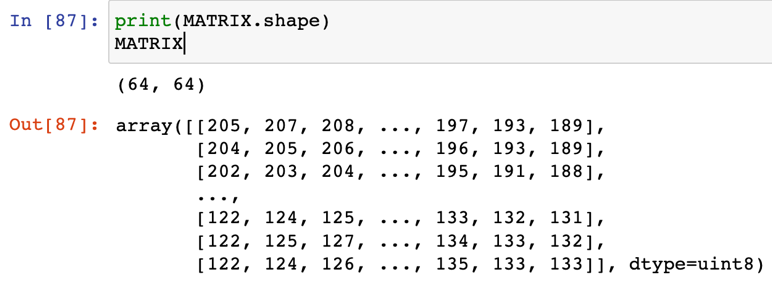
### Уменьшение размеров изображения

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.

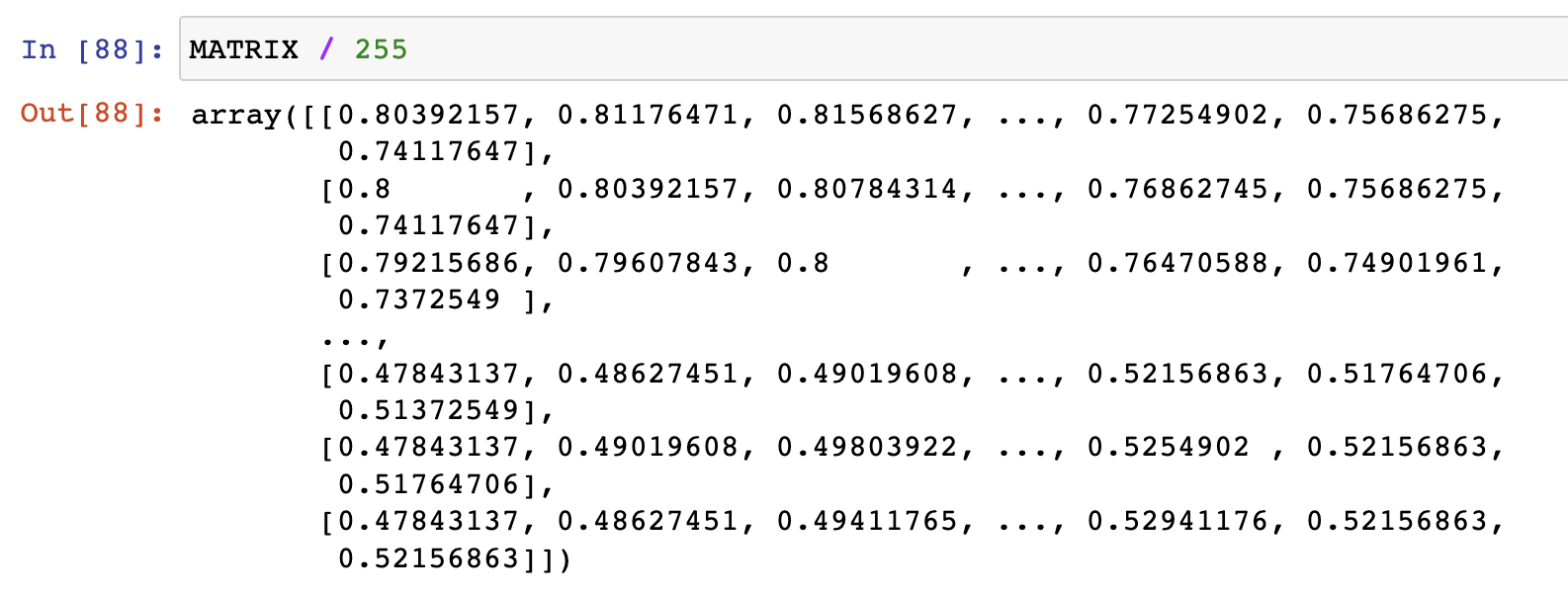


### Нормализация изображений

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 автомобилей и машин Стэнфордского университета необходимо разделить выборки на обучающую и проверочную. У каждого из них достаточно изображений (10980 для собственного, 558 для Стэнфордского), выборки делим в соотношении 80% на 20% с помощью метода scikit-learn train\_test\_split.

X\_train, X\_test, y\_train, y\_test = train\_test\_split(X, Y, test\_size=0.2, random\_state=0, stratify=Y)

В аргументе X передаём numpy.array с нормализованными матрицами изображений автомобилей, в Y – номера классов, которые эти изображения характеризуют, в test\_size – размер проверочной выборки, random\_state=0 - то псевдослучайные величины при генерации будут иметь одни и те же значения при каждом вызове. При указании аргумента stratify метод производит разбиение таким образом, что доля значений в полученной выборке будет такой же, как и доля значений, предоставленных параметру stratify.

После обработки данного метода интерпретатором, получим 4 выборки:

* X\_train – матрицы изображений обучающей выборки
* X\_test – матрицы изображений проверочной выборки
* Y\_train – классы, характеризующие изображения обучающей выборки
* Y\_test – классы, характеризующие изображения проверочной выборки

# IV. СОЗДАНИЕ АЛГОРИТМОВ И ОБУЧЕНИЕ МОДЕЛЕЙ

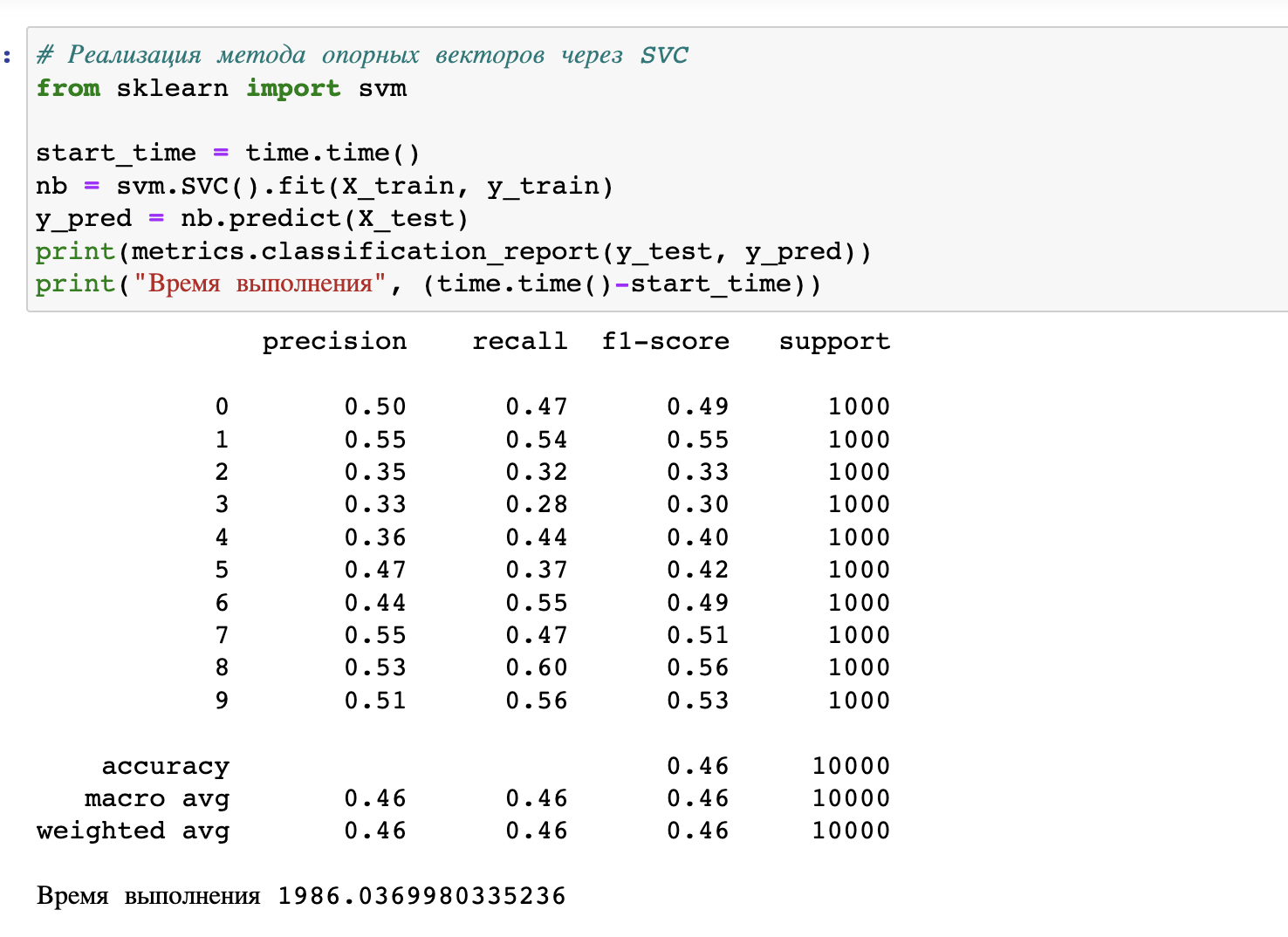
## Алгоритмы машинного обучения

### 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. На других датасетах результаты следующие:

|  |  |  |
| --- | --- | --- |
| **Датасет** | **Точность** | **Время** |
| CIFAR-10 | 0,46 | 1986,03 |
| Собственный | 0,24 | 361,42 |
| Stanford | 0,35 | 1,06 |

### Метод Байеса

Наивный байесовский классификатор является одним из наиболее простых и быстрых алгоритмов машинного обучения, который используется в качестве базового достаточно часто, причем достаточно часто даёт хорошие показатели метрик в задачах классификации.

Наивным алгоритм называют прежде всего из-за того, что по определению он считает каждый фактор равнозначным. Например, при изображении разрешением каждый из 4096 пикселей будет влиять на результат равнозначно, что не совсем так. Достаточно часто исходный объект на изображении находится ближе к центру, при этом слева и справа будут также располагаться пиксели, которые не относятся к нему.



Метод показал относительно невысокие результаты точности в районе 23%, зато время выполнения алгоритма составило менее секунды, что является самым быстрым результатом среди всех рассмотренных алгоритмов.

|  |  |  |
| --- | --- | --- |
| **Датасет** | **Точность** | **Время** |
| CIFAR-10 | 0,23 | 0,51 |
| Собственный | 0,16 | 0,3 |
| Stanford | 0,29 | 0,05 |

### Метод k-ближайших соседей

Алгоритм k-ближайших соседей, также известный как KNN, является непараметрическим классификатором контролируемого обучения, который использует близость для классификации или прогнозирования группировки отдельных точек данных. Обычно он применяется в качестве алгоритма классификации, исходя из предположения, что похожие объекты могут быть найдены рядом друг с другом.

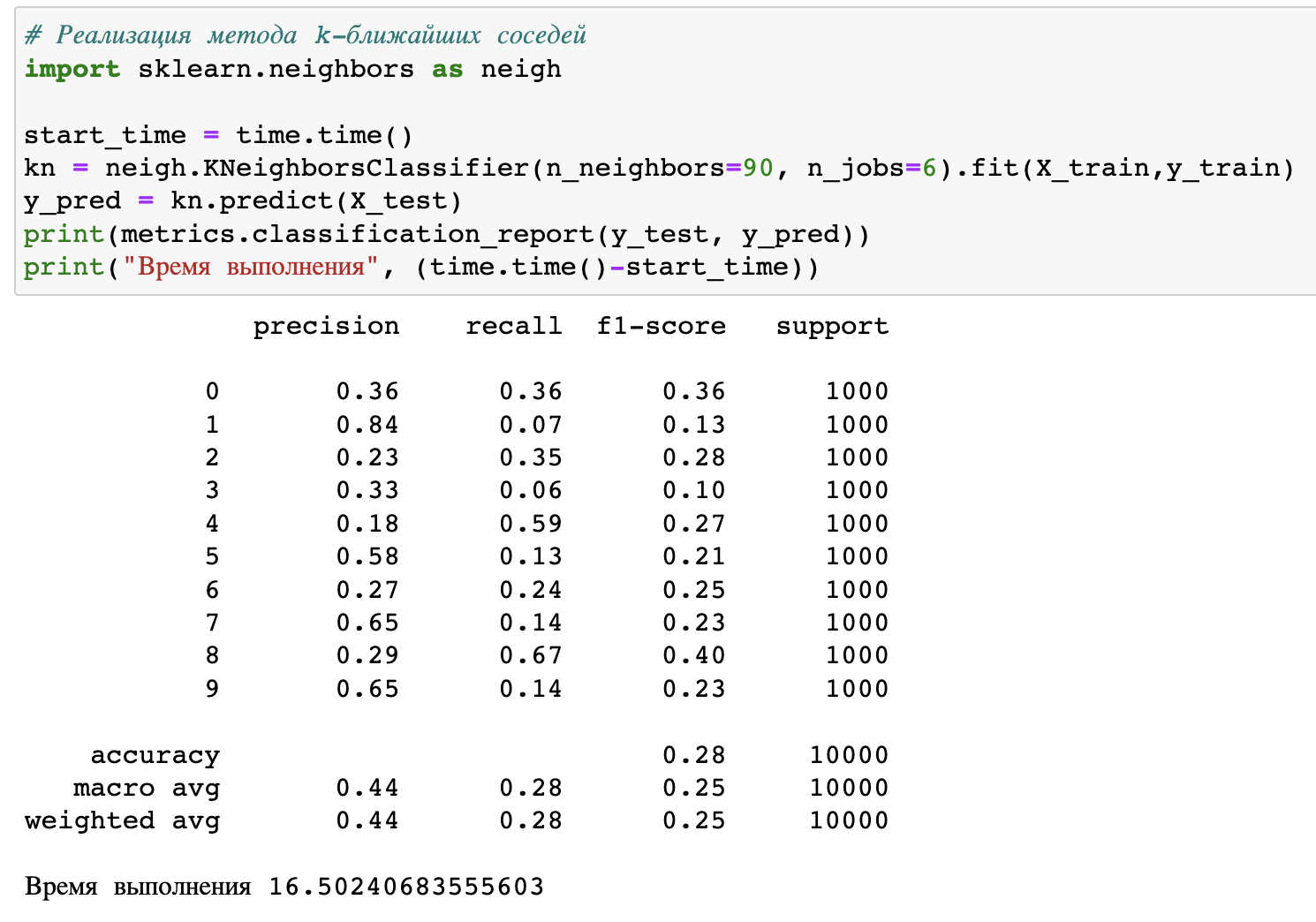
Метод достаточно прост: для каждого объекта происходит сравнение со всеми другими объектами в выборке и путем нахождения косинуса угла между векторами вычисляется расстояние между ними.

(10)

Затем определяется аргумент k, который в обучающей выборке определяет количество ближайших объектов для текущего элемента b.

. Далее элемент заносят в тот класс, у которого релевантность выше заданного порога:

(11)



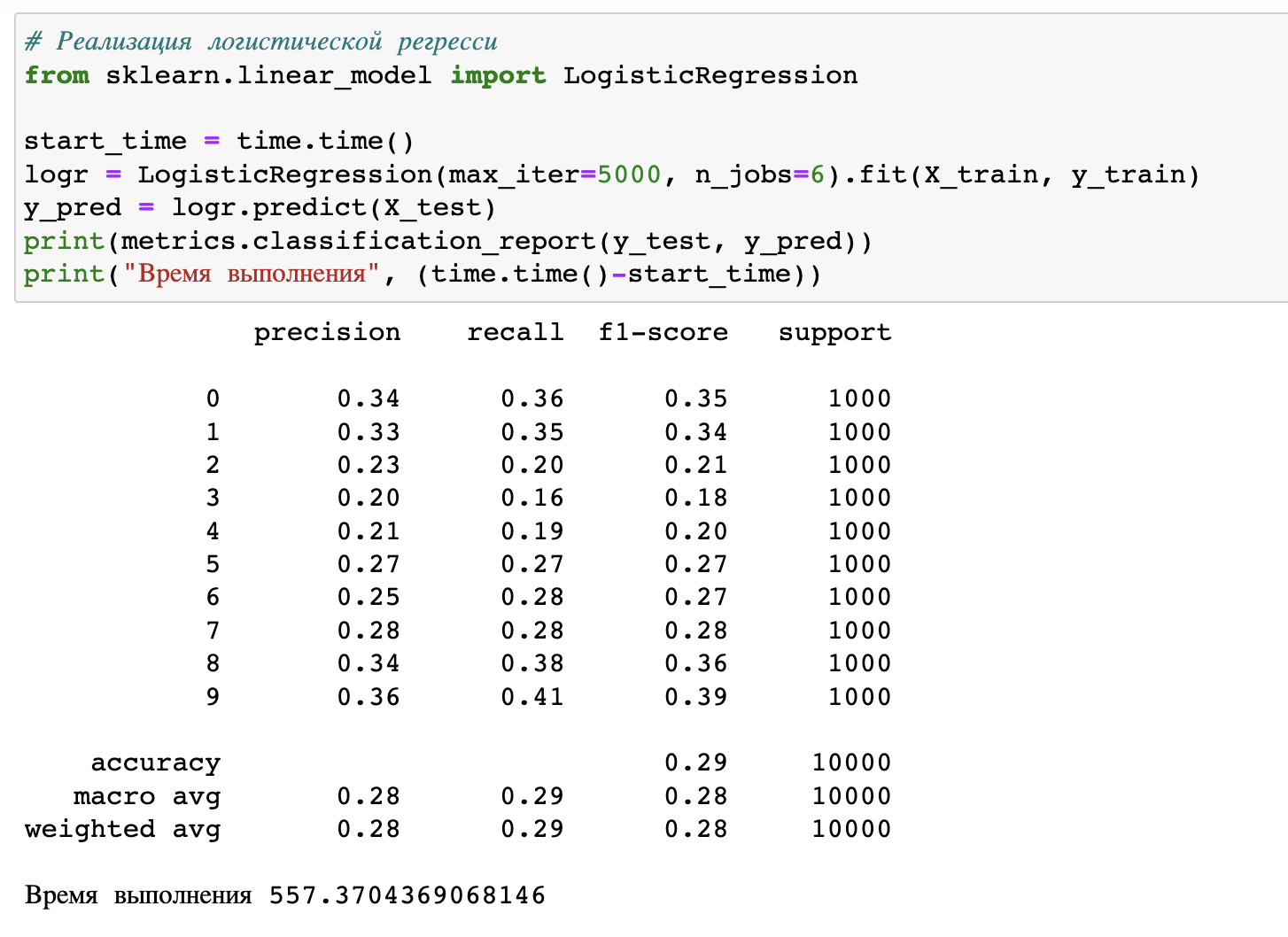
Метод выполнялся чуть дольше наивного байесовского классификатора: 16 секунд, получил точность чуть более высокую 28% вместо 23% у Байеса.

|  |  |  |
| --- | --- | --- |
| **Датасет** | **Точность** | **Время** |
| CIFAR-10 | 0,28 | 16,5 |
| Собственный | 0,21 | 3,18 |
| Stanford | 0,28 | 0,11 |

### Логистическая регрессия

Логистическая регрессия – алгоритм контролируемого обучения, используемый для прогнозирования зависимой категориальной целевой переменной.

Логистическая регрессия требует, чтобы зависимая переменная была категориальной. Алгоритм основывается на вероятностях принадлежности данных к определённому классу, которые моделируются с помощью логистической функции, которая выступает кумулятивной функцией распределения. Задача с непрерывным результатом, например, прогнозирование цены акции или запаса топлива в генераторе не подходит для использования логистической регрессии, для задач классификации ее использование возможно.



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.

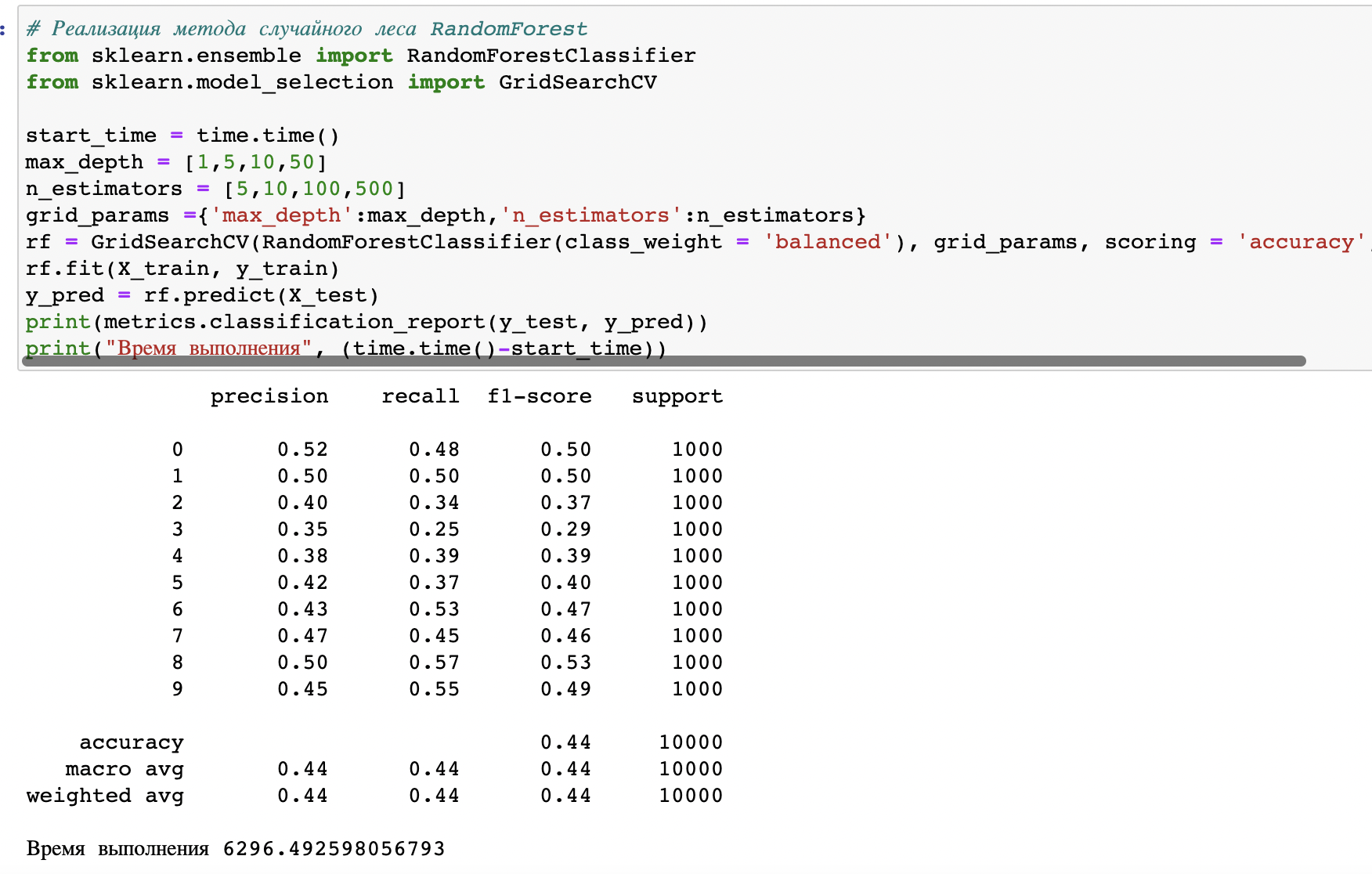
|  |  |  |
| --- | --- | --- |
| **Датасет** | **Точность** | **Время** |
| CIFAR-10 | 0,29 | 557,37 |
| Собственный | 0,21 | 534,2 |
| Stanford | 0,22 | 12,69 |

### Метод «Cлучайного леса»

RF (random forest) — это множество решающих деревьев. В задаче регрессии их ответы усредняются, в задаче классификации принимается решение голосованием по большинству. Все деревья строятся независимо по следующей схеме:

* Выбирается подвыборка обучающей выборки размера samplesize– по ней строится дерево (для каждого дерева — своя подвыборка).
* Для построения каждого расщепления в дереве просматриваем max\_features случайных признаков (для каждого нового расщепления — свои случайные признаки).
* Выбираем наилучшие признак и расщепление по нему (по заранее заданному критерию). Дерево строится, как правило, до исчерпания выборки (пока в листьях не останутся представители только одного класса), но в современных реализациях есть параметры, которые ограничивают высоту дерева, число объектов в листьях и число объектов в подвыборке, 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%.

|  |  |  |
| --- | --- | --- |
| **Датасет** | **Точность** | **Время** |
| CIFAR-10 | 0,44 | 6296,49 |
| Собственный | 0,25 | 1340,82 |
| Stanford | 0,29 | 71,4 |

### Линейный классификатор с SGD-обучением

Линейный классификатор - алгоритм классификации, основанный на построении линейной разделяющей поверхности. В случае двух классов разделяющей поверхностью является гиперплоскость, которая делит пространство признаков на два полупространства. В случае большего числа классов разделяющая поверхность кусочно-линейна.

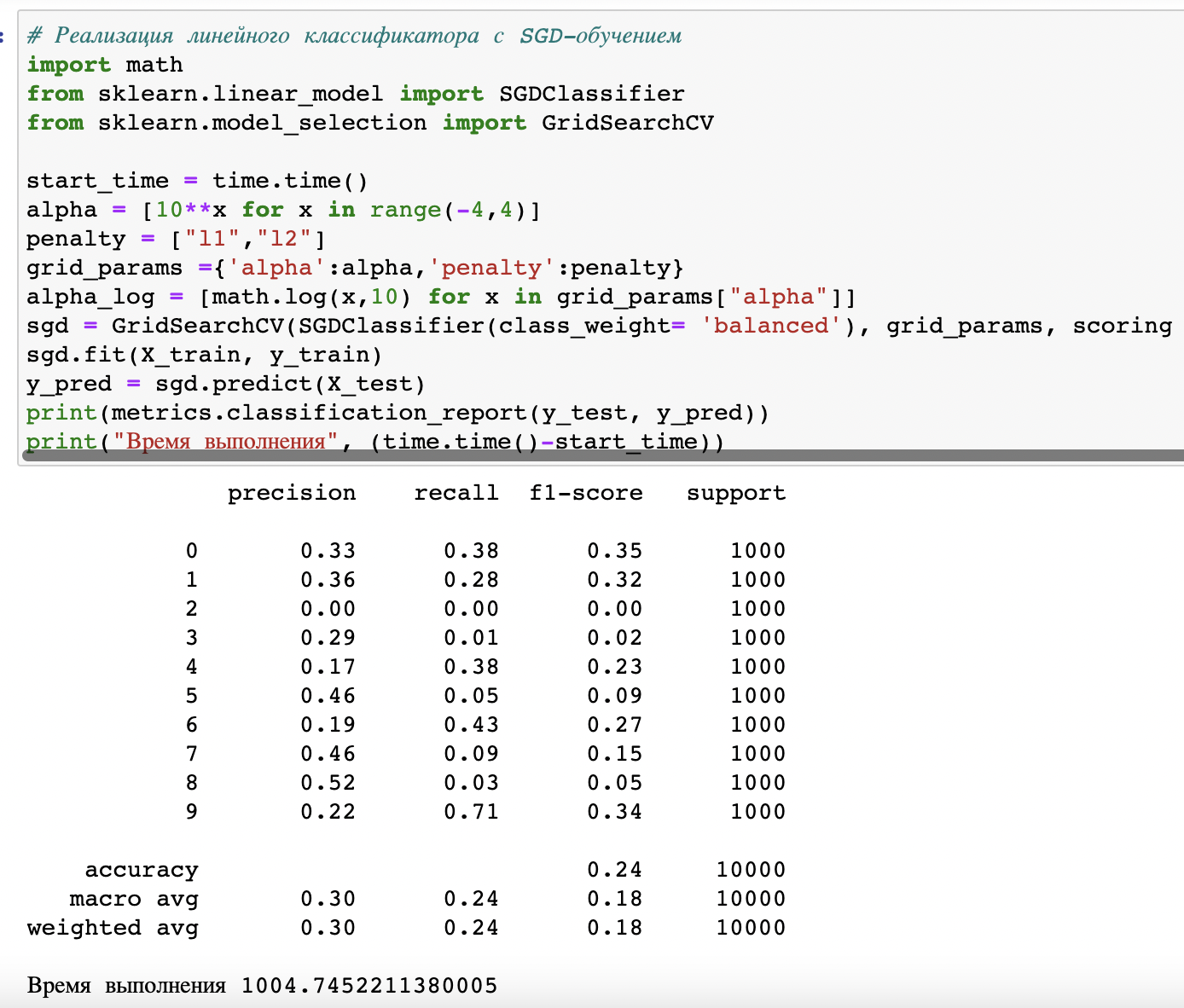
Пусть объекты описываются числовыми признаками , n. Тогда пространство признаковых описаний объектов есть . Пусть конечное множество номеров (имён, меток) классов.

В случае двух классов, Положим .  
Линейным классификатором называется алгоритм классификации вида

где вес -го признака, порог принятия решения, вектор весов, скалярное произведение признакового описания объекта на вектор весов. Предполагается, что искусственно введён «константный» нулевой признак: .

В случае произвольного числа классов линейный классификатор определяется выражением

где каждому классу соотвествует свой вектор весов . Реализуем линейный классификатор с помощью scikit-learn:



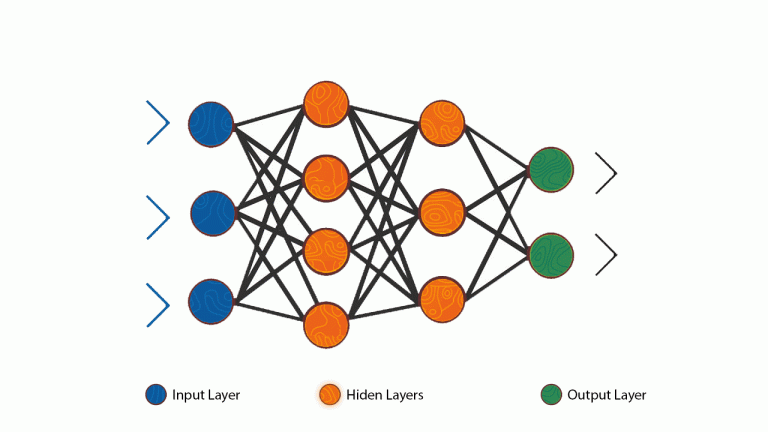
Линейный классификатор с SGD-обучением по точности аналогичен наивному методу Баеса (24% распознавания), но при этом работает 16 минут 44 секунды, что примерно в 1000 раз дольше.

|  |  |  |
| --- | --- | --- |
| **Датасет** | **Точность** | **Время** |
| CIFAR-10 | 0,24 | 1004,74 |
| Собственный | 0,2 | 1431,24 |
| Stanford | 0,24 | 32,73 |

## Алгоритмы глубокого обучения

### Нейронная сеть прямого распространения ANN

#### Один перцептрон (или нейрон) можно представить как логистическую регрессию. Нейронная сеть прямого распространения, 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.



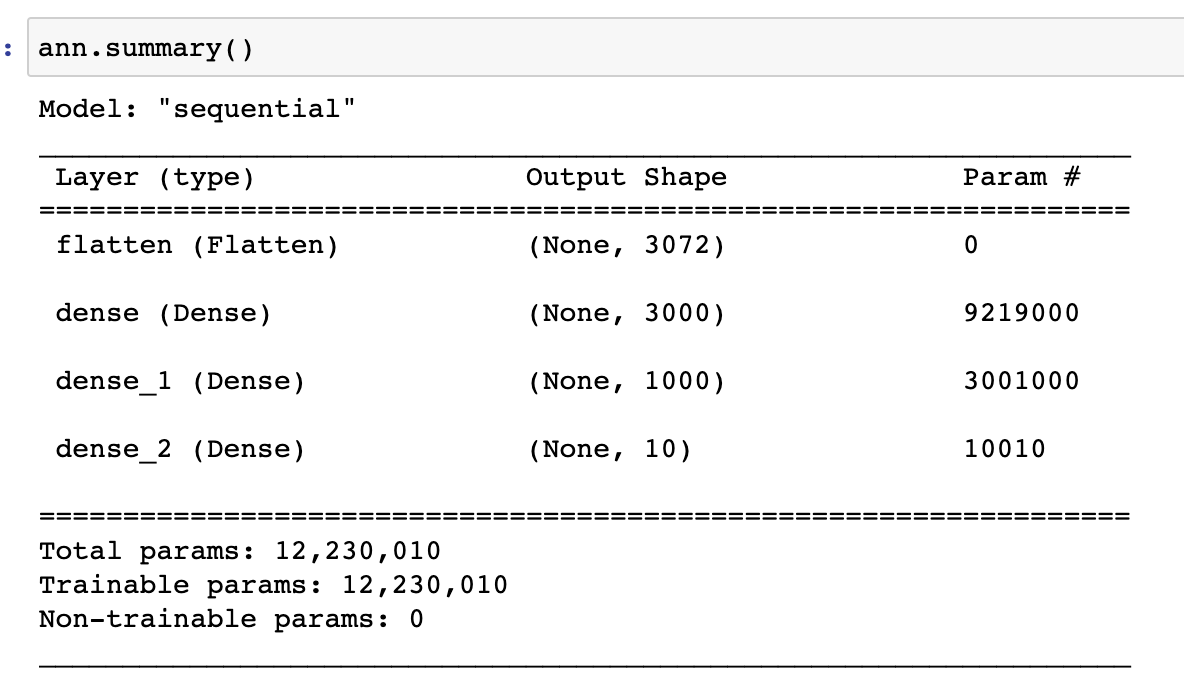
ANN состоит из 3 типов слоев - входного, скрытого и выходного. Входной слой принимает входные данные, скрытый слой обрабатывает входные данные, а выходной слой выдает результат. По сути, каждый слой пытается выучить определенные веса.

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 следующая сеть прямого распространения:





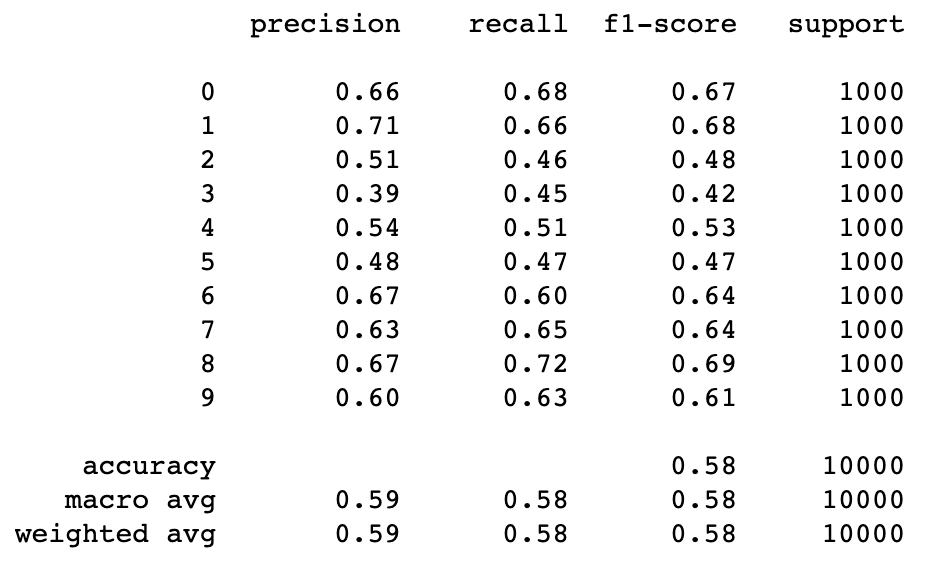
Данная сеть имеет следующие типы слоёв:

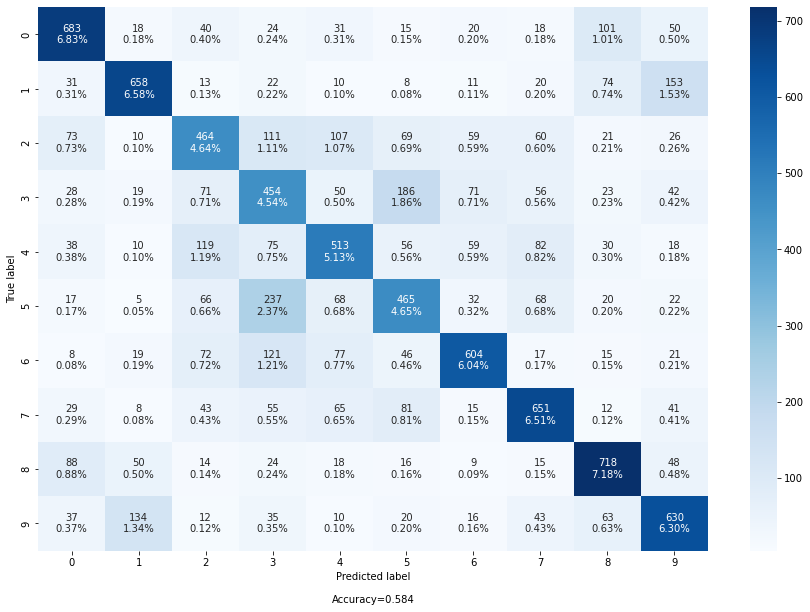
* Входной слой Flatten. Flatten используется для конвертации входящих данных в меньшую размерность.
* Dense слой. Dense-слой получает информацию со всех узлов предыдущего слоя. Вычисляет вывод следующим образом: output=activation(dot(input,kernel)+bias), где activation — это активатор, а kernel — взвешенная матрица, применяемая к входящим тензорам. bias — это константа, помогающая настроить модель наилучшим образом.

В Dense-слоях используются следующие функции активации:

* RELU (Rectified Linear Unit) — это наиболее часто используемая функция активации при глубоком обучении. Данная функция возвращает 0, если принимает отрицательный аргумент, в случае же положительного аргумента, функция возвращает само число т.е. может быть записана как Функция нелинейна, можно использовать в нейронных сетях с множеством слоев.
* Softmax – функция, превращающая наборы чисел в вероятности, причем сумма которых равна единице. Функция выводит в качестве результата вектор, представляющий распределения вероятностей списка потенциальных результатов. Это также основной элемент, используемый в задачах классификации глубокого обучения

После обучения в 100 эпох получаем точность 58%, время обучения – 1770 секунд.



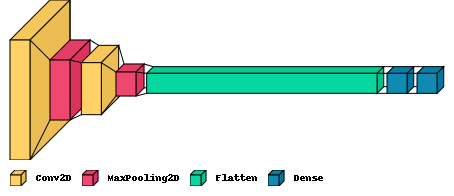


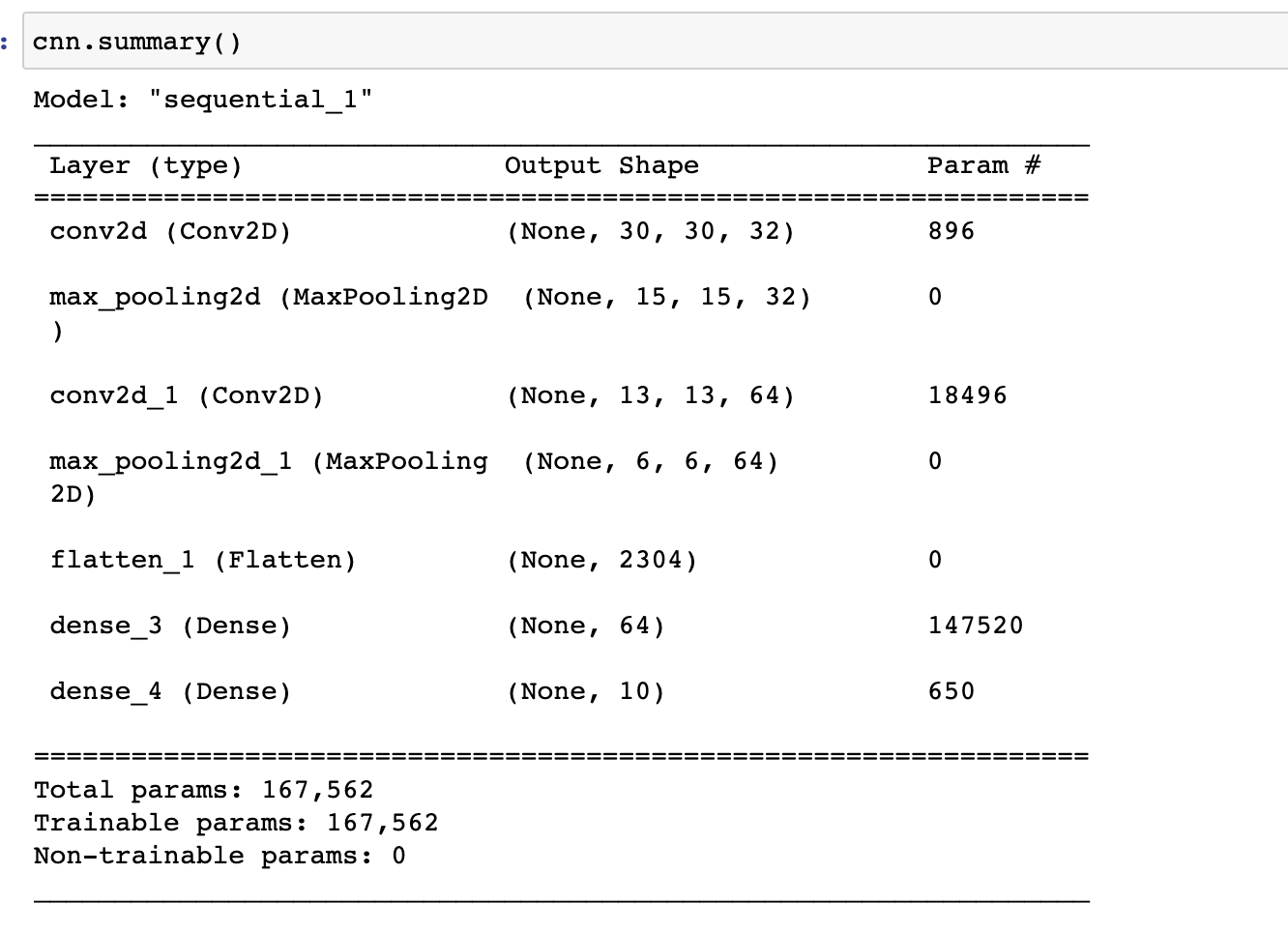
### Сверточная нейронная сеть CNN

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.

С помощью библиотеки Keras была спроектирована следующая сверточная нейронная сеть:





Данная сеть имеет следующие типы слоёв:

* Conv2D. Этот слой подобен Dense-слою, и содержит веса и смещения, которые подвергаются оптимизации (подбору). Conv2D-слой так же содержит фильтры («ядра»), значения которых тоже оптимизируются. В Conv2D-слое значения внутри матрицы фильтра и есть переменные, которые подвергаются оптимизации.
* Max pooling2D. Операция максимального объединения для двумерных пространственных данных. Уменьшает выборку входных данных по пространственным измерениям, беря максимальное значение в окне ввода для каждого канала входных данных.
* Flatten и Dense, аналогично использованные и описанные при построении сети прямого распространения в предыдущем пункте.

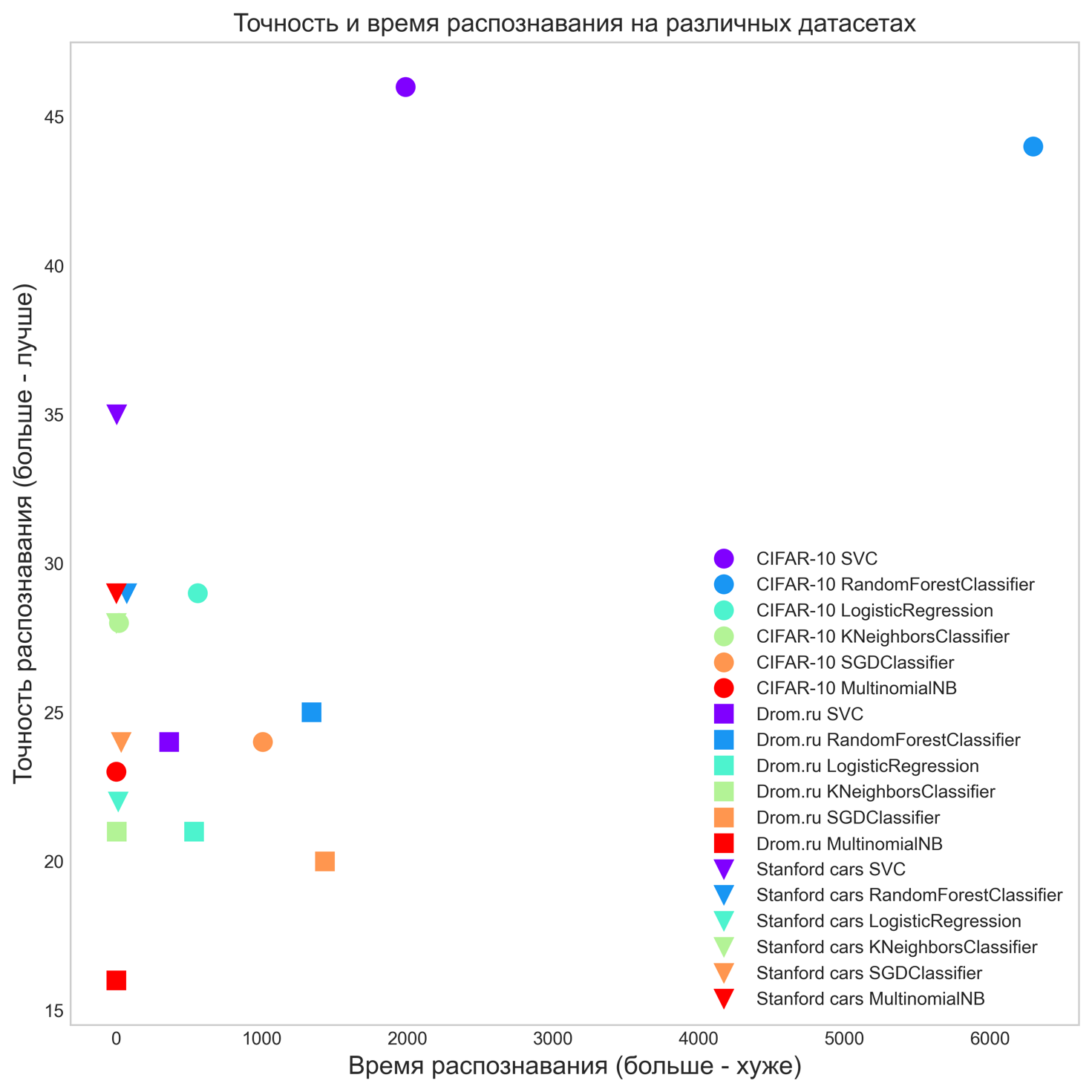
После обучения 12 эпох получаем точность 70%, время обучения – 189 секунд.





# V. СРАВНЕНИЕ МОДЕЛЕЙ

В ходе испытаний были получены следующие данные на не-глубоких алгоритмах по всем 3 датасетам:

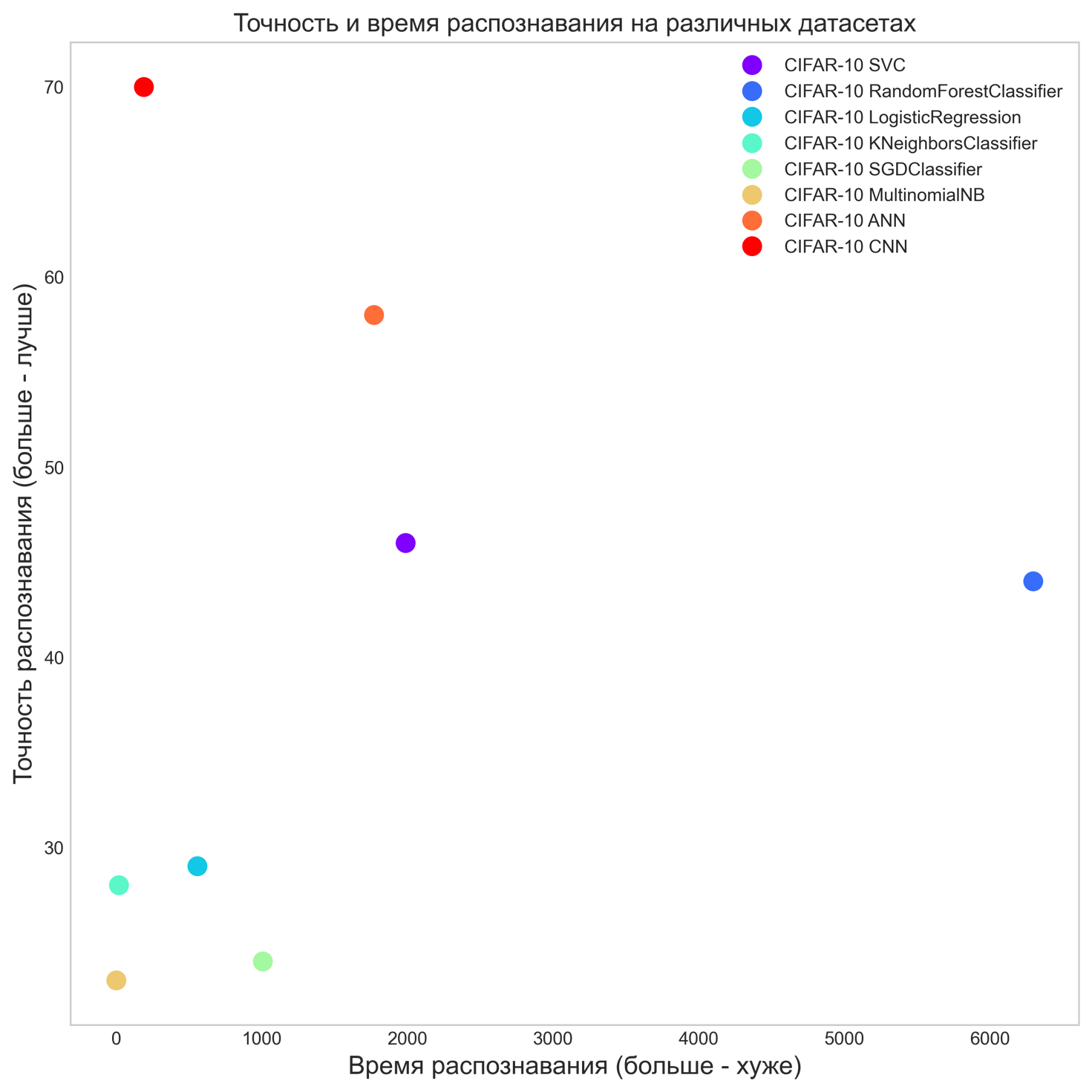


Также данные можно представить в виде таблицы:

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
|  | **CIFAR-10** | | **Drom.ru** | | **Stanford cars** | |
| **Алгоритм** | **Точность** | **Время, сек** | **Точность** | **Время, сек** | **Точность** | **Время, сек** |
| SVC | 46% | 1986,03 | 24% | 361,42 | 35% | 1,06 |
| RandomForestClassifier | 44% | 6296,49 | 25% | 1340,82 | 29% | 71,4 |
| LogisticRegression | 29% | 557,37 | 21% | 534,2 | 22% | 12,69 |
| KNeighborsClassifier | 28% | 16,5 | 21% | 3,18 | 28% | 0,11 |
| SGDClassifier | 24% | 1004,74 | 20% | 1431,24 | 24% | 32,73 |
| MultinomialNB | 23% | 0,51 | 16% | 0,3 | 29% | 0,05 |

Лучше всего с задачей по классификации исходных данных справились модели, основанные на машине опорных векторов и случайного леса, но последняя отрабатывает в 3,2 раза дольше, что для большого количества данных критично (на CIFAR-10 33 минуты против 1 часа 44 минуты соответственно), поэтому наилучшим вариантом будет метод опорных векторов, если важна именно точность. Стоит учитывать лишь относительную точность между алгоритмами по определенной выборке. т.к. при уменьшении количества классов абсолютная точность возрастает. Если же при работе нам необходима максимальная скорость обработки данных, то следует использовать наивный байесовский алгоритм, где время не превышало полсекунды.

Если же мы говорим об алгоритмах глубокого обучения на примере данных CIFAR-10, то они действительно являются наилучшими на сегодняшний день, особенно отличается решение на базе сверточных нейронных сетей, где процент распознавания 70%, а обучение занимает всего 3 минуты 9 секунд.



# V. ЗАКЛЮЧЕНИЕ

Я кот кот кот

Точность классификации сильно зависит от исходных данных, в то числа от методов разбалансировки классов. Классически алгоиртмы ахахах

# 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).

# XI. ПРИЛОЖЕНИЯ

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

Тут приложение