МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ
РОССИЙСКОЙ ФЕДЕРАЦИИ

Федеральное государственное бюджетное образовательное
учреждение высшего образования
«САНКТ-ПЕТЕРБУРГСКИЙ ГОСУДАРСТВЕННЫЙ
ЭКОНОМИЧЕСКИЙ УНИВЕРСИТЕТ»

Институт магистратуры

Кафедра прикладной математики

и экономико-математических методов

ГОДОВАЯ КУРСОВАЯ РАБОТА

Тема:

Алгоритмы классификации музыкальных произведений

Направление магистерской подготовки:

01.04.02 «Прикладная математика и информатика»

Магистерская программа:

Математическое и компьютерное моделирование в экономике и управлении

Магистрант: Солуянова Наталия Павловна_____

Группа ПМ-1841 Подпись _____

Руководитель:

Лебедева Людмила Николаевна, к. ф.-м. н.,

доцент кафедры прикладной математики и экономико-математических методов

Оценка _____ Подпись _____

Санкт-Петербург

2019 г.

Содержание

ВВЕДЕНИЕ

Задача классификации музыкальных произведений актуальна для любого стримингового сервиса, поскольку очень важно учитывать интересы пользователя и либо составлять списки рекомендаций, основанные на похожих жанрах, либо позволять ему самостоятельно осуществлять поиск по вкусу в базе музыкальных композиций.

При написании годовой курсовой работы мне было необходимо ознакомиться с существующими подходами к классификации музыкальных композиций, рассмотреть и сравнить различные алгоритмы и предложить какие-то свои.

Целью работы являлось определение лучшего алгоритма многоклассовой классификации музыкальных композиций по нескольким метрикам качества. Для достижения цели необходимо было решить следующие задачи:

- Подготовка данных
- Обработка данных
- Построение алгоритмов классификации
- Сравнение алгоритмов

Работа проводилась для данных, взятых из открытого источника “Free Music Archive” [1]. Авторами этого архива были собраны различные статистические, амплитудно-частотные и просто информационные характеристики более 100 тысяч песен разных лет и разных жанров, и на основании этих показателей после некоторых преобразований мною были построены алгоритмы классификации, которые подробно описаны в последующих разделах.

1. ОПИСАНИЕ ОБЪЕКТА ИССЛЕДОВАНИЯ

1.1. Аналитический обзор источников

Задача музыкальной классификации решается чаще всего в целях коммерческого использования, то есть исследователи делятся результатом, но не подробностями его достижения, поэтому в данном разделе будут рассмотрены подходы к решению задач не только музыкальной классификации, но и других сфер, где применимы те же алгоритмы анализа данных.

Например, в работе [4] описано применение логистической регрессии при прогнозировании вероятности наступления инфаркта у пациентов, уже перенёсших инфаркт, имеющих предрасположенность к инфаркту, а также у здоровых пациентов двух типов. Сама по себе логистическая регрессия предсказывает вероятность принадлежности к классу, что подходит для решения задачи многоклассовой классификации.

Теоретические аспекты и пример использования алгоритма XGBoost (EXtreme Gradient Boosting) изложены в статье [5], а в статье [6] приведён простой пример работы этого алгоритма для диагностики болезни Паркинсона.

Алгоритм “Случайный лес” (Random Forest) достаточно подробно описан в статье [7]. Это один из самых популярных и, что немаловажно, универсальных методов машинного обучения, который подходит для большинства задач.

Также был рассмотрен алгоритм бэггинга, описанный в статьях [8] и [9]. Бэггинг – это метод классификации, использующий композиции элементарных классификаторов, обучающихся независимо друг от друга, при этом результат определяется голосованием, как и у случайного леса. Классификаторы не исправляют ошибки друг друга, а компенсируют их при голосовании.

Все эти методы были применены в решении задачи многоклассовой классификации музыкальных композиций по разным жанрам, и было произведено сравнение результатов работы по нескольким метрикам, о чём будет рассказано далее.

1.2. Обработка данных

С сайта [2] был сохранён большой объём данных о 106574 песнях 16341 артиста, находящихся в 14854 альбомах и иерархически разделённых на 161 жанр. Во время дальнейшей работы использовались только два csv-файла: features.csv, содержащий техническую информацию о каждой песне (амплитудно-частотные характеристики и различные статистические показатели; как они устроены и получены, подробнее описано в источниках [11] и [12]), и tracks.csv, где находится информация о дате выхода песни, альбома, об исполнителе, его геолокации, основном жанре и поджанрах.

Исследование проводилось при помощи Jupyter Notebook – интерактивной среды языка Python. На рисунке 1 показано подключение необходимых для работы библиотек.

```
import numpy as np
import pandas as pd

import matplotlib.pyplot as plt
import seaborn as sns
%matplotlib inline
import re
import itertools
import warnings
import string
import math

warnings.filterwarnings('ignore')

plt.rcParams["figure.figsize"] = (20,8)
```

Рисунок 1 – Подключение библиотек

После подключения библиотек были прочитаны вышеназванные файлы, но некоторые столбцы таблиц, то есть признаки, не несли никакой полезной для классификации информации, поэтому были удалены. Это коснулось только файла с преимущественно текстовой информацией о песне или об артисте. Также нас не интересовала точная дата выхода трека или альбома (тем более что много где она была неизвестна или стояло 1 января), и был оставлен только год, а из жанров взят только самый главный, без входящих в него. На рисунке 2 с комментариями показаны эти преобразования.

```
df = pd.read_csv('features.csv', sep=',')#features
dt = pd.read_csv('tracks.csv', sep=',')#tracks
#dt.info()#сколько ненулевых
#dt.columns#имена столбцов
dt=dt.dropna(subset=['album.2'])#убрали песни без года выпуска
#dt.info()#сколько теперь ненулевых

for i in range(2,70295):
    try:
        dt['album.2'][i]=dt['album.2'][i][:4]#оставим только год
    except:
        i+=1

#dt['track.19'][:4]#смотрим на содержимое
#удаляем ненужное
del dt['album']
del dt['album.1']
del dt['album.3']
del dt['album.4']
del dt['album.5']
del dt['album.6']
del dt['album.7']
del dt['album.8']
del dt['album.9']
del dt['album.10']
del dt['album.11']
del dt['album.12']
del dt['artist']
del dt['artist.1']
del dt['artist.2']
del dt['artist.3']
del dt['artist.4']
del dt['artist.5']
del dt['artist.6']
del dt['artist.7']
del dt['artist.9']
del dt['artist.11']
del dt['artist.12']
del dt['artist.13']
del dt['artist.14']
del dt['artist.15']
del dt['artist.16']
del dt['set']
#del dt['set.1']
del dt['track']
del dt['track.1']
del dt['track.2']
del dt['track.3']
del dt['track.4']
del dt['track.5']
del dt['track.6']
del dt['track.8']
del dt['track.9']
del dt['track.10']
del dt['track.11']
del dt['track.12']
del dt['track.13']
del dt['track.15']
del dt['track.16']
del dt['track.17']
```

Рисунок 2 – Первое преобразование данных

На рисунке 3 с подробными комментариями показано соединение двух таблиц в одну для дальнейшей работы.

```
cols = dt.columns.tolist()
cols1=['Unnamed: 0','album.2','artist.8','artist.10','set.1','track.14','track.7']#меняем порядок столбцов
dt=dt[cols1]
#dt.head(6)
df = df[df["feature"] != 'number']#убираем лишнюю строчку
dt.loc[0][0]='statistics'#меняем значение ячейки, чтобы было такое же, как в df
dt.rename(columns={'Unnamed: 0':'feature'},inplace=True)#меняем имя, чтобы было как в dt
ft=pd.merge(df, dt, on='feature', how='outer')#склеиваем два датафрейма по столбцу feature с track_id
ftnames=[ft.columns[i] for i in range(525)]#имена столбцов
#ft[ftnames[500:]].info()#видим,что из 100К размечены только 30К
ft=ft.dropna(subset=['track.7'])#оставляем только те песни, где указан жанр
#ft.head()
#ft[ftnames[500:]].info()#снова смотрим и видим, что показатели собраны у 26961 песни
ft=ft.dropna(subset=['zcr'])#оставляем только те песни, у которых собраны показатели
#ft[ftnames[500:]].info()#снова смотрим и видим, что есть два признака с большим количеством отсутствующих наблюдений
#ft['artist.8'][0]#это оказываются широта и долгота

#пусть широта, долгота и подвыборка пока не нужны
del ft['artist.8']
del ft['artist.10']
del ft['set.1']

ftnames1=[ft.columns[i] for i in range(522)]
#ft[ftnames1[500:]].info()#теперь все данные есть
colnames1=ft.columns
```

Рисунок 3 – Составление одной таблицы из двух датафреймов

Из рисунка 3 видно, что у большей части песен жанр неизвестен, поэтому были оставлены только те, у которых эта самая важная для данной задачи информация есть.

Этап подготовки данных завершён, теперь можно приступать к их анализу.

2. АНАЛИЗ ДАННЫХ

2.1. Представление данных в наглядном виде

На рисунке 4 представлено, сколько песен какого жанра в наших данных. Видно, что Rock, Experimental и Electronic заметно превалируют над остальными, а также есть совсем малочисленные в масштабах 30 тысяч треков Country, Blues и Easy Listening.

```
#ft['track.7'].unique()#сколько есть жанров
interest=[i for i in range(1,522)]#интересующие нас столбцы - все, кроме первого с track_id
interestnames=[colnames1[i] for i in interest]

#можно посмотреть на распределения
#for i in range(len(interest)):
#    plt.figure(figsize=(3,3))
#    sns.distplot(pd.to_numeric(ft[colnames1[i]][2:].dropna(), errors='ignore'),color='red',bins=50)
#    plt.title('Распределение по признаку "%s"' % interestnames[i])
#    plt.xlabel('Значение')
#    plt.ylabel('Частота')
ft1=ft[interestnames].loc[2:]#убрали вторую строку заголовка
ft1=pd.DataFrame([pd.to_numeric(ft1[i], errors='coerce') for i in interestnames[:-1]]).transpose()#превращаем значения в числовы
ft1[interestnames[-1]]=ft[interestnames[-1]]#значения таргета не меняются
#ft1.head()
#ft1[interestnames1[500:]].info()#теперь всё в порядке
ft1['track.7'].value_counts()#классы не сбалансированы
```

```
Rock              7566
Experimental      6726
Electronic        5095
Hip-Hop           2243
Folk              1308
Pop               1065
Instrumental      1016
International      629
Classical          542
Spoken             300
Jazz               249
Soul-RnB           112
Country             52
Blues               41
Easy Listening      16
```

Рисунок 4 – Распределение песен по жанрам

Наверняка такое различие в количестве композиций с чем-то связано, поэтому стоит посмотреть, сколько песен какого жанра выходило каждый год. На рисунке 5 показана произведённая при помощи функции LabelEncoder() кодировка жанров, чтобы таблица не оказалась слишком широкой, а на рисунке 6 – собственно таблица с интересующей информацией.

```
[['Blues', 0],
 ['Classical', 1],
 ['Country', 2],
 ['Easy Listening', 3],
 ['Electronic', 4],
 ['Experimental', 5],
 ['Folk', 6],
 ['Hip-Hop', 7],
 ['Instrumental', 8],
 ['International', 9],
 ['Jazz', 10],
 ['Pop', 11],
 ['Rock', 12],
 ['Soul-RnB', 13],
 ['Spoken', 14]]
```

Рисунок 5 – Кодировка жанров

```
t=ft1.pivot_table(values=['track.14'], index=['album.2'],columns=['track.7'], aggfunc='count',fill_value='')#Вместо track.14
t
```

	track.14														
track.7	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14
album.2															
1968.0							12								
1971.0													7		
1973.0													5		
1976.0						2					2		9		
1979.0													4		
1980.0													2		
1981.0													6		
1982.0													19		
1984.0													11		
1985.0													8		
1986.0													10		
1987.0												14			
1990.0													10		
1991.0													6		
1992.0													15		
1993.0													61		
1994.0						3							17		
1995.0					15	5					2		83		
1996.0						9			5				29		
1997.0						6	1		11	11			23		
1998.0						66							15		
1999.0					5	4	3						15		
2000.0						17	1				3		37		
2001.0					6	19	12		1		3		3	9	23
2002.0					26	65	4						23		
2003.0					27	20	2	1		8	22	9	22	13	
2004.0					12	80			13				63	6	
2005.0	3	11			73	74	3		4	5			41	8	
2006.0	8			2	132	77	3		1	15	21	1	86	2	
2007.0		12			146	88	4	45	39	13		25	107		1
2008.0		40			193	162	17		27	4	7	18	145		
2009.0	7	16			266	207	69	76	7	12	30	19	331	6	75
2010.0		171			630	736	162	190	112	44	37	103	723	13	17
2011.0	2	18			618	726	129	278	98	161	42	136	1158	8	54
2012.0		182	19		589	778	125	311	34	46	31	120	954	9	27
2013.0	5	12	1	6	561	901	128	251	95	31	2	132	868	4	8
2014.0					224	362	83	149	16	62	5	12	319		9
2015.0													2		
2016.0						7									

Рисунок 6 – Распределение количества песен по жанрам каждый год

На рисунке 7 показана гистограмма распределения песен за всё время, то есть уже известная общая информация в наглядной форме.

```
plt.figure(figsize=(20,4))
sns.countplot(x=ft1['track.7'], data=ft1)
plt.xticks(rotation='horizontal')
plt.ylabel('Количество', fontsize=12)
plt.xlabel('Жанр', fontsize=12)
plt.title('Количество песен во всех жанрах за всё время')
plt.show()
```

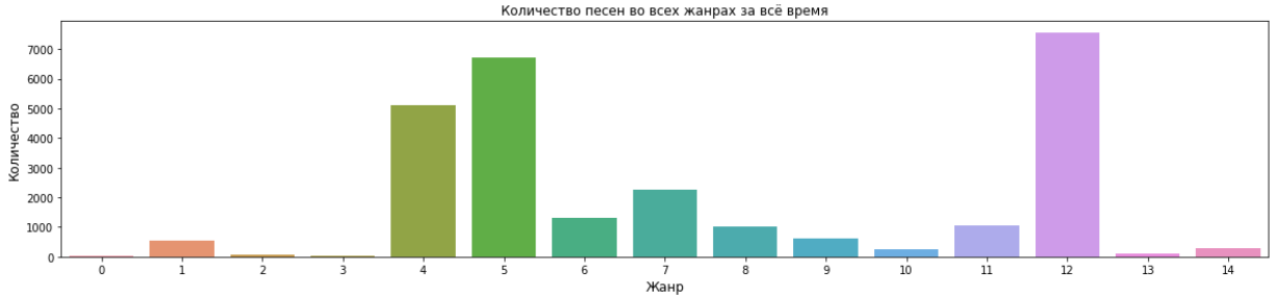

Рисунок 7 – Распределение количества песен по жанрам за всё время

Можно строить такие гистограммы за каждый год, но из таблицы на рисунке 6 видно, что очень долго будет практически один рок и развитие разнообразия жанров в данном датасете начнётся к концу девяностых-началу двухтысячных годов, поэтому интереснее построить гистограммы по каждому жанру, чтобы посмотреть более конкретно, когда они начали развиваться и приобретать популярность. Следующие рисунки 8-22 как раз это демонстрируют.

```
#Года по жанрам
for i in list(ft1['track.7'].unique()):
    plt.figure(figsize=(20,4))
    sns.countplot(x=ft1[ft1['track.7']==i]['album.2'], data=ft1[ft1['track.7']==i])
    plt.xticks(rotation='horizontal')
    plt.ylabel('Количество', fontsize=12)
    plt.xlabel('Год', fontsize=12)
    plt.title('Количество песен по годам в жанре %s'%names[i][0])
    plt.show()
```

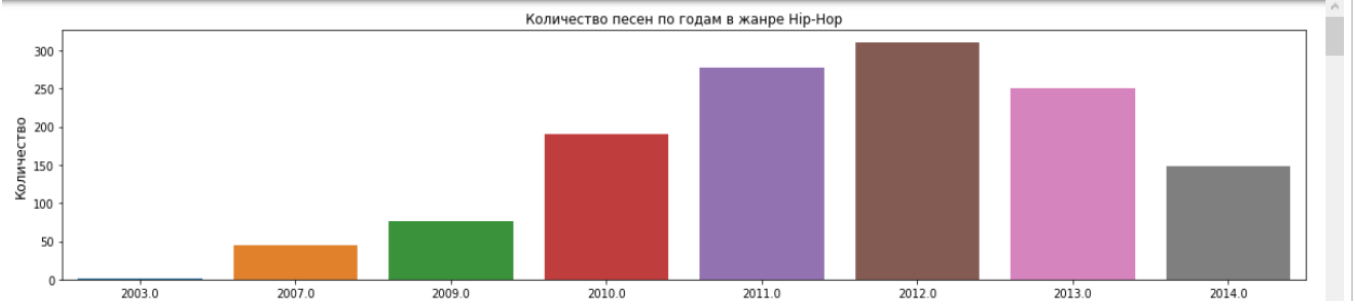

Рисунок 8 – Количество песен по годам в жанре Hip-Hop

Из рисунка 8 видно, что хип-хоп начал набирать популярность во второй половине нулевых и продолжил в десятых.

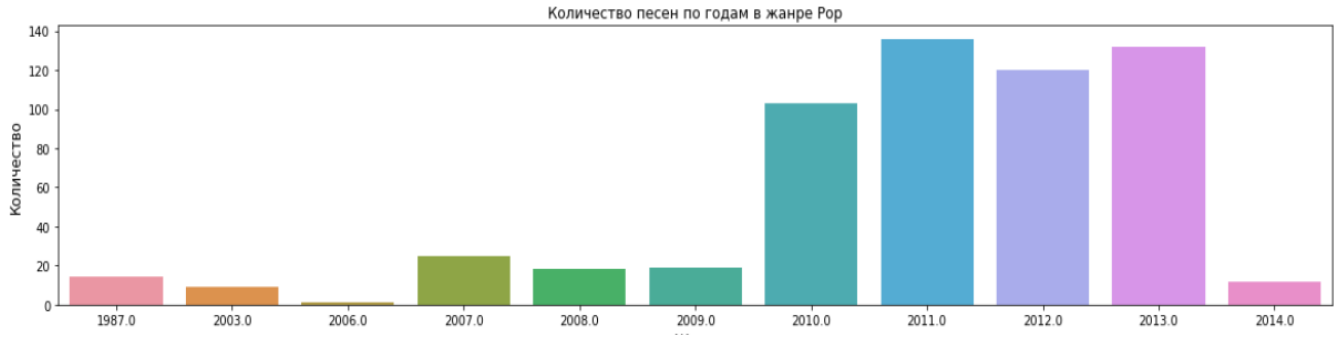

Рисунок 9 – Количество песен по годам в жанре Pop

На рисунке 9 показано, что количество поп-песен в данном датасете резко возросло, начиная с 2010 года.

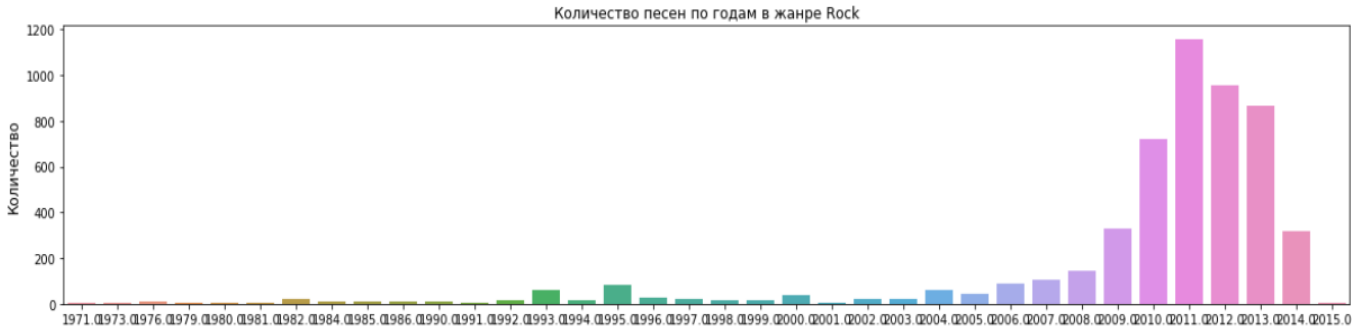

Рисунок 10 – Количество песен по годам в жанре Rock

Из рисунка 10 видно, что практически 45 лет подряд за каждый год в таблице присутствуют рок-композиции, однако заметно много их стало тоже в начале десятых годов.

Рисунок 11 – Количество песен по годам в жанре Experimental

Похожая ситуация с популярностью жанра Experimental на рисунке 11: количество песен заметно возросло в десятых годах.

Рисунок 12 – Количество песен по годам в жанре Folk

Жанр Folk на рисунке 12 тоже не удивляет ничем новым.

Рисунок 13 – Количество песен по годам в жанре Jazz

Джазовые композиции на рисунке 13 как раз удивляют отсутствием своего наличия во второй половине прошлого века и ростом количества в десятых годах этого.

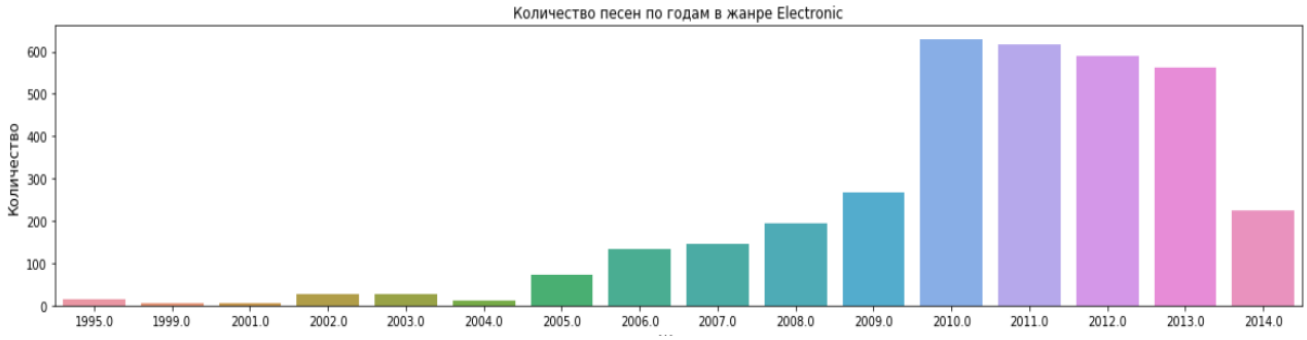

Рисунок 14 – Количество песен по годам в жанре Electronic

Электронная музыка, как видно из рисунка 14, популяризировалась уже с середины нулевых, но количество песен резко выросло в десятых.

Рисунок 15 – Количество песен по годам в жанре Spoken

Жанр Spoken на рисунке 15 представлен отнюдь не большим количеством песен, и нельзя однозначно утверждать, с чем связано именно такое их распределение по годам.

Рисунок 16 – Количество песен по годам в жанре International

Как видно из рисунка 16, про жанр International можно сказать, что большая часть этих песен была выпущена в 2011 году и их количество после 2010 заметно больше, чем раньше.

Рисунок 17 – Количество песен по годам в жанре Soul-RnB

Ввиду своей малочисленности жанр Soul-RnB (рисунок 17) в сравнении с другими имеет гораздо меньшие различия от года к году.

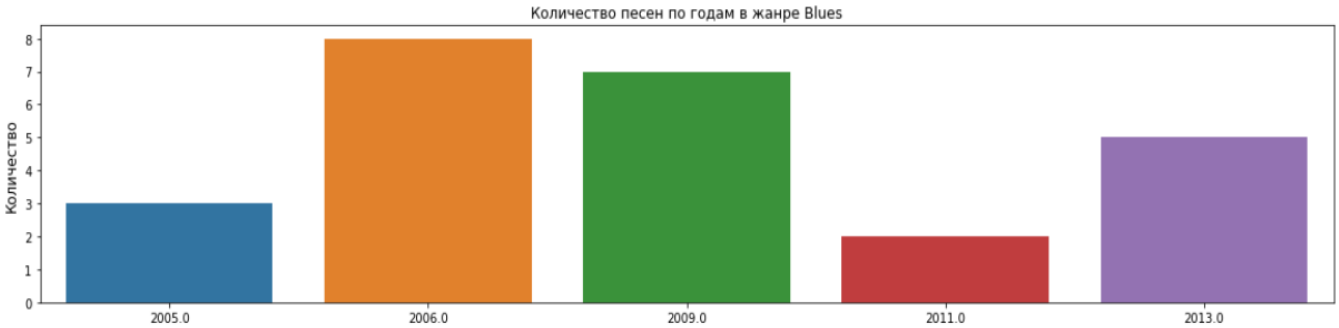

Рисунок 18 – Количество песен по годам в жанре Blues

В некоторых музыкальных сервисах (например, «ВКонтакте») жанры джаз и блюз объединены, в данном датасете – нет, поэтому оба достаточно малочисленны. Впрочем, и при объединении ситуация изменилась бы не сильно, поскольку блюзовых композиций всего 41 (рисунок 18).

Рисунок 19 – Количество песен по годам в жанре Classical

Из рисунка 19 не очень понятно, как классическая музыка попала в XXI век, но всё же много песен в этом жанре было выпущено в 2010 и 2012 годах.

Рисунок 20 – Количество песен по годам в жанре Instrumental

Инструментальная музыка (рисунок 20) занимает примерно 3%-ю долю от всех треков, то есть этот жанр не является совсем малочисленным, и больше песен было выпущено после 2010 года.

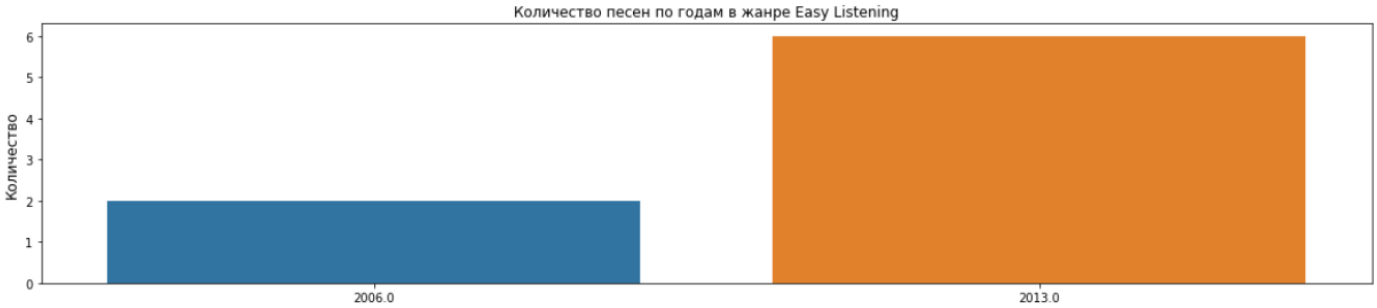

Рисунок 21 – Количество песен по годам в жанре Easy Listening

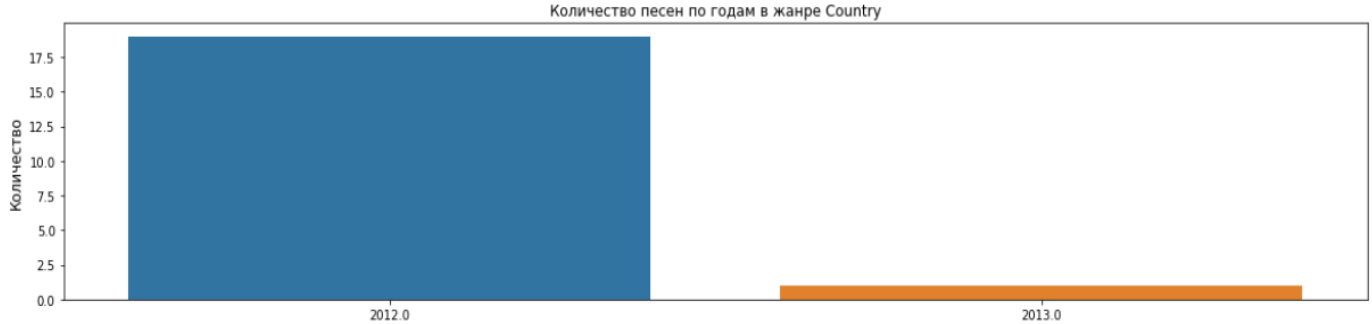

Рисунок 22 – Количество песен по годам в жанре Country

На рисунках 21 и 22 представлены самые малочисленные жанры Easy Listening и кантри, и их распределения не поддаются объяснению.

Посмотрев на распределения по годам каждого жанра, можно предположить, что, вероятно, заметный скачок в количестве композиций во всех жанрах обусловлен местом, откуда были взяты данные: скорее всего, его начали активно заполнять в десятых годах нынешнего века – и именно преимущественно современными песнями.

На рисунке 23 можно увидеть распределение по жанрам в 2013 году, это один из немногих лет, когда выпускались песни всех жанров.

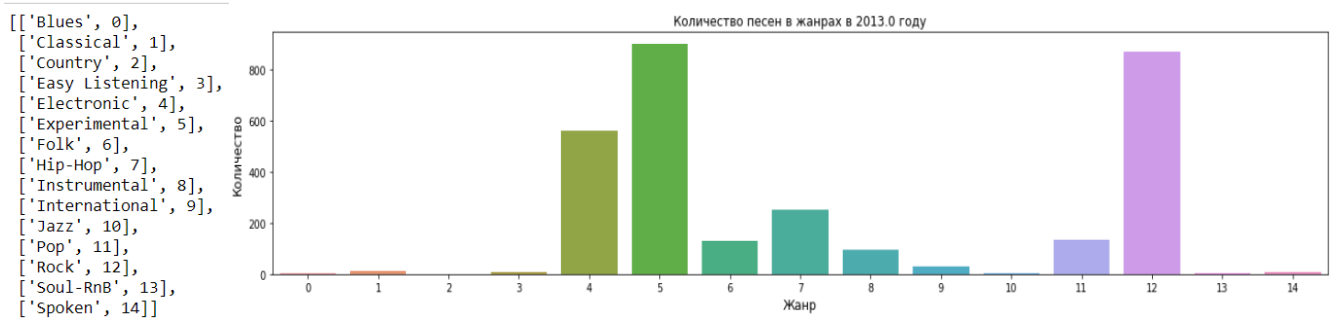

Рисунок 23 – Количество песен во всех жанрах в 2013 году

Этап так называемого знакомства с данными завершён, теперь можно приступать к отбору признаков.

2.2. Отбор признаков

В наборе данных для каждой песни собран 521 признак, и подавляющее большинство из них – технические характеристики. Чтобы разобраться в природе каждого из них, нужно иметь обширные знания из области физики звука и устройства звуковых волн в целом; также вполне возможно, что 521 – это слишком много для обучения классификаторов, есть риск или переобучения, или больших временных затрат при наличии очень тесно коррелирующих друг с другом признаков, поэтому стоит рассмотреть несколько вариантов их отбора.

Чтобы выбрать «правильные» характеристики наблюдений, нужно познакомиться с ними всеми хотя бы в общих чертах, что и было сделано. На рисунке 24 представлено общее описание признаков, подробнее о них можно узнать в справке библиотеки librosa [11].

		Острота пика	Макс	МО	Медиана	Мин	Скос	Ст. откл.		
	Признак\Описательная статистика	**kurtosis**	**max**	**mean**	**median**	**min**	**skew**	**std**	**итого**	
Вектор цветности	**chroma_cens**	12	12	12	12	12	12	12	**84**	
Преобразование Constant-Q	**chroma_cqt**	12	12	12	12	12	12	12	**84**	
Хроматограмма	**chroma_stft**	12	12	12	12	12	12	12	**84**	
Мел-частотные кепстральные к-ты	**mfcc**	20	20	20	20	20	20	20	**140**	
СКО для каждого сэмпла песни	**rmse**	1	1	1	1	1	1	1	**7**	
Полоса пропускания	**spectral_bandwidth**	1	1	1	1	1	1	1	**7**	
Средняя точка распр-я энергии	**spectral_centroid**	1	1	1	1	1	1	1	**7**	
Спектральный контраст	**spectral_contrast**	7	7	7	7	7	7	7	**49**	
Спектральный спад	**spectral_rolloff**	1	1	1	1	1	1	1	**7**	
Центр тяжести тона	**tonnetz**	6	6	6	6	6	6	6	**42**	
Частота перехода через ноль	**zcr**	1	1	1	1	1	1	1	**7**	
										518

Рисунок 24 – Признаки, имеющиеся в наборе данных

Авторы, выложившие архив в открытый доступ, отмечают, что для некоторых признаков по каждой описательной статистике собрано 12 показателей – по одному для каждой ноты.

Для отбора наиболее важных показателей воспользуемся тремя основными подходами: удаление коллинеарных признаков, удаление признаков с малой дисперсией и удаление признаков, исходя из их важности в случайном лесу. На рисунке 25 представлены соответствующие функции.

```python
#удаление признаков с высокими коэффициентами корреляции
def remove_collinear_features(x,y,threshold):
    '''
    Objective:
        Remove collinear features in a dataframe with a correlation coefficient
        greater than the threshold. Removing collinear features can help a model
        to generalize and improves the interpretability of the model.

    Inputs:
        threshold: any features with correlations greater than this value are removed

    Output:
        dataframe that contains only the non-highly-collinear features
    '''

    # Calculate the correlation matrix
    corr_matrix = x.corr()
    iters = range(len(corr_matrix.columns) - 1)
    drop_cols = []

    # Iterate through the correlation matrix and compare correlations
    for i in iters:
        for j in range(i):
            item = corr_matrix.iloc[j:(j+1), (i+1):(i+2)]
            col = item.columns
            row = item.index
            val = abs(item.values)

            # If correlation exceeds the threshold
            if val >= abs(threshold):
                # Print the correlated features and the correlation value
                # print(col.values[0], "|", row.values[0], "|", round(val[0][0], 2))
                drop_cols.append(col.values[0])

    # Drop one of each pair of correlated columns
    drops = set(drop_cols)
    x = x.drop(columns = drops)

    return x

#удаление признаков с малой дисперсией
def variance_threshold_selector(data, threshold):
    selector = VarianceThreshold(threshold)
    selector.fit(data)
    return data[data.columns[selector.get_support(indices=True)]]

#удаление признаков, исходя из важности в случайном лесу
def extra_trees_selector(X,y,nest):
    clf = ExtraTreesClassifier(n_estimators=nest,random_state=42,class_weight='balanced')
    clf = clf.fit(X, y)
    model = SelectFromModel(clf, prefit=True)
    return X[X.columns[model.get_support(indices=True)]]
```

Рисунок 25 – Способы отбора признаков

Функция remove_collinear_features является несколько модифицированной одноимённой встроенной: отличие лишь в том, что удаляются также и признаки с отрицательным коэффициентом корреляции, если он больше какого-либо порогового значения threshold.

Функции для удаления признаков с малой дисперсией или не важных по результатам обучения случайного леса – встроенные.

На рисунке 26 показаны результаты отбора по каждому методу с разными пороговыми значениями.

```
X, y = ft1[interestnames[:-3]], ft1['track.7']
X_new11 = remove_collinear_features(X,y,0.5)#(26960, 159) 310.49404800799994 seconds
X_new12 = remove_collinear_features(X,y,0.6)#(26960, 231) 321.4249692210001 seconds
X_new13 = remove_collinear_features(X,y,0.7)#
X_new14 = remove_collinear_features(X,y,0.8)#
X_new21 = variance_threshold_selector(X, .5 * (1 - .5))#(26960, 281) 3.0337050810001074 seconds
X_new22 = variance_threshold_selector(X, .6 * (1 - .6))#(26960, 281) 0.2549910939999336 seconds
X_new23 = variance_threshold_selector(X, .7 * (1 - .7))#(26960, 282) 0.24523688899989793 seconds
X_new24 = variance_threshold_selector(X, .8 * (1 - .8))#(26960, 286) 0.2620619290000832 seconds
X_new25 = variance_threshold_selector(X, .9 * (1 - .9))#(26960, 300) 0.25380156900018846 seconds
X_new31 = extra_trees_selector(X,y,50)#(26960, 232) 12.627768252999886 seconds
X_new32 = extra_trees_selector(X,y,100)#(26960, 229) 24.964343564000046 seconds
X_new33 = extra_trees_selector(X,y,500)#(26960, 232) 125.92107139399991 seconds
```

Рисунок 26 – Результаты удаления признаков

Отчётливо заметно как сокращение размерности массива данных, так и увеличение времени работы алгоритмов при желании пользователя получить большую точность.

Для дальнейшей работы возьмём признаки, отобранные случайным лесом: этот алгоритм сложнее, чем удаление признаков с высокой корреляцией, но тоже довольно сильно фильтрует их количество. Сравнение выбора случайных лесов из 50 и 100 деревьев (переменные X_new31 и X_new32) представлено на рисунке 27.

```
#метод, похожий на head и tail, но для столбцов
def interv(self, m, n):
    return self.iloc[:, m:n]

pd.DataFrame.interv = interv
```

```
X_new31.shape,X_new32.shape
```

```
((26960, 232), (26960, 229))
```

```
ta=[X_new31,X_new32]
clnames=[list(i.columns) for i in ta]
indx=['X_new31','X_new32']
rest=pd.DataFrame(clnames, index=indx)
```

```
rest.interv(20,40)#разные параметры не сильно влияют на результат
```

	20	21	22	23	24	25	26	27	28	29
X_new31	chroma_cqt.76	chroma_cqt.77	chroma_cqt.79	chroma_cqt.80	chroma_cqt.81	chroma_cqt.82	chroma_stft.24	chroma_stft.25	chroma_stft.26	chroma_stft.28
X_new32	chroma_cqt.74	chroma_cqt.75	chroma_cqt.76	chroma_cqt.77	chroma_cqt.78	chroma_cqt.79	chroma_cqt.80	chroma_cqt.81	chroma_cqt.82	chroma_stft.24

Рисунок 27 – Признаки, отобранные случайными лесами с разным количеством деревьев

3. ОБУЧЕНИЕ МОДЕЛЕЙ

3.1. Построение алгоритмов классификации и выбор метрик качества

В разделе 1 уже было сказано об алгоритмах классификации, построенных в данной работе. На рисунке 28 показаны необходимые для этого библиотеки и функции.

```
from sklearn.model_selection import train_test_split

from sklearn import metrics
from sklearn.metrics import f1_score,precision_score,recall_score,confusion_matrix
import itertools

def plot_confusion_matrix(cm, classes,
                          title='Confusion matrix',
                          cmap=plt.cm.Blues,
                          normalize=True):
    if normalize:
        cm = cm.astype('float') / cm.sum(axis=1)[:, np.newaxis]

    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)

    thresh = cm.max() / 2.
    for i, j in itertools.product(range(cm.shape[0]), range(cm.shape[1])):
        plt.text(j, i, format(cm[i, j], 'f'),
                 horizontalalignment='center',
                 color='white' if cm[i, j] > thresh else 'black')

    plt.ylabel('True label')
    plt.xlabel('Predicted label')
    plt.tight_layout()

from sklearn import linear_model
from sklearn.linear_model import LogisticRegression

!pip install xgboost
!pip install scikit-learn
from xgboost import XGBClassifier
from sklearn import preprocessing

from sklearn.ensemble import RandomForestClassifier

!pip install imbalanced-learn
from imblearn.ensemble import BalancedBaggingClassifier, EasyEnsemble
```

Рисунок 28 – Импорт библиотек для построения классификаторов

Стоит подробнее остановиться на метриках, по которым будет сравниваться качество работы классификаторов: precision, recall и f-score, которые рассчитываются для бинарной классификации, то есть переменная-таргет с пятнадцатью возможными классами превращается в 15 бинарных переменных по названиям соответствующих классов.

Recall (полнота) = $\frac{TP}{TP+FN}$, где TP – верно классифицированные значения для класса 1, а FN – ошибочно классифицированные как 0 значения класса 1, то есть отношение классифицированных правильно как 1 к общему фактическому количеству 1.

Precision (точность) = $\frac{TP}{TP+FP}$, где TP – верно классифицированные значения для класса 1, а FP – ошибочно классифицированные как 1 значения класса 0, то есть отношение классифицированных правильно как 1 к общему количеству классифицированных как 1.

F-score (f-мера) = $\frac{2*\text{точность}*\text{полнота}}{\text{точность}+\text{полнота}}$. Это совместная оценка точности и полноты, их среднее гармоническое.

Самая простая и популярная метрика accuracy (доля правильно классифицированных объектов) не использовалась, так как её оптимизация может привести к большому количеству ошибок второго рода, когда происходит ложноотрицательное срабатывание.

После выбора метрик необходимо определиться с алгоритмом отбора признаков, и было принято решение оставить 229 отобранных случайным лесом из 100 деревьев. Пошагово структурированная функция обучения классификатора представлена на рисунке 29.

```
def classifier(X,y,modelname):
    X_train, X_test, y_train, y_test = train_test_split(X,
                                                        y,
                                                        test_size=0.2,
                                                        random_state=42,
                                                        shuffle=True
                                                        )
    X_train, X_val, y_train, y_val = train_test_split(X_train, y_train, test_size=0.15, random_state=42,shuffle=True)

    modelL = LogisticRegression(class_weight='balanced')
    modelX = XGBClassifier()
    modelR=RandomForestClassifier(class_weight='balanced',
                                  n_estimators=200,
                                  max_depth=10,
                                  random_state=1)
    modelB = BalancedBaggingClassifier(base_estimator=modelR,
                                       n_estimators=200,
                                       random_state=1)

    pip_resampled = make_pipeline(Imputer(strategy="mean"),
                                  RobustScaler(),
                                  modelB)

    if modelname=='L':
        model=modelL
    elif modelname=='X':
        model=modelX
    elif modelname=='R':
        model=modelR
    elif modelname=='B':
        model=pip_resampled

    model.fit(X_train, y_train)

    predictions = model.predict(X_train)
    predictionst = model.predict(X_test)
    predictionsv = model.predict(X_val)

    cnf=confusion_matrix(y_train,predictions)
    f1=f1_score(y_train, predictions,average='weighted')
    pr=precision_score(y_train, predictions,average='weighted')
    rec=recall_score(y_train, predictions,average='weighted')
    #print("Для train: f1-score",f1,"precision",pr,"recall",rec)

    cnft=confusion_matrix(y_test,predictionst)
    f1_t=f1_score(y_test, predictionst,average='weighted')
    pr_t=precision_score(y_test, predictionst,average='weighted')
    rec_t=recall_score(y_test, predictionst,average='weighted')
    #print("Для test: f1-score",f1_t,"precision",pr_t,"recall",rec_t)

    cnfv=confusion_matrix(y_val,predictionsv)
    f1_v=f1_score(y_val, predictionsv,average='weighted')
    pr_v=precision_score(y_val, predictionsv,average='weighted')
    rec_v=recall_score(y_val, predictionsv,average='weighted')
    #print("Для val: f1-score",f1_v,"precision",pr_v,"recall",rec_v)

    #print("Для train:\n",cnf)
    #print("Для test:\n",cnft)
    #print("Для val:\n",cnfv)
    return [[f1,pr,rec],[f1_t,pr_t,rec_t],[f1_v,pr_v,rec_v],[cnf,cnft,cnfv]]
```

Рисунок 29 – Классификатор

На первом этапе данная функция разбивает выборку на обучающую (80% наблюдений) и тестовую (20%), затем обучающая выборка разбивается ещё раз на обучающую (85%) и валидационную (15%) выборки, чтобы при задании другого ядра датчика случайных чисел random_state можно было убедиться, что модели, натренированные на любой части обучающей выборки, не будут сильно отличаться друг от друга.

На втором шаге обозначаем логистическую регрессию как modelL, XGBoost – modelX, случайный лес – modelR, бэггинг – modelB, после чего обучаем соответствующую поданной на вход букве модель.

Завершающим шагом является построение confusion matrix (матрицы ошибок или неточностей) для обучающей, валидационной и тестовой выборок, а также расчёт описанных выше метрик качества на каждой из них. Всё это возвращает функция classifier.

На рисунке 30 представлена таблица результатов работы всех алгоритмов по всем метрикам на всех выборках.

train		f1-score	precision	recall
LogisticRegression		0.622510	0.658189	0.610572
XGBoost		0.724122	0.746514	0.738054
RandomForest		0.730205	0.757963	0.728889
Balance BaggingClassifier		0.457909	0.611829	0.460015
test		f1-score	precision	recall
LogisticRegression		0.600718	0.643871	0.581973
XGBoost		0.649825	0.666412	0.673961
RandomForest		0.568828	0.600008	0.569733
Balance BaggingClassifier		0.434634	0.588333	0.433605
val		f1-score	precision	recall
LogisticRegression		0.605578	0.649471	0.584672
XGBoost		0.645051	0.659277	0.671508
RandomForest		0.574557	0.599976	0.576638
Balance BaggingClassifier		0.445010	0.593932	0.443449

Рисунок 30 – Результаты работы алгоритмов классификации

Результаты на тестовой и валидационной выборках несколько хуже, чем на обучающей, что является хорошим знаком, то есть можно предположить об отсутствии переобучения. Лучше всех по метрикам качества оказался XGBoost, несколько от него отстаёт RandomForest. Логистическая регрессия не показала сопоставимый результат на обучающей выборке, однако оказалась самой стабильной и на тестовой, и на валидационной. Balance BaggingClassifier по всем параметрам оказался хуже всех.

Стоит посмотреть на матрицы ошибок и сделать выводы о том, какие музыкальные жанры лучше поддаются классификации, а какие вводят алгоритмы в заблуждение. Рассмотрим каждый алгоритм отдельно.

3.2. Логистическая регрессия

Посмотрим на матрицу ошибок логистической регрессии на обучающей выборке (рисунок 31).

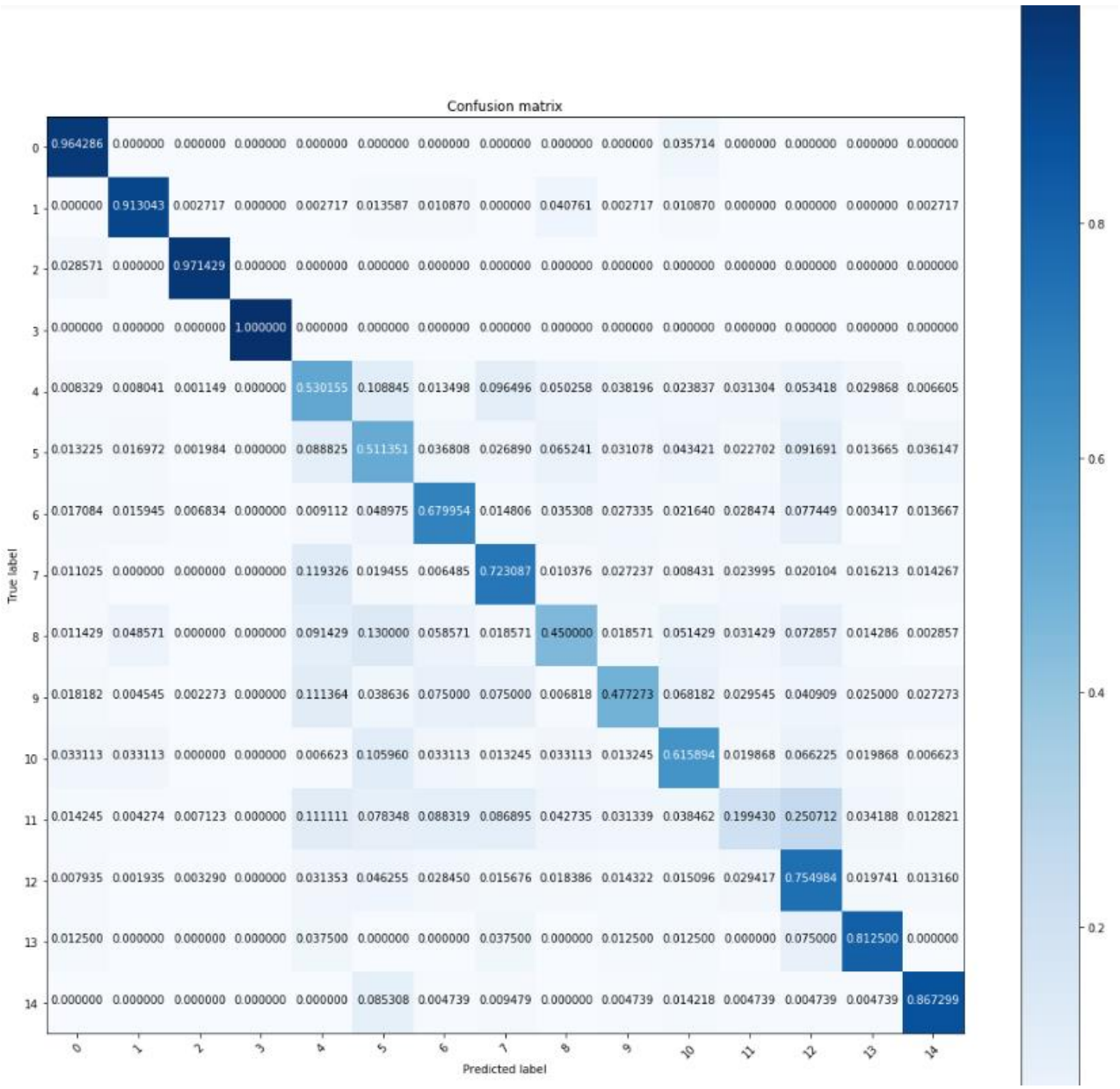

Рисунок 31 – Матрица ошибок логистической регрессии на обучающей выборке

Из матрицы на рисунке 31 видно, что логистическая регрессия очень хорошо классифицировала блюз (класс 0), классическую музыку (1), кантри (2), жанр Easy Listening (3), Soul R-n-B (13) и Spoken (14). С роком получилось несколько хуже (12), но в контексте большого количества композиций в этом жанре результат довольно неплохой. Хуже всех алгоритм справился с поп-музыкой (11), правильно отнеся у ней лишь 20% песен, тогда как 25% отправил в рок (12) и 11% – в электронную музыку (4). Это вполне объяснимо, потому что поп-музыка на то и популярная, что

может сочетать в себе звучание, характерное для совершенно разных жанров, и включает в себя поджанры, например, поп-рок (если приводить конкретные примеры, то сразу на ум приходят российские исполнители группа «Звери» или Максим Свобода) или электропоп (Depeche Mode, Lady Gaga). С электронной (4), экспериментальной (5) и инструментальной (8) музыкой, а также с жанром International (9) логистическая регрессия угадала в половине случаев. Фолк (6) был угадан в двух случаях из трёх, джаз (10) – в трёх из пяти, хип-хоп (7) – в ¾ случаев.

Посмотрим на такую же матрицу для валидационной выборки (рисунок 32).

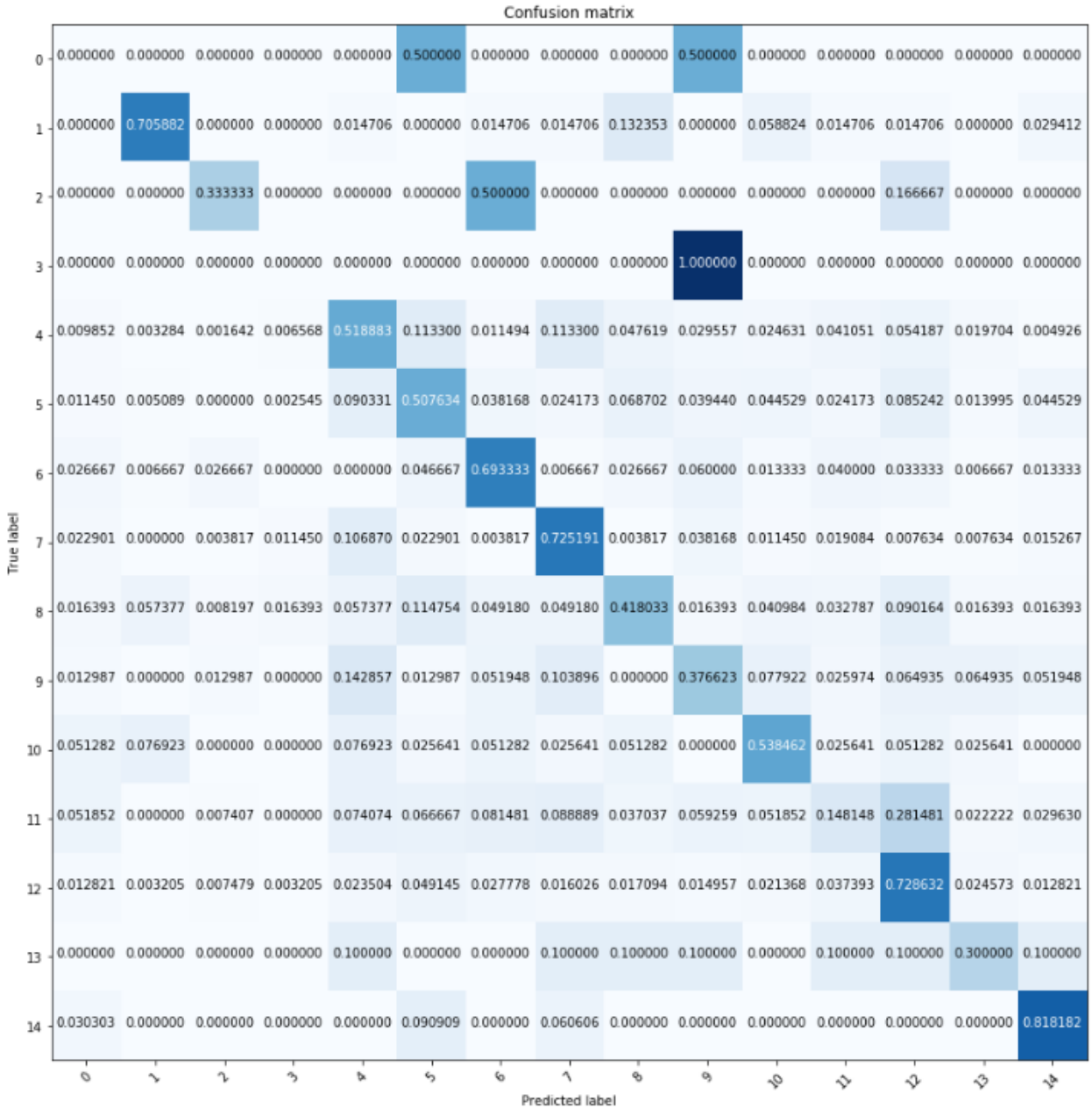

Рисунок 32 – Матрица ошибок логистической регрессии на валидационной выборке

Результаты предсказания классов логистической регрессией на валидационной выборке уже не впечатляют: в случае с блюзом (0) алгоритм отнёс половину песен к экспериментальной музыке (5) и к международной (9), хотя с какой-то стороны можно действительно так обобщить, но в данной ситуации всё дело в том, что блюзовых композиций очень мало по сравнению с другими жанрами и алгоритмам сложно их учитывать. С классикой (1) логистическая регрессия справилась в 70,6% случаев и отнесла 13% классических композиций к инструментальной музыке (8), с чем очень трудно спорить. Жанр кантри (2) был угадан всего лишь в трети случаев, половину этих песен алгоритм отнёс к фолку (6) и 1/6 – к року. С самым малочисленным жанром Easy Listening (3) тоже не получилось ничего хорошего: все песни этого жанра были приняты за международную музыку (9). Лучше всех дела обстоят с жанром Spoken (14) и довольно неплохо с фолком (6), хип-хопом (7) и роком (12). Также стабильно неправильно были угаданы поп-композиции (11), относительное большинство которых алгоритм принял за рок (12). С остальными жанрами дела обстоят плохо.

На данным момент можно предположить о том, что причины столь неоднозначного результата предсказаний кроются в малочисленности классов и/или в непропорциональном разбиении на обучающую, валидационную и тестовую выборки, но для полноты картины стоит построить матрицу ошибок для тестовой выборки (рисунок 33).

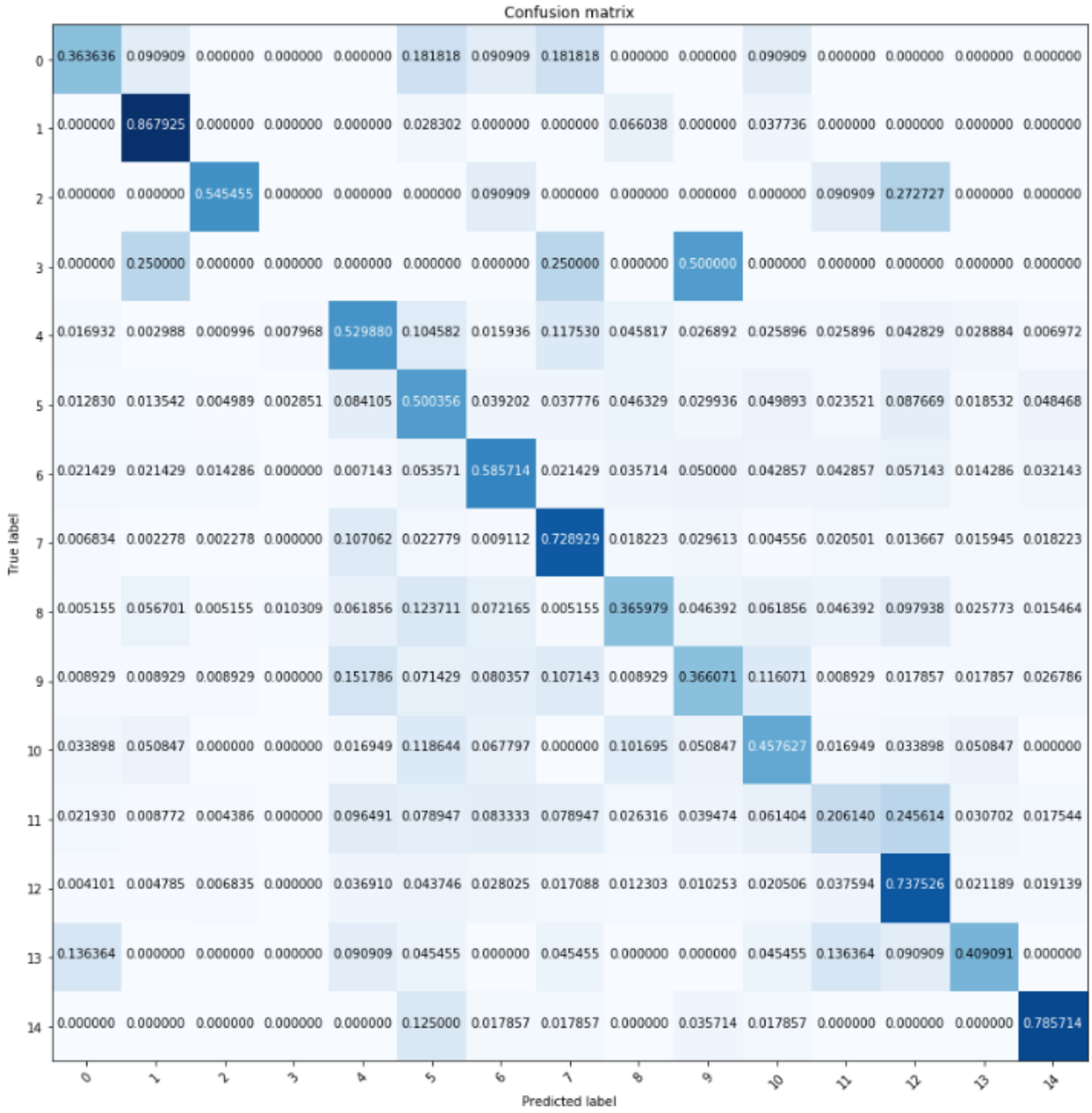

Рисунок 33 – Матрица ошибок логистической регрессии на тестовой выборке

Результаты работы логистической регрессии на тестовой выборке будто бы сочетают в себе результаты на обучающей и валидационной: например, алгоритм оказался неточным для жанра Easy Listening (3), но показал себя хорошо для классики (1), хип-хопа (7), рока (12) и жанра Spoken (14), всё ещё стабильно плохо отличая поп-композиции (11) и практически случайно угадывая электронную (4) и экспериментальную (5) музыку.

Посмотрим на точность предсказаний логистической регрессии на всех выборках рядом по каждому жанру (рисунок 34).

			Логистическая регрессия		
		Песен	train	val	test
Блюз	0	41	96%	0%	36%
Классика	1	542	91%	71%	87%
Кантри	2	52	97%	33%	55%
Easy Listening	3	16	100%	0%	0%
Электронная	4	5095	53%	52%	53%
Экспериментальная	5	6726	51%	51%	50%
Фолк	6	1308	68%	69%	59%
Хип-хоп	7	2243	72%	73%	73%
Инструментальная	8	1016	45%	42%	37%
Международная	9	629	48%	38%	37%
Джаз	10	249	62%	54%	46%
Поп	11	1065	20%	15%	21%
Рок	12	7566	75%	73%	74%
Soul-RnB	13	112	81%	30%	41%
Spoken	14	300	87%	82%	79%

Рисунок 34 – Сравнение точности логистической регрессии для всех выборок

Глядя на сравнительную таблицу на рисунке 34, можно сделать вывод, что логистическая регрессия переобучается на малочисленных классах, однако количества песен около 300, как у жанра Spoken (14), уже вполне достаточно для достижения сопоставимого уровня точности на всех выборках. Как уже было сказано выше, такую проблему можно пытаться решить либо разделением на выборки с сохранением пропорций, либо объединением малочисленных классов с более крупными, например, джаз объединить с блюзом, как сделано в музыкальной части «ВКонтакте», или Soul-RnB объединить с хип-хопом, потому что, опять же, довольно часто жанры хип-хоп и R’n’B идут вместе. Но это уже следующий этап анализа, который не рассматривается в рамках данного исследования; его необходимо производить уже после выбора алгоритма классификации и принятия каких-либо мер для повышения качества предсказаний, а пока что была рассмотрена только логистическая регрессия. Перейдём к методу XGBoost.

3.3. XGBoost

Посмотрим на матрицу ошибок алгоритма XGBoost на обучающей выборке (рисунок 35).

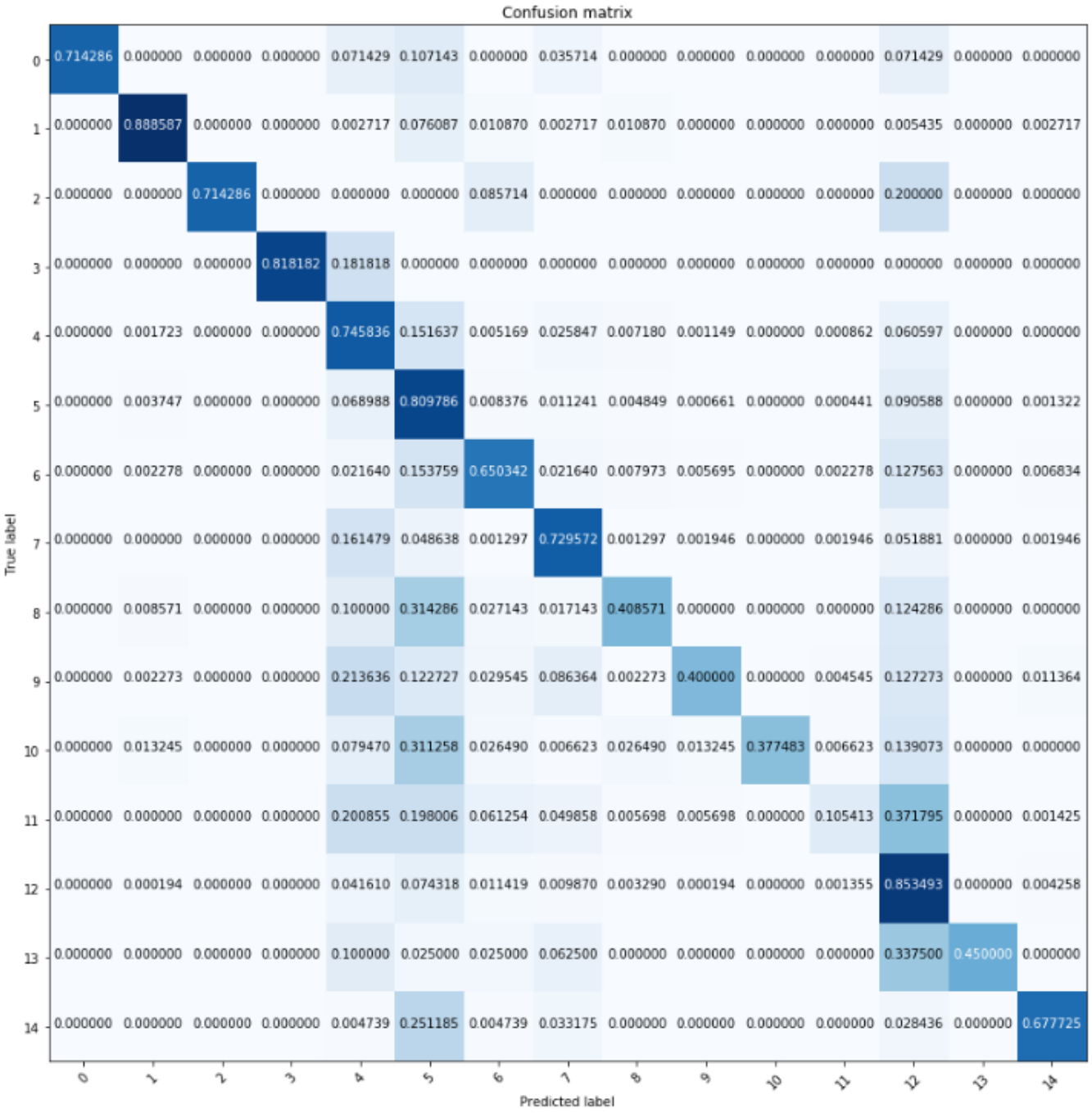

Рисунок 35 – Матрица ошибок алгоритма XGBoost на обучающей выборке

Из матрицы на рисунке 35 видно, что алгоритм довольно хорошо классифицировал большинство жанров, однако 31% инструментальной музыки (8) ошибочно отнёс к экспериментальной (5), угадав лишь 41%; похожая ситуация и с жанром International (9): угадано 40%, ошибочно отнесено к электронной (4) – 21%. Далее наблюдается некоторое чередование ошибок: джазовые композиции (10)

часто принимались за экспериментальные (5), популярные (11) – за электронные (4) и в 37% случаев – за рок-музыку (12). Soul-RnB (13) часто предсказывался как рок (12), а песни жанра Spoken (14) хоть и были угаданы в 2/3 случаев, но в четверти случаев были приняты за экспериментальную музыку (5). Жанр кантри (2) в 20% случаев был принят за рок (12).

Ошибки выглядят похожими с результатами логистической регрессией, только с поп-музыкой дела ещё хуже. Стоит посмотреть на валидационную выборку на рисунке 36.

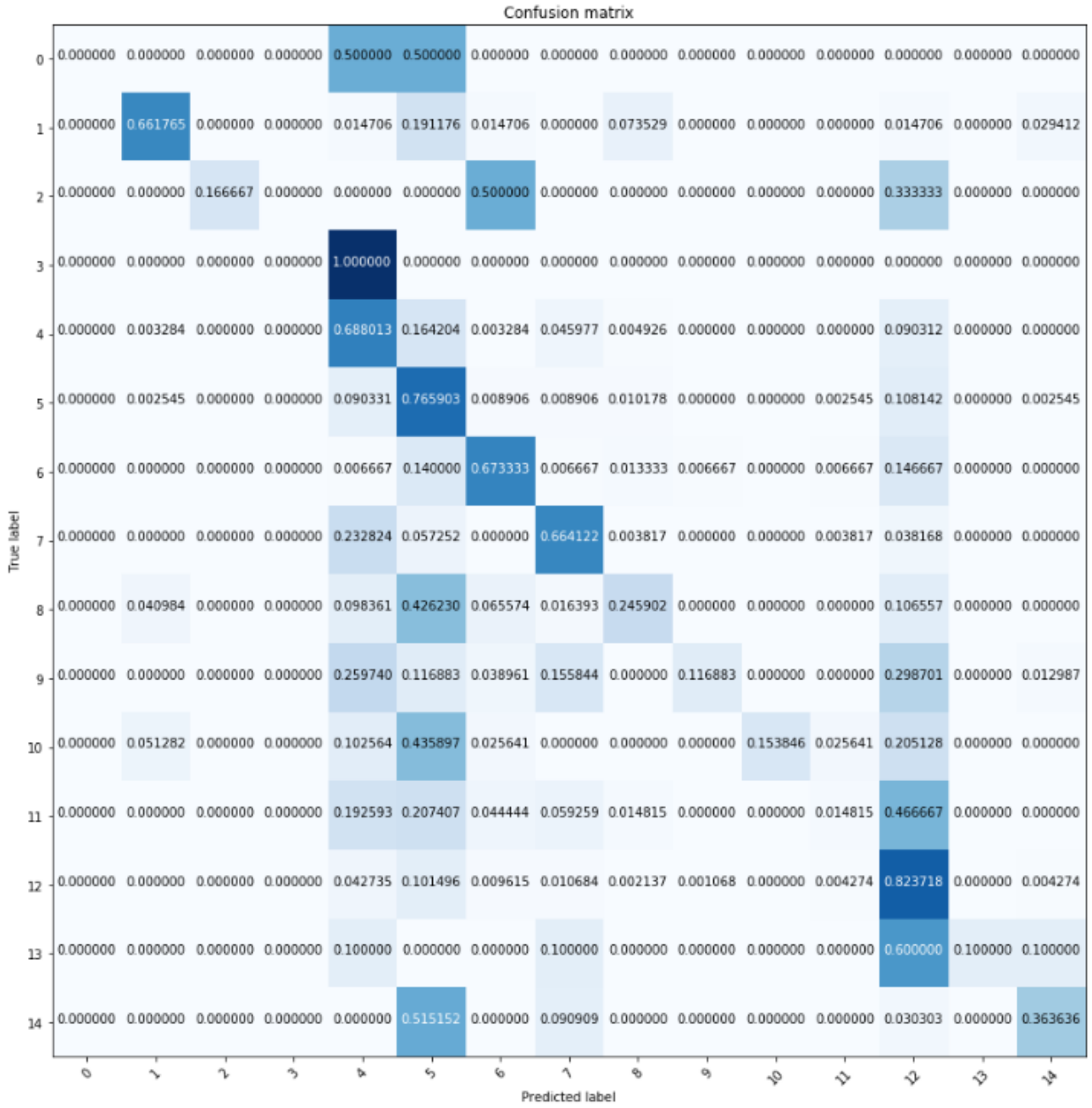

Рисунок 36 – Матрица ошибок алгоритма XGBoost на валидационной выборке

Как и у логистической регрессии, на валидационной выборке результаты XGBoost’а заметно хуже: ни разу не угадан блюз (0), по половине наблюдений принято за электронную (4) и экспериментальную (5) музыку, половина песен жанра кантри (2) принята за фолк (6), треть – за рок (12); все песни жанра Easy Listening (3) приняты за электронную музыку (4). Более 40% инструментальной музыки (8) и джаза (10) ошибочно классифицировано как экспериментальная музыка (5). У поп-композиций (11) дела обстоят совсем плохо: 47% принято за рок (12), по 20% – за электронную (4) и экспериментальную (5) музыку. 60% песен жанра Soul-RnB (13) также принято за рок (12). Посмотрим, что будет происходить на тестовой выборке (рисунок 37).

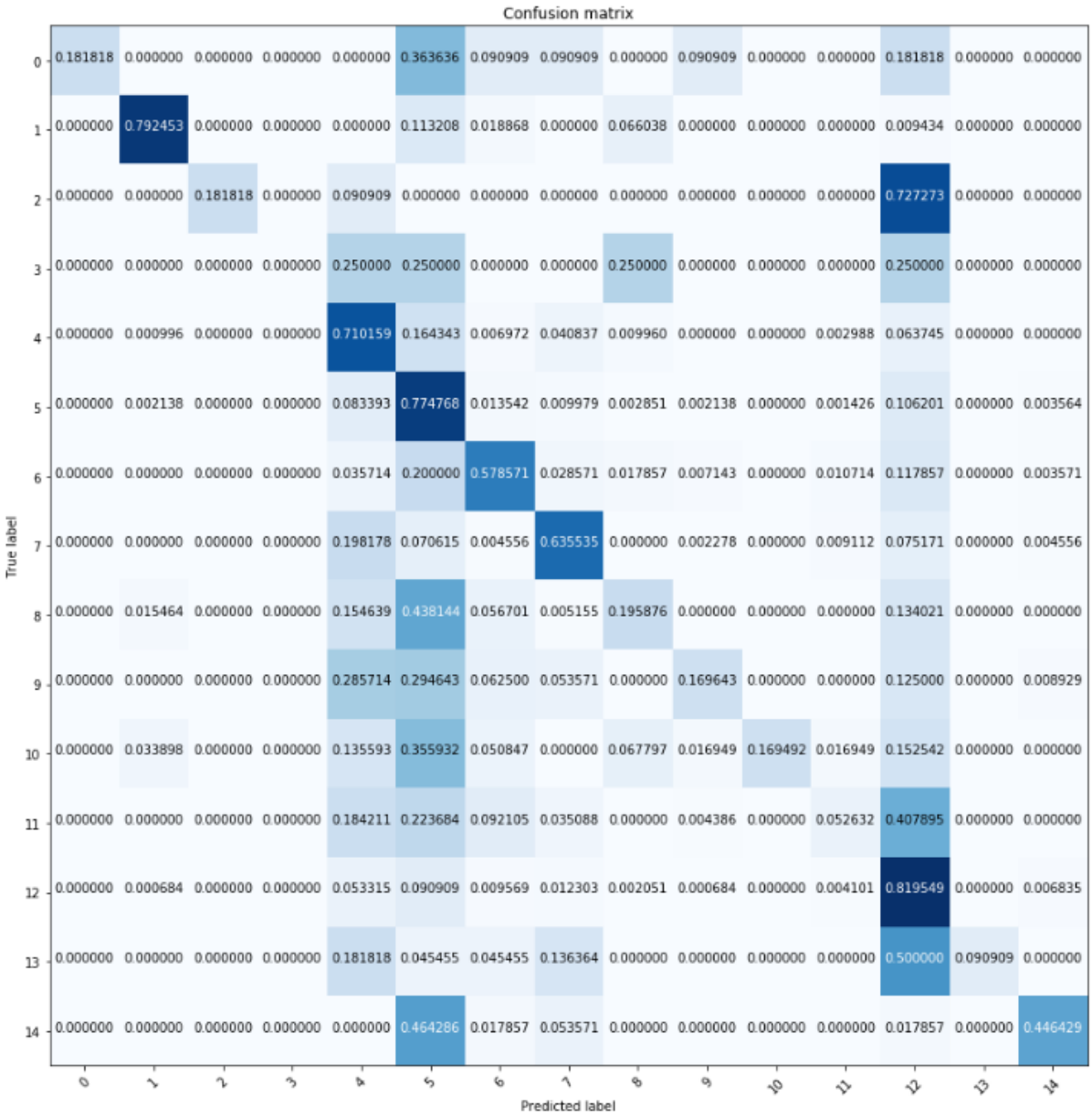

Рисунок 37 – Матрица ошибок алгоритма XGBoost на тестовой выборке

На тестовой выборке очень плохо угадан блюз (0), 73% кантри-композиций (2) приняты за рок (12), жанр Easy Listening (3) не был угадан ни разу, результаты одинаково распределились на электронную (4), экспериментальную (5), инструментальную (8) и рок-музыку (12). 44% инструментальных песен (8) было отнесено к экспериментальной музыке (5), почти по 30% песен жанра International (9) были приняты за электронную (4) и также экспериментальную (5) музыку. 35% джаза (10) и 46% песен жанра Spoken (14) также классифицированы как экспериментальные (5); 50% песен жанра Soul-RnB (13) и 40% поп-треков (11) классифицированы как рок (12). Хорошо XGBoost отработал для классической (1), электронной (4), экспериментальной (5) и рок-музыки (12), довольно неплохо – для хип-хопа (7) и фолка (6).

На данный момент кажется, что XGBoost заметно переобучился на превалирующих по количеству песен жанрах (рок, экспериментальная и электронная музыка), однако стоит посмотреть на рисунок 38, где проведено сравнение результатов работы алгоритма на всех выборках.

			XGBoost		
		Песен	train	val	test
Блюз	0	41	71%	0%	18%
Классика	1	542	89%	66%	79%
Кантри	2	52	71%	17%	18%
Easy Listening	3	16	82%	0%	0%
Электронная	4	5095	75%	69%	71%
Экспериментальная	5	6726	81%	77%	77%
Фолк	6	1308	65%	67%	58%
Хип-хоп	7	2243	73%	66%	64%
Инструментальная	8	1016	41%	25%	20%
Международная	9	629	40%	12%	17%
Джаз	10	249	38%	15%	17%
Поп	11	1065	11%	1%	5%
Рок	12	7566	85%	82%	82%
Soul-RnB	13	112	45%	10%	9%
Spoken	14	300	68%	36%	45%

Рисунок 38 – Сравнение точности алгоритма XGBoost для всех выборок

Из рисунка 38 видно, что XGBoost плохо классифицирует малочисленные классы и хорошо – классы с большим количеством наблюдений, однако из многочисленности класса не следует, что песни данного жанра будут хорошо

классифицированы, о чём свидетельствуют просто провальные результаты для поп-музыки (11), хотя песен в этом жанре – 1065.

Промежуточный итог аналогичен итогу после сравнения результатов логистической регрессии: стоит повысить многочисленность классов, хотя до сих пор не понятно, что делать с поп-музыкой.

Перейдём к построению случайного леса.

3.4. Случайный лес

Посмотрим на матрицу ошибок случайного леса на обучающей выборке (рисунок 39).

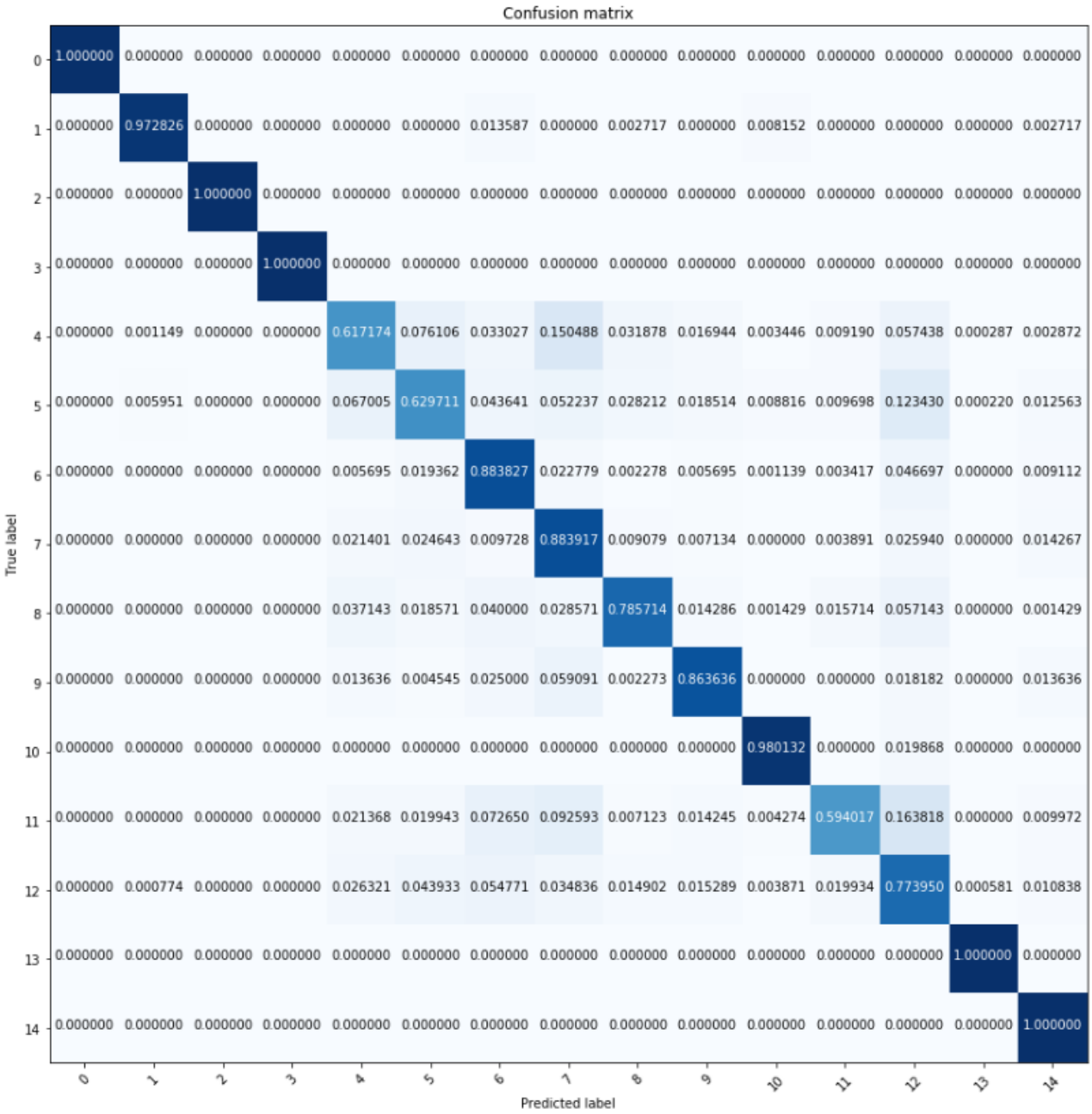

Рисунок 39 – Матрица ошибок случайного леса на обучающей выборке

Как видно из рисунка 39, на обучающей выборке случайный лес показывает просто феноменальные по точности результаты: абсолютно верно классифицированы такие жанры, как блюз (0), кантри (2), Easy Listening (3), Soul-RnB (13) и Spoken (14), очень точно – классика (1) и джаз (10), на 86-88% точно – фолк (6), хип-хоп (7) и International (9), на уровне 77-79% – инструментальная музыка (8) и рок (12). Худшие результаты алгоритм показал на электронной (4), экспериментальной (5) и популярной (11) музыке, но эти «худшие» результаты – от 59% до 63% точности, что очень заметно лучше, чем у алгоритмов, рассмотренных раньше. Однако к сравнению точности алгоритмов перейдём позже, сейчас рассмотрим результаты работы случайного леса на валидационной выборке (рисунок 40).

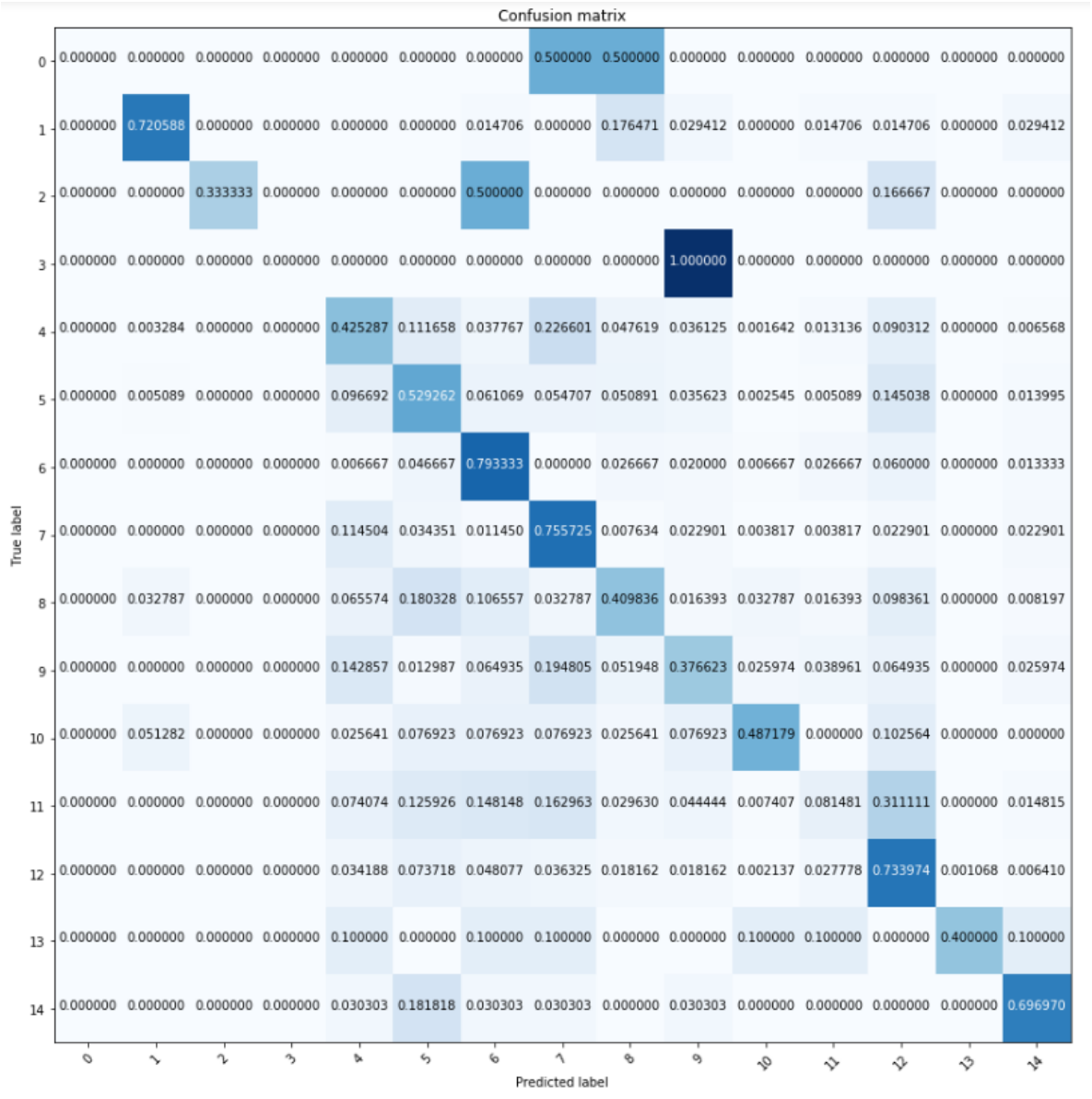

Рисунок 40 – Матрица ошибок случайного леса на валидационной выборке

Предсказание случайного леса на валидационной выборке радует отнюдь не так сильно, как на обучающей: блюз (0) в половине случаев превратился в инструментальную музыку (8), а в другой половине – в международную (9); жанр кантри (2) был угадан лишь в трети случаев, тогда как в половине случаев алгоритм его принял за фолк (6); с жанром Easy Listening (3) полное попадание в International (9). По опыту двух предыдущих алгоритмов ожидаемо исчезло качество предсказания поп-музыки (11): 31% песен алгоритм посчитал роком (12). Неплохой уровень точности около 70-80% случайный лес показал на классической музыке (1), фолке (6), хип-хопе (7), роке (12) и жанре Spoken (14), точность на остальных жанрах – в районе 40-50%, что сопоставимо со случайным выбором. Дорисуем картину настроений относительно случайного леса и посмотрим на результаты его работы на тестовой выборке (рисунок 41).

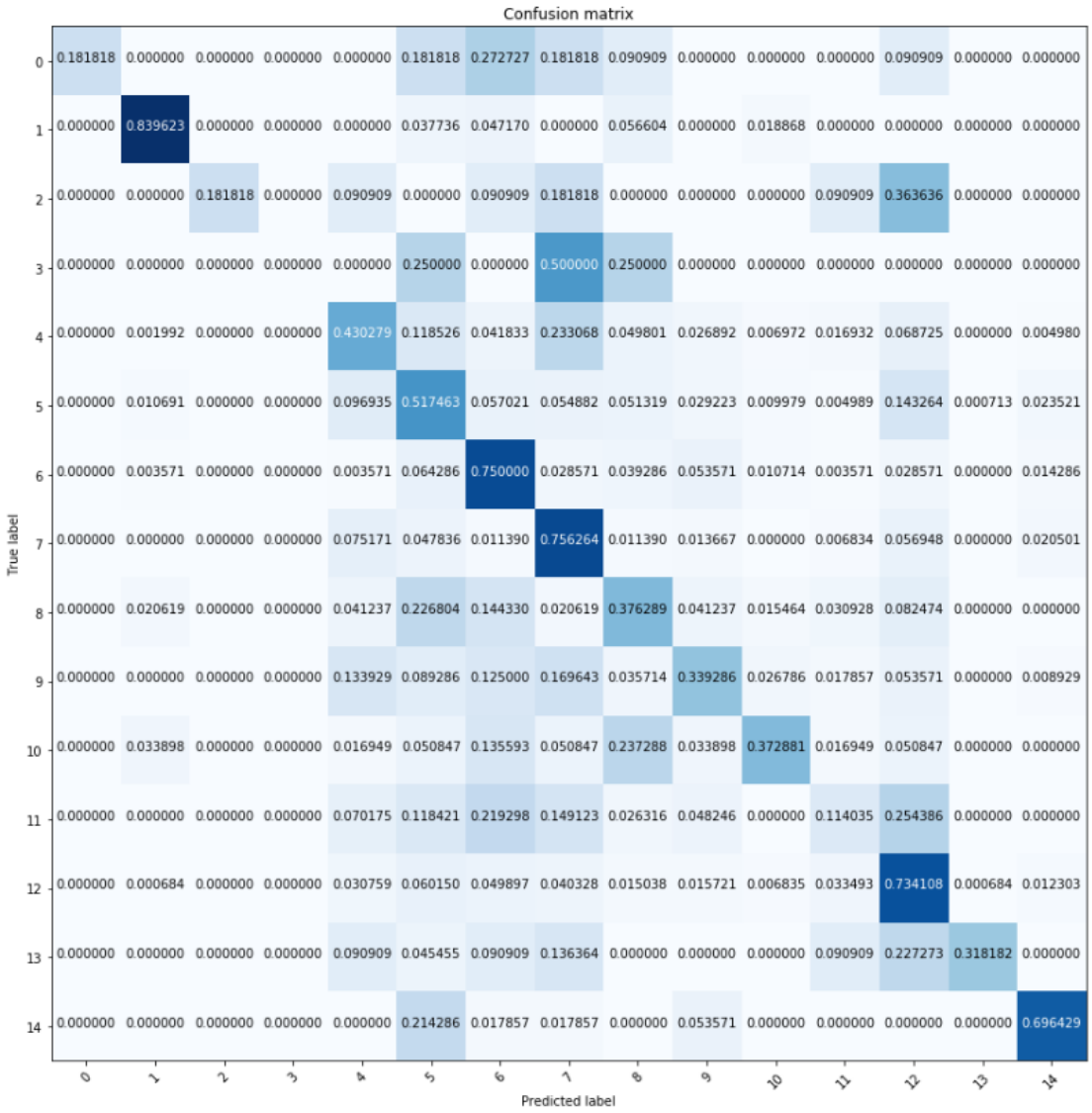

Рисунок 41 – Матрица ошибок случайного леса на тестовой выборке

Диагональ матрицы на рисунке 41 выглядит более синей, чем на рисунке 40, но на самом деле надеяться на лучшее не приходится: хорошая точность для классики (1), фолка (6), хип-хопа (7), рока (12) и жанра Spoken (14). Любопытно, что почти четверть песен жанра Soul-RnB (13) и поп-музыки (11) алгоритм отнёс к року (12), а четверть Easy Listening (3), Instrumental (8) и Spoken (14) – к экспериментальной музыке (5). Также интересно, что 22% поп-треков (11) случайный лес принял за фолк (6), чего не наблюдалось у описанных выше алгоритмов.

Промежуточный вывод: трудности с классификацией поп-музыки – перманентны, однако перепутать её можно с очень разными жанрами. Посмотрим на рисунок 42, чтобы увидеть, где и как переобучился случайный лес.

			Случайный лес		
		Песен	train	val	test
Блюз	0	41	100%	0%	18%
Классика	1	542	97%	72%	84%
Кантри	2	52	100%	33%	18%
Easy Listening	3	16	100%	0%	0%
Электронная	4	5095	62%	43%	43%
Экспериментальная	5	6726	63%	53%	52%
Фолк	6	1308	88%	79%	75%
Хип-хоп	7	2243	88%	76%	76%
Инструментальная	8	1016	79%	41%	38%
Международная	9	629	86%	38%	34%
Джаз	10	249	98%	49%	37%
Поп	11	1065	59%	8%	11%
Рок	12	7566	77%	73%	73%
Soul-RnB	13	112	100%	40%	32%
Spoken	14	300	100%	70%	70%

Рисунок 42 – Сравнение точности алгоритма RandomForest для всех выборок

Очевидно переобучение на малочисленных классах, однако, как и в случае с алгоритмом XGBoost, большое количество песен в классе не означает высокую точность предсказания. Стабильно хорошо угадывается фолк, хип-хоп, рок и классика, что вполне естественно: эти жанры сильно отличаются друг от друга и от любых жанров в целом, но весь секрет в том, что если их как-то совместить, то

получится поп-музыка, которая сама по себе и есть всевозможное объединение разных звучаний. Ведь популярность означает, что песня нравится большому количеству слушателей, у которых разные музыкальные вкусы, но каждому из них западает в душу что-то своё: будь то основная мелодия, расположение ударных или акустическая гитара на заднем плане в левом ухе после первого припева.

Кстати говоря, жанр Spoken не является широко распространённым, но его название «говорящее»: «Spoken word — это особый жанр в рок-музыке, фокусирующийся на эстетике слова и интонациях голоса. Его смысл — в декламационном исполнении, где упор делается не на инструментале и энергии, а на атмосфере и лирическом герое текстов» [13], – то есть в целом это чтение стихов под рок-аранжировку. Высокая точность предсказания этого жанра объясняется его отличием от рока в том, что нет вокальной мелодии, которая в рок-музыке играет важную роль, а от хип-хопа –наличием в целом живого аккомпанемента, что характерно в первую очередь для рок-музыки.

Вывод: случайный лес хорошо угадывает жанры, в которых представлено много песен и которые заметно отличаются друг от друга. Можно даже сказать, что у этого алгоритма несколько стереотипное мышление: минимально разбирающийся в музыке человек сможет различить классику, рок-музыку с большим количеством гитар и глубокими текстами, рэп-треки с отсутствием вокала как такового и фольклорные мотивы, но если он услышит песню, в которой гармонично сочетается чуть ли не всё вышеперечисленное, то он затруднится, как и затрудняется случайный лес. А таких песен сейчас всё больше и больше, потому что музыканты в поисках разнообразия творят порой совершенно не тривиальные вещи.

Рассмотрев наиболее популярные алгоритмы классификации, перейдём к бэггингу, хотя его шансы нас удивить крайне малы.

3.5. Balanced Bagging

Посмотрим на матрицу ошибок алгоритма бэггинга на обучающей выборке (рисунок 43).

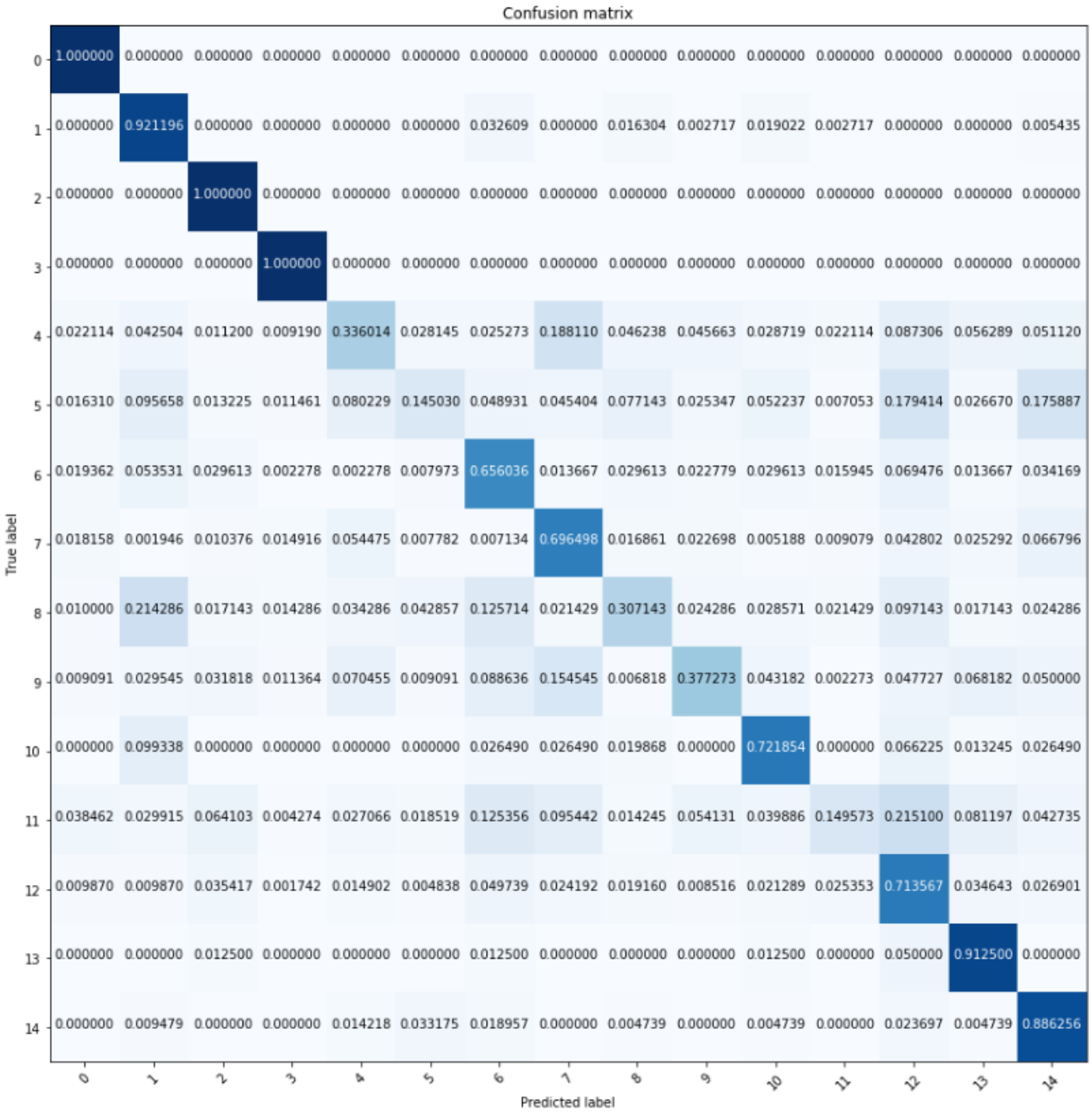

Рисунок 43 – Матрица ошибок бэггинга на обучающей выборке

Из матрицы ошибок на рисунке 43 видно, что на обучающей выборке у бэггинга не возникло больших проблем с блюзом (0), классикой (1), кантри (2), Easy Listening (3), Soul-RnB (13) и Spoken (14), также на неплохом уровне были предсказаны такие жанры, как фолк (6), хип-хоп (7), джаз (10) и рок (12). Электронная (4), инструментальная (8) и интернациональная (9) музыка была

угадана лишь в трети случаев. Электронную (4) и интернациональную (9) музыку алгоритм часто путал с хип-хопом (7), что несколько удивительно, а инструментальную (8) – с классикой (1), чем грешила и логистическая регрессия. С поп-музыкой (11) всё те же проблемы, и её бэггинг путал с роком (12) и фолком (6). Экспериментальную музыку (5) алгоритм в 18% случаев принимал за рок (12) или Spoken (14). По сравнению с результатами на обучающей выборке других алгоритмов у бэггинга больше неточностей, хотя некоторые из них новы и любопытны. Посмотрим, что изменится на валидационной выборке (рисунок 44).

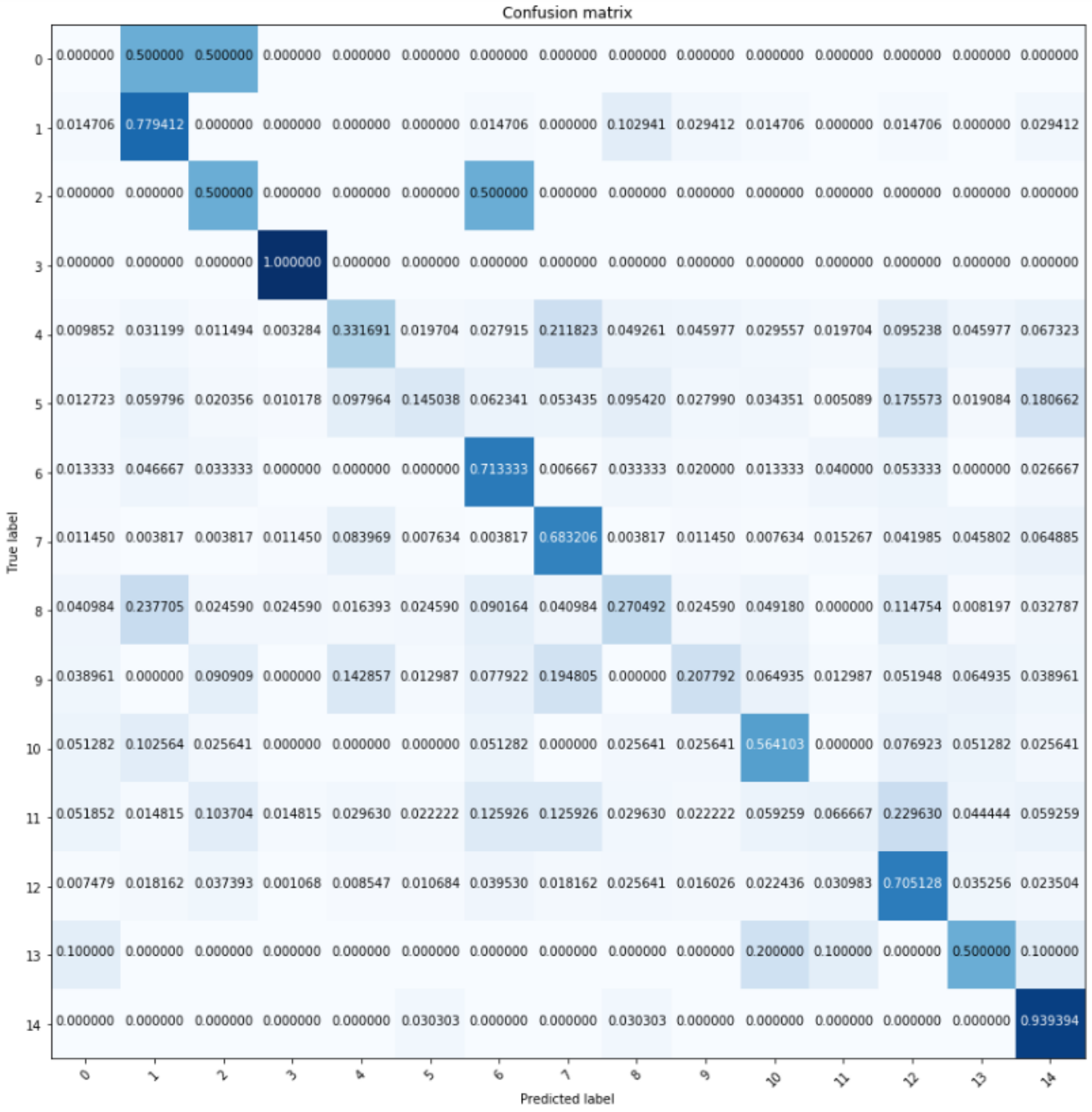

Рисунок 44 – Матрица ошибок бэггинга на валидационной выборке

На валидационной выборке результаты бэггинга ожидаемо хуже, чем на обучающей: с блюзом (0) он совсем не справился, путая его с классикой (1) и кантри (2) одинаково часто, поп-музыка (11) ещё больше перепутана с роком (12), фолком (6), хип-хопом (7) и даже кантри (2), хотя, как уже говорилось выше, на самом деле всё это имеет место быть. Жанр кантри (2) в половине случаев был угадан, а в половине – принят за фолк (6), и это тоже имеет место быть в реальности. Удивительно, но точность распознавания жанра Spoken (14) даже возросла по сравнению с обучающей выборкой. Как и на обучающей выборке, неплохие результаты для классики (1), фолка (6), хип-хопа (7) и рока (12), остальные жанры либо сопоставимы с рандомом, либо вообще вводят алгоритм в заблуждение. Посмотрим на тестовую выборку (рисунок 45).

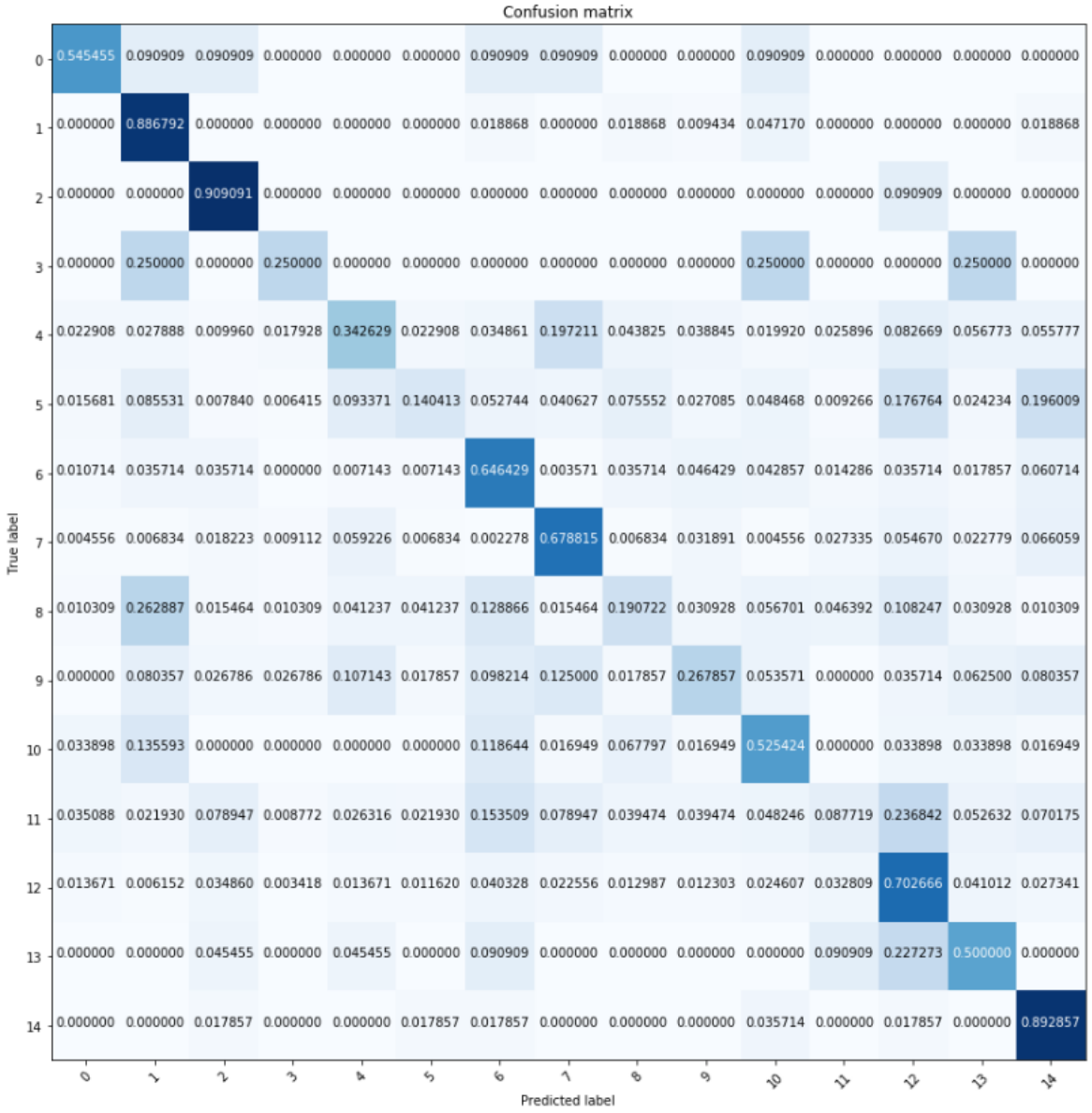

Рисунок 45 – Матрица ошибок бэггинга на тестовой выборке

Предсказания бэггинга на тестовой выборке неожиданно кажутся даже лучше, чем на валидационной (хотя это может быть обманчиво): очень хорошо угадана классика (1), кантри (2) и жанр Spoken (14), на стабильно неплохом уровне – фолк (6), хип-хоп (7) и рок (12), чуть точнее случайного выбора – блюз (0) и джаз (10). Инструментальная музыка (8) снова перепутана с классикой (1) и фолком (6), а интернациональную (9) бэггинг путает со всеми жанрами понемногу. Поп-музыка (11) опять перепутана с фолком (6) и роком (12), электронная (4) – с хип-хопом (7), а с классификацией жанра Soul-RnB (13) вместо бэггинга вполне мог бы справиться на том же уровне простой рандомайзер. Посмотрим на сравнение результатов работы алгоритма на всех выборках (рисунок 46).

			Balanced Bagging		
		Песен	train	val	test
Блюз	0	41	100%	0%	55%
Классика	1	542	92%	78%	89%
Кантри	2	52	100%	50%	81%
Easy Listening	3	16	100%	100%	25%
Электронная	4	5095	34%	33%	34%
Экспериментальная	5	6726	15%	15%	14%
Фолк	6	1308	66%	71%	65%
Хип-хоп	7	2243	70%	68%	68%
Инструментальная	8	1016	31%	27%	19%
Международная	9	629	38%	21%	27%
Джаз	10	249	72%	56%	53%
Поп	11	1065	15%	7%	9%
Рок	12	7566	71%	71%	70%
Soul-RnB	13	112	91%	50%	50%
Spoken	14	300	89%	94%	89%

Рисунок 46 – Сравнение точности алгоритма BalancedBagging для всех выборок

Общий итог работы бэггинга такой же, как и у случайного леса – недаром в их основе похожий принцип: алгоритм хорошо различает так называемые уникальные жанры, которые трудно спутать друг с другом, но плохо работает на жанрах, сочетающих в себе много направлений, даже если песен предостаточно.

Наконец настало время сравнить между собой все алгоритмы, что и сделано в следующем разделе.

3.6. Сравнение алгоритмов классификации

На рисунке 47 представлена сравнительная таблица всех описанных выше алгоритмов классификации музыкальных композиций.

			Логистич. регрессия			XGBoost			Случайный лес			Balanced Bagging		
		Песен	train	val	test	train	val	test	train	val	test	train	val	test
Блюз	0	41	96%	0%	36%	71%	0%	18%	100%	0%	18%	100%	0%	55%
Классика	1	542	91%	71%	87%	89%	66%	79%	97%	72%	84%	92%	78%	89%
Кантри	2	52	97%	33%	55%	71%	17%	18%	100%	33%	18%	100%	50%	81%
Easy Listening	3	16	100%	0%	0%	82%	0%	0%	100%	0%	0%	100%	100%	25%
Электронная	4	5095	53%	52%	53%	75%	69%	71%	62%	43%	43%	34%	33%	34%
Экспериментальная	5	6726	51%	51%	50%	81%	77%	77%	63%	53%	52%	15%	15%	14%
Фолк	6	1308	68%	69%	59%	65%	67%	58%	88%	79%	75%	66%	71%	65%
Хип-хоп	7	2243	72%	73%	73%	73%	66%	64%	88%	76%	76%	70%	68%	68%
Инструментальная	8	1016	45%	42%	37%	41%	25%	20%	79%	41%	38%	31%	27%	19%
Международная	9	629	48%	38%	37%	40%	12%	17%	86%	38%	34%	38%	21%	27%
Джаз	10	249	62%	54%	46%	38%	15%	17%	98%	49%	37%	72%	56%	53%
Поп	11	1065	20%	15%	21%	11%	1%	5%	59%	8%	11%	15%	7%	9%
Рок	12	7566	75%	73%	74%	85%	82%	82%	77%	73%	73%	71%	71%	70%
Soul-RnB	13	112	81%	30%	41%	45%	10%	9%	100%	40%	32%	91%	50%	50%
Spoken	14	300	87%	82%	79%	68%	36%	45%	100%	70%	70%	89%	94%	89%

Рисунок 47 – Сравнение алгоритмов классификации

Если расфокусировать зрение, то при взгляде на часть таблицы, где находится логистическая регрессия, видно, что слева направо нет резкого контраста, а это значит, что результаты на валидационной и тестовой выборках не очень сильно отличаются от результата на обучающей, то есть можно предположить об отсутствии переобучения, чего не скажешь о голосующих классификаторах Random Forest и Balanced Bagging.

Все алгоритмы переобучились на малочисленных классах (блюз, Easy Listening, кантри, Soul-RnB, джаз) и не справились с поп-музыкой, хотя случайный лес на обучающей выборке дал призрачную надежду, но тут же разбил её о валидационную и добил о тестовую. Но нельзя не сказать и о хорошем: все алгоритмы справились с классикой, фолком, хип-хопом и роком.

Также нельзя не отметить, что у случайного леса оказалась самая большая склонность к переобучению по всем классам, и бэггинг от него недалеко ушёл. XGBoost, между прочим, переобучался ещё меньше, чем логистическая регрессия, но в его части таблицы контраст больше ввиду резкого снижения качества на валидационной выборке относительно обучающей по сравнению с логистической регрессией.

Посмотрим на эту же таблицу, но без малочисленных классов, чтобы больше сосредоточиться на основных (рисунок 48).

			Логистич. регрессия			XGBoost			Случайный лес			Balanced Bagging		
		Песен	train	val	test	train	val	test	train	val	test	train	val	test
Классика	1	542	91%	71%	87%	89%	66%	79%	97%	72%	84%	92%	78%	89%
Электронная	4	5095	53%	52%	53%	75%	69%	71%	62%	43%	43%	34%	33%	34%
Экспериментальная	5	6726	51%	51%	50%	81%	77%	77%	63%	53%	52%	15%	15%	14%
Фолк	6	1308	68%	69%	59%	65%	67%	58%	88%	79%	75%	66%	71%	65%
Хип-хоп	7	2243	72%	73%	73%	73%	66%	64%	88%	76%	76%	70%	68%	68%
Инструментальная	8	1016	45%	42%	37%	41%	25%	20%	79%	41%	38%	31%	27%	19%
Международная	9	629	48%	38%	37%	40%	12%	17%	86%	38%	34%	38%	21%	27%
Поп	11	1065	20%	15%	21%	11%	1%	5%	59%	8%	11%	15%	7%	9%
Рок	12	7566	75%	73%	74%	85%	82%	82%	77%	73%	73%	71%	71%	70%
Spoken	14	300	87%	82%	79%	68%	36%	45%	100%	70%	70%	89%	94%	89%

Рисунок 48 – Сравнение алгоритмов классификации на превалирующих классах

С классикой все алгоритмы справились одинаково хорошо, бэггинг – немного лучше. С электронной музыкой единственный справился XGBoost – на среднем уровне, тогда как логистическая регрессия лишь чуть точнее случайного выбора, случайный лес переобучился, а бэггинг стабильно угадывал лишь в одном из трёх случаев. Похожая ситуация и с экспериментальной музыкой: достойно справился XGBoost, чуть лучше рандома – логистическая регрессия, переобучился случайный лес, совсем плохо – бэггинг. С фолком справились все алгоритмы, но случайный лес заметно лучше остальных. Аналогично и с хип-хопом, но случайный лес лучше уже не так заметно. С инструментальной и международной музыкой не справился ни один алгоритм, но делала это стабильно только логистическая регрессия. С поп-музыкой также не справился ни один алгоритм, но логистическая регрессия угадывала популярные песни хотя бы в одном случае из пяти, тогда как остальные алгоритмы – вдвое реже. С рок-музыкой справились все алгоритмы, точнее других оказался XGBoost. С интересным жанром Spoken справились логистическая регрессия и, как ни странно, бэггинг – ещё и лучше. Случайный лес переобучился, хотя и не так сильно, как на других классах, XGBoost тоже, но ещё хуже.

Стоит отметить, что, несмотря на схожий принцип работы, бэггинг не является ухудшенной версией случайного леса: он как раз таки меньше склонен к переобучению (во всяком случае в данном эксперименте) и правильно классифицирует те же жанры, что и случайный лес, причём ненамного хуже, а где-то – даже лучше (классика и Spoken).

Вывод: чтобы уметь предсказывать большее количество классов, стоит объединить несколько классификаторов – например, для рока, электронной и экспериментальной музыки взять XGBoost, для фолка и хип-хопа – случайный лес, а затем добавить к ним бэггинг для классической музыки и жанра Spoken. С другой стороны, нельзя отметать тот факт, что логистическая регрессия оказалась самой стабильной при разбиении выборок, то есть вполне можно оптимизировать какие-то её параметры и тоже включить в композицию алгоритмов. Впрочем, есть поговорка «лучшее – враг хорошего»: необходимо как минимум эмпирически неоднократно проверить, приведёт ли сложный составной классификатор к улучшению точности предсказания.

ЗАКЛЮЧЕНИЕ

В рамках годовой курсовой работы было построено четыре алгоритма классификации музыкальных композиций: логистическая регрессия, XGBoost, случайный лес и Balanced Bagging. Было проведено тестирование каждого из них на обучающей, валидационной и тестовой выборках данных, после чего были выявлены наиболее подходящие для классификации каждого жанра алгоритмы, либо было установлено, что ни один из них для классификации конкретного жанра не подходит. Сравнение производилось как по метрикам качества (f-score, precision и recall), так и в наглядном виде – в виде матриц ошибок, что позволило сделать предположение о необходимости создания ансамбля классификаторов, хотя, как уже было сказано выше, не факт, что композиция алгоритмов даст лучший результат. Это отдельная большая задача, о подходах к её решению можно прочитать в статьях [14] и [15].

Общий итог: хорошо поддаются классификации те музыкальные жанры, которые легко отличит друг от друга любой человек. Они практически не пересекаются по используемым инструментам, их обработке, по технике или вообще наличию вокала. Популярную музыку выделить из других жанров в данной работе не получилось ввиду её похожести на все другие жанры сразу. С инструментальной музыкой аналогичная ситуация, потому что к этому жанру относятся любые аранжировки.

Для решения задачи классификации музыкальных композиций на уровне выпуска алгоритма как приложения на рынок необходимо провести ещё много экспериментов, варьируя параметры алгоритмов, следя за сбалансированностью классов и вообще их составом. В задачах машинного обучения никогда нет заранее известной пошаговой инструкции для достижения цели, всё получается эмпирическим путём. Или не получается. Но отсутствие результата – тоже результат, хотя такая формулировка очень удивительна для уха человека, привыкшего, что в любой задаче должен быть ответ.

СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ

1. Michaël Defferrard, Kirell Benzi, Pierre Vandergheynst, Xavier Bresson. FMA: A Dataset For Music Analysis [Электронный ресурс]. – 2017. – URL: https://github.com/mdeff/fma (дата обращения: 16.11.2018)
2. Michaël Defferrard, Kirell Benzi, Pierre Vandergheynst, Xavier Bresson. FMA: A Dataset For Music Analysis [Электронный ресурс]. – 2017. – URL: https://arxiv.org/abs/1612.01840 (дата обращения: 16.11.2018)
3. Brian McFee, Colin Raffel, Dawen Liang, Daniel P.W. Ellis, Matt McVicar, Eric Battenberg, Oriol Nieto. Librosa: Audio and Music Signal Analysis in Python [Электронный ресурс]. – 2015. – URL: http://conference.scipy.org/proceedings/scipy2015/pdfs/brian_mcfee.pdf (дата обращения: 18.11.2018)
4. Мотренко А. П., Стрижов В. В. Многоклассовая логистическая регрессия для прогноза вероятности наступления инфаркта / А. П. Мотренко, В. В. Стрижов // Известия Тульского государственного университета/ Естественные науки. – 2012. – Вып. 1. – С. 153-162.
5. Northern Eurasia Contests [Электронный ресурс]. – URL: https://neerc.ifmo.ru/wiki/index.php?title=XGBoost (дата обращения: 02.12.2018).
6. 10 строк для диагностики болезни Паркинсона при помощи XGBoost [Электронный ресурс]. – 2018. – URL: https://proglib.io/p/xgboost/ (дата обращения: 03.12.2018).
7. Дьяконов Александр. Случайный лес (Random Forest) [Электронный ресурс]. – 2016. – URL: https://dyakonov.org/2016/11/14/случайный-лес-random-forest/ (дата обращения: 27.11.2018).
8. Open Data Science. Открытый курс машинного обучения. Тема 5. Композиции: бэггинг, случайный лес. – 2017. – URL: https://habr.com/ru/company/ods/blog/324402/ (дата обращения: 05.12.2018).
9. MachineLearning.ru. Профессиональный информационно-аналитический ресурс, посвященный машинному обучению, распознаванию образов и

интеллектуальному анализу данных [Электронный ресурс]. – URL: http://www.machinelearning.ru/wiki/index.php?title=Бэггинг (дата обращения: 05.12.2018).
10. Scikit-learn. Machine learning in Python [Электронный ресурс]. – URL: https://scikit-learn.org/0.20/supervised_learning.html#supervised-learning (дата обращения: 17.11.2018).
11. Справка библиотеки librosa [Электронный ресурс]. – URL: https://librosa.github.io/librosa/feature.html (дата обращения: 05.12.2018).
12. Constant-Q Transform and Chroma [Электронный ресурс]. – URL: https://musicinformationretrieval.com/chroma.html (дата обращения: 05.12.2018).
13. Анна Курдюкова. 5 групп, после которых вы полюбите spoken word [Электронный ресурс]. – URL: http://rockcult.ru/po/spoken-word-bands/ (дата обращения: 12.12.2018).
14. Дьяконов Александр. Стекинг (Stacking) и блендинг (Blending) [Электронный ресурс]. – 2017. – URL: https://dyakonov.org/2017/03/10/стекинг-stacking-и-блендинг-blending/ (дата обращения: 27.04.2019).
15. Дьяконов Александр. Ансамбли в машинном обучении [Электронный ресурс]. – 2019. – URL: https://dyakonov.org/2019/04/19/ансамбли-в-машинном-обучении/ (дата обращения: 27.04.2019).