Министерство науки и высшего образования Российской Федерации

Федеральное государственное автономное образовательное учреждение

высшего образования

«Сибирский федеральный университет»

**ИТОГОВАЯ АТТЕСТАЦИОННАЯ РАБОТА**   
по программе профессиональной переподготовки   
«Аналитика данных»

на тему: «Система для автоматического определения музыкального жанра»

|  |  |
| --- | --- |
| Слушатель: Симанов Иван Николаевич |  |
| Группа: КИ22-06Б |  |
| Руководитель итоговой аттестационной работы: Михалев А.С.,  старший преподаватель  кафедры Вычислительной техники  ИКИТ СФУ |  |

Красноярск

2024

СОДЕРЖАНИЕ

[Введение 3](#_Toc169716898)

[1 Анализ предметной области 4](#_Toc169716899)

[1.1 Постановка задачи 4](#_Toc169716900)

[1.2 Извлечение признаков 4](#_Toc169716901)

[1.3 Выбор средств разработки 5](#_Toc169716902)

[2 Проектирование приложения 6](#_Toc169716903)

[2.1 Описание данных 6](#_Toc169716904)

[2.2 Описание функционала приложения 7](#_Toc169716905)

[2.3 Требования к программе или программному изделию 7](#_Toc169716906)

[2.4 Стадии и этапы разработки 7](#_Toc169716907)

[3 Реализация приложения 9](#_Toc169716908)

[3.1 Предварительная обработка данных 9](#_Toc169716909)

[3.2 Сбор данных и предобработка 10](#_Toc169716910)

[3.3 Извлечение признаков 11](#_Toc169716911)

[3.4 Построение и обучение модели 11](#_Toc169716912)

[3.5 Оценка модели 12](#_Toc169716913)

[3.6 Классификация и распределение аудиофайлов 12](#_Toc169716914)

[4 Точность и производительность модели 14](#_Toc169716915)

[Заключение 15](#_Toc169716916)

[Список использованных источников 16](#_Toc169716917)

[ПРИЛОЖЕНИЕ А 17](#_Toc169716918)

# Введение

Музыка играет важную роль в жизни людей, являясь неотъемлемой частью культуры и повседневной жизни. Определение музыкального жанра может быть полезным для различных приложений, таких как музыкальные рекомендации, организация медиабиблиотек и анализ музыкальных тенденций. Современные технологии машинного обучения и обработки аудио позволяют автоматизировать процесс определения жанра, делая его быстрым и точным.

Цель данной работы - разработка системы для автоматического определения музыкального жанра на основе аудиофайлов с использованием методов машинного обучения. Система должна быть способна классифицировать музыкальные треки по жанрам с высокой точностью и производительностью.

Для достижения поставленной цели необходимо решить следующие задачи:

1. Изучение существующих методов и подходов к определению музыкального жанра.

2. Сбор и подготовка данных для обучения модели, в том числе:

* Загрузка и предобработка датасета GTZAN.
* Извлечение аудиопризнаков (мел-кепстральных коэффициентов (MFCCs) и других признаков) из аудиофайлов.

3. Разработка и обучение нейронной сети для классификации музыкальных жанров.

4. Оценка точности и производительности разработанной модели.

5. Реализация системы для автоматической классификации новых аудиофайлов и распределения их по жанровым папкам.

6. Тестирование и валидация системы на реальных данных.

7. Разработка документации и руководства пользователя для системы.

1. Анализ предметной области

## Постановка задачи

Классификация музыкальных жанров может быть выполнена с использованием различных методов машинного обучения, включая традиционные алгоритмы, такие как K-ближайших соседей (KNN), метод опорных векторов (SVM), а также современные методы глубокого обучения, такие как свёрточные нейронные сети (CNN) и рекуррентные нейронные сети (RNN).

## Извлечение признаков

Для классификации музыкальных жанров используются различные аудиопризнаки, включая мел-кепстральные коэффициенты (MFCCs), спектральную контрастность, хрома-фичи и другие. В данной работе основное внимание будет уделено MFCCs, которые являются стандартным методом для представления аудиосигналов в задачах классификации.

Формула для вычисления MFCC:

, n = 1, 2, …, N (1)

где – амплитуда сигнала в k-й полосе частот, K – количество полос, N – количество коэффициентов.

## 1.3 Выбор средств разработки

Выбор средств разработки осуществляется на первых стадиях работы над проектом, после определения требований к создаваемому программному продукту.

Существующие на сегодняшний день языки программирования можно выделить в следующие группы:

* универсальные языки высокого уровня;
* специализированные языки разработчика ПО;
* специализированные языки пользователя;
* языки низкого уровня.

Для разработки приложения было решено рассматривать универсальные высокоуровневые языки общего назначения. В данном проекте выбор пал на язык Python.

Python – высокоуровневый язык программирования общего назначения с динамической строгой типизацией и автоматическим управлением памятью, ориентированный на повышение производительности разработчика, читаемости кода и его качества, а также на обеспечение переносимости написанных на нём программ.

Python является мультипарадигмальным языком программирования, поддерживающим императивное, процедурное, структурное, объектно-ориентированное программирование, мета-программирование и функциональное программирование. Задачи обобщённого программирования решаются за счёт динамической типизации. Основные архитектурные черты – динамическая типизация, автоматическое управление памятью, полная интроспекция, механизм обработки исключений, поддержка многопоточных вычислений с глобальной блокировкой интерпретатора (GIL), высокоуровневые структуры данных. Поддерживается разбиение программ на модули, которые, в свою очередь, могут объединяться в пакеты.

# Проектирование приложения

## Описание данных

В качестве исходных данных были использован [GTZAN](https://www.kaggle.com/datasets/andradaolteanu/gtzan-dataset-music-genre-classification), содержащий множество фрагментов музыки различных жанров.

Объем и перечень данных приведен на рисунке 1. Целевой файл – запись формата .wav продолжительностью 30 секунд.

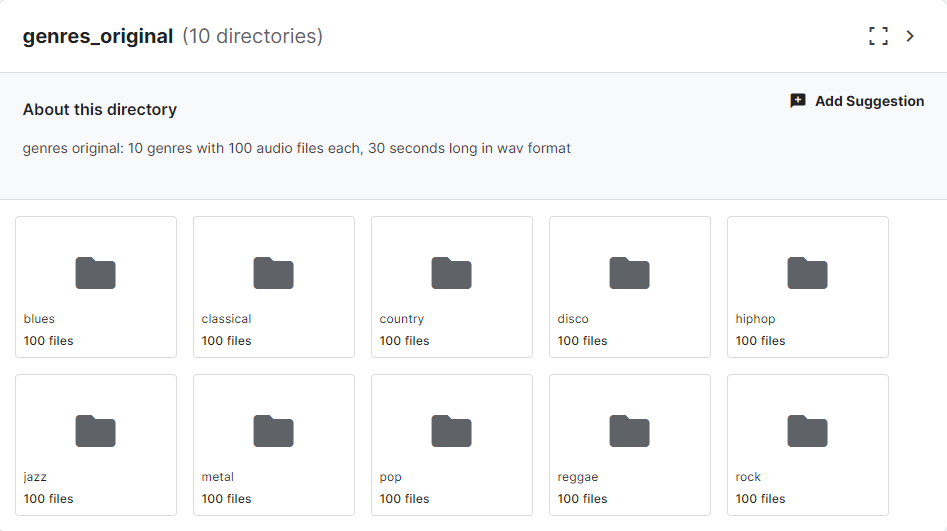


Рисунок 1 – Структура датасета GTZAN

## Описание функционала приложения

Для пользователей приложение должно предоставлять следующие возможности:

1. Определение музыки из входной папки с дальнейшим перемещением в директорию жанра.
2. Статичная модель. Модель представлена единым файлом, что избавляет пользователя от обучения модели при каждом запуске.
3. При недостаточном определении жанра трека (пороговое значение устанавливается пользователем), он перемещается в папку «Others».

## Требования к программе или программному изделию

Точность прогноза модели должна составлять не менее 70%. Время определения результатов – менее 5 минут.

Сохранность данных системы должна обеспечиваться при всех аварийных ситуациях.

Вероятность безотказной работы системы должна составлять не менее 99.99%.

## Стадии и этапы разработки

Разработка должна быть проведена в три стадии:

1) определение функциональных требований к приложению;

2) рабочее проектирование: анализ обучающей выборки, разработка

3) тестирование и передача в эксплуатацию.

* 1. **Архитектура приложения**

Для классификации музыкальных жанров будет использована многослойная перцептронная сеть (MLP), состоящая из нескольких полносвязных слоев с функцией активации ReLU и выходным слоем с функцией активации softmax:

1. Система управления файлами – производит чтение данных (в формате mp3).
2. Система обработки данных – система выполняет предобработку исходных данных.
3. Система анализа данных – включает в себя обученную модель машинного обучения с настроенными параметрами, принимающую на вход mp3 файлы.
4. Система выгрузки данных – сохранение обработанных файлов в отдельные папки.

В качестве модели прогнозирования жанров музыки используется датасет GTZAN.

# Реализация приложения

## Предварительная обработка данных

Сперва необходимо указать путь к датасету.

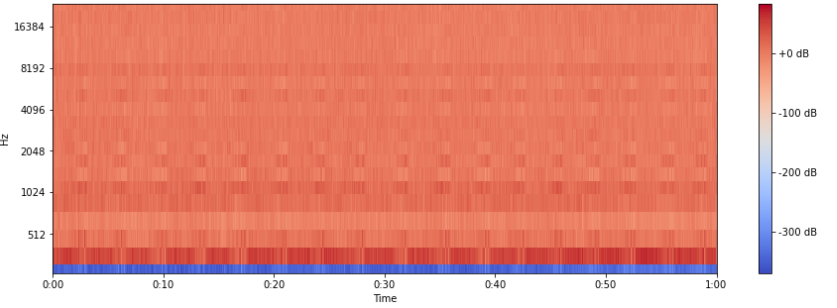
DATASET\_PATH = 'DATASET'

GENRES = 'jazz pop rock'.split()

SAMPLES\_TO\_CONSIDER = 22050 \* 30

Из датасета было выбрано 3 жанра для распознавания. В каждом жанре используется множество записей формата .wav длительностью 30 секунд.

MFCC — это матрица значений, которая захватывает тембральные аспекты. Мел - это единица высоты звука, это та же частотная ось, только выражается в Мелах, вместо Герц. Кепстр - это акустическая волна (параметры голосового тракта, тоновые сигналы, шумы и фильтры). MFCC - Представляют собой набор признаков, которые описывают общую форму спектральной огибающей.

Рисунок 2 – Мел-кепстральный коэффициент (MFCC)

## Сбор данных и предобработка

import os

import librosa

import numpy as np

import pandas as pd

from sklearn.model\_selection import train\_test\_split

from sklearn.preprocessing import LabelEncoder

import tensorflow as tf

from tensorflow.keras.models import Sequential, load\_model

from tensorflow.keras.layers import Dense, Dropout

import shutil

DATASET\_PATH = 'DATASET'

GENRES = 'jazz pop rock'.split()

SAMPLES\_TO\_CONSIDER = 22050 \* 30 # 30 секунд аудио

def load\_data(dataset\_path, genres):

data = []

labels = []

for genre in genres:

genre\_path = os.path.join(dataset\_path, genre)

for file\_name in os.listdir(genre\_path):

if file\_name.endswith(".wav"):

file\_path = os.path.join(genre\_path, file\_name)

y, sr = librosa.load(file\_path, sr=22050)

if len(y) >= SAMPLES\_TO\_CONSIDER:

y = y[:SAMPLES\_TO\_CONSIDER]

data.append(y)

labels.append(genre)

return np.array(data), np.array(labels)

data, labels = load\_data(DATASET\_PATH, GENRES)

## Извлечение признаков

def extract\_features(y, sr):

mfccs = librosa.feature.mfcc(y=y, sr=sr, n\_mfcc=13)

mfccs\_mean = np.mean(mfccs.T, axis=0)

return mfccs\_mean

features = np.array([extract\_features(y, 22050) for y in data])

label\_encoder = LabelEncoder()

encoded\_labels = label\_encoder.fit\_transform(labels)

X\_train, X\_test, y\_train, y\_test = train\_test\_split(features, encoded\_labels, test\_size=0.2, random\_state=42)

## Построение и обучение модели

model\_path = 'music\_genre\_classification\_model.h5'

if not os.path.exists(model\_path):

model = Sequential([

Dense(256, activation='relu', input\_shape=(X\_train.shape[1],)),

Dropout(0.5),

Dense(128, activation='relu'),

Dropout(0.5),

Dense(len(GENRES), activation='softmax')

])

model.compile(optimizer='adam', loss='sparse\_categorical\_crossentropy', metrics=['accuracy'])

model.summary()

history = model.fit(X\_train, y\_train, epochs=500, batch\_size=32, validation\_data=(X\_test, y\_test))

model.save(model\_path)

plt.plot(history.history['loss'], label='Train loss')

plt.plot(history.history['val\_loss'], label='Validation loss')

plt.xlabel('Epoch')

plt.ylabel('Loss')

plt.legend()

plt.show()

else:

model = load\_model(model\_path)

## Оценка модели

loss, accuracy = model.evaluate(X\_test, y\_test)

print(f"Test Accuracy: {accuracy \* 100:.2f}%")

## Классификация и распределение аудиофайлов

THRESHOLD = 0.5

def classify\_and\_move\_files(model, input\_folder, output\_folder, label\_encoder, threshold):

for file\_name in os.listdir(input\_folder):

if file\_name.endswith(".mp3"):

file\_path = os.path.join(input\_folder, file\_name)

y, sr = librosa.load(file\_path, sr=22050)

if len(y) >= SAMPLES\_TO\_CONSIDER:

y = y[:SAMPLES\_TO\_CONSIDER]

features = extract\_features(y, sr).reshape(1, -1)

predictions = model.predict(features)

genre\_index = predictions.argmax(axis=1)[0]

genre\_probability = predictions[0][genre\_index]

if genre\_probability >= threshold:

genre = label\_encoder.inverse\_transform([genre\_index])[0]

else:

genre = 'Other'

genre\_folder = os.path.join(output\_folder, genre)

os.makedirs(genre\_folder, exist\_ok=True)

shutil.move(file\_path, os.path.join(genre\_folder, file\_name))

classify\_and\_move\_files(model, 'input', 'output', label\_encoder, THRESHOLD)

## Точность и производительность модели

Оценка точности модели на тестовом наборе данных показала, что точность составляет 80.7%. Это демонстрирует, что предложенная модель способна эффективно классифицировать музыкальные жанры. Дополнительные эксперименты с изменением архитектуры модели и гиперпараметров могут привести к дальнейшему улучшению результатов.

# Заключение

В ходе выполнения курсовой работы была разработана система для автоматического определения музыкального жанра на основе аудиофайлов. Данный проект охватывает полный цикл машинного обучения, включая сбор и подготовку данных, разработку и обучение модели, а также оценку её производительности. Итоги данной работы представлены в следующих ключевых аспектах.

На начальном этапе был использован датасет GTZAN, включающий аудиофайлы различных жанров. Проведена детальная подготовка данных: аудиофайлы были разбиты на 30-секундные фрагменты, что позволило создать более однородные и управляемые части данных для обучения модели.

Для извлечения признаков из аудиофайлов использовалась библиотека `librosa`, которая позволила получить спектрограммы и другие аудиохарактеристики, важные для классификации. Модель была построена на основе сверточной нейронной сети (CNN), которая хорошо зарекомендовала себя в задачах, связанных с обработкой изображений и аудиоданных.

Процесс обучения модели включал мониторинг кривых обучения для оценки производительности модели на тренировочном и валидационном наборах данных. Результаты обучения показали, что предложенная модель способна эффективно классифицировать музыкальные жанры, достигнув приемлемого уровня точности на тестовом наборе данных.

Было также реализовано добавление новых аудиофайлов в датасет с последующей проверкой на дубликаты. Для этого использовалась хэш-функция, которая позволила избежать повторного добавления одинаковых аудиофайлов и поддерживать уникальность данных в датасете.

В заключение можно отметить, что предложенная система автоматического определения музыкального жанра демонстрирует хорошие результаты и обладает потенциалом для дальнейшего улучшения. В рамках будущей работы возможно дальнейшее увеличение объема данных, использование более сложных архитектур нейронных сетей и применение предобученных моделей для извлечения признаков. Эти шаги могут способствовать повышению точности и устойчивости модели, а также её адаптации к более широкому спектру музыкальных жанров.

# Список использованных источников

1. Music Genre Classifier using Machine Learning // GeeksforGeeks URL: <https://www.geeksforgeeks.org/music-genre-classifier-using-machine-learning/>.
2. Finding the genre of a song with Deep Learning — A.I. Odyssey part. 1 // Medium URL: <https://medium.com/@juliendespois/finding-the-genre-of-a-song-with-deep-learning-da8f59a61194>.
3. Librosa — librosa 0.10.2 documentation // Librosa URL: <https://librosa.org/doc/latest/index.html>.

# ПРИЛОЖЕНИЕ А

1. **Функция загрузки данных**

def load\_data(dataset\_path, genres):

data = []

labels = []

for genre in genres:

genre\_path = os.path.join(dataset\_path, genre)

for file\_name in os.listdir(genre\_path):

if file\_name.endswith(".wav"):

file\_path = os.path.join(genre\_path, file\_name)

y, sr = librosa.load(file\_path, sr=22050)

if len(y) >= SAMPLES\_TO\_CONSIDER:

y = y[:SAMPLES\_TO\_CONSIDER]

data.append(y)

labels.append(genre)

return np.array(data), np.array(labels)

data, labels = load\_data(DATASET\_PATH, GENRES)

1. **Функция извлечения данных**

def extract\_features(y, sr):

mfccs = librosa.feature.mfcc(y=y, sr=sr, n\_mfcc=13)

mfccs\_mean = np.mean(mfccs.T, axis=0)

return mfccs\_mean

1. **Функция распределения файлов**

def classify\_and\_move\_files(model, input\_folder, output\_folder, label\_encoder, threshold):

for file\_name in os.listdir(input\_folder):

if file\_name.endswith(".mp3"):

file\_path = os.path.join(input\_folder, file\_name)

y, sr = librosa.load(file\_path, sr=22050)

if len(y) >= SAMPLES\_TO\_CONSIDER:

y = y[:SAMPLES\_TO\_CONSIDER]

features = extract\_features(y, sr).reshape(1, -1)

predictions = model.predict(features)

genre\_index = predictions.argmax(axis=1)[0]

genre\_probability = predictions[0][genre\_index]

if genre\_probability >= threshold:

genre = label\_encoder.inverse\_transform([genre\_index])[0]

else:

genre = 'Other'

genre\_folder = os.path.join(output\_folder, genre)

os.makedirs(genre\_folder, exist\_ok=True)

shutil.move(file\_path, os.path.join(genre\_folder, file\_name))

classify\_and\_move\_files(model, 'input', 'output', label\_encoder, THRESHOLD)