Определение эмоций по голосу.

Данные:

1. набор данных TESS, отражает каждую из семи эмоций (гнев, отвращение, страх, счастье, приятный сюрприз, грусть, нейтральный). Женская озвучка. https://www.kaggle.com/ejlok1/toronto-emotional-speech-set-tess
2. набор данных SAVEE, отражает каждую из семи эмоций (гнев, отвращение, страх, счастье, приятный сюрприз, грусть, нейтральный). Мужская озвучка.

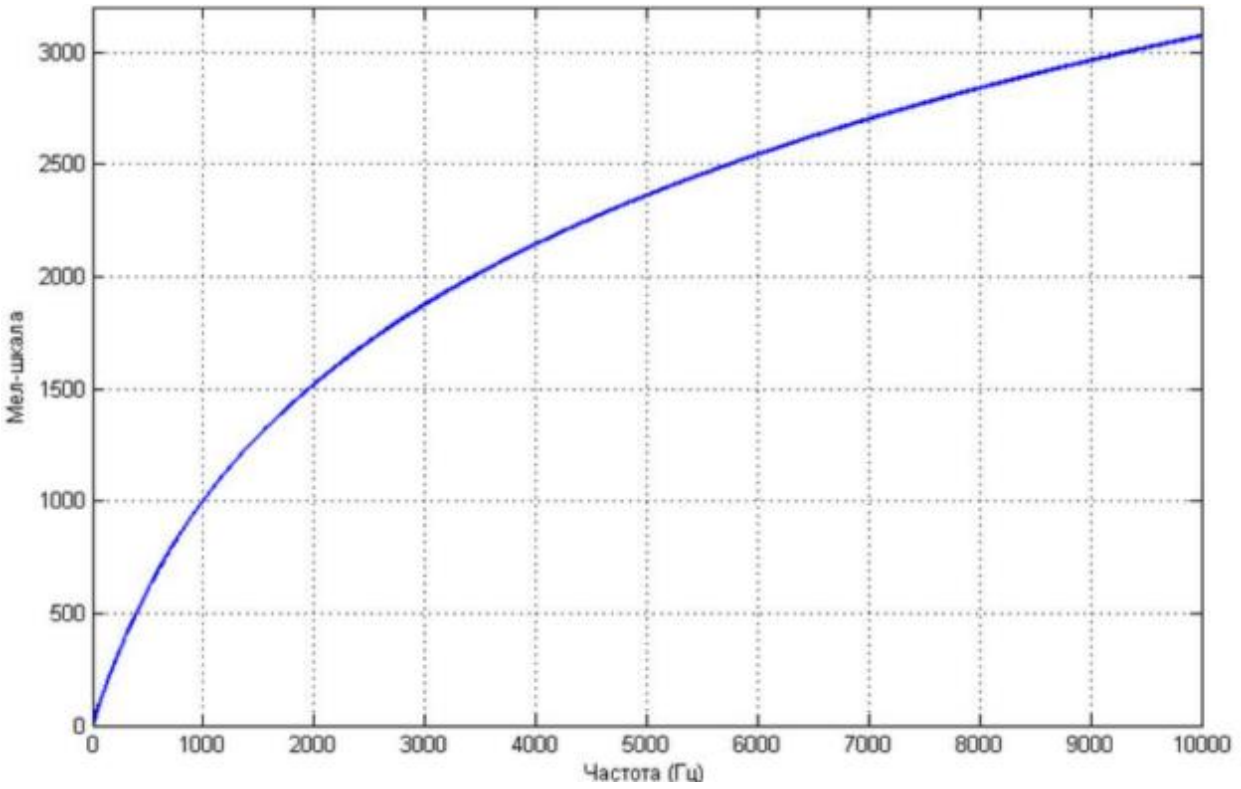
Цель: разработать алгоритм, позволяющий диагностировать эмоции обучаемых по их голосу.

Задания.

Задание 1. Преобразовать аудиофайлы в мел-кепстральные коэффициенты.

Справка:

Что же такое мел-кепстральные коэффициенты? Очевидно, что это понятие состоит из трех слов. Что ж, давайте пойдем по порядку и будем объяснять значение каждого, для появления смысла у всего выражения. Первое слово в этом не простом определении – «Мел». Согласно определению из словаря, это единица высоты звука, определяемая человеческим ухом. Высота звука не совсем линейно зависит от его частоты, если посмотреть на график, он больше напоминает логарифмическую зависимость:



Данные единицы измерения очень часто применяются в распознавании речи, потому что они наиболее похожи на человеческие инструменты восприятия голоса. А, как я уже говорил ранее, человека еще никто не переплюнул в вопросах распознавания речи. Высота тона является одной из характеристик речевого сигнала и измеряется как частота сигнала. Шкала Мел - это шкала, которая связывает воспринимаемую частоту тона с фактической измеренной частотой. Он

масштабирует частоту, чтобы более точно соответствовать тому, что слышит человеческое ухо (люди лучше распознают небольшие изменения в речи на более низких частотах). Эта шкала была получена из серии экспериментов на людях. Позвольте мне дать вам интуитивное объяснение того, именно показывает мел-шкала.

Для очень простого понимания кепстр - это информация о скорости изменения спектральных полос. При обычном анализе сигналов времени любой периодический компонент (например, эхо-сигналы) обнаруживается в виде резких пиков в соответствующем частотном спектре (то есть спектре Фурье. Это получается путем применения преобразования Фурье к сигналу времени). Это можно увидеть на следующем изображении.

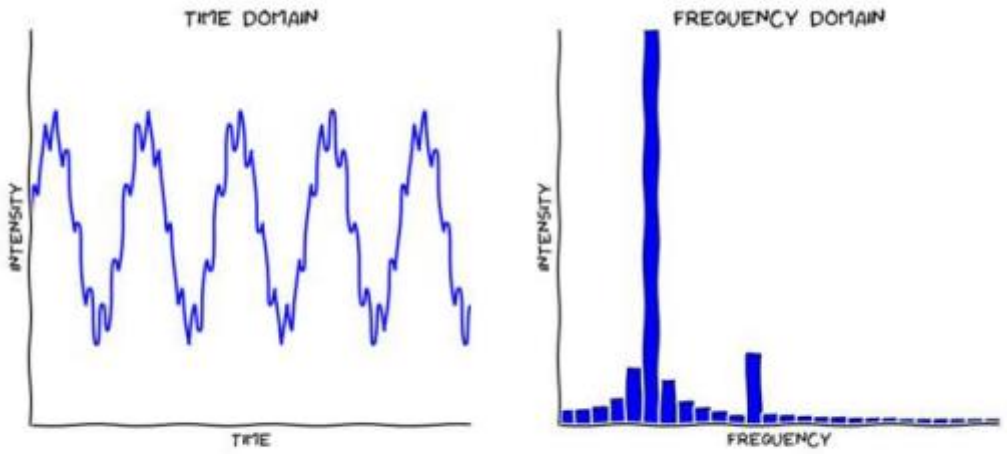


Рисунок 3. Преобразование сигнала времени в спектр Фурье.

Взяв логарифм величины этого спектра Фурье, а затем снова взяв спектр этого логарифма с помощью косинусного преобразования (я знаю, это звучит сложно, но потерпите меня, пожалуйста!). Мы наблюдаем пик там, где есть периодический элемент в исходном сигнале времени. Поскольку мы применяем преобразование к самому частотному спектру, результирующий спектр не находится ни в частотной области, ни во временной области. И этот спектр логарифма спектра временного сигнала получил название кепстр (та-да!). Кстати, интересное замечание, что кепстр был впервые введен для характеристики сейсмического эха, возникающего в результате землетрясений. Для лучшего понимания процесса, давайте разберем весь процесс «добычи» мел-кепстральных коэффициентов на примере. В качестве «подопытного» возьмем маленькое, простое слово «ОДИН», состоящее из двух слогов. Вот его временное представление:

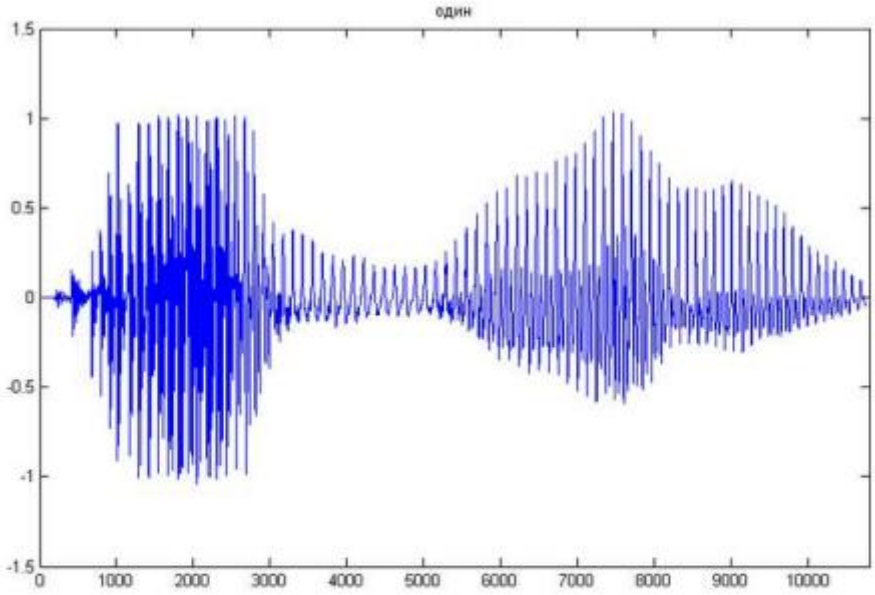


Рисунок 4. Временное представление слова "один".

Первым шагом наших действий будет преобразование Фурье, применим его, для получения спектра аудио-сигнала:

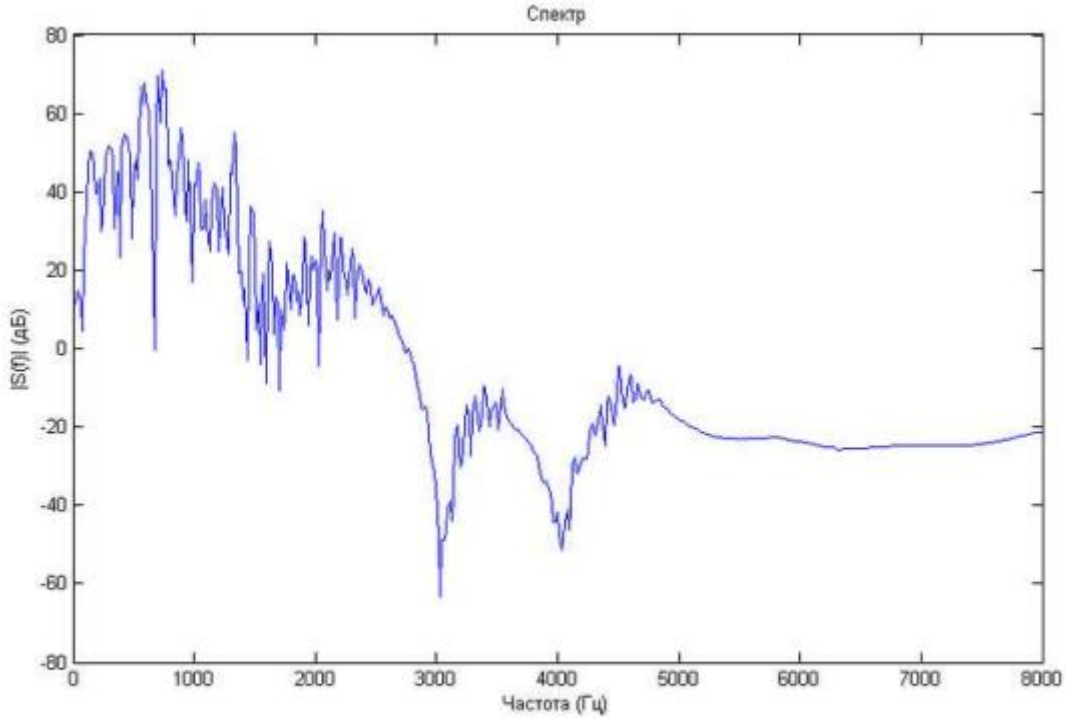


Рисунок 5. Спектр слова "один" после преобразования Фурье.

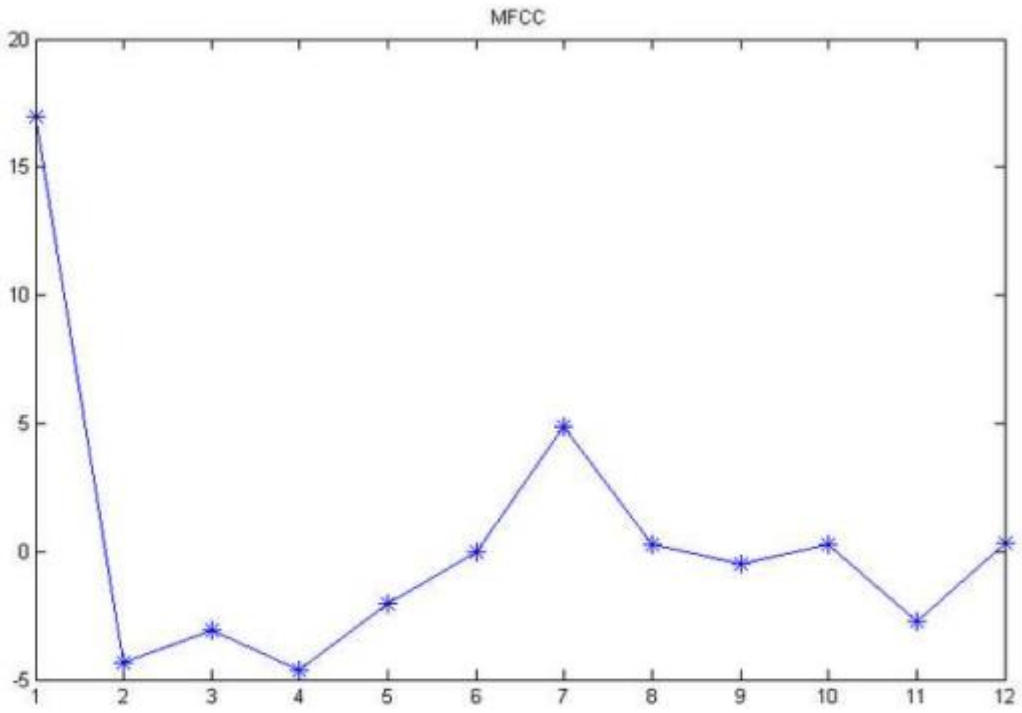


Рисунок 8. Кепстральные коэффициенты слова "один".

Как прекрасно, что мне не придется проделывать все эти математические преобразования вручную и я могу просто использовать встроенные инструменты python, для добычи этих самых мел-кепстральных коэффициентов.

Библиотека librosa – для преобразования аудио-сигнала.

На следующем рисунке представлен код преобразования

```
path = './drive/My Drive/TESS'
start = time.time()

data = []

for subdir, dirs, files in os.walk(path):

    for file in files:
        print(file)
        target = str(file.split('_')[2])[:-4]

        y, sr = lr.load(os.path.join(subdir, file), res_type='kaiser_fast')
        mfccs = np.mean(lr.feature.mfcc(y=y, sr=sr, n_mfcc=30).T, axis=0)
        sample = mfccs, target
        data.append(sample)

end = time.time()
print(f'Writing ended in {end - start} seconds')
```

```
YAF_limb_angry.wav
YAF_loaf_angry.wav
```

После выполнения данного кода у нас будет двумерный список «data», каждый элемент которого будет содержать два объекта: набор mfcc и название эмоции, которую этот набор описывает. Преобразуем список в следующий датафрейм:

```
X, y = zip(*data)
dataset_TESS = pd.DataFrame(X,(y))
dataset_TESS.rename(columns={
    'index': 'target'
}, inplace=True)
dataset_TESS
```

	target	0	1	2	3	4	5	6	7	
0	angry	-288.719238	62.625057	-26.012300	23.225218	-28.039703	1.384188	-5.769329	-26.590179	-8.33716
1	angry	-336.208311	36.680333	-15.951548	16.189888	-38.497388	2.420249	-6.803961	-17.491220	-10.96015
2	angry	-290.566195	40.682354	-2.650168	20.170969	-22.214548	0.604162	-4.247833	-12.081742	-13.03240
3	angry	-322.739539	28.126587	-22.118054	31.071080	-32.006755	-5.765176	-5.862902	-15.413687	-15.99879
4	angry	-298.179893	26.212140	-17.609021	17.530507	-41.873195	1.158211	-11.125472	-14.612097	-13.10275
...	...	...	...	...	...	...	...	...	...	...
2795	sad	-414.832950	61.244351	24.871053	47.345543	-4.288396	13.609492	-9.357812	-4.297829	-3.56024
2796	sad	-428.959335	87.337324	22.923531	45.436908	1.115740	13.786090	-7.202889	-3.144844	-4.43602
2797	sad	-401.938887	73.464665	13.728966	38.863063	-9.918325	10.162456	-12.362507	-3.936528	-8.78457
2798	sad	-409.457717	96.403305	18.716205	38.404808	-8.669222	9.568859	-11.840629	-9.701875	-3.40942

Наши данные почти готовы, но есть одна проблема. Компьютер не понимает, что такое “sad”, “happy”, “angry” и т. д. поэтому нам необходимо каждой эмоции присвоить ее номер.

Для этого:

1. Создаем словарь, где ключами были – числовые значения, а словами – наши эмоции из столбца target.

```
[ ]  emotions = {
          '00': 'neutral',
          '01': 'happy',
          '02': 'sad',
          '03': 'angry',
          '04': 'fear',
          '05': 'disgust',
          '06': 'ps'
      }
```

2. Поменяем в словаре слова и ключи местами, чтобы по названию эмоции мне выдавало ее числовое значение. И прошелся по уже готовому двумерному списку “data” меняя каждое название эмоции на ее числовое значение, затем снова преобразуем в новый датасет:

```
[ ]  X, y = zip(*data)
     Y = []
     emotions_inv = {value: key for key, value in emotions.items()}
     for i in range(len(y)):
       Y.append(int(emotions_inv[y[i]]))
```

```
dataset = pd.DataFrame(X,(Y))

dataset.rename(columns={
    'index': 'target'
}, inplace=True)
dataset
```

	target	0	1	2	3	4	5	6	7	
0	3	-288.719238	62.625057	-26.012300	23.225218	-28.039703	1.384188	-5.769329	-26.590179	-8.3
1	3	-336.208311	36.680333	-15.951548	16.189888	-38.497388	2.420249	-6.803961	-17.491220	-10.9
2	3	-290.566195	40.682354	-2.650168	20.170969	-22.214548	0.604162	-4.247833	-12.081742	-13.0
3	3	-322.739539	28.126587	-22.118054	31.071080	-32.006755	-5.765176	-5.862902	-15.413687	-15.9
4	3	-298.179893	26.212140	-17.609021	17.530507	-41.873195	1.158211	-11.125472	-14.612097	-13.1

Повторяем эти аналогичные операции для датасета SAVEE

Если у вас возникли ошибки, то можете воспользоваться готовыми датасетами - https://drive.google.com/drive/folders/1WQflU_1ZYsO4EuCJx9SRB5lUGKDtgEOL?usp=sharing

Задание 2. Используя стандартные алгоритмы машинного обучения решите задачу классификации для двух датасетов (случайный лес, логистическая регрессия).

Сравните результаты с приеденными на рисунке

```
print(classification_report (y_test, y_pred))
```

	precision	recall	f1-score	support
0	0.71	0.77	0.74	44
1	0.50	0.67	0.57	12
2	0.41	0.27	0.33	26
3	0.38	0.38	0.38	13
4	0.35	0.26	0.30	23
5	0.25	0.38	0.30	13
6	0.31	0.31	0.31	13
accuracy			0.48	144
macro avg	0.42	0.44	0.42	144
weighted avg	0.47	0.48	0.47	144

Какая из эмоций определяется хуже всего?

Задание 3. Решите задачи классификации используя deep learning.

Качество обучения сравните с результатами на рисунке ниже

```
Epoch 344/350
384/384 [==============================] - 0s 431us/step - loss: 0.1731 - accuracy: 0.9401 - mae:
Epoch 345/350
384/384 [==============================] - 0s 427us/step - loss: 0.1830 - accuracy: 0.9219 - mae:
Epoch 346/350
384/384 [==============================] - 0s 421us/step - loss: 0.1810 - accuracy: 0.9427 - mae:
Epoch 347/350
384/384 [==============================] - 0s 424us/step - loss: 0.1599 - accuracy: 0.9323 - mae:
Epoch 348/350
384/384 [==============================] - 0s 409us/step - loss: 0.1864 - accuracy: 0.9245 - mae:
Epoch 349/350
384/384 [==============================] - 0s 428us/step - loss: 0.1845 - accuracy: 0.9375 - mae:
Epoch 350/350
384/384 [==============================] - 0s 448us/step - loss: 0.2475 - accuracy: 0.9141 - mae:
```