­­МИНИСТЕРСТВО ОБРАЗОВАНИЯ И НАУКИ РОССИЙСКОЙ ФЕДЕРАЦИИ

Федеральное государственное бюджетное образовательное учреждение

высшего образования

**«КУБАНСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ»**

**(ФГБОУ ВО «КубГУ»)**

**Факультет компьютерных технологий и прикладной математики**

**Кафедра вычислительных технологий**

**КУРСОВАЯ РАБОТА**

**РЕАЛИЗАЦИЯ И СРАВНИТЕЛЬНЫЙ АНАЛИЗ МЕТОДОВ МАШИННОГО ОБУЧЕНИЯ ДЛЯ КЛАССИФИКАЦИИ МУЗЫКАЛЬНЫХ КОМПОЗИЦИЙ ПО ЖАНРАМ**

Работу выполнил \_\_\_\_\_ Н.С. Лысенко

(подпись, дата) (инициалы, фамилия)

Факультет компьютерных технологий и прикладной математики 3 курс

Направление подготовки 02.03.03 Математическое обеспечение и администрирование информационных систем

Направленность Технология программирования

Научный руководитель

доц., канд.т.н. \_\_\_\_\_\_\_ Т.А. Приходько

(подпись, дата) (инициалы, фамилия)

Нормоконтролер

ассистент \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ А.А. Климец

(подпись, дата) (инициалы, фамилия)

Краснодар 2020

СОДЕРЖАНИЕ

[Введение 3](#_Toc41462211)

[1 Задача классификации и методы её решения 4](#_Toc41462212)

[1.1 Линейные методы классификации 4](#_Toc41462213)

[1.2 Нелинейные методы классификации 5](#_Toc41462214)

[1.3 Логистическая регрессия 6](#_Toc41462215)

[1.4 Скрытые марковские модели 8](#_Toc41462216)

[2 Описание используемых методов, функций и библиотек 11](#_Toc41462217)

[2.1 Извлечение признаков из звуковых файлов 11](#_Toc41462218)

[2.2 Выбор метода классификации 14](#_Toc41462219)

[2.3 Возможности Python по анализу данных 15](#_Toc41462220)

[3 Алгоритм классификации 18](#_Toc41462221)

3.1 Описание исходных данных

[3.2 Первичный визуальный анализ данных 18](#_Toc41462222)

[3.3 Описание алгоритма классификации 22](#_Toc41462223)

[4 Анализ полученных результатов. Сравнение точности линейного и нелинейного классификатора 24](#_Toc41462224)

[Заключение 29](#_Toc41462225)

[Список использованных источников 30](#_Toc41462226)

[Приложение 31](#_Toc41462227)

# ВВЕДЕНИЕ

В настоящее время нейросетевые классификаторы широко используются во многих медиа, поисковых, развлекательных интернет-сервисах, что существенно облегчает работу с данными любых типов, будь то документы, изображения, видео или аудио. Также это позволяет значительно повысить уровень автоматизации таких сервисов или информационных систем и снизить влияние человеческого фактора на процессы управления данными. В данной курсовой работе будет рассмотрена задача классификации именно музыкальных композиций, так как она считается весьма актуальной в связи с увеличением числа веб-приложений с музыкальной тематикой и ускоренным темпом роста их популярности. Также многие методы распознавания музыкальных композиций применяются в задачах распознавания голоса. Классификаторы на основе машинного обучения обладают способностью обучаться на ограниченном множестве входных данных и имеют большое количество эффективных алгоритмов обучения. Существует множество методов классификации, которые используют различный математический аппарат и различные подходы при реализации.

Цель курсовой работы: изучить линейные и нелинейные методы машинного обучения, реализовать собственный классификатор музыкальных композиций по жанрам на основе логистической регрессии на языке программирования Python, а также сравнить результаты работы линейного и нелинейного классификаторов.

# Задача классификации и методы её решения

Классификацией в теории машинного обучения называется такая процедура, в которой объекты распределяются по группам (классам) в соответствии со значениями их признакового описания, представленных в численном виде. [1] На основе этих признаков строится обучающая выборка, то есть данные, которые будут использоваться для обучения алгоритма. При этом метка в классификации принимает набор дискретных значений, эквивалентных номерам классов. Идея классификации состоит в том, чтобы для каждого нового объекта из имеющейся обучающей выборки определить метку класса этого объекта. Геометрически, данная задача эквивалентна построению разделяющей поверхности в многомерном признаковом пространстве. Среди подходов к решению подобных задач, как правило, выделяют классификацию с обучением и без него. Если для всех объектов исходного набора известно, к какому классу они принадлежат, то такая постановка задачи называется классификацией с учителем (или с обучением). Обучение без учителя происходит в том случае, когда принадлежность объектов в исходном наборе данных заранее не известна.

В зависимости от типа разделяющей точки обучающей выборки границы, существуют следующие виды методов классификации: вероятностные, метрические, логистические, линейные, нелинейные, регрессия.

## Линейные методы классификации

Разделяющая объекты поверхность в данных методах является линейной гиперплоскостью. Эта плоскость задается в зависимости от имеющихся исходных данных и обучающего алгоритма. Идея линейной классификации состоит в том, чтобы провести линейную гиперплоскость через пространство объектов таким образом, чтобы она разделяла его на две соответствующие классам области, как на рисунке 1. Для корректной работы алгоритма точки исходных данных (объекты) должны быть линейно разделяемыми.

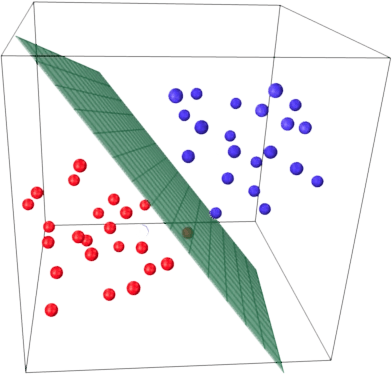


Рисунок 1 – разделение объектов гиперплоскостью на два класса.

Подобная разделяющая плоскость называется линейным дискриминантом ввиду линейности функции, которая её описывает, и позволяет модели производить разделение, то есть дискриминацию точек на различные классы. Решение в задачах линейной классификации принимается на основании линейного оператора над входными данными. [2]

К линейным алгоритмам классификации относятся линейная регрессия, логистическая регрессия, машина опорных векторов. Такие методы довольно эффективны на малом объёме данных, и могут работать лучше сложных алгоритмов, например, таких как деревья решений или deep learning, но при большом количестве входных данных результаты заметно ухудшаются.

## Нелинейные методы классификации

Правило классификации образов, основанное на выборке, в идеальном случае должно верно классифицировать точки самой выборки. Это не всегда достижимо, если ограничивается возможная форма правила классификации. То есть, если в правиле применяется линейное преобразование пространства признаковых описаний, то точная классификация осуществима только в том случае, когда можно провести гиперплоскости между каждой парой классов так, чтобы все точки одного класса лежали по одну сторону гиперплоскости, а все точки другого класса — по другую. [3]

Когда же предположение о линейной отделимости описаний нарушаются, используются нелинейные алгоритмы классификации. В данных методах граница, разделяющая объекты на соответствующие классы, является нелинейной. Классическими алгоритмами такого рода дискриминации объектов являются метод k-ближайших соседей, деревья решений, нейронные сети, скрытые марковские модели.

## Логистическая регрессия

Один из старейших линейных методов классификации в задачах прогнозирования. В отличие от обычной регрессии, в логистической регрессии не производится предсказание значения числовой переменной (признака класса) исходя из выборки исходных значений. Вместо этого, значением функции является *вероятность* принадлежности исходного значения к определенному классу, то есть логистическая регрессия прогнозирует вероятность отнесения вектора описаний к классу «**+**».

Математически, алгоритм прогнозирования осуществляется следующим образом:

1. Вычисляется значение линейной функции:

(1)

где *w* – вектор параметров модели (в машинном обучении эти параметры также называют весами), *x –* вектор признаков.

1. Вычисляется алгоритм отношения шансов (вероятностей):

(2)

Отношение вероятностей , то есть вероятность того, произойдёт событие *Х* или не произойдёт.

1. Имея теперь прогноз шансов на принадлежность *х* к классу «***+***», вычисляется с помощью зависимости:

(3)

Теперь перейдём к вопросу об обучении модели. Вероятность принадлежности к классу « – » находится аналогично:

(4)

Объединяя шаги вычисления и , можем получить:

(5)

Выражение называется отступом классификации на объекте , и если он положителен или равен 0, то модель верно спрогнозировала класс для данного объекта, иначе, если отступ отрицательный, то классифицирован неверно. Следует отметить, что отступ определён именно для объектов обучающей выборки, для которых известны метки класса – . [2]

Геометрически, отступ – это перпендикуляр к разделяющей точки выборки гиперплоскости, и, если его значение велико (по модулю), то чем дальше от данной границы объект, тем уверенней он классифицируется. Если же значение отступа малое (по модулю), то точка находится быстро к разделяющей гиперплоскости. Верность классификации определяется знаком отступа (рисунок 2).

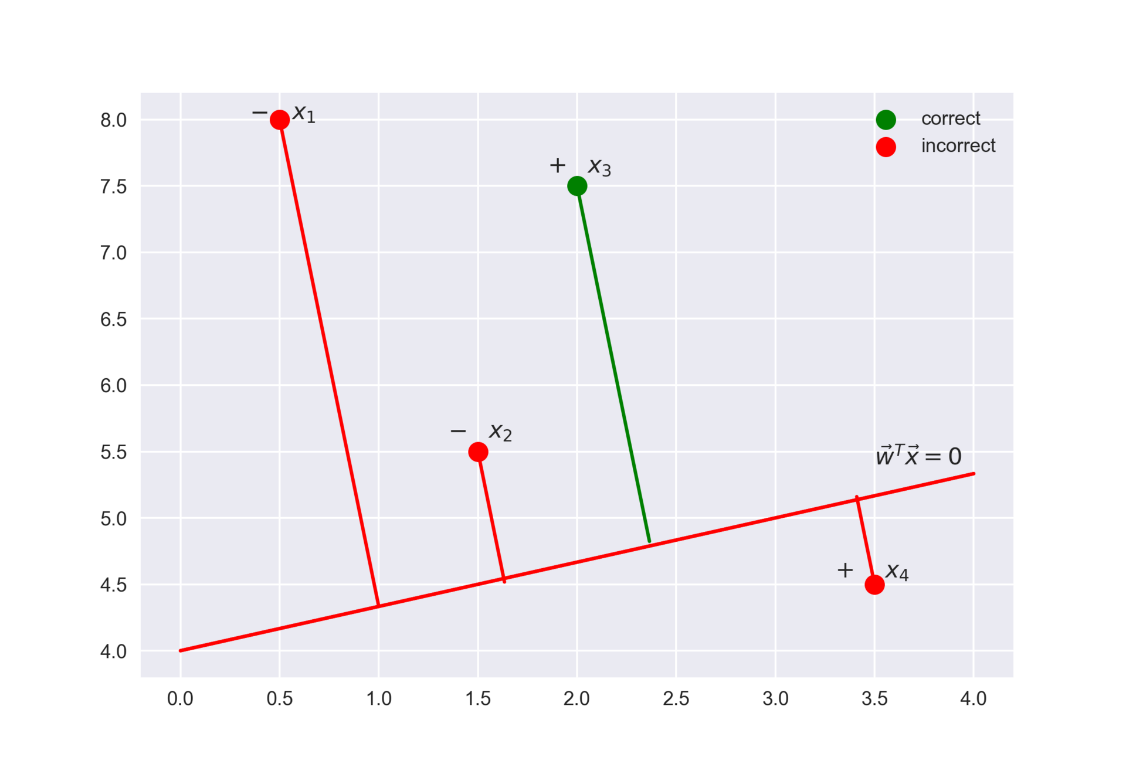


Рисунок 2 – геометрическая интерпретация классификации

Таким образом, проводится оценка вероятности верной классификации точки обучающей выборки данной моделью. Поэтому, среднее значение для всей обучающей выборки показывает вероятность того, что случайная точка данных будет корректно классифицирована независимо от возможного класса.   
То есть, механизм обучения логистической регрессии старается максимизировать среднее значение функции от точки обучающей выборки, иначе говоря, применяется метод максимального правдоподобия.

## Скрытые марковские модели

Метод машинного обучения, который часто используется в задачах распознавания речи, то есть по сути звуковых сигналов, поэтому из нелинейных методов в задаче идентификации жанра он является наиболее подходящим. Понятие скрытых марковских моделей тесно связано с понятием Марковского процесса. Марковскийпроцесс — случайный процесс, эволюция которого после любого заданного значения временного параметра t не зависит от эволюции, предшествовавшей t, при условии, что значение процесса в этот момент фиксировано («будущее» процесса зависит от «прошлого» лишь через «настоящее»). Например, по оси абсцисс случайным образом перемещается точка. Причём её перемещение происходит так: в момент времени *t=0* она находится в начале координат и остается там в течение одной секунды. Через секунду бросается монета — если выпал герб, то точка перемещается на одну единицу длины вправо, если решка — влево. Через секунду снова бросается монета и производится такое же случайное перемещение, и так далее. Процесс изменения положения точки является случайным процессом с дискретным временем *t=0, 1, 2,* … и счётным множеством состояний. Такое перемещение точки и представляет собой марковский процесс, потому что следующее состояние точки зависит только от текущего состояния и не зависит от прошлых состояний. [4]

Теперь перейдём к определению скрытых марковских моделей. Скрытая марковская модель (СММ) — статистическая модель, имитирующая работу процесса, похожего на марковский процесс с неизвестными параметрами, и задачей является разгадывание неизвестных параметров на основе наблюдаемых. Полученные параметры могут быть использованы в дальнейшем анализе, например, для распознавания образов.

В обычном марковском процессе состояние видимо наблюдателю, поэтому имеется единственный параметр – вероятности переходов. В скрытой же Марковской модели мы можем следить лишь за переменными, на которые оказывает влияние данное состояние. Каждое состояние имеет вероятностное распределение среди всех возможных выходных значений. Поэтому последовательность символов, сгенерированная СММ, даёт информацию о последовательности состояний.

Структура скрытых марковских моделей в общем случае может быть представлена как конечный автомат с перепадами между любыми парами состояний, как показано на рисунке 3.

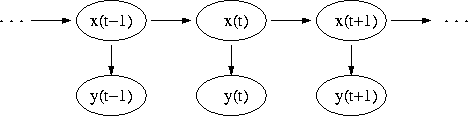


Рисунок 3 – Конечный автомат, представляющий структуру скрытых марковских моделей

Здесь случайная переменная *x ( t ) {\displaystyle x(t)} x(t)* представляет собой значение скрытой переменной в момент времени *t {\displaystyle t} t*. Случайная переменная *y ( t ) {\displaystyle y(t)} y(t)* — это значение наблюдаемой переменной в момент времени *t {\displaystyle t} t*. Стрелки на диаграмме символизируют некоторые условные зависимости. Из диаграммы становится ясно, что значение скрытой переменной x ( t ) x(t) зависит только от значения скрытой переменной *x ( t − 1 ) {\displaystyle x(t-1)} x(t-1)* в момент t − 1 времени *t-1*. Это называется свойством Маркова. Хотя в то же время значение наблюдаемой переменной *y ( t ) {\displaystyle y(t)} y(t)* зависит только от значения скрытой переменной *x ( t ) {\displaystyle x(t)} x(t)*. Для нахождения неизвестных параметров СММ используется Алгоритм Баума-Велша, который занимается обучением модели. [5]

# Описание используемых методов, функций и библиотек

## Извлечение признаков из звуковых файлов

Так как обучающая выборка строится на основе признаков классифицируемых объектов, то необходимо выделить такие признаки в музыкальных композициях, по которым их можно было бы однозначно отнести к определённому классу. В качестве описаний объектов были выбраны мел-частотные кепстральные коэффициенты (Mel-Frequency Cepstral Coefficients, MFCC).

В задачах распознавания речи и звуковых сигналов часто используется способ кодирования энергетического спектра звука, то есть распределения энергии, содержащейся в частотах, который называется метод мел-частотных кепстральных коэффициентов (MFCC). Эти коэффициенты – спектральные характеристики, уникальные для каждого сигнала. Иначе говоря, это представление энергии сигнала в виде особого спектра, из которого путём различных фильтраций и преобразований удалены незначительные для человеческого уха компоненты и шумы. Спектр проецируется на мел-шкалу. Подобное представление сохраняет волновую «природу» сигнала и позволяет выделить наиболее значимые для человеческого уха частоты.

Вычисление мел-частотных кепстральных коэффициентов включает в себя следующие шаги [6]:

* 1. Необходимо разделить исходный сигнал на кадры. Их размер обычно выбирается в диапазоне от 10 до 40 мс. Кадры накладываются друг на друга.
  2. Полагаем, что звуковой сигнал конечен и не является периодическим, поэтому из-за разрывов на его концах, когда применяется преобразование Фурье, может произойти так называемый эффект утечки. В дискретной области так называют эффекты, связанные с искажениями в Фурье анализе, вызванные конечностью выборки. В случае периодического дискретного процесса этих эффектов удается избежать, если специальным образом выбирать шаг дискретизации и длину временного ряда. То есть, можно контролировать «ширину» этой утечки или, концентрируя её вокруг главной частоты (сильно «размывается» спектр, зато не мешает соседним частотам), или, растягивая её на бесконечности (но в этом случае размытие пиков уменьшается, а «шум» сильно растёт и вследствие этого растёт погрешность измерения амплитуды отдельных частот). Как правило, чтобы снизить влияние эффекта утечки на результат, к каждому кадру применяется оконная функция (весовая функция). Часто «конечный» сигнал, используемый для преобразования Фурье, в реальности является частью возможно бесконечного сигнала, такого, как синусоида, например. В этой ситуации применяют дополнение конечного отрезка нулями: считают, что исходный сигнал имеет бесконечно большую длину, но затем умножается на некоторую взвешивающую функцию — «окно», обращающуюся в ноль вне доступного для измерения отрезка. В простейших случаях роль «окна» играет прямоугольная функция, с помощью которой мы, по сути, дополняем отрезок слева и справа бесконечным количеством нулей. В данном случае используем оконную функцию Хемминга:

*w(n) = 0.54 – 0.46cos(2)* (6)

* 1. К каждому кадру применяется преобразование Фурье, получая тем самым спектр сигнала. Далее вычисляется периодограмма – оценка спектральной плотности мощности сигнала:

*Pi (k) = |Yi(k)|2/N* (7)

* 1. Полученное представление сигнала в частотной области разбивают на диапазоны с помощью блока треугольных пересекающихся фильтров, расположенных наиболее плотно в области низких частот. Границы фильтров рассчитывают в мел-шкале. Данная шкала является результатом исследований по способности человеческого уха к восприятию звуков на различных частотах. Количество таких фильтров - 26. Для расчёта фильтров обычно выбираются верхняя и нижняя частоты, а затем осуществляется переход от частотной шкалы к мел-шкале по формуле (8):

*M(f) = 1127\*ln(1+f/700)* (8)

* 1. Между полученными значениями на мел-шкале выбираются точки, расположенные линейно, как правило, для 26 фильтров — 28 точек. После этого переход обратно в частотную шкалу осуществляется по обратной формуле 9:

*F(m) = 700(exp(m/1125) - 1)*  (9)

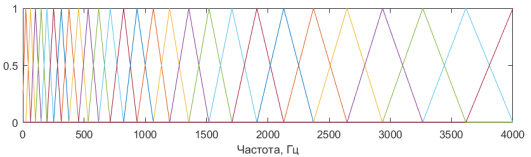


Рисунок 2 – График частотной шкалы

Из графика видно, что окна сжимаются в большей степени в области низких частот (рисунок 2). Далее простым перемножением векторов спектра сигнала и оконной функции находится энергия сигнала, которая попадает в каждое из анализируемых окон. Получаем некоторый набор коэффициентов. Полученные значения энергии сигнала возводятся в квадрат и логарифмируются, после чего применяется дискретное косинусное преобразование. В итоге, у нас есть 26 коэффициентов для каждого кадра (окна) — это и есть мел-частотные кепстральные коэффициенты. В задачах распознавания речи обычно используются первые 13 из них, как наиболее информативные для речевого сигнала.

В результате последовательность коэффициентов выглядит так, как показано на рисунке 3.

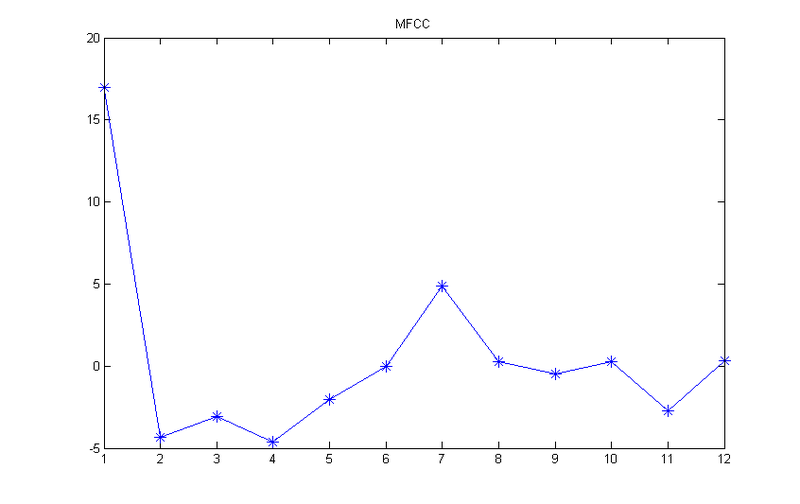


Рисунок 3 – График расположения спектральных частотных коэффициентов (MFCC) на мел-шкале

Существуют и другие характеристики звукового сигнала, которые можно было бы взять в качестве признаковых описаний объекта, но для решения задачи классификации по жанрам вектора из мел-частотных кепстральных коэффициентов будет вполне достаточно, чтобы обеспечить высокие показатели точности идентификации, потому что эти значения несут в себе много важной информации об уникальности звукового сигнала.

## Выбор метода классификации

В общем случае, выбор зависит от вида пространства точек и их расположения в плоскости обучающей выборки. Для нашей задачи идентификации жанра в качестве описаний каждого объекта выбран вектор из MFCC, следовательно, в алгоритм классификации будет поступать матрица признаков объектов в качестве обучающей выборки с соответствующей меткой класса.

Количество классов явно больше двух, потому что существуют десятки различных жанров, значит, данная задача относится к полиномиальной классификации. Также предполагается, что классы не пересекаются. В таком случае, алгоритм линейной регрессии не пододёт, так как он рассчитан на бинарную классификацию. Для дискриминации объектов по нескольким группам из линейных алгоритмов часто используют логистическую регрессию (Logistic Regression, LR) или метод опорных векоторов (Support Vector Machine, SVM). Эти методы аналогичны; отличаются только функциями потерь и тем, что SVM старается максимизировать разницу между ближайшими векторами, а LR – вероятность апостериорного класса. Исходя из этого, считается, что метод опорных векторов работает немного лучше, то есть выдаёт более высокие по точности классификации результаты. Но всё-таки разница в точности незначительная, и логистическая регрессия имеет достойные показатели в полиномиальной классификации. К тому же она лучше реализовывается в языке программирования Python и хорошо работает на небольшом количестве признаковых описаний объектов, поэтому в поставленной задаче идентификации жанра используется именно этот линейный метод.

Из нелинейных методов машинного обучения для решения проблемы классификации звуковых сигналов лучше всего подходит алгоритм Скрытых Марковских Моделей, который часто используется при распознавании диктора. Метод k-ближайший соседей больше применяется в компьютерном зрении, пространство объектов в котором выглядит совершенно иначе, нежели в данной задаче.

## Возможности Python по анализу данных

Для исследования данных и реализации алгоритма классификации используется язык программирования Python, у которого в настоящее время имеется большое количество библиотек и фреймворков для реализации машинного обучения, анализа данных и их визуализации. В работе используются облачный сервис Google Colaboratory. Это новый облачный сервис, направленный на упрощение исследований в области обучения нейронных сетей и deep learning. Используя Colaboratory, можно получить удаленный доступ к машине с подключенной видеокартой и процессором Tesla K80, что приносит большую пользу, когда приходится обучать нейронные сети на больших объёмах данных, или, когда возникают проблемы с установкой и подключением библиотек в интерпретаторах на обычном компьютере. Можно считать, что она является некоторым аналогом google-документов для Jupyter Notebook. В Google Colaboratory содержатся все библиотеки Python, необходимые для написания программ визуализации данных, для сложных математических расчётов, при создании различных обучаемых моделей, глубоких нейронных сетей и многого другого. Также в ней присутствует функция разбиения написанного кода на независимые блоки, вследствие чего появляется возможность запуска отдельных частей программы, что очень удобно для выявления ошибок в коде и при отладке программы. Этот функционал получилось реализовать, потому что Python является интерпретируемым языком (то есть команды исполняются построчно).

Для визуализации данных в работе используется метод pyplot из библиотеки matplotlib.

В качестве пакета для работы со звуком было решено выбрать Librosa. Она позволяет создать полноценную систему извлечения музыкальной информации. Также её преимущества в том, что Librosa предназначена именно для обработки музыки и хорошо документирована.

Машинное обучение осуществляется с помощью библиотеки Scikit-Learn, в которой реализовано большое количество линейных и нелинейных способов машинного обучения. Также в Scikit-Learn содержатся средства для загрузки и нормализации данных, отбора признаков, оптимизации параметров алгоритма, оценки качества работы алгоритма.

Реализация метода логистической регрессии осуществляется с помощью объекта LogisticRegression библиотеки Scikit-Learn. Функция LogisticRegression() создаёт модель логистической регрессии. Она имеет параметры, настраиваемые под определённую задачу классификации. В нашем случае параметр multi\_class='multinomial', так как производится полиномиальная классификация. Параметр max\_iter отвечает за максимальное количество итераций и по умолчанию имеет значение, равное 100. Если классификация не бинарная, то следует указать больше итераций, например, 1000. Параметр penalty='l2' позволяет выбрать функцию регуляризации логистических потерь, и в случае, когда количество классов больше двух, должен быть равен ‘l2’. Все остальные параметры можно на первых порах оставить такими же, какие они есть, но после получения результатов классификации некоторые из них, скорее всего, будет необходимо изменить для улучшения качества работы метода. Чтобы обучить модель, нужно вызвать функцию fit(X,y), где X – это двумерный массив признаков, y – метка класса для каждого набора описаний объекта из X. Также в работе используется библиотека NumPy — библиотека с открытым исходным кодом для языка программирования Python. Основной возможностью этой библиотеки является поддержка многомерных массивов (включая матрицы) и поддержка высокоуровневых математических функций, предназначенных для работы с многомерными массивами. Эта библиотека использует параллельные вычисления, и содержит алгоритмы для решения задач линейной алгебры. Numpy входит в библиотеку для научных вычислений SciPy.

Для работы с каталогами, где хранятся данные для обучения и тестирования в виде файлов в формате .wav применяются средства пакета os. Модуль os предоставляет множество функций для работы с операционной системой, причём их поведение, как правило, не зависит от ОС, поэтому программы остаются переносимыми. В данной реализации с помощью os проверяется наличие каталогов, подкаталогов и файлов в хранилище данных.

# Алгоритм классификации

**3.1** Описание исходных данных

Работа проводилась с набором данных GTZAN, предназначенном специально для спектрального анализа. [7] Это архив, содержащий 10 жанров, из которых в обучающую выборку включены музыкальные композиции 6 наиболее популярных жанров: classical, hiphop, jazz, pop, rock, metal. В наборе данных 100 произведений каждого жанра, каждое из которых имеет длительность 30 секунд и представлено в формате «.wav». Несмотря на то, что MP3 – более привычный нам формат хранения аудиофайлов, ведь он сущетвенно экономит память устройства, и музыки можно хранить в несколько раз больше, с точки зрения спектрального анализа он не походит, потому что является форматом сжатия с существенной потерей информации о сигнале. Поэтому для классификации используются аудиофайлы именно в «.wav» формате. К тому же, в Python можно считывать их с помощью пакета scipy.io.wavfile, который позволяет извлекать из звукового файла отсчёты, представимые в виде одномерного массива и частоту, с которой они берутся. В нашем случае все «.wav» файлы в одноканальном (моно) формате, поэтому массив одномерный, если бы формат был стерео, то массив отсчётов был бы двумерным.

## Первичный визуальный анализ данных

Из документации модуля Logistic Regression библиотеки Scikit-Learn известно, что на вход классификатору должны подаваться двумерный массив признаков и массив меток для каждого класса. Как было сказано в разделе 2.1 и 2.2, в качестве признаков выбраны мел-частотные кепстральные коэффициенты, которые несут в себе важную информацию об уникальности звукового сигнала, а в качестве признаков – названия жанров. Тогда предполагается, что у произведений одинаковых жанров должны быть похожие коэффициенты. Самый удобный способ это проверить – произвести визуальных анализ данных, иными словами – изобразить график этих коэффициентов для музыкальных композиций, который также можно назвать спектрограммой MFCC. Визуализация MFCC изображена на рисунках 4,5,6.

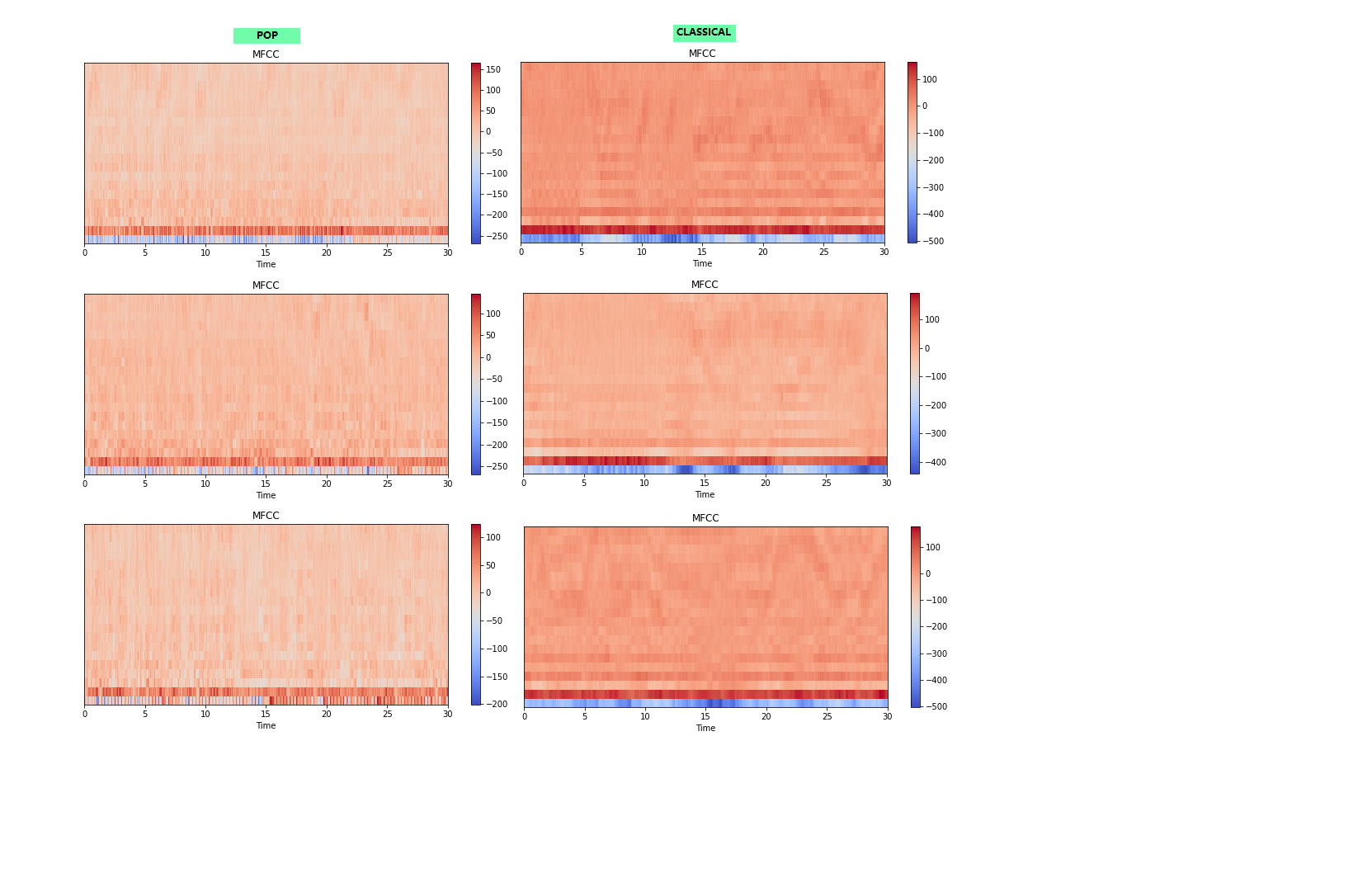


Рисунок 4 – спектрограмма MFCC для композиций жанров pop и classical

Можно заметить, что у произведений одного жанра есть некоторое сходство в расположении MFCC на временной шкале: в классических произведениях частота изменения значений MFCC меньше, чем в композициях жанра pop, например. На это указывает текстура рисунка спектра: там, где чаще встречаются кадры с мел-частотными коэффициентами, текстура больше напоминает некую «рябь». Это видно, например, в мел-спектрограммах жанров pop, hiphop, rock, metal (рисунки 4,5,6), потому что помимо инструментального звучания в них присутствует и вокал, а MFCC изначально направлены как раз таки на кодирование речевого спектра.

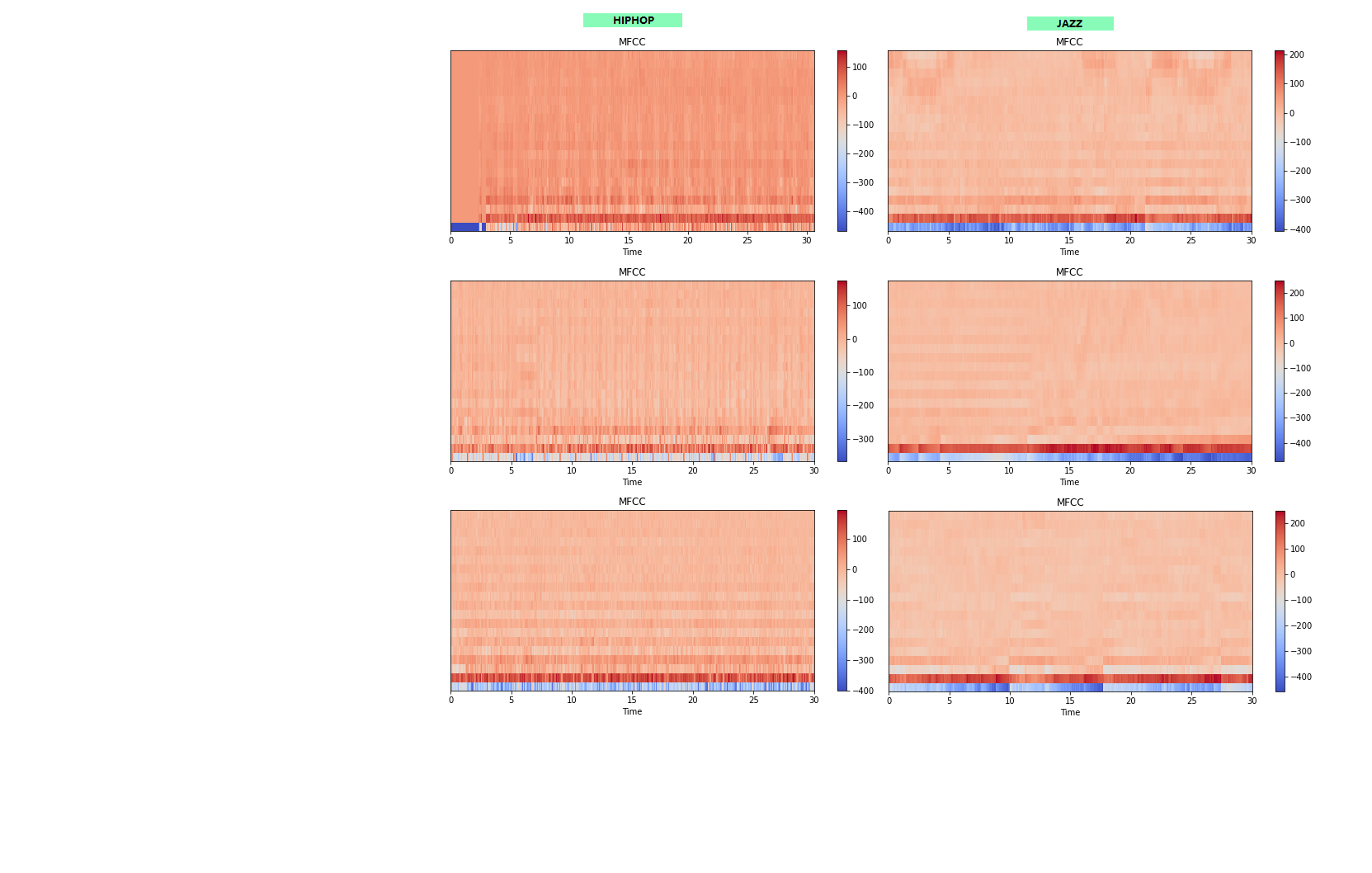


Рисунок 5 – спектрограмма MFCC для композиций жанров hiphop и jazz

Джазовые и классические композиции имеют в основном только интрументальное звучание, к тому же они не такие энергичные, как произведения других четырёх жанров нашей выборки. Поэтому MFCC в них изменяют свои значения не так часто, и диапазон значений начиная с четвёртого или пятого коэффициента не особо варьируется (более однородная цветовая гамма мел-спектрограммы наблюдается ближе к верхней границе рисунка).

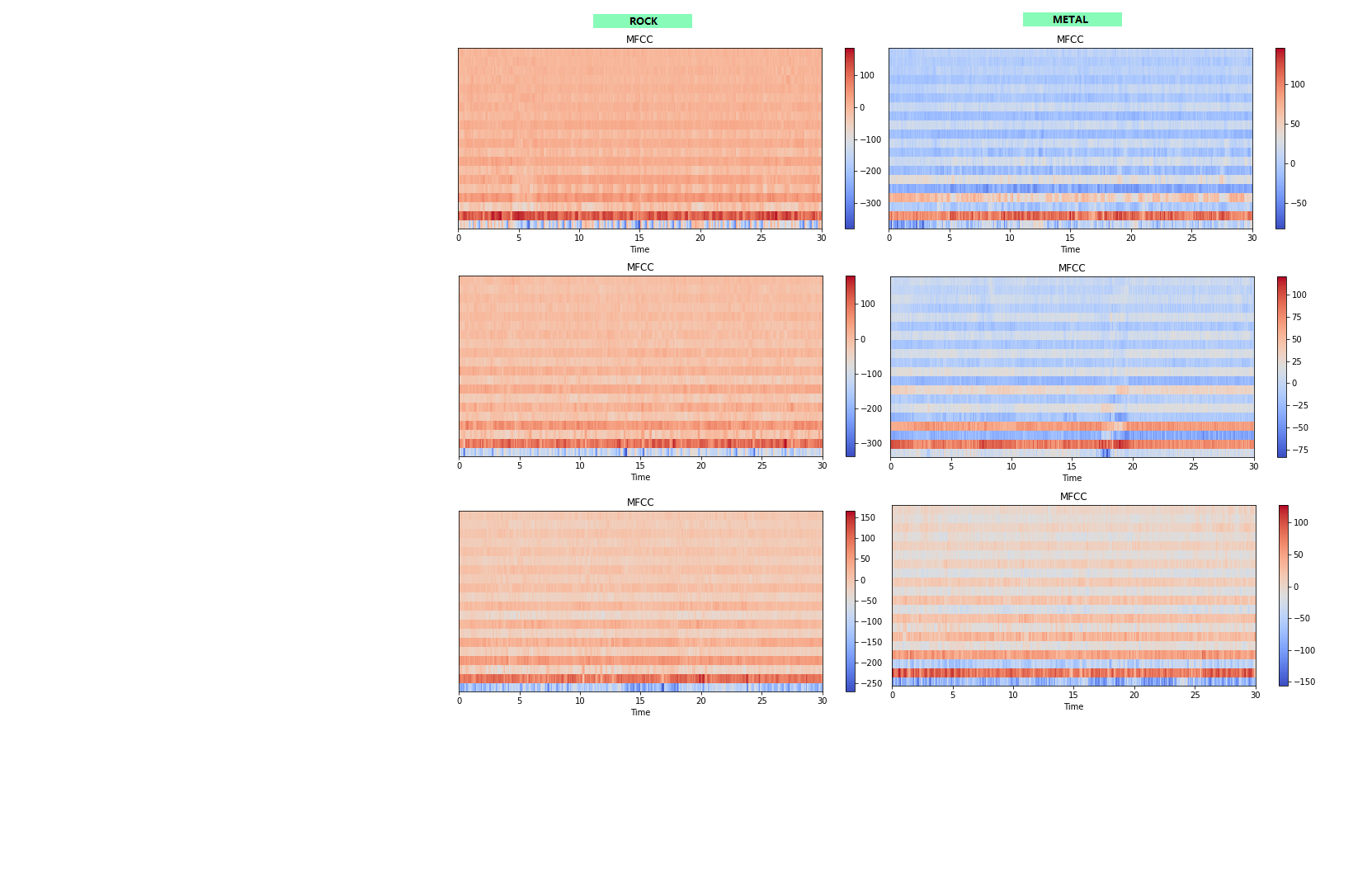


Рисунок 6 – спектрограмма MFCC для композиций жанров rock и metal

Достаточно бросить беглый взгляд на изображения мел-частотных коэффициентов, скажем, классических произведений и металла, чтобы заметить разницу не только в частоте их расположения на временной шкале, но и в их значениях, которые соответствуют интенсивности энергии спектра. На это указывает преобладание тёплых и холодных тонов на спектрограммах.

После проведения первичного визуального анализа изображений мел-спектра некоторых музыкальных композиций можно прийти к выводу о том, что значения MFCC в произведениях одного жанра действительно похожи, а также можно увидеть общие черты в частоте их расположения относительно временной шкалы. Данное заключение указывает на то, что эти характеристики звуковых сигналов однозначно можно использовать в качестве признаков объектов из обучающей выборки для их классификации.

## Описание алгоритма классификации

В ходе данной курсовой работы был реализован классификатор музыкальных композиций по жанрам на основе метода логистической регрессии. Этот алгоритм хорошо работает на небольшом количестве признаков обучающей выборки и удобно реализуется в языке программирования Python. Как и во многих случаях решения задач машинного обучения, большинство времени ушло на обработку данных, предназначенных для обучающей выборки.

На вход классификатору планировалось подать массив из 13 мел-частотных коэффициентов для каждого из 4135 кадров музыкального произведения. Но такой объём данных слишком велик для линейного алгоритма, если учитывать то, что в обучающей выборке более 500 файлов. Вместо этого коэффициенты усредняются по всем кадрам. Также предполагается, что несколько первых и последних секунд звучания в меньшей степени определяют жанр, чем середина, поэтому 10% звучания можно проигнорировать. Это нужно для более корректной работы классификатора. В итоге массив признаков для аудиофайла содержит небольшое число коэффициентов, равное числу коэффициентов, извлекаемых из каждого кадра.

Обучающие данные для классификатора подготовлены вручную и имеют структуру небольшого каталога: для каждого жанра создаётся папка с его именем в качестве названия, содержащая аудиофайлы в формате «.wav». Затем эти папки помещаются в родительскую папку. Причём в ней должны быть только подпапки-жанры.

В общем случае, алгоритм классификации работает следующим образом. Для каждой композиции в папке считаются мел-кепстральные коэффициенты, усредняются по всем кадрам и заносятся в массив, а название папки – жанра – заносится в массив признаков в качестве метки для данного файла. Коэффициенты для каждого произведения сохраняются в папке с помощью функции np.save() в файле с названием аудиофайла и разрешением «.npy». Это необходимо, чтобы не считать их каждый раз и избежать переобучения классификатора. Далее создается матрица признаков, содержащая массивы MFCC для композиций всех жанров и вместе с массивом меток подаётся алгоритму логистической регрессии для обучения.

Для тестовых композиций MFCC извлекаются и усредняются по аналогичной схеме и заносятся в матрицу признаков, а затем вызывается функция predict(X\_test, Y\_test), которая для каждого произведения рассчитывает вероятность принадлежности его к каждому классу. На выходе получаем массив меток – распознанные жанры.

# Анализ полученных результатов. Сравнение точности линейного и нелинейного классификатора

По окончании работы алгоритма печатается массив предсказанных жанров и значение точности классификации. Но в многоклассовых задачах важно не только узнать, насколько верно определён жанр произведения, но и выяснить, какие жанры путаются. Для этого можно воспользоваться одной из метрик оценки качества классификации – матрицей неточностей (confusion matrix). В ней содержится распределение меток, предсказанных на тестовом наборе для каждого жанра. На рисунке 7 представлена матрица ошибок для классификации 39 произведений.

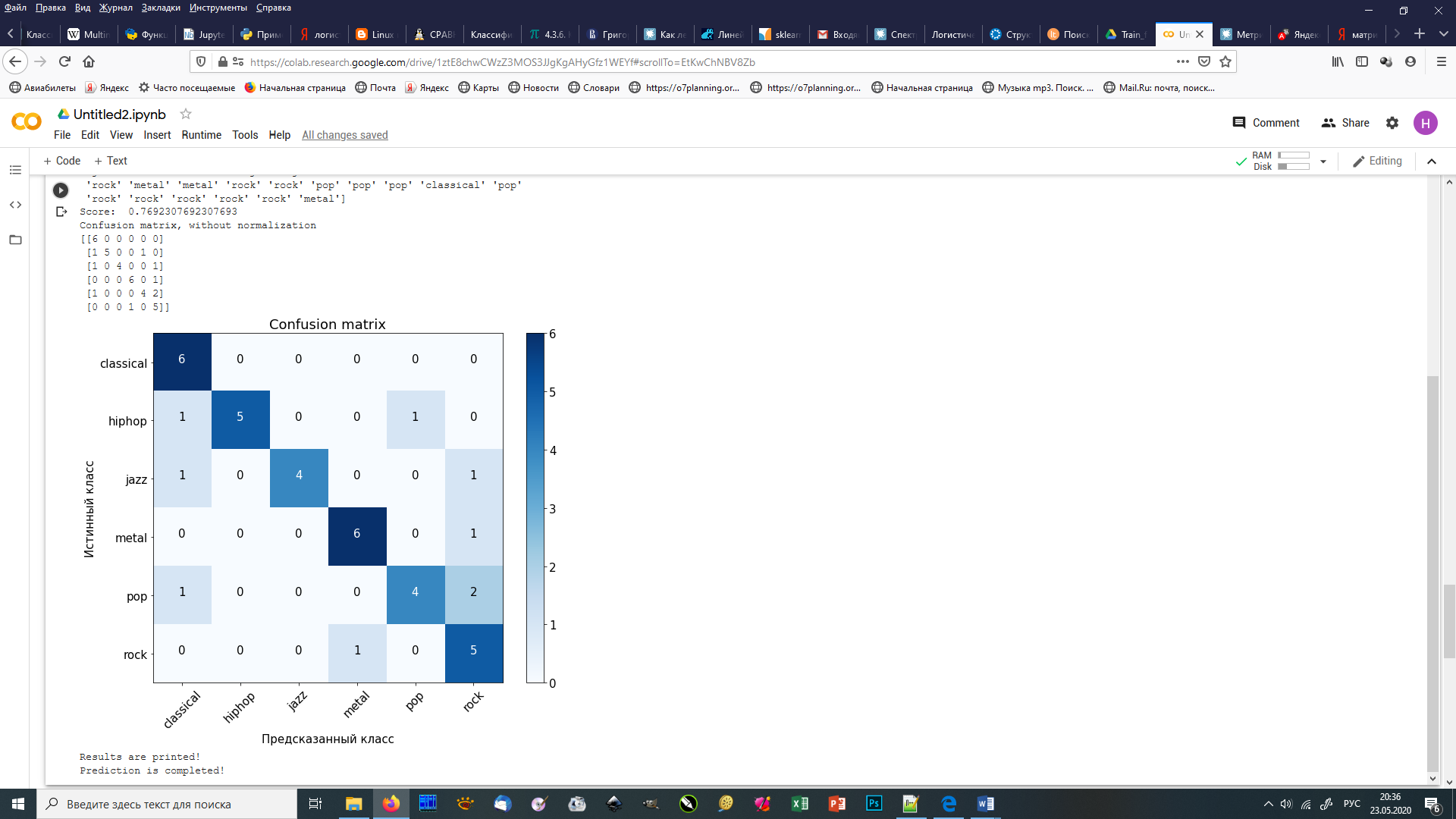


Рисунок 7 – матрица неточностей для линейного классификатора

Размер матрицы 6×6, так как в выборке шесть жанров. На диагонали находятся правильные результаты. Первая строка означает, что 6 из 6 классических произведений классифицированы верно, во второй строке 5 файлов жанра hiphop из 7 были распознаны верно, 1 – как классика и 1 – как pop. Итого из 39 произведений 30 были идентифицированы правильно. Точность классификации составляет 76,9%, что является достойным результатом для многоклассовой задачи. Поскольку наглядное представление воспринимается гораздо легче, чем просто напечатанные двумерные массивы nympy, лучше визуализировать матрицу неточностей (рисунок 8).

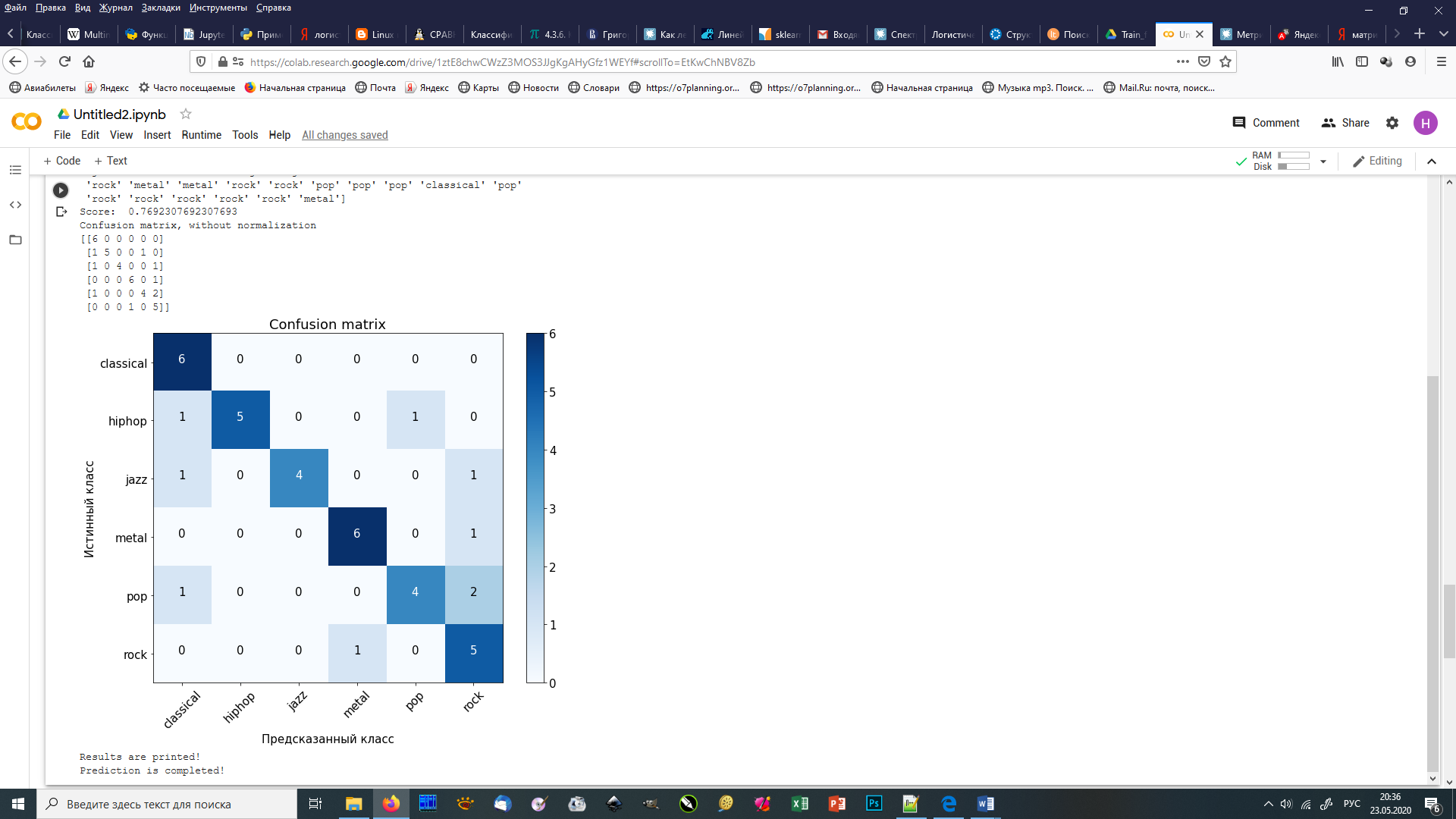


Рисунок 8 – визуализированная матрица ошибок

В первом случае мы извлекали по 13 коэффициентов из каждого аудиофайла. Попробуем увеличить их количество до 20, тогда на вход классификатору подаётся по 20 MFCC на каждое музыкальное произведение. Это немного увеличивает время обучения, но зато результаты, изображенные на рисунке 9, немного отличаются от предыдущих.

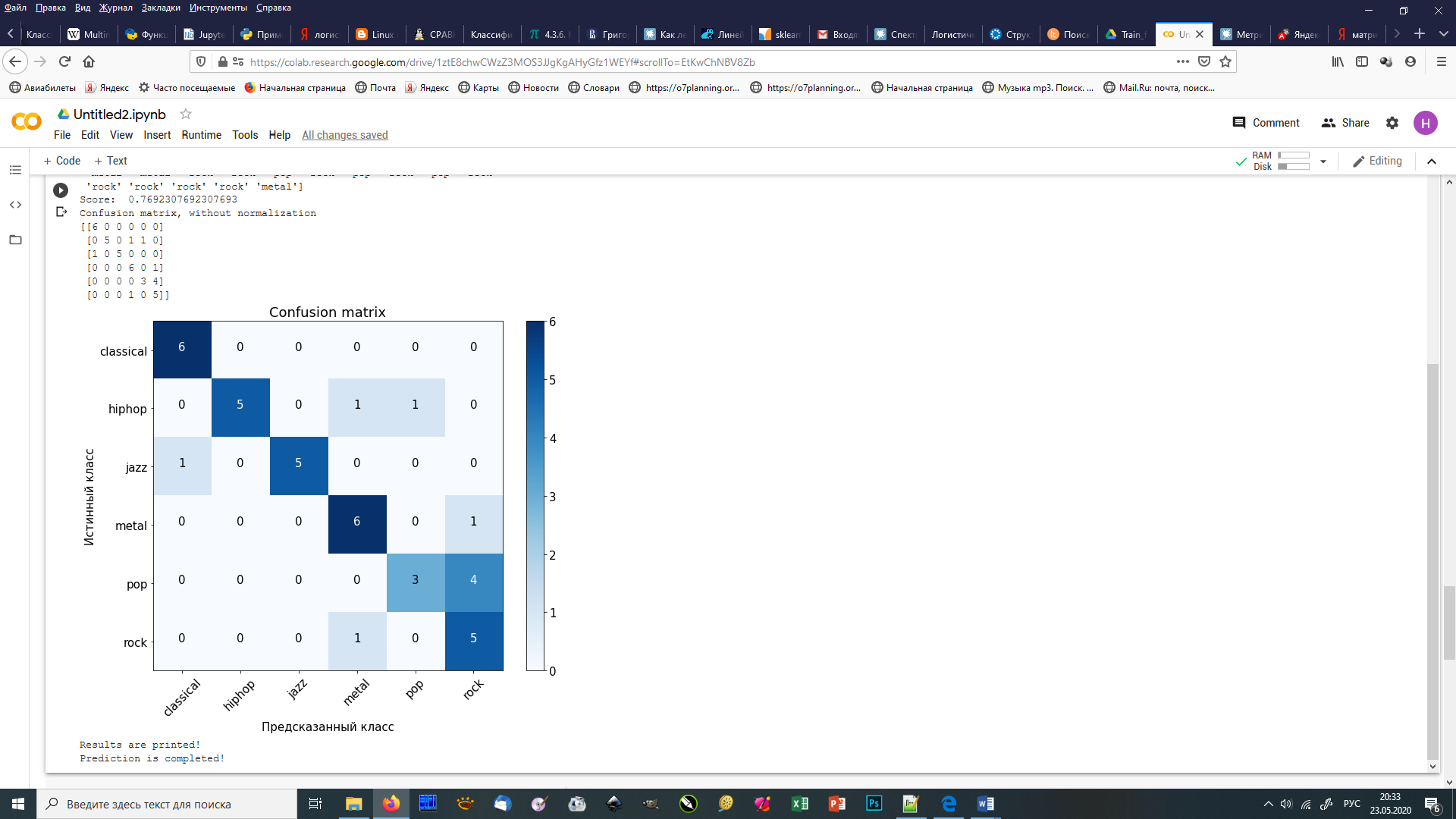


Рисунок 9 – матрица неточностей при количестве MFCC = 20

Не трудно заметить, что для жанра jazz предсказания стали на одну композицию точнее, тогда как в жанре pop увеличилось число неверно распознанных треков. Поэтому в данном случае точность также составляет 76%. Классификатор явно путает жанры pop и hiphop, хотя в реальности даже для человеческого слуха они могут быть неотличимы, а он является самым совершенным идентификатором на данный момент.

Теперь протестируем на том же наборе тестовых и обучающих данных нелинейный классификатор на основе Скрытых Марковских Моделей (СММ) [8]. Аналогично, результаты занесём в матрицу ошибок. Получаем гораздо более точный прогноз для тестовых композиций, представленный на рисунке 10.

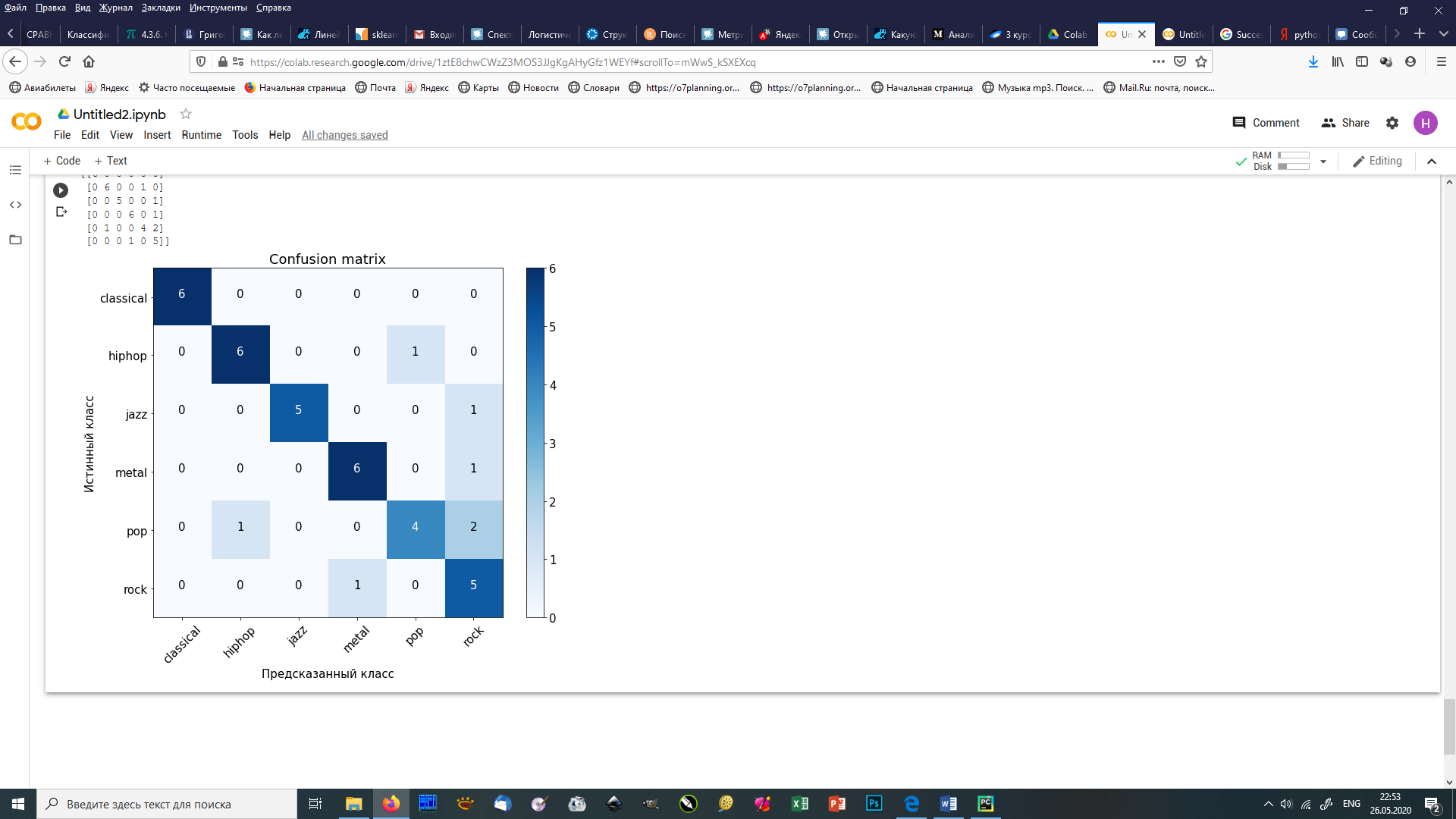


Рисунок 10 – результаты классификатора на основе СММ

Показатель точности идентификации составляет уже 82%, хотя алгоритм СММ верно распознал всего на две композиции больше, чем логистическая регрессия, то есть 32 из 39 треков. Коэффициенты в данном алгоритме не усредняются по всем кадрам, соответственно матрица признаков имеет очень большие размеры, ведь нелинейные методы машинного обучения как раз и рассчитаны на большое количество численных описаний объектов. Но стоит отметить, что время обучения и тестирования в данном случае в несколько раз превышает время работы линейного классификатора. Если классификатору на основе логистической регрессии требуется около 5 минут, чтобы обучиться и напечатать результат, то нелинейному классификатору со сложной моделью машинного обучения требуется на это около часа.

Теперь в тестовую выборку включим больше файлов, чтобы её объём был примерно 30% от всего объёма данных. В таких пропорциях обычно разделяют данные в машинном обучении: 30% тестовых и 70% обучающих. В каталоге 175 музыкальных композиций и 20 MFCC в массиве для каждого трека были получены результаты работы линейного классификатора, изображенные на рисунке 11.

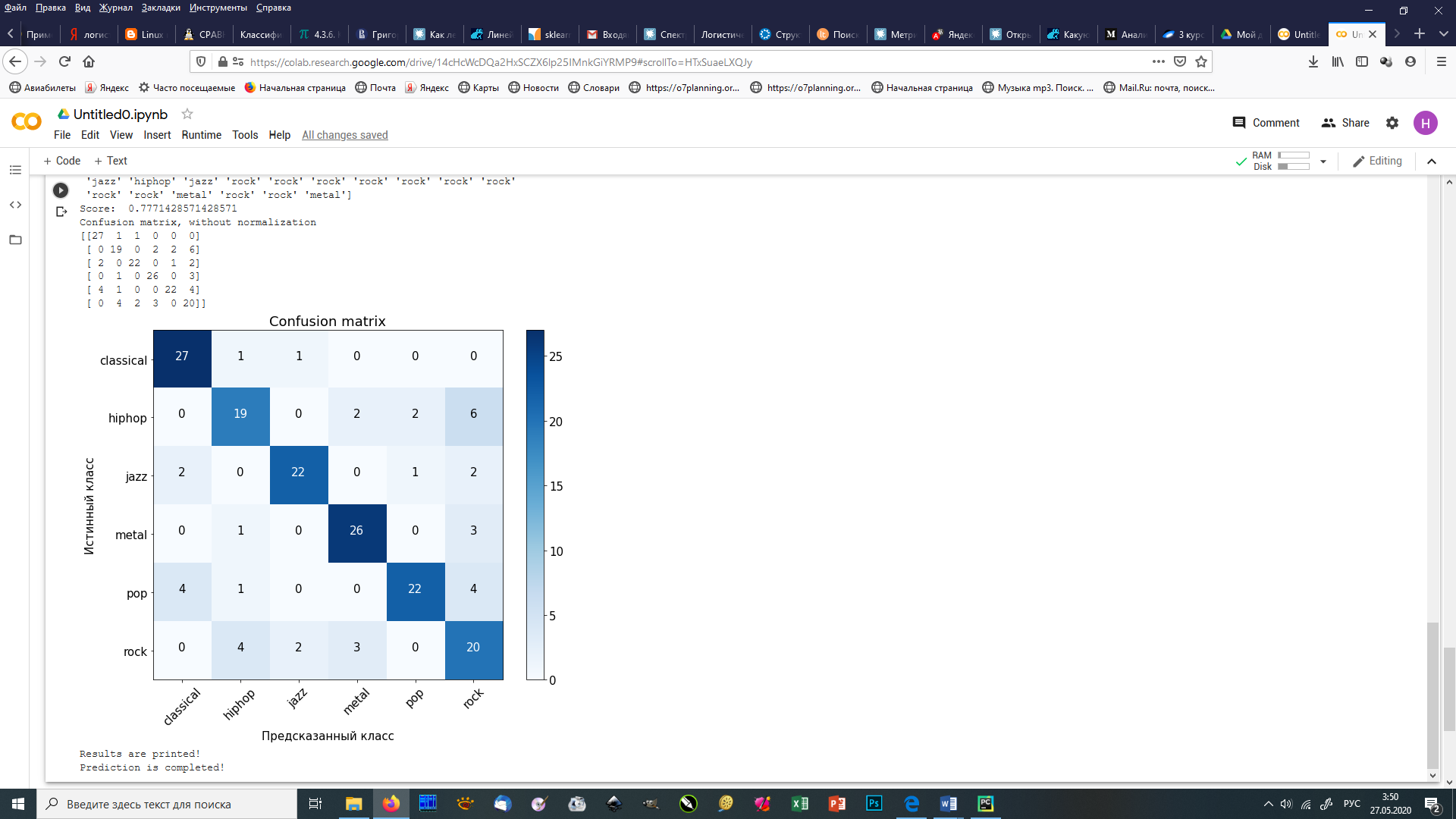


Рисунок 11 – результаты работы для 175 аудиофайлов

Теперь верно распознаны 136 из 175 треков и точность составляет 77,7%. Процесс обучения и тестирования классификатора также занимает относительно немного времени – около 15 минут.

На тех же данных классификатор на основе Скрытых Марковских моделей распознал верно 143 из 175 музыкальных произведений, матрица неточностей представлена на рисунке 12.

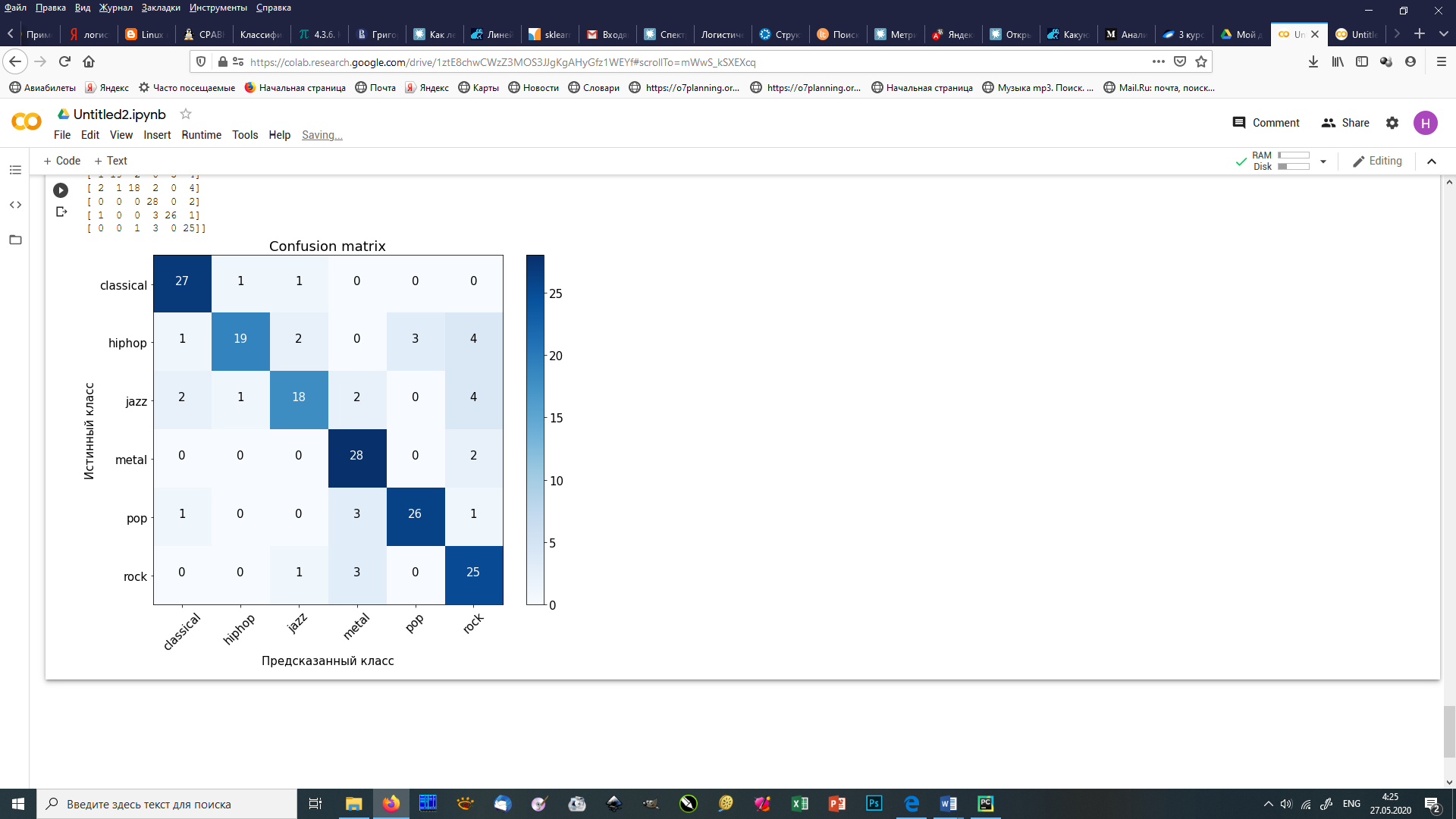


Рисунок 12 – результаты распознавания композиций нелинейным классификатором

Точность выше 81% является очень хорошим показателем работы алгоритма идентификации.

Можно было бы утверждать, что нелинейный классификатор, который составляет сложные модели для каждого класса, более гибкий с точки зрения точности, чем классификатор на основе логистической регрессии. Но построение подробных моделей отнимает очень много времени: обучение классификатора на основе СММ длилось около 10 часов. Достоинство линейных методов классификации в экономии времени обучения и проверки и в хорошей интерпретации.

# ЗАКЛЮЧЕНИЕ

В процессе выполнения работы был реализован на языке программирования Python классификатор музыкальных композиций по жанрам на основе логистической регрессии, проведено его обучение и тестирование на различных объемах данных. Также произведено сравнение результатов работы данного линейного классификатора и нелинейного на основе Скрытых Марковских моделей, в ходе которого сделан вывод о том, что линейные методы классификации показывают достаточно высокую точность распознавания жанра, лучше интерпретируются и имеют преимущество в минимальных временных затратах на обучение и проверку.

Полученные знания в анализе линейных и нелинейных алгоритмов машинного обучения в будущем планируется применить к совершенно иному набору данных. Они будут использоваться для распознавания объектов на фотографиях и видеозаписях, составляющих класс задач в новой предметной области – машинное зрение.

Задача идентификации музыкальных композиций является весьма сложной, потому что часто не существует общепринятых классификационных признаков того или иного жанра. Также сложность её состоит в том, что даже человек допускает ошибки в определении исполнителя или жанра композиции, а он является примером идеального идентификатора на данных момент.

В связи с ростом числа музыкальных интернет-сервисов и их постоянной заинтересованностью улучшить рекомендации для своих пользователей, подобная задача классификации музыкальных композиций по жанрам является весьма актуальной в настоящее время.

# СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ

1. Померанцев А. Классификация [Электронный ресурс]. − URL: <https://www.chemometrics.ru/old/Tutorials/classification.htm> (Дата обращения – 25.04.2020)
2. Открытый курс машинного обучения. Линейные методы классификации и регрессии [Электронный ресурс]. − URL: <https://habr.com/ru/company/ods/blog/323890> (Дата обращения – 25.04.2020)
3. Э. Хант Искусственный интеллект / пер. с англ. Д. А. Белова и Ю. Н. Крюкова, под ред. В.Л. Стефанюка. Издательство «Мир», 1978. – 558 с.
4. Скрытая Марковская Модель [Электронный ресурс]. URL: https://ru.wikipedia.org/wiki/Скрытая\_марковская\_модель (Дата обращения – 27.04.2020)
5. Алгоритм Баума-Велша [Электронный ресурс]. − URL: https://ru.wikipedia.org/wiki/Алгоритм\_Баума\_—\_Велша (Дата обращения – 27.04.2020)
6. Мел-кепстральные коэффициенты и распознавание речи [Электронный ресурс]. URL: https://habr.com/ru/post/140828/ (Дата обращения – 27.04.2020)
7. Коэльо Л.П., Ричарт В. Построение систем машинного обучения на языке Python. 2-е издание/ пер. с англ. Слинкин А.А. – М.: ДМК Пресс, 2016. – 302 с.: ил. ISBN 978-5-97060-330-7
8. Классификация музыкальных композиций по исполнителям с помощью скрытых марковских моделей [Электронный ресурс]. − URL: https://m.habr.com/ru/post/351462/ (Дата обращения – 29.11.2019)

# ПРИЛОЖЕНИЕ

# Листинг программы:

import librosa

from librosa.feature import mfcc

from scipy.io import wavfile

import sklearn.preprocessing

from sklearn.metrics import confusion\_matrix

import numpy as np

import os

from os import path

import random

from sklearn.linear\_model import LogisticRegression

import matplotlib.pyplot as plt

import itertools

import warnings

warnings.simplefilter('ignore', DeprecationWarning)

genre\_list = ['classical', 'hiphop', 'jazz', 'metal', 'pop', 'rock']

number\_of\_mfcc=24

model=LogisticRegression(C=10.0, class\_weight=None, dual=False, fit\_intercept=True,

                            intercept\_scaling=1, l1\_ratio=None, max\_iter=1000,

                            multi\_class='multinomial', n\_jobs=None, penalty='l2',

                            random\_state=None, solver='lbfgs', tol=0.0001, verbose=0,warm\_start=False)

#сохраняем mfcc в файл с разрешением .ceps

def write\_mfcc(mfcc\_array, filename):

    base\_fn,ext = os.path.splitext(filename)

    data\_fn=base\_fn+".ceps"

    np.save(data\_fn,mfcc\_array)

    print("Written to %s" % data\_fn)

# Визуализация матрицы неточностей:

def plot\_confusion\_matrix(cm, classes,

                          normalize=False,

                          title='Матрица неточностей',

                          cmap=plt.cm.Blues):

    """

    This function prints and plots the confusion matrix.

    Normalization can be applied by setting `normalize=True`.

    """

    plt.imshow(cm, interpolation='nearest', cmap=cmap)

    plt.title(title)

    plt.colorbar()

    tick\_marks = np.arange(len(classes))

    plt.xticks(tick\_marks, classes, rotation=45)

    plt.yticks(tick\_marks, classes)

    if normalize:

        cm = cm.astype('float') / cm.sum(axis=1)[:, np.newaxis]

        print("Normalized confusion matrix")

    else:

        print('Confusion matrix, without normalization')

    print(cm)

    thresh = cm.max() / 2.

    for i, j in itertools.product(range(cm.shape[0]), range(cm.shape[1])):

        plt.text(j, i, cm[i, j],

                 horizontalalignment="center",

                 color="white" if cm[i, j] > thresh else "black")

    plt.tight\_layout()

    plt.ylabel('Истинный класс')

    plt.xlabel('Предсказанный класс')

scale=True

def getFeaturesFromWAV(filename):

    audio, sampling\_freq = librosa.load(filename, sr=None, res\_type='scipy')

    features = librosa.feature.mfcc(audio, sampling\_freq, n\_mfcc=number\_of\_mfcc, n\_fft=1024, hop\_length=512)

    if scale:

        features = sklearn.preprocessing.scale(features)

    return features.T

def Training(dataFolder):

    if debug\_mode:

        print('Current folder is ' + dataFolder)

    lstdirs = [os.path.join(dataFolder, dirname) for dirname in os.listdir(dataFolder) if os.path.isdir(os.path.join(dataFolder, dirname))]

    noSubFolders = not lstdirs

    if noSubFolders:

        lstdirs.append(dataFolder)

    X=np.array([]) #общий массив признаков

    y1=[] #массив меток

    count\_of\_songs=0

    for subfolder in lstdirs:

        if not os.path.isdir(subfolder):

            continue

        if debug\_mode:

            print(subfolder + ' training is started!')

        label = subfolder[subfolder.rfind('/') + 1:] #label=genre=subfolder

        #featureMatrix = np.array([]) # матрица признаков для каждого жанра

        #для каждого файла из subfolder считаем мфсс и кешируем их

        for filename\_n in [x for x in os.listdir(subfolder) if x.endswith('.wav')]:

            filepath = os.path.join(subfolder, filename\_n)

            featureMatrix = np.array([])

            count\_of\_songs +=1

            features=getFeaturesFromWAV(filepath)

            num\_mfcc=len(features)

            featureMatrix = np.append(featureMatrix, features, axis=0) if len(featureMatrix) != 0 else features

              #усредняем коэффициенты по всем кадрам

            averaged\_mfcc=np.mean(featureMatrix[int(num\_mfcc\*0.1):int(num\_mfcc\*0.9)],axis = 0)

            print('averaged\_mfcc = ', averaged\_mfcc)

            write\_mfcc(averaged\_mfcc,filepath)

            # записываем коэффициенты для всех файлов класса в массив признаков

            X=np.append(X,averaged\_mfcc,axis=0)

            y1.append(label) #метка присваивается каждому файлу

        all\_mfcc=len(X)

    X=[X[i:i+number\_of\_mfcc] for i in range(0,all\_mfcc,number\_of\_mfcc)]

    y = np.array(y1)

    model.fit(X,y) #обучаем модель

    if noSubFolders and debug\_mode:

        print(dataFolder + ' training is completed!')

def Prediction(folder\_name):

    kol\_vo\_songs=0

    if debug\_mode:

        print('Tests are started!')

    X\_test=np.array([]) #общий массив признаков

    Y\_true\_labels=np.array([]) #массив истинных значений для X\_test

    for input\_file in [x for x in os.listdir(folder\_name) if x.endswith('.wav')]:

        if debug\_mode:

            print('Current file is ' + input\_file)

        filepath = os.path.join(folder\_name, input\_file)

        kol\_vo\_songs+=1

        featureMatrix = np.array([]) # матрица признаков для каждого файла

        features = getFeaturesFromWAV(filepath)

        num\_mfc=len(features)

        #находим имя файла каждой песни (без разрешения)

        strr=''

        short\_name = path.splitext(input\_file)[0]

        short\_name = path.splitext(short\_name)[0]

        strr = strr+short\_name

        lab = [strr]

        Y\_true\_labels =np.append(Y\_true\_labels,lab,axis=0)

        featureMatrix = np.append(featureMatrix, features, axis=0) if len(featureMatrix) != 0 else features

        #усредняем коэффициенты по всем кадрам

        averaged\_mfc=np.mean(featureMatrix[int(num\_mfc\*0.1):int(num\_mfc\*0.9)],axis = 0)

        X\_test=np.append(X\_test,averaged\_mfc,axis=0)

    print('Y\_true\_labels', Y\_true\_labels)

    all\_coef=len(X\_test)

    # разделяем массив коэффициентов на кол-во частей = кол-ву файлов

    X\_test=[X\_test[i:i+number\_of\_mfcc] for i in range(0,all\_coef,number\_of\_mfcc)] #((адекватно сделать))

    predict\_y=model.predict(X\_test)

    #predict\_proba\_y=model.predict\_proba(X\_test)

    print('predict\_y',predict\_y)

    #print(predict\_proba\_y)

    res=model.score(X\_test,Y\_true\_labels) #???

    print('Score: ',res)

    # Матрица неточностей

    font = {'size' : 15}

    plt.rc('font', \*\*font)

    #plt.savefig("conf\_matrix.png")

    plt.figure(figsize=(10, 8))

    cnf\_matrix = confusion\_matrix(Y\_true\_labels,predict\_y)

    plot\_confusion\_matrix(cnf\_matrix, genre\_list, title='Confusion matrix')

    plt.show()

    # вызываем функцию Визуализации матрицы неточностей

    if debug\_mode:

        print('Results are printed!')

#configi

genre\_dir = '/content/drive/My Drive/Train\_folder'

test\_dir = '/content/drive/My Drive/Test\_2'

#main

debug\_mode=True

Training(genre\_dir)

print('Training is completed!')

Prediction(test\_dir)

print('Prediction is completed!')