ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ АВТОНОМНОЕ
ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ
ВЫСШЕГО ОБРАЗОВАНИЯ
«НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ УНИВЕРСИТЕТ
«ВЫСШАЯ ШКОЛА ЭКОНОМИКИ»

Факультет информатики, математики и компьютерных наук

**Программа подготовки бакалавров по направлению
01.03.02 Прикладная математика и информатика**

Гараев Рамазан Арзу оглы

КУРСОВАЯ РАБОТА

Улучшение метода обнаружения дрона в звуковом потоке

Научный руководитель
Старший преподаватель

А.А. Пономаренко

Нижний Новгород, 2024

Содержание

1 Введение

Аудиоанализ представляет собой развивающуюся область применения машинного обучения, включающую цифровую обработку сигналов, автоматическое распознавание речи, классификацию, тегирование и генерацию музыки и другие направления. Одним из возможных применений машинного обучения в аудиоанализе является обнаружение различных объектов, в том числе беспилотных летательных аппаратов (дронов). В последнее время дроны получили широкое распространение, что привело к необходимости разработки методов их обнаружения для обеспечения безопасности и контроля в охраняемых зонах.

Стоит отметить, что количество дронов и случаев их использования растет, создавая большие объемы звуковых данных, которые требуют автоматизированных методов анализа для эффективной обработки. Обнаружение дронов по звуковым сигналам становится актуальной задачей, так как дроны издают характерные звуки во время полета, что позволяет использовать эти звуковые сигнатуры для их идентификации.

Поэтому цель данной курсовой работы – разработка программного обеспечения для обнаружения дронов в звуковом потоке с использованием методов машинного обучения. В частности, предлагается использовать алгоритм XGBoost для классификации звуковых данных с целью выделения звуков дронов.

Обозначения:

Пусть у нас есть (X_{train}, Y_{train}) и (X_{test}, Y_{test}) - обучающая и тестовая выборки,

такие что $X_{train}, X_{test} \subset X$ и $Y_{train}, Y_{test} \subset Y$,

где X – матрица признаков, Y – список меток;

$X = (x_1, x_2, .., x_i, \ldots, x_n)$, $x_i \in \{k_0, k_1, \ldots\}$, k_j - признак, $|x_i|$ - *кол-во признаков*

$Y = (y_1, y_2, ..., y_i, ..., y_n)$, $y_i \in \{0, 1\}$, (сигнал: 0 - не дрон, 1 - дрон)

$n = \frac{t_{len} - window_{size}}{step_{size}} + 1$, где t_{len} - длина файла,

$window_{size}$ - размер окна, $step_{size}$ - шаг смещения окна

В этом исследовании были использованы данные [1], содержащие реальные аудиозаписи, издаваемые дронами. Датасет включает три основные папки:

Binary_Drone_Audio: Здесь содержатся аудиозаписи, разделенные на два класса: с дроном и без дрона. В папке находится 1333 аудиозаписей с дронами и 10373 без них.

Micro_yes_drone: Эта папка содержит 205 аудиозаписей дронов, которые были записаны лично автором на микрофон.

Multiclass_Drone_Audio: Здесь также присутствуют аудиозаписи, разделенные на два класса: с дроном и без дрона. В данной папке также содержится 1333 аудиозаписей с дронами и 10373 без них.

Этот набор данных обеспечивает достаточное количество примеров для обучения и тестирования моделей обнаружения дронов в звуковом потоке.

Таким образом в работе ставятся следущие задачи:

1. Изучить признаки, извлекаемые из аудиосигнала и методы их получения.
2. Обучить модель машинного обучения, используя признаки.
3. Узнать наиболее весомые признаки для модели.
4. Протестировать обученную модель.
5. Проанализировать проделанную работу.

2 Детали реализации

При работе использовался интерпретатор Python3.10; среда разработки – Jupyter Notebook, GoogleCollab.

2.1 Разметка данных

Поскольку наша задача заключается в создании модели с высокой точностью прогнозирования, мы планируем в дальнейшем улучшить результаты, фокусируясь на анализе отдельных частей файла, а не его общей обработке.

Однако в файле, представленном в [1], метки присваиваются для всего файла целиком. Поскольку длительность каждого аудиофайла варьируется от 0,5 до 5 секунд, присутствие звуков дрона занимает лишь часть этого времени. Поэтому мы будем работать с окнами и смещениями, чтобы наша модель могла обрабатывать аудио любой длительности в процессе обучения.

Для более качественной обработки аудиофрагментов пересмотрена также сбалансирование данных, так как звуков дрона в 10 раз меньше, чем звуков класса «не дрон». В противном случае, модель будет определять один из классов лучше, чем другой. Также это чревато недообучением модели или даже переобучением.

Для работы со звуком будет использоваться библиотка ***librosa***. Прежде всего необходимо преобразовать звуковой файл в цифровой формат.

```
In [7]: import librosa
        sound , sr = librosa.load('/Users/midasxlr/Desktop/B_S2_D1_067-bebop_001_.wav')
        print(sound)
        print(sr)

        [ 0.02102794 -0.05801165 -0.12614687 ... -0.02967592 -0.06080377
          0.        ]
        22050
```

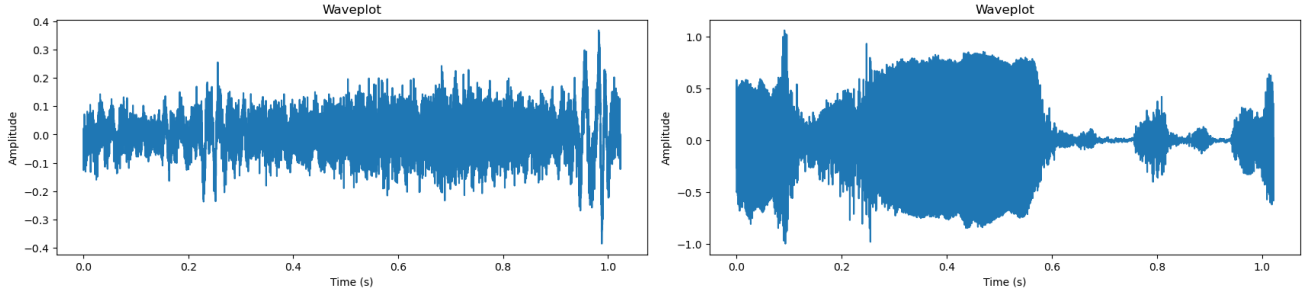
В результате получается numpy массив с частотой дикретизации sr = 22050 Гц. В данном случае sr измеряется в кол-ве семплов в секунду. Зададим окно размером 1 секунда, а сдвиг - 0.25 секунды.

Таким образом, для каждого аудиофайла мы извлекаем признаки из неперекрывающихся окон длительностью 1 секунду с шагом в 0.25 секунды.

Далее в построении графиков используются аудиофайлы **(1) B_S2_D1_067-bebop_001_.wav** - с дроном **1-5996-A-60.wav** - без дрона.

Перед извлечением признаков из рассматриваемых звуковых дорожек, представим их графически с помощью:

```
# Построение waveplot
plt.figure(figsize=(10, 4))
time = np.arange(0, len(sound)) / sr
plt.plot(time, sound)
plt.xlabel('Time (s)')
plt.ylabel('Amplitude')
plt.title('Waveplot')
plt.show()
```



(1) - дрон (2) - не дрон

Рис. 1: Осциллограммы рассматриваемых аудиофайлов.

На Рис. 1 показана зависимость амплитуды звука от времени. Взглянув на них мы можем сделать некие выводы.

Первый график:

1. Амплитуда звукового сигнала находится в пределах от -0.4 до 0.4.
2. Сигнал выглядит более равномерным и шумоподобным, без явно выраженных изменений в структуре.
3. Такой сигнал может соответствовать звуку работающего дрона, так как звуки дронов часто имеют постоянную частоту и могут быть шумными из-за работы моторов и пропеллеров.

Второй график:

1. Амплитуда сигнала находится в пределах от -1 до 1.
2. Сигнал имеет более выраженные изменения в структуре, видно несколько участков с разными уровнями амплитуды.
3. Сигнал начинается с высокого уровня амплитуды, затем уменьшается, затем снова увеличивается и вновь уменьшается.
4. Такой сигнал больше похож на какой-то другой звуковой источник, который имеет более сложную временную структуру. Это может быть речь, музыка или другой звук, не связанный с дроном.

Все это конечно гипотезы, в самой сути при таком анализе помогают в основном спектрограммы.

```
# Построение спектрограммы
S = librosa.feature.melspectrogram(y=sound, sr=sr, n_mels=128)
S_dB = librosa.power_to_db(S, ref=np.max)

plt.figure(figsize=(10, 4))
librosa.display.specshow(S_dB, sr=sr, x_axis='time', y_axis='mel')
plt.colorbar(format='%+2.0f dB')
plt.title('Mel-Spectrogram')
plt.xlabel('Time (s)')
plt.ylabel('Frequency (Hz)')
plt.show()
```

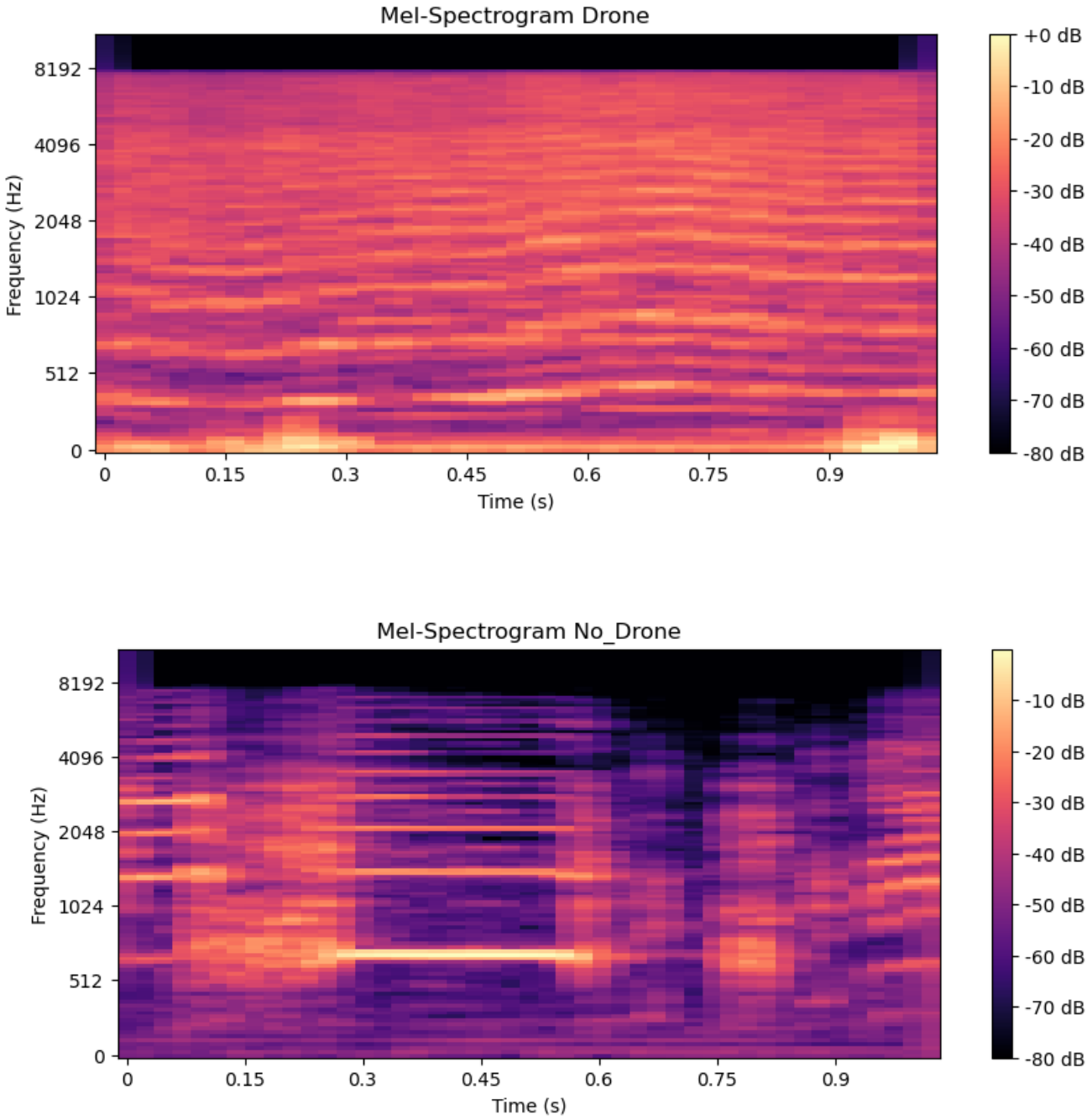


Рис. 2: Спектограммы рассматриваемых аудиофайлов.

На вертикальной оси представлены частоты (от 0 до 8192 Гц), а на горизонтальной — время.

На спектрограмме на Рис. 2 'Mel-Spectrogram Drone' отображен сигнал, издаваемый пропеллерами дрона, на протяжении всего аудиофайла. На Рис. 2 'Mel-Spectrogram No_Drone' виден небольшой фрагмент, похожий на звук пропеллера дрона, однако он сопровождается значительным количеством шума, что может затруднить правильное предсказание модели.

Для обучения модели простого дискретного представления аудиофайла недостаточно, поэтому рассмотрим некоторые возможные признаки, которые можно извлечь из звука.

2.2 Извлечение признаков из аудио сигналов

На примере предыдущих двух аудиофайлов рассмотрим извлечение признаков.

1. Мел-частотные кепстральные коэффициенты (MFCC)

Mel-spectrogram представляет распределение энергии по частотам и учитывает особенности человеческого слуха. Для обнаружения дронов мел-спектрограмма помогает визуализировать энергетическое распределение звука в частотном диапазоне, где обычно работают двигатели дронов.

```
def plot_mfcc(file_path):
    # Загрузка аудиофайла
    audio, sr = librosa.load(file_path, sr=None)

    # Извлечение MFCC
    mfccs = librosa.feature.mfcc(y=audio, sr=sr, n_mfcc=13)

    # Визуализация MFCC
    plt.figure(figsize=(10, 4))
    librosa.display.specshow(mfccs, x_axis='time')
    plt.colorbar()
    plt.title(f'MFCCs для {file_path}')
    plt.tight_layout()
    plt.show()
```

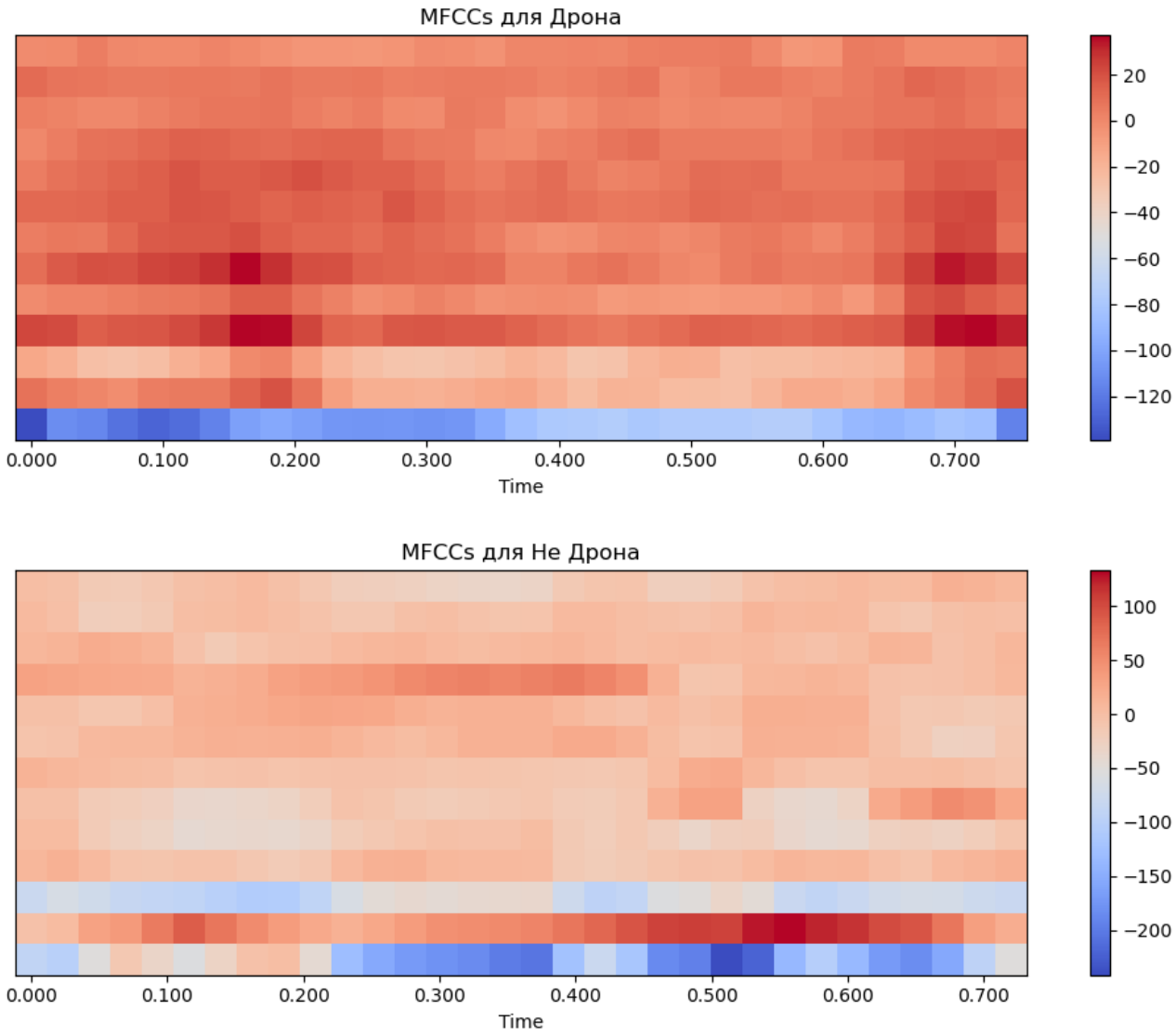


Рис. 2: Графики mfcc рассматриваемых аудиофайлов.

Как мы видим звук дрона более массивен, чем второй звук для наших ушей.

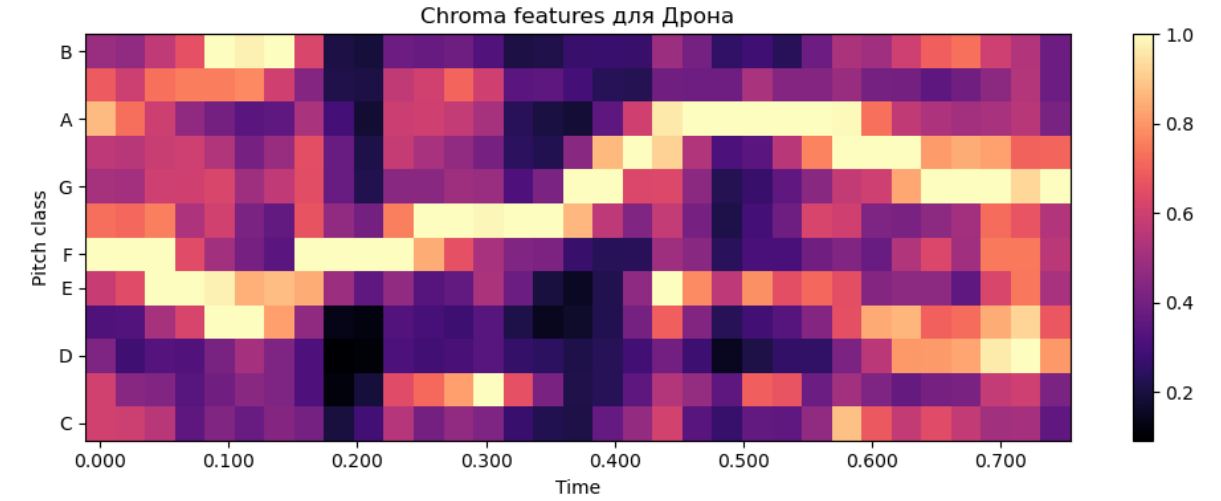
2. Хроматические признаки (Chroma features)

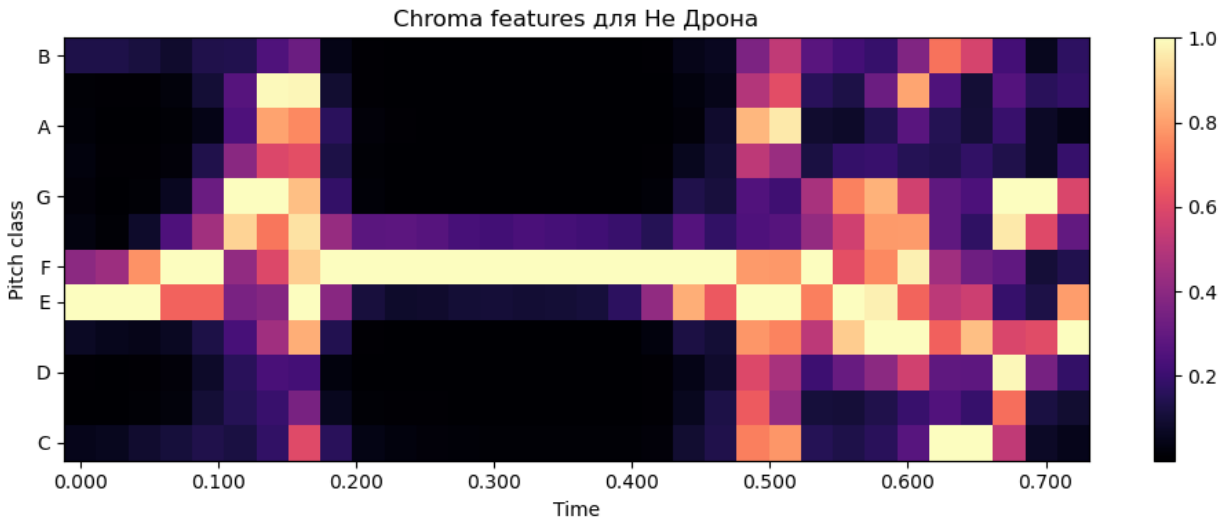
Chroma features (хроматические признаки) представляют собой 12 различий в частотах, соответствующих 12 полутонов в октаве. Эти признаки полезны для анализа гармонической структуры звука. В контексте обнаружения дронов, хроматические признаки могут помочь выявить характерные гармоники, создаваемые роторами дронов.Рис. 2: Графики mfcc рассматриваемых аудиофайлов.

```python
def plot_chroma(file_path):
    # Загрузка аудиофайла
    audio, sr = librosa.load(file_path, sr=None)

    # Извлечение хроматических признаков
    chroma = librosa.feature.chroma_stft(y=audio, sr=sr)

    # Визуализация хроматических признаков
    plt.figure(figsize=(10, 4))
    librosa.display.specshow(chroma, y_axis='chroma', x_axis='time')
    plt.colorbar()
    plt.title(f'Chroma features для {file_path}')
    plt.tight_layout()
    plt.show()
```





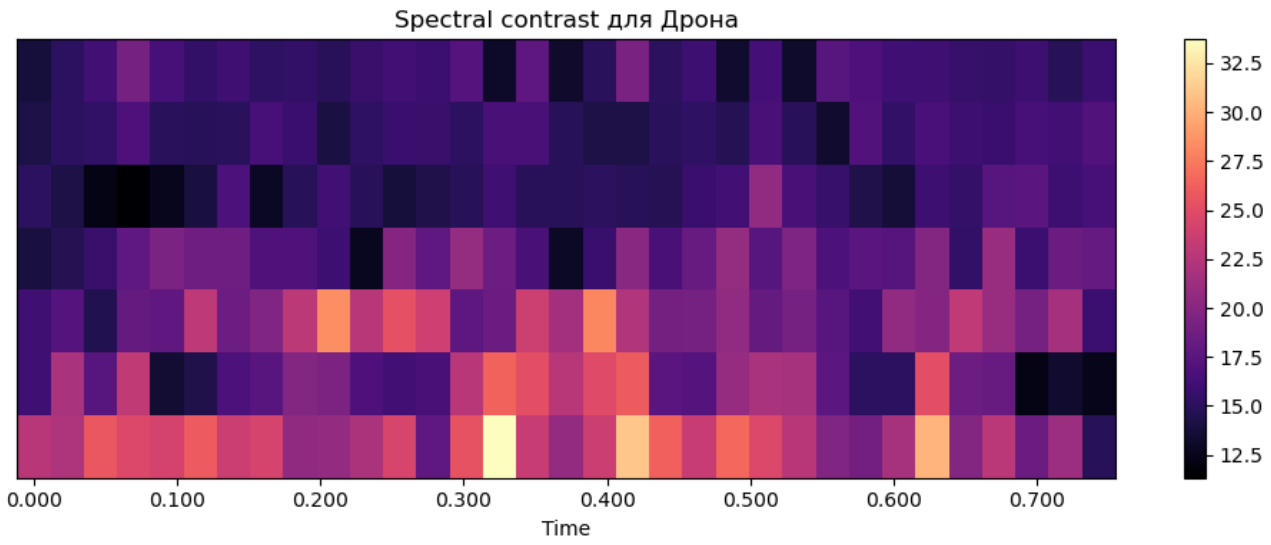
3. Спектральный контраст (Spectral Contrast)

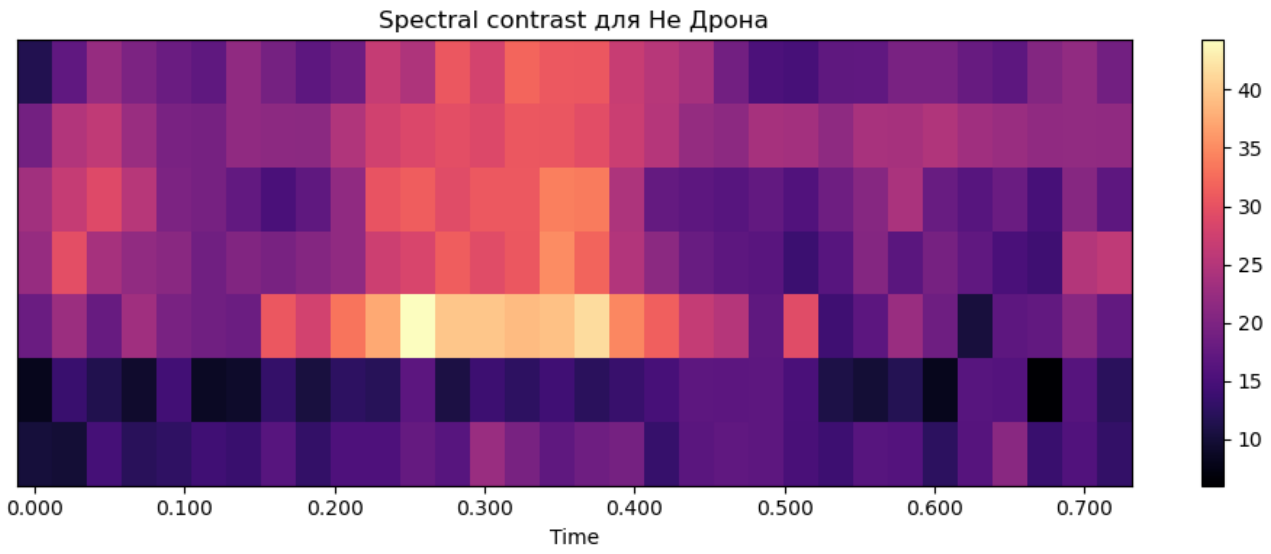
Spectral contrast измеряет разницу в амплитуде между пиками и впадинами в спектре. Этот признак может быть полезен для различения звуков дронов, так как у них характерные спектральные особенности, связанные с шумом пропеллеров и двигателей.

```python
def plot_spectral_contrast(file_path):
    # Загрузка аудиофайла
    audio, sr = librosa.load(file_path, sr=None)

    # Извлечение спектрального контраста
    spectral_contrast = librosa.feature.spectral_contrast(y=audio, sr=sr)

    # Визуализация спектрального контраста
    plt.figure(figsize=(10, 4))
    librosa.display.specshow(spectral_contrast, x_axis='time')
    plt.colorbar()
    plt.title(f'Spectral contrast для {file_path}')
    plt.tight_layout()
    plt.show()
```





Мы разобрали все признаки, благодаря которым научим нашу модель определять звук дрона.Теперь перейдем к следующему этапу.

3 Тренировка модели

3. 1 Обучение модели XGBoost

В качестве модели был использован XGBoost, а именно XGBRegressor. Обучим модель с помощью полученных ранее признаков. В данном случае, использовался XGBoost в виде классификатора (xgb.XGBClassifier()), который обучается на данных и выполняет классификацию на основе обученной модели.

Приступим к написанию самой модели:

- **Импорт нужных библиотек**

Мы используем **os** для работы с файловой системой, **librosa** для анализа аудиофайлов, **numpy** для работы с массивами, **train_test_split** и метрики из **sklearn** для подготовки и оценки данных, **xgboost** для построения моделей градиентного бустинга, и **RandomUnderSampler** и **RandomOverSampler** из **imblearn** для работы с несбалансированными данными.

```
In [1]: import os
        import librosa
        import numpy as np
        from sklearn.model_selection import train_test_split
        from sklearn.metrics import accuracy_score, classification_report
        import xgboost as xgb
        from imblearn.under_sampling import RandomUnderSampler
        from imblearn.over_sampling import RandomOverSampler
```

- **Сбор признаков**

Это функция **extract_features**, которая извлекает признаки из аудиофайлов. Она принимает путь к файлу, длительность окна и смещение. Внутри функции аудиофайл разбивается на сегменты, и из каждого сегмента извлекаются признаки: MFCC, chroma, mel-спектрограмма и контраст. Затем эти признаки объединяются и возвращаются.

```python
def extract_features(file_path, duration=1, offset=0.25):
    features = []
    audio, sr = librosa.load(file_path, sr=None, duration=duration, offset=offset)

    num_segments = int(np.ceil(len(audio) / (sr * duration)))

    for i in range(num_segments):
        start = int(sr * duration * i)
        end = int(min(len(audio), sr * duration * (i + 1)))

        segment = audio[start:end]

        mfccs = librosa.feature.mfcc(y=segment, sr=sr)
        chroma = librosa.feature.chroma_stft(y=segment, sr=sr)
        mel = librosa.feature.melspectrogram(y=segment, sr=sr)
        contrast = librosa.feature.spectral_contrast(y=segment, sr=sr, fmin=100.0, n_bands=6)

        features.append(np.mean(mfccs, axis=1))
        features.append(np.mean(chroma, axis=1))
        features.append(np.mean(mel, axis=1))
        features.append(np.mean(contrast, axis=1))

    if len(features) > 0:
        return np.concatenate(features)
    else:
        return None
```

Также не забываем установить директорию , в которой находятся аудиофайлы, а также определить массивы для хранения признаков и меток, и параметры длительности окна и смещения.

- **Установка меток (1 - дрон, 0 - не дрон) и начало сбора всех признаков.**

```
features = []
labels = []

duration = 1  # длина окна в секундах
offset = 0.5  # шаг прохода окна в секундах

for root, dirs, files in os.walk(data_dir):
    base_dir = os.path.basename(root)
    if base_dir in ["yes_drone", "bebop_1", "membo_1","Micro"]:
        label = 1  # Звук дрона
    elif base_dir == "unknown":
        label = 0


    for file in files:
        if file.endswith(".wav"):
            file_path = os.path.join(root, file)
            feature = extract_features(file_path, duration, offset)
            if feature is not None:
                features.append(feature)
                labels.append(label)


X = np.array(features)
```

Этот блок кода проходит по директориям, собирает аудиофайлы и извлекает из них признаки с помощью extract_features. После извлечения признаков они добавляются в массив features, а соответствующие метки - в массив labels.

- **Балансировка данных**

```
X = np.array(features)
y = np.array(labels)

# Разбиение на тренировочный и тестовый наборы данных
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=108)

# Создание объекта RandomUnderSampler для undersampling
under_sampler = RandomUnderSampler(sampling_strategy='majority', random_state=108)

# Применение undersampling к тренировочным данным
X_resampled_under, y_resampled_under = under_sampler.fit_resample(X_train, y_train)

# Создание объекта RandomOverSampler для oversampling
over_sampler = RandomOverSampler(sampling_strategy='minority', random_state=108)

# Применение oversampling к данным после undersampling
X_resampled, y_resampled = over_sampler.fit_resample(X_resampled_under, y_resampled_under)
```

Логическое объяснение использования **RandomUnderSampler** и **RandomOverSampler**

RandomUnderSampler используется для уменьшения дисбаланса классов в тренировочных данных путем случайного удаления примеров класса большинства. Это помогает избежать ситуации, когда модель учится только на данных класса большинства и игнорирует класс меньшинства.

RandomOverSampler используется для увеличения количества примеров класса меньшинства путем их дублирования. Это также помогает справиться с дисбалансом классов, но с другой стороны — увеличивая количество примеров класса, который реже встречается в данных.

Вместе эти методы помогают создать более сбалансированный набор данных для обучения модели, что может улучшить ее способность правильно классифицировать примеры всех классов, а не только те, которые изначально были в большинстве.

- **Обучение модели модели**

```
model = xgb.XGBClassifier()
model.fit(X_train, y_train)
```

Только что мы с вами обучили нашу модель, но работа пока не закончена. Прежде чем смотреть на то ,как работает модель на деле, мы должны оценить ее работу на тестовых данных. Все это мы увидим на следующей странице.

3.2 Средства проверки качества модели

Для оценивания моделей были использованы метрики из библиотеки *sklearn*.

Так как мы работаем с несбалансированным датасетом, то мы будем смотреть на оценки сначала эффективность модели с несбалансированным датасетом, а после уже с балансированным датасетом. Для этого мы будем использовать метрики:

1. Precision (Точность): Доля предсказанных положительных случаев, которые действительно являются положительными.

2. Recall (Полнота): Доля фактических положительных случаев, которые были правильно предсказаны.

3. F1-Score: Гармоническое среднее между точностью и полнотой.

4. Accuracy (Точность): Доля всех правильно предсказанных случаев среди всех предсказаний.

5. Macro avg (Среднее значение макро): Среднее арифметическое значений метрик для всех классов, без учета их пропорций.

6. Weighted avg (Среднее значение с весами): Среднее значение метрик для всех классов, взвешенное по количеству экземпляров в каждом классе.

Начнем с несбалансированного датасета

```
Точность модели на тестовых данных: 0.9989411266412537

Отчет о классификации на несбалансированных данных:
              precision    recall  f1-score   support

           0       1.00      1.00      1.00      4152
           1       1.00      0.99      1.00       570

    accuracy                           1.00      4722
   macro avg       1.00      1.00      1.00      4722
weighted avg       1.00      1.00      1.00      4722
```

Вывод:

Модель демонстрирует практически идеальные результаты на несбалансированных данных, что может указывать на её превосходную производительность, но также и на возможное переобучение на большинстве классов.

Перейдем к результатам после балансирования данных

```
Точность модели на тестовых данных сбалансированными данными: 0.9896230410842863

Отчет о классификации на сбалансированных данных:
              precision    recall  f1-score   support

           0       1.00      0.99      0.99      4152
           1       0.92      1.00      0.96       570

    accuracy                           0.99      4722
   macro avg       0.96      0.99      0.98      4722
weighted avg       0.99      0.99      0.99      4722
```

Вывод:

Модель показывает высокую точность на сбалансированных данных, что свидетельствует о её способности хорошо предсказывать как мажоритарный, так и миноритарный классы. Более реалистичные значения метрик, особенно для класса 1, указывают на улучшенную сбалансированность предсказаний.

Таким образом, мы создали модель классификации, которая продемонстрировала высокие показатели точности и сбалансированности. Модель успешно справляется с задачей классификации и демонстрирует отличные результаты на как несбалансированных, так и сбалансированных данных, обеспечивая высокую точность и надежность предсказаний.

Можем перейти к практической части моей работы.

4. Создание устройства по распознанию дрона в звуковом потоке

В ходе создания прототипа мы напишем небольшой проект. Суть его будь такова, что мы сделаем первую версию этого устройства из нашего компьютера. Для этого мы должны:

1. Написать код, который будет также работать в встроенным микрофоном.
2. Анализ звуков благодаря нам известным признакам.
3. Отображение присутствия дрона в звуковом поле.

Первым делом, мы подключим все нужные библиотеки. Также напишем небольшую функцию record_sound, которая и будет отвечать за прием всего звука и делить его на фрагменты размером 0.5 сек и передавать нашей модели, которую мы подключили заранее.

```python
import pyaudio
import numpy as np
import xgboost as xgb
import librosa
import signal
import soundfile as sf

model = xgb.Booster(model_file='/Users/midasxlr/Desktop/Drone_Classification/code and models/XGBoost')


1 usage
def record_sound(min_duration=0.5, sr=44100, frames_per_buffer=512):
    p = pyaudio.PyAudio()
    stream = p.open(format=pyaudio.paFloat32, channels=1, rate=sr, input=True, frames_per_buffer=frames_per_buffer)
    frames = []
    recorded_duration = 0

    try:
        while recorded_duration < min_duration:
            data = stream.read(frames_per_buffer)
            frames.append(np.frombuffer(data, dtype=np.float32))
            recorded_duration += frames_per_buffer / sr
    except OSError as e:
        print("Произошла ошибка записи звука:", e)

    stream.stop_stream()
    stream.close()
    p.terminate()
```

Затем мы напишем функцию, которая собирает признаки в полученных фрагментах. Она точно такая же, что мы использовали в обучении модели.

После этого нужно написать небольшую функцию detect_drone_sound
в 4 строки, задачей которой будет просто передавать полученный звук в модель и оценивать , что это за звук и полученный результат передавать в функцию, код которого ниже.

И последняя часть нашего кода- это вывод результата.

```
CHUNK_SIZE = 512
FORMAT = pyaudio.paInt16
CHANNELS = 1
RATE = 44100

p = pyaudio.PyAudio()
stream = p.open(format=FORMAT, channels=CHANNELS, rate=RATE, input=True, frames_per_buffer=CHUNK_SIZE)

while True:
    try:
        sound_data = record_sound(min_duration=0.5, sr=RATE, frames_per_buffer=CHUNK_SIZE)
        recorded_sounds.append(sound_data)
        sr = RATE
        is_drone, probability = detect_drone_sound(sound_data, sr)
        if is_drone:
            print(f"Обнаружен звук дрона! Вероятность: {probability:.2f}")
        else:
            print("Звука дрона не обнаружено.")
    except KeyboardInterrupt:
        print("\nПрерывание программы.")
        break

stream.stop_stream()
stream.close()
p.terminate()
```

Результат:

Микрофон слушает и определят звуки дрона. Я включаю программу и первые две секунды просто фоновой шум и ничего, потом включаю аудио в длиной 3 секунды не из датасета и он выводит «Обнаружен звук дрона!».

Таким образом, получилось создать прототип устройства, который может стать началом для хорошей идеи, а в дальнейшем полезным проектом.

5. Заключение

В рамках данной курсовой работы были проведены экспериментальные исследования характеристик случайной величины x, представляющей собой набор признаков, извлекаемых из звукового потока. Также был разработан код для их извлечения.

Кроме того, в ходе работы были изучены методы фильтрации звука, а также проведен балансирование датасета. После балансировки датасета были сравнены метрики модели с несбалансированным датасетом, что позволило оценить влияние балансировки на качество результатов. Был разработан код для тренировки и тестирования модели, а также код для обнаружения дронов в реальном времени.

Код и результаты экспериментов доступны по адресу:

https://github.com/Ramzzz10/Drone_Classification

Полученную модель машинного обучения, а также прототип устройства можно считать основным результатами работы.

Список литературы

[1] Sara A Al-Emadi, Abdulla K Al-Ali, Abdulaziz Al-Ali, Amr Mohamed. "Audio Based Drone Detection and Identification using Deep Learning». IWCMC 2019 Vehicular Symposium (IWCMC-VehicularCom 2019), Tangier, Morocco, 2019.
url: https://saraalemadi.com/2019/01/12/drone-audio-dataset/

[2] "XGBoost". GeeksforGeeks, 2023.
url: https://www.geeksforgeeks.org/xgboostysclid=lwdkta7n80518721819

[3] Oleg Sedukhin. "Practical application of XGBoost on the example of a binary classification problem." 2022.
url: https://habr.com/ru/users/boygenius/