Санкт–Петербургский государственный университет

Козлова Анна Олеговна

Магистерская диссертация

Объяснимый искусственный интеллект для распознавания речи

Уровень образования: магистратура
Направление 01.04.02 «Прикладная математика и информатика»
Основная образовательная программа ВМ.5504.*
«Исследование операций и системный анализ»

Научный руководитель:
доцент, кафедра математического моделирования энергетических систем, канд. физ.-мат. наук Петросян Ованес Леонович

Рецензент:
главный специалист - аналитик отдела страховой статистики, АО Совкомбанк страхование, Бойко Алина Владимировна

Санкт-Петербург
2021 г.

Содержание

Введение

В настоящее время искусственный интеллект все чаще применяется для извлечения пользы из различных задач машинного обучения. В последние годы такие системы стокнулись с проблемой потери "прозрачности"и понятности, в особенности для конечных пользователей. В этой работе будут исследованы методы обьяснения искусственного интеллекта на основе обученной модели для распознавания речи.

XAI или обьяснимый искусственный интеллект, это область ИИ, которая имеет набор интрументов, алгоритмов и методов, которые могут генерировать интуитивно понятные и интерпретируемые обьяснения для человека.

Глубокие нейронные сети используются в системах которые напрямую влияют на качество жизни человека. Примерами тому являются здравоохранение и беспилотный транпорт. Закрытые системы принятия решений, называются черными ящиками, и на данный момент не пользуются доверием у пользователей.[1]

Интерпретируемость глубоких нейронных сетей, область машинного обучения которая появилась совсем недавно. Результат ее работы нацелен на лучшее понимание того как модели проводят отбор признаков, а также получают решения своих задач[2].

Ежегодно публикуется неколько статей-обзоров, с посследними достижениями в этой области, подробробнее остановимся на "Opportunities and Challenges in Explainable Artificial Intelligence (XAI): A Survey"[1]. Ключевыми этапами развития для таких популярных алгоритмов, как LIME и SHAP обозначены усовершенствания классических моделей и применение существующих методов, в новых, неисследованных областях. Примерами новых интерпретаций алгоритмов, могут служить: SLIME, QLIME, KernelSHAP и др.

Основная сфера иссследования и примения агоритмов обьяснения моделей машинного обучения, это медицина. Алгоритмы отлично себя проявили во многих задачах от работы с изображениями до обработки естественного языка.[3] В этом исследовании, объединены последние тенденции

XAI и применение его для распознавания эмоций, которые в свою очередь могут использоваться в медицине для помощи в постановке диагноза для людей страдающих психическими расстройствами.

Для проведения исследования используется два набора данных. Первый включает в себя аудиозаписи актеров, разных полов: CREMA-D[4], RAVDES[5], SAVEE[6], TESS[7]. Второй DAIC-WOZ Database Description содержит клинические интервью, предназначенные для поддержки диагностики психологических дистресс-состояний, таких как тревога, депрессия и посттравматическое стрессовое расстройство[8].

Свидетельством того, что это направление набирает обороты, можно связать с запуском первой глобальной конференции, посвященной исключительно этой развивающейся дисциплине, Международной совместной конференции по искусственному интеллекту: семинар по объяснимому искусственному интеллекту (XAI)[9].

Европейский Союз ввел право на объяснение в Общем праве на защиту данных (GDPR) как попытку справиться с потенциальными проблемами, вытекающими из растущей важности алгоритмов. В Соединенных Штатах страховые компании должны быть в состоянии объяснить свои решения о ставках.[1] Другим примером является, использование XAI в колл-центре для классификации звонков в соответствии с эмоциями, в последствиии это применяется в качестве параметра производительности для анализа разговоров, таким образом идентифицируя неудовлетворенного клиента, удовлетворенность клиентов и т. д. Примение XAI способно оказать помощь компаниям в улучшении их услуг. Он также может быть использован в бортовой системе автомобиля, основанной на информации о психическом состоянии водителя, которая может быть предоставлена системе для инициирования его/ее безопасности, предотвращающей несчастные случаи.[2]

Постановка задачи

Исследование можно разделить на две части:
1) решение задачи классификации распознавания речевых эмоций;
2) использование алгоритма XAI для объяснения результатов.

В первой части используются наборы аудиовизуальных данных: Crema-D, Ravdess, Savee, Tess. Строим волновые диаграммы и спектограммы, извлекаем признаки и строим модель нейронной сети.

Вторая часть включает обьяснение результатов алгоримов XAI и приведение результатов.

Обзор литературы

Работа началась со знакомства с популярным предшественником объяснимого ИИ, рекомендательными системами. Популярные в 70-80 гг. прошлого века, они включали в себя механизм вывода и базу знаний. Область медицинской диагностики, ощутила на себе в числе первых, где они были опробованы. Ранней версией экспертной системы была MYCIN, разработанная в 70х в Стэнфордском университете. MYCIN разрабатывалась для работы с бактериями, а именно диагностировались те, что вызывают тяжелые болезни, например, менингит. В том числе изучалось количество необходимых антибиотиков, для благоприятного исхода болезни. Реализация была простой и включала в себя ответы «Да» или «Нет» на вопросы медиков, в конце выдавалась рекомендация по дальнейшим действиям. Система была описана в работе "Rule-based Expert System – The MYCIN Experiments of the Stanford Heuristic Programming Project". [10] Одним из критериев новых систем подразумевалась понятность решений для профессионалов своей области, а не только для IT специалистов.

Например, применение интерпретируемости в медицинских исследованиях, улучшит работу многих специалистов:

1) практикующие врачи смогут впоследствии использовать эти методы для рекомендаций в постановке диагнозов;

2) понимание решений ИИ, впоследствии обернется большим количеством идей для реализации в медицинской практике.[11]

Открытыми остаются вопросы: «Кто несет ответственность за неправильный прогноз и лечение?». Тот же вопрос задает автор работы "“Why Should I Trust You?” Explaining the Predictions of Any Classifier"[12]. В качестве пути решения, автор предлагает побороть недоверие к модели или прогнозу. В роли инструментов для объяснения результатов, алгоритмы XAI, дающие понятную интерпретацию работы модели.

Для анализа полученных результатов будут использованы алгоритмы SHAP и LIME. На их основе в обученной модели распознавания речи будут определены признаки объясняющие тот или иной результат. Методология работы алгоритмов была описана в работе «Opportunities and Challenges in

Explainable Artificial Intelligence (XAI): A Survey»[1], в ней же представлены перспективы развития отрасли. На основе проделанной работы своих коллег, автор заключает, что будущее за расширением классических методов и применением их в новых отраслях.

Важность этой сферы можно оценить по нарастающему интересу к ней ученого сообщества, которое организует конференции посвященные XAI.[9]

Раскрываемая тема, касается сложной природы объяснения результатов обучения нейронной сети. Поэтому регулированием использования ИИ в различных сферах обеспокоились мировые технологические лидеры: страны ЕС и США. Таким образом они вводят законы включающие «право на объяснение».[1]. Эти меры позволяют контролировать справедливое использование алгоритмов. Подобные решения улучшают не только качество жизни людей, но и обеспечивают их безопасность.

Глава 1. Распознавание эмоций в аудиоданных с использованием глубоких нейронных сетей

1.1 Предварительная обработка

Распознавание эмоций играет важную роль в межличностной коммуникации, но даже среди людей понимание человеческих эмоций сильно отличается. Цель этой главы распознать эмоции в аудио. Для работы взят набор данных состоящий из:

CREMA-D - это набор данных из 7442 оригинальных клипов от 91 актера. Эти клипы были сняты 48 мужчинами и 43 женщинами в возрасте от 20 до 74 лет, представляющими различные расы и этнические группы (афроамериканцы, азиаты, кавказцы, латиноамериканцы и неуказанные). Актеры говорили из подборки из 12 предложений. Предложения были представлены с использованием одной из шести различных эмоций (гнев, отвращение, страх, счастье, нейтральность и печаль) и четырех различных уровней эмоций (низкий, средний, высокий и неопределенный).[4]

RAVDESS - эта часть содержит 1440 файлов: 60 испытаний на одного актера x 24 актера = 1440, включая 24 профессиональных актера (12 женщин, 12 мужчин), озвучивающих два лексически подобранных высказывания с нейтральным североамериканским акцентом. Речевые эмоции включают выражения спокойствия, радости, печали, гнева, страха, удивления и отвращения. Каждое выражение производится на двух уровнях эмоциональной интенсивности (нормальный, сильный), с дополнительным нейтральным выражением.[5]

SAVEE - база данных была получена от четырех мужчин-носителей английского языка (идентифицированных как DC, JE, JK, KL), аспирантов и исследователей Университета Суррея в возрасте от 27 до 31 года. Эмоции были описаны психологически в отдельных категориях: гнев, отвращение, страх, счастье, печаль и удивление. Также добавлена нейтральная категория для записи седьмой категории эмоций.Текстовый материал состоял из 15 предложений TIMIT на эмоцию: 3 общих, 2 специфичных для эмоций и 10 общих предложений, которые были разными для каждой эмоции

и фонетически сбалансированными. 3 общих и $2 \times 6 = 12$ предложений, специфичных для эмоций, были записаны как нейтральные, чтобы дать 30 нейтральных предложений. В результате на одного оратора пришлось в общей сложности 120 высказываний.[6]

TESS - набор из 200 целевых слов, произнесенных в несущей фразе "Скажи слово_"двумя актрисами (в возрасте 26 и 64 лет), и были сделаны записи набора, изображающие каждую из семи эмоций (гнев, отвращение, страх, счастье, приятное удивление, печаль и нейтральность). Всего существует 2800 точек данных (аудиофайлов).

Набор данных организован таким образом, что каждая из двух женщин-актеров и их эмоции содержатся в отдельной папке. И в нем можно найти все 200 целевых слов аудиофайла. Формат аудиофайла WAV.[7]

В первую очередь визуализируем данные в виде диаграмм и спектограмм. Преобразованный файл дает результат в виде эмоции которую испытывает актер когда говорит это предложение. Модели предсказывают только одну из пяти эмоций: «злость», «отвращение», «страх», «счастье», «нейтральный тон».

Подход распознавания эмоций с помощью спектограмм, заключается в представлении изображения звукового сигнала, которое состоит из трех компонентов:

1) Время по оси абсцисс.

2) Частота по оси ординат.

3) Интенсивность мощности.

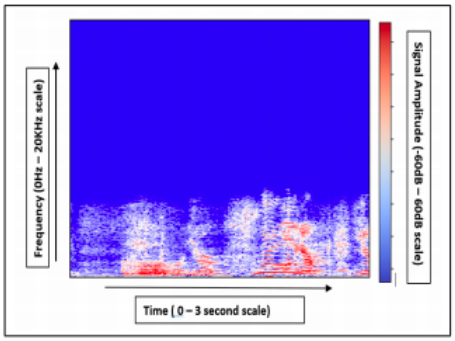

Рис. 1: Пример аудиоспектрограммы эмоции гнева. Оригинальная шкала времени без очистки от шума.

1.2 Выделение признаков

Корпус данных содержит аудиофайлы wav с различной продолжительностью времени и с маркировкой фактической метки эмоций для соответствующего временного сегмента. Звуковая спектрограмма извлекается из wav-файла с помощью пакета librosa.[13] Архитектура нейронной сети - CNN (сверточная нейронная сеть). Сеть состоит из четырех 1D сверточных слоев и maxpooling слоев за которыми следует три полносвязных слоя.

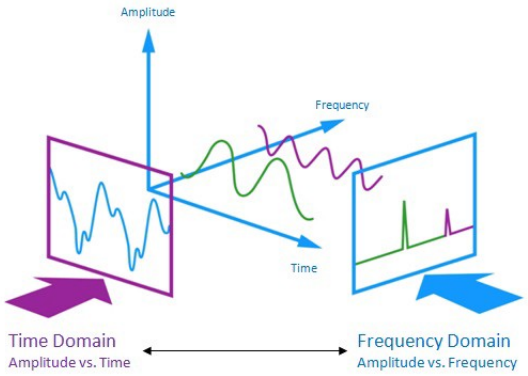

Рис. 2: Аудиосигнал представляет собой трехмерный сигнал, в котором три оси представляют время, амплитуду и частоту.

Перед тем как передать файлы в модель, необходимо выделить признаки. Для этого были использованы следующие функции:

1) Zero-crossing rate.[14]
2) Chroma_stft.[15]
3) Mel-frequency cepstrum.[16]
4) Root mean square[17]
5) MelSpectogram.[18]

Рассмотрим каждый из пунктов.Скорость пересечения нуля или Zero-crossing rate, это способ измерения гладкости сигнала, вычисление числа пересечений нуля в пределах сегмента этого сигнала. Голосовой сигнал колеблется медленно. Например, сигнал 100 Гц будет пересекать ноль 100 раз в секунду, тогда как “немой” фрикативный сигнал может иметь 3000 пересечений нуля в секунду.

Более высокие значения наблюдаются в таких высоко ударных звуках, как в металле и роке. Теперь визуализируем этот процесс и рассмотрим

вычисление скорости пересечения нуля.

```
def extract_features(data):
    # ZCR
    result = np.array([])
    zcr = np.mean(librosa.feature.zero_crossing_rate(y=data).T, axis=0)
    result=np.hstack((result, zcr)) # stacking horizontally
```

Рис. 3: Вычисление Zero-crossing rate в librosa

Chroma_stft (признак или вектор цветности) обычно представлен вектором признаков из 12 элементов, в котором указано количество энергии каждого высотного класса C, C#, D, D#, E, . . . , B в сигнале. Используется для описания меры сходства между музыкальными произведениями.

```
# Chroma_stft
stft = np.abs(librosa.stft(data))
chroma_stft = np.mean(librosa.feature.chroma_stft(S=stft, sr=sample_rate).T, axis=0)
result = np.hstack((result, chroma_stft)) # stacking horizontally
```

Рис. 4: Вычисление Chroma_stft rate в librosa

Mel-frequency cepstrum или Мел-частотные кепстральные коэффициенты (MFCC). Представляют собой небольшой набор признаков (обычно около 10–20), которые кратко описывают общую форму спектральной огибающей. Они моделируют характеристики человеческого голоса.

```
# MFCC
mfcc = np.mean(librosa.feature.mfcc(y=data, sr=sample_rate).T, axis=0)
result = np.hstack((result, mfcc)) # stacking horizontally
```

Рис. 5: Вычисление Mel-frequency cepstrum в librosa

Root mean square (среднее квадратичное значение). RMS количественно определяет средний уровень сигнала, более тесно коррелирует с нашим слухом.

MelSpectogram, это спектрограмма, где частота выражена не в Гц, а в мелах. Итогом преобразования служит изменение высоких частот из более широкого диапазона, их значение усредняется, низкие частоты остаются почти без изменений.

```
# Root Mean Square Value
rms = np.mean(librosa.feature.rms(y=data).T, axis=0)
result = np.hstack((result, rms)) # stacking horizontally
```

Рис. 6: Вычисление Root mean square в librosa

```
# Root Mean Square Value
rms = np.mean(librosa.feature.rms(y=data).T, axis=0)
result = np.hstack((result, rms)) # stacking horizontally
```

Рис. 7: Вычисление MelSpectogram в librosa

На этом предобработка аудиофайлов завершается. В следующей главе будет рассмотрено ее применение в сверточной нейронной сети.

Глава 2. Нейронная сеть

2.1 Сверточная нейронная сеть

Сверточные нейронные сети (CNN) состоят из нейронов с обучаемыми весами и сдвигами. Каждый нейрон получает на вход данные, далее выполняется скалярное произведение и при необходимости добавляет нелинейную оптимизацию. Вся сеть выражает одну дифференцируемую функцию оценки. Также есть функция потерь, включенная на последнем слое, например Softmax.

В качестве оптимизационного алгоритма выбран Adamax. С помощью него вычисляется инерционный момент распределения градиентов произвольной степени p.

$$u_t = \beta_2^p v_{t-1} + (1 - \beta_2^p)|g_t|^p \tag{1}$$

Чтобы использовать значение из формулы (1), требуется извлечь корень $u_t = v_t^{\frac{1}{p}}$, выводим решающее правило, при $p \to \infty$ и развернув под корнем u_t при помощи формулы (1):

$$u_t = \lim_{p\to\infty}[\beta_2^p v_{t-1} + (1 - \beta_2^p)|g_t|^p]^{\frac{1}{p}}$$

$$= \lim_{p\to\infty}[(1 - \beta_2^p)\sum_{i=1}^{t}\beta_2^{p(t-i)}|g_i|^p]^{\frac{1}{p}}$$

$$= \lim_{p\to\infty}(1 - \beta_2^p)^{\frac{1}{p}}(\sum_{i=1}^{t}\beta_2^{p(t-i)}|g_i|^p)^{\frac{1}{p}}$$

$$= \lim_{p\to\infty}(\sum_{i=1}^{t}\beta_2^{p(t-i)}|g_i|^p)^{\frac{1}{p}}$$

$$= max(\beta_2^{t-1}|g_1|, \beta_2^{t-2}|g_2|, ..., \beta_2|g_{t-1}|, |g_t|) \tag{2}$$

Помещаем u_t в уравнение обновления:

$$\theta_{t+1} = \theta_t - \frac{\eta}{u_t}\hat{m}_t \tag{3}$$

Хорошие значения по умолчанию: $\eta = 0{,}002$, $\beta_1 = 0{,}9$ и $\beta_2 = 0{,}999$.

Переходим к постороению нейронной сети. Архитектура сверточной сети принимает на вход данные в качестве одномерного массива. Релизация нашей сети будет проходить в Google Colab.

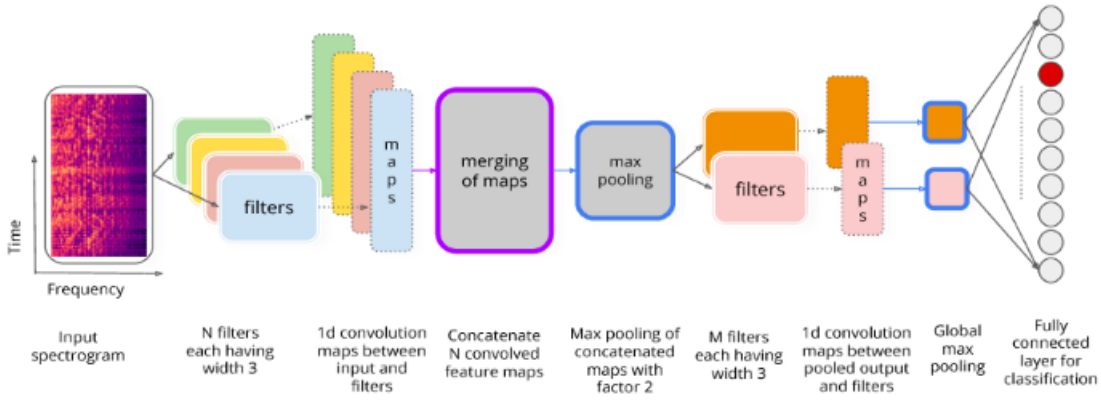

Рис. 8: Схема архитектуры CNN

Для дальнейшей работы будут использованы библиотеки sklearn и Keras. С помощью train_test_split библиотеки sklearn 90% данных будет использоваться для обучения, 10% для теститрования.

Также перед обучением проводим увеличение данных. Увеличение данных-это процесс, с помощью которого мы создаем новые синтетические выборки данных, добавляя небольшие возмущения в начальный обучающий набор. Чтобы генерировать синтаксические данные для аудио, мы можем применять инъекцию шума, время сдвига, изменение высоты тона и скорости. Цель состоит в том, чтобы сделать нашу модель инвариантной к этим возмущениям и повысить ее способность к обобщению. Для того, чтобы это работало, добавление возмущений должно сохранять ту же метку, что и исходный обучающий образец. Для увеличения данных используется:
1)Noise Injection.
2)Stretching.
3)Shifting.

4)Pitch.

```
def noise(data):
    noise_amp = 0.035*np.random.uniform()*np.amax(data)
    data = data + noise_amp*np.random.normal(size=data.shape[0])
    return data

def stretch(data, rate=0.8):
    return librosa.effects.time_stretch(data, rate)

def shift(data):
    shift_range = int(np.random.uniform(low=-5, high = 5)*1000)
    return np.roll(data, shift_range)

def pitch(data, sampling_rate, pitch_factor=0.7):
    return librosa.effects.pitch_shift(data, sampling_rate, pitch_factor)
```

Рис. 9: Процесс увеличения данных

Создаем сверточную нейронную сеть:

```
model=Sequential()
model.add(Conv1D(256, kernel_size=28, strides = 1, padding='same', activation='relu', input_shape=(x_train.shape[1], 1)))
model.add(MaxPooling1D(pool_size=5, strides = 2, padding = 'same'))
model.add(Dropout(0.3))

model.add(Conv1D(256, kernel_size=28, strides=1, dilation_rate = 2, padding='same', activation='tanh'))
model.add(MaxPooling1D(pool_size=5, strides = 2, padding = 'same'))
model.add(Dropout(0.3))

model.add(Conv1D(128, kernel_size=20, strides=1, dilation_rate = 2, padding='same', activation='relu'))
model.add(MaxPooling1D(pool_size=5, strides = 2, padding = 'same'))
model.add(Dense(units=64,kernel_regularizer=keras.regularizers.l1_l2(l1=0.0001, l2=0.0001), activation='tanh'))
model.add(Dropout(0.3))

model.add(Conv1D(64, kernel_size=20, strides=1, kernel_regularizer=keras.regularizers.l1_l2(l1=0.0001, l2=0.0001), padding='same', activation='relu'))
model.add(MaxPooling1D(pool_size=5, strides = 2, padding = 'same'))
model.add(Dropout(0.3))

model.add(Flatten())
model.add(Dense(units=64,kernel_regularizer=keras.regularizers.l1_l2(l1=0.0001, l2=0.0001), activation='tanh'))
model.add(Dropout(0.3))

model.add(Dense(units=5, activation='softmax'))
model.compile(optimizer = 'adamax' , loss = 'categorical_crossentropy', metrics = ['accuracy'])

model.summary()
```

Рис. 10: Одномерная сверточная нейронная сеть

Обучаем сеть с помощью adamax, количество эпох 200. После обучения, переходим к оценке модели, потеря составляет 1.0212, точность 0.7624.

Переходим к прогнозированию на тестовом наборе данных.

	Predicted Labels	Actual Labels
0	disgust	happy
1	angry	angry
2	angry	angry
3	happy	happy
4	angry	angry
5	happy	fear
6	angry	angry
7	happy	happy
8	neutral	neutral
9	disgust	neutral

Рис. 11: Фактические и предсказанные значения

Результаты можно представить также в виде confusion matrix:

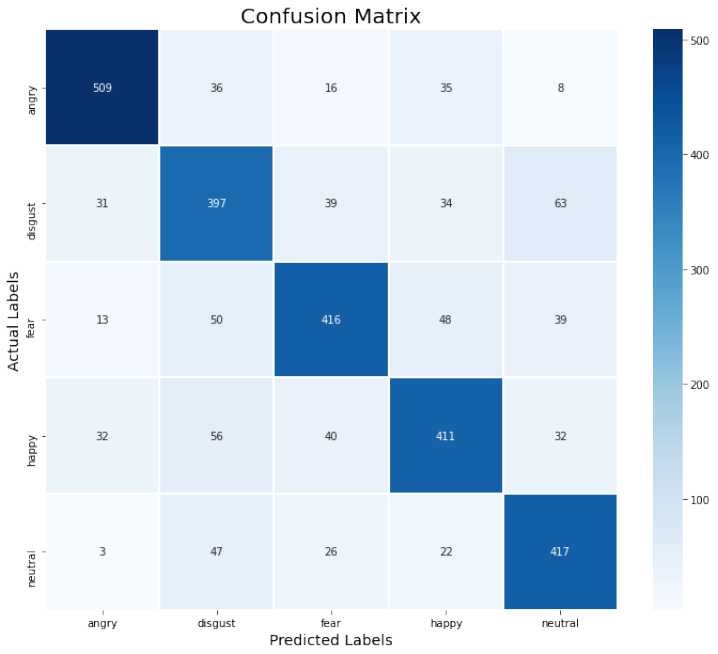

Рис. 12: Confusion matrix для предсказанных значений

Элементы по диагонали представляют те данные, в которых предсказания совпали с имеющейся меткой. Остальные значения показывают ошибки выявленные классификатором. Большинство правильных прогнозов приходится на эмоцию angry. Эмоции happy, fear и neutral находятся примерно в равных условиях. Хуже всего показатели у disgust.

Перейдем к отчету о классификации. Здесь можно увидеть следующие метрики:

- precision (точность) можно рассматривать как меру точности классификатора. Для каждого класса он определяется как отношение истинных положительных результатов к сумме истинных и ложных положительных результатов. Другими словами, «какой процент был правильным для всех случаев, классифицированных как положительные?».

-recall, это мера полноты классификатора; умение классификатора правильно находить все положительные примеры. Для каждого класса он определяется как отношение истинных положительных результатов к сумме истинных положительных и ложно отрицательных результатов. Другими словами, «для всех случаев, которые были действительно положительными, какой процент был классифицирован правильно?».

-f1 score, оценка f1 представляет собой среднее значение precision и recall, так что наилучшая оценка составляет 1,0, а наихудшая - 0,0. Как правило, для сравнения моделей классификаторов следует использовать именно значение f1, а не общую точность.

-support, это количество фактических вхождений класса в указанный набор данных. Несбалансированная поддержка в обучающих данных может указывать на структурные недостатки сообщаемых баллов классификатора и может указывать на необходимость стратифицированной выборки или повторного балансирования. Support не меняется между моделями, а вместо этого диагностирует процесс оценки.[19]

-macro avg, определяет точность классификатора.

-weighted avg, показывает среднее значение для модели, объединяет прогнозы для каждой из них в равной степени и часто приводит к более высокой производительности в среднем, чем данная отдельная модель.

Рассмотрим имеющиеся в таблице значения. Наилучшие значения показывает angry, f1-score больше всех приближено к единице, наименьшее значение принимает эмоция disgust.

	precision	recall	f1-score	support
angry	0.87	0.84	0.85	604
disgust	0.68	0.70	0.69	564
fear	0.77	0.73	0.75	566
happy	0.75	0.72	0.73	571
neutral	0.75	0.81	0.78	515
accuracy			0.76	2820
macro avg	0.76	0.76	0.76	2820
weighted avg	0.76	0.76	0.76	2820

Рис. 13: Classification report для предсказанных значений

Общая точность модели 0,76. Macro avg и weighted avg также имеют значения 0,76, а значит точность классификатора и параметры модели выбраны оптимальные.

2.2 Предсказание на реальных данных

Для предсказания значений на основе обученной модели используются встроенные циклы. В качестве входных данных массивы Numpy.

Для анализа был взят набор DAIC-WOZ Depression Database — база данных депрессии ВОЗ. Эта база данных является частью более крупного корпуса Distress Analysis Interview Corpus (DAIC), который содержит клинические интервью. Эти интервью были собраны в рамках усилий по созданию компьютерного агента, который опрашивает людей и идентифицирует вербальные и невербальные показатели психического заболевания. Собранные данные включают в себя аудио и видеозаписи и обширные ответы на анкеты; эта часть корпуса включает в себя интервью Волшебника из страны Оз, проведенные анимированным виртуальным интервьюером по имени Элли, контролируемым человеком -интервьюером в другой комнате. Данные были расшифрованы и аннотированы для различных вербальных и невербальных особенностей. [8]

Пакет включает в себя 189 папок сеансов 300-492 с не размеченными данными. Для нашего исследования сеансы были ограничены с 300 до 350. Каждый сеанс был на разделен на аудиофайлы с временным отрезком в 4 секунды, для более точного выделения признаков которые превалируют в том или ином сеансе.

Далее будут представлены полученные результаты по каждому из сенсов в отдельности и набору данных в целом.

```
angry: 28.52%
neutral: 7.22%
happy: 26.28%
disgust: 28.80%
fear: 9.18%
```

Рис. 14: Предсказанные значения для всего набора данных

Из процентного соотношения положительных и негативных эмоций, можно скать что предсказание сделано верно, т.к. эмоции angry, fear и disgust в большей степени отражают депрессивные состояния и составляют 66,5% от общего колличества данных.

```
------------------------------------------------------------
Summary for path /content/drive/MyDrive/Delenie/300_AUDIO:
  angry -> 40.10%
  neutral -> 1.45%
  happy -> 16.43%
  disgust -> 14.01%
  fear -> 28.02%
------------------------------------------------------------
Summary for path /content/drive/MyDrive/Delenie/301_AUDIO:
  happy -> 12.50%
  angry -> 52.65%
  disgust -> 28.03%
  fear -> 5.30%
  neutral -> 1.52%
------------------------------------------------------------
Summary for path /content/drive/MyDrive/Delenie/302_AUDIO:
  happy -> 11.11%
  disgust -> 52.67%
  fear -> 19.34%
  angry -> 14.40%
  neutral -> 2.47%
------------------------------------------------------------
Summary for path /content/drive/MyDrive/Delenie/303_AUDIO:
  angry -> 49.84%
  fear -> 9.84%
  disgust -> 18.73%
  happy -> 20.00%
  neutral -> 1.59%
```

Рис. 15: Пример предсказанных значений для сеансов 300-303

Большая заинтересованность в использовании такого анализа существует у ведущих исследовательских центров в России, например, Национальный медицинский исследовательский центр имени В. А. Алмазова проводит на их основе исследование совместно с коллегами из СПбГУ, для определения у больных конкретных психических отклонений используя аудиоанализ для выявления характерных речевых отклонений. В том числе такой анализ позволяет осуществить поиск неочевидных факторов, например определение хрипоты и т.п.

Глава 3. XAI алгоритмы

3.1 Обзор популярных методов XAI

Модели машинного обучения в большистве своем являются черными ящиками. Понимание причин, лежащих в основе прогнозов, весьма важно при оценке доверия, что имеет фундаментальное значение, если вы планируется предпринять действия на основе прогноза. Одной из целей XAI является разработка инновационных алгоритмов объяснения, которые обещают дать новое представление о современных моделях черного ящика машинного обучения и тем самым помочь пользователю лучше понять и доверять системе искусственного интеллекта[2].

Можно отметить следующие факторы, способствующие распространению алгоритмического принятия решений на основе искусственного интеллекта:

1) потребность в обработке различных объемных данных;

2) наличие мощных вычислительных ресурсов (например, вычислений на GPU, облачных вычислений);

3) мощные и новые алгоритмы.

Также вводится законодательное регулирование для контроля за реализацией направления:

1)Общая защита данных Европейского союза, положение (GDPR) (“право на объяснение”);

2) Закон об алгоритмической подотчетности правительства США от 2019г.;

3) Этические принципы Министерства обороны США для искусственного интеллекта и решения проблем, связанных со справедливостью, подотчетностью и прозрачностью, с помощью автоматизированных систем принятия решений.

Перечислим некоторые из популярных инструментов для XAI: LIME, DeepVis Toolbox, TreeInterpreter, Keras-vis, Microsoft InterpretML, MindsDB, SHAP, Tensorboard WhatIf, Tensorflow Lucid, Cleverhans Tensorflow и т. д.

На общем уровне каждый из предложенных подходов имеет схожие концепции, такие как важность признаков их взаимодействие, четкие цен-

ности, частичная зависимость, суррогатные модели, вливание знаний и др.

Возобновления интереса к исследованиям XAI, после экспертных систем проистекает из недавних достижений в области ИИ, его применения в широком спектре областей, опасений по поводу неэтичного использования, отсутствия прозрачности и нежелательных искажений в моделях. Кроме того, недавние законы, принятые различными правительствами, требуют проведения дополнительных исследований в области XAI.

В 2019 году Мюллер[20] и др. представляет всесторонний обзор подходов, используемых рядом типов "систем объяснения", и характеризует их на три поколения:

1) системы первого поколения - например, экспертные системы начала 70—х годов;

2) системы второго поколения - например, интеллектуальные системы обучения;

3) системы третьего поколения - инструменты и методы недавно появившиеся, начиная с 2015 года.

Системы первого поколения пытаются четко выразить внутренний рабочий процесс системы путем встраивания экспертных знаний в правила, часто получаемые непосредственно от экспертов (например, путем преобразования правил в выражения на естественном языке). Системы второго поколения можно рассматривать как человеко-компьютерную систему, созданную на основе человеческих знаний и способности рассуждать для обеспечения когнитивной поддержки. Например, организовать интерфейс таким образом, чтобы он дополнял знания, которых не хватает пользователю. Подобно системам первого поколения, системы третьего поколения также пытаются прояснить внутреннюю работу систем. Но на этот раз эти системы в основном являются "черным ящиком" (например, глубокие сети, ансамблевые подходы).

В широком смысле методы объяснения классифицируются на три вида:

1)внутренние интерпретируемые методы;

2)модельно-агностические объяснения;

3)основанные на примерах объяснения.

3.1.1 Внутренне интерпретируемые методы.

В линейной регрессии прогнозируемая цель состоит из взвешенной суммы входных признаков. Таким образом, вес или коэффициент линейного уравнения можно использовать в качестве средства объяснения предсказания, когда число признаков невелико.

Логистическая регрессия, это расширение линейной регрессии для задачи классификации. Она моделирует вероятности для задач классификации. Интерпретация логистической регрессии отличается от линейной регрессии, поскольку она дает вероятность от 0 до 1, где вес может не точно представлять линейную связь с предсказанной вероятностью. Однако вес дает представление о направлении влияния (отрицательном или положительном) и факторе влияния между классами, хотя он и не является аддитивным к общему прогнозу.

Модели на основе дерева решений разделяют данные несколько раз на основе порога отсечения на каждом узле, пока они не достигнут конечного узла. В отличие от логистической и линейной регрессии, она работает даже тогда, когда связь между входными и выходными данными нелинейна и даже когда объекты взаимодействуют друг с другом (корреляция между объектами). Однако древовидные объяснения не могут выразить линейную связь между входными объектами и выходными данными. Им также не хватает плавности; небольшие изменения на входе могут оказать большое влияние на прогнозируемый результат. Кроме того, для одной и той же проблемы может быть несколько разных деревьев. Как правило, чем больше узлов или глубина дерева, тем сложнее его интерпретация.

Правила принятия решений (простые условия IF-THEN-ELSE) также являются неотъемлемой моделью объяснения. Хотя правила (IF-THEN) просты в интерпретации, они в основном ограничены проблемами классификации (т.е. не поддерживают проблему регрессии) и неадекватны при описании линейных отношений. Кроме того, алгоритм RuleFit [21] имеет внутреннюю интерпретацию, поскольку он изучает разреженные линейные модели, которые могут обнаруживать эффекты взаимодействия в виде правил принятия решений. Правила принятия решений состоят из комбинации

разделенных решений по каждому из путей принятия решений. Однако, помимо оригинальных функций, он также изучает некоторые новые функции, чтобы запечатлеть эффекты взаимодействия оригинальных функций. Обычно интерпретируемость ухудшается с увеличением числа функций.

Другие интерпретируемые модели включают расширение линейных моделей, таких как обобщенные линейные модели (GLMs) и обобщенные аддитивные модели (GAMs). Они помогают справиться с некоторыми допущениями линейных моделей (например, целевой результат y и заданные функции следуют распределению Гаусса; отсутствие взаимодействия между функциями). Однако эти расширения делают модели более сложными (т.е. добавляют взаимодействия), а также менее интерпретируемыми. Кроме того, существуют Наивный байесовский классификатор, основанный на теореме Байеса, где вероятность классов для каждого из признаков вычисляется независимо (при условии строгой независимости функций), и K-Nearest Neighbours, которые используют ближайших соседей как точки данных для прогнозирования (регрессии или классификации).

3.1.2 Модельно-агностические объяснения

Методы, не зависящие от модели, отделяют объяснение от модели машинного обучения, позволяя методу объяснения быть совместимым с различными моделями. Это разделение имеет некоторые очевидные преимущества, такие как:
1) способ толкования может работать с несколькими моделями машинного обучения;
2) предоставляет различные формы объяснимости (например, визуализация важности признаков) для конкретной модели;
3) существует возможность для гибкого представления, например текстовый классификатор использует абстрактное встраивание слов для классификации, но с использованием реальных слов для объяснения. Некоторые из инструментов модельно-агностических объяснений: Partial Dependence Plot (PDP), Individual Conditional Expectation (ICE), Accumulation Local Effects (ALE) Plot, Feature Interaction, Feature Importance, Global Surrogate, Local Surrogate (LIME), and Shapley Values (SHAP).

Partial Dependence Plot (PDP) или график PD показывает предельное влияние одного или двух признаков (в лучшем случае трех признаков в 3-D) на прогнозируемый результат модели ML [21]. Это глобальный метод, поскольку он показывает общее поведение модели и способен показывать линейные или сложные отношения между целью и объектом/объектами). Он предоставляет функцию, которая зависит только от объекта/обьектов, отображаемого путем выделения других объектов таким образом, чтобы они включали взаимодействия между ними. PDP обеспечивает четкую причинно-следственную связь интерпретации путем предоставления изменений в прогнозе в связи с изменениями в конкретных функциях. Однако PDP предполагает , что объекты не коррелируют с остальными. Кроме того, существует практический предел только двух функций, которые PD может четко объяснить одновременно. Кроме того, это глобальный метод, поскольку он отображает среднее влияние (всех экземпляров) объекта/объектов на прогноз, а не для всех объектов в конкретном экземпляре.

Individual Conditional Expectation(ICE). В отличие от PDP, ICE строит по одной строке на экземпляр, показывая, как объект влияет на изменения в прогнозе.

Accumulation Local Effects(ALE). Аналогично диаграммам PD графики ALE описывают, как особенности в среднем влияют на прогноз. Однако, в отличие от PDP, график ALE достаточно хорошо работает с коррелированными функциями и сравнительно быстрее. Хотя график ALE не смещен в сторону коррелированных признаков, трудно интерпретировать изменения в прогнозе, когда признаки сильно коррелированы и анализируются изолированно. В этом случае только графики, показывающие изменения в обоих коррелированных признаках вместе, имеют смысл для понимания изменений в прогнозе.

Одним из методов является Feature Interaction. Когда функции взаимодействуют друг с другом, отдельные эффекты функций не суммируются с общими эффектами функций от всех функций вместе взятых. H-статистика (т.е. критерий Фридмана) помогает обнаруживать различные типы взаимодействия, даже с тремя или более функциями. Сила взаимодействия между двумя функциями - это разница между частичным функция зависимости для этих двух функций вместе и сумма частных функций зависимости для каждой функции отдельно.

Следующий метод определяется как Feature Importance. Обычно важность признака заключается в увеличении ошибки прогнозирования модели, когда мы переставляем значения признака, чтобы нарушить связь между признаком и результатом. Если после перетасовки значений функции ошибки увеличиваются, то функция важна. Позже была введена важность функции, основанная на перестановках, для случайных лесов. Еще позже была расширена работа до версии, не зависящей от модели. Важность функций обеспечивает сжатое и глобальное представление о поведении модели ML. Хотя важность признаков учитывает как основной их эффект, так и взаимодействие. Это является недостатком, поскольку оно включено в важность коррелированных признаков. В результате при наличии взаимодействия между функциями, важность функций не влияет на общее снижение производительности. Кроме того, неясно, следует ли использо-

вать тестовый набор или обучающий набор для оценки важности объектов, поскольку он демонстрирует различия от запуска к запуску в перетасованном наборе данных. Необходимо отметить, что важность функции также подпадает под глобальные методы.

Global Surrogate. Глобальная суррогатная модель пытается аппроксимировать общее поведение модели "черного ящика", используя интерпретируемую модель ML. Другими словами, суррогатные модели пытаются аппроксимировать функцию прогнозирования модели черного ящика, используя интерпретируемую модель как можно точнее, учитывая что предсказание поддается интерпретации.Он также известен как метамодель, приближенная модель, модель поверхности отклика или эмулятор. Недостатком являются различные объяснения одного и того же "черного ящика", такие как несколько возможных деревьев решений с различными структурами.

Shapley Values еще один метод локального объяснения. Шепли ввел это значение в 1953 году [22]. Он основан на теории кооперативных игр, которая помогает справедливо распределить важность функций между участвующими игроками. Здесь предполагается, что каждое значение функции экземпляра является игроком в игре, а прогноз - это общая выплата, которая распределяется среди игроков (то есть функций) в соответствии с их вкладом к общей выплате (то есть прогнозу). Shapely Value—это средний вклад в прогнозирование по всем возможным коалициям объектов, что делает его вычислительно дорогостоящим, когда имеется большое количество объектов, например, для k-го числа объектов будет 2k-е число коалиций. В отличие от LIME, Shapely Value-это метод объяснения с твердой теорией , которая дает полные объяснения. Однако он также страдает от проблемы коррелированных признаков. Кроме того, значение Shapely возвращает одно значение для каждого объекта; нет никакого способа сделать заявление об изменениях в выходных данных, вызванных в них изменениями. Одна из реализаций Shapley Values библиотека SHAP.

3.1.3 Объяснения на основе примеров.

Методы объяснения на основе примеров используют конкретные экземпляры из набора данных для объяснения поведения модели и распределения данных агностическим способом. Это можно выразить так: “X похож на Y, и Y вызвал Z, поэтому предсказание говорит, что X вызовет Z”. Далее рассмотрим [23], несколько методов объяснения, которые подпадают под эту категорию.

Adversarial метод указывает на необходимые изменения на стороне ввода, которая будет иметь значительные изменения в предсказании / выводе. Контрфактические объяснения могут объяснить отдельные предсказания. Например, он может дать объяснение, описывающее прямую зависимость ситуации, такую как “Если бы A не произошло, B не произошло бы”. Хотя контрфактические объяснения удобны для человека, они страдают от “эффекта Рашомона”, когда каждое контрфактическое объяснение рассказывает другую историю, чтобы достичь предсказания. Другими словами, существует несколько истинных объяснений (контрфактических) для каждого предсказания уровня экземпляра, и проблема заключается в том, как выбрать лучшее из них. Контрфактические методы не требуют доступа к данным или моделям и могут работать с системой, которая вообще не использует машинное обучение. Кроме того, этот метод плохо работает для категориальных переменных со многими значениями.

Adversarial метод способен перевернуть решение, используя контрфактические примеры, чтобы обмануть машинное обучение (т.е. небольшие преднамеренные возмущения во входных данных, чтобы сделать ложный прогноз). Однако примеры adversarial могут помочь обнаружить скрытые уязвимости, а также улучшить модель.

Prototypes состоят из выбранного набора экземпляров, которые очень хорошо представляют данные. И наоборот, набор экземпляров , которые плохо представляют данные, получают критические замечания.

Influential Instances-это точки данных из обучающего набора , которые влияют на прогноз и определение параметров модели. Хотя это помогает отладить модель и лучше понять поведение модели, определение пра-

вильной точки отсечения для разделения влиятельных или невлиятельных экземпляров является сложной задачей.

K-nearest Neighbors Model, предсказание модели k-ближайшего соседа может быть объяснено точками данных k-соседа (соседи, которые были усреднены для прогнозирования). Визуализация отдельного кластера, содержащего аналогичные экземпляры, обеспечивает интерпретацию того, почему экземпляр является членом определенной группы или кластера.

3.2 SHAP

Аддитивные объяснения Шепли (SHAP): это теоретико-игровой подход для объяснения результатов любой модели машинного обучения, использующее значения Шепли.

Был предложен Лундбергом и его коллегами [24]. SHAP объясняет предсказания входного x путем вычисления вклада отдельных функций в это выходное предсказание. Формулируя функции данных как игроков в коалиционной игре, можно вычислить значения Шепли, чтобы научиться справедливо распределять выплаты.

В методе SHAP данные могут быть отдельными категориями. в табличных данных или в группах суперпикселей в изображениях, подобных LIME. Затем SHAP выводит задачу как набор линейных функций, где объяснение также является линейной функцией[25].

Если мы рассмотрим g как модель объяснения модели машинного обучения $f, z' \in \{0,1\}^M$ как вектор коалиции, M максимальный размер коалиции, а $\phi_j \in R$ - атрибуция признака j, $g(z')$ - это сумма смещения и вкладов отдельных функций такой, что:

$$g(z') = \phi_0 + \sum_{j=1}^{M} \phi_j z'_j \tag{4}$$

Лундберг [24] описывает несколько вариантов к базовому методу SHAP, например KernelSHAP, который уменьшает количество оценок, необходимых для больших входных данных в любой модели машинного обучения, LinearSHAP, который оценивает значения SHAP используя весовые коэффициенты модели с учетом независимых входных характеристик, Low-Order SHAP, который эффективен для небольшого максимума с размером коалиции M и DeepSHAP, который адаптирует DeepLIFT метод для использования глубоких нейронных сетей для улучшения атрибуции. Поскольку KernelSHAP применим ко всем алгоритмам машинного обучения, мы описываем это в алгоритме 2 (рисунок 16). Общая идея KernelSHAP состоит в том, чтобы выполнить метод атрибуции аддитивных признаков случайным образом, сделав выборку коалиций путем удаления признаков из входных

данных и линеаризации влияния модели с использованием ядер SHAP.

Algorithm 2 KernelSHAP Algorithm for Local Explanations

Input: classifier f, input sample x
Output: explainable coefficients from the linear model
1: $z_k \leftarrow$ SampleByRemovingFeature(x)
2: $z_k \leftarrow h_x(z_k) \triangleright h_x$ is a feature transformation to reshape to x
3: $y_k \leftarrow f(z_k)$
4: $W_x \leftarrow SHAP(f, z_k, y_k)$
5: `LinearModel(`W_x`).fit()`
6: Return `LinearModel.coefficients()`

Рис. 16: KernelSHAP Algorithm for Local Explanations.

SHAP также широко использовался исследовательским сообществом, был применен напрямую и улучшился во многих аспектах. Рассмотрим некоторые примеры использования SHAP в области медицины для объяснения принятия клинических решений и некоторые недавние работы, которые имеют большое значение:

• Антварг и его коллеги [26], разработали расширенный метод SHAP для объяснения автоэнкодеров, используемых для обнаружения аномалий. Авторы классифицируют аномалии с использованием автокодировщика путем сравнения фактических экземпляров данных с реконструированным выходом. Поскольку окончательный результат - реконструкция, авторы предполагают, что объяснения должны основываться на ошибке реконструкции. Значения SHAP найдены для наиболее эффективных функций и разделены на вносящих вклад и компенсирующих аномалии.

• Сундарараджан и его коллеги [27], выразили различные недостатки метода SHAP, например, генерирование нелогичных объяснений для случаев, когда определенные функции не важны. Это свойство «уникальности» метода атрибуции улучшено с использованием базового метода Шепли (BShap). Авторы дополнительно расширяют метод, используя интегрированные градиенты для непрерывных областей.

• Аас и его коллеги[28], исследовали зависимость между значениями SHAP путем расширения метода KernelSHAP для обработки зависимых особенностей. Авторы также представили метод значения кластера Шепли, соответствующий зависимым признакам. Было проведено тщательное исследование метода KernelSHAP, осуществленное четырьмя предложен-

ными методами для замены условных распределений метода KernelSHAP с использованием эмпирического подхода и либо Гауссовского.

• Лундберг и его коллеги [29], описали расширение SHAP метода для деревьев в рамках под названием TreeExplainer, где структуру глобальной модели обьясняют с помощью локальных объяснения. Авторы описали алгоритм вычисления локального объяснения деревьев за полиномиальное время на основе точных значений Шепли.

• Вега Гарсия и его коллеги [30], описали метод на основе SHAP для объяснения предсказаний сигналов временных рядов, включающих сети с долговременной краткосрочной памятью (LSTM). Авторы использовал алгоритм DeepSHAP для объяснения отдельных случаев в тестовом наборе, основанном на наиболее важных функциях из обучающего набора. Однако никаких изменений в методе SHAP не было сделано, и объяснения генерировались для каждого экземпляра ввода.

3.3 LIME

LIME относится к категории локально обьяснимых алгоритмов. Как правило такой метод фокусируется на одном экземпляре данных для генерации объяснений с использованием различных данных. Здесь нас интересует генерация g для объяснения решения, принятые f для единственного входного экземпляра x.[Рис. 17]

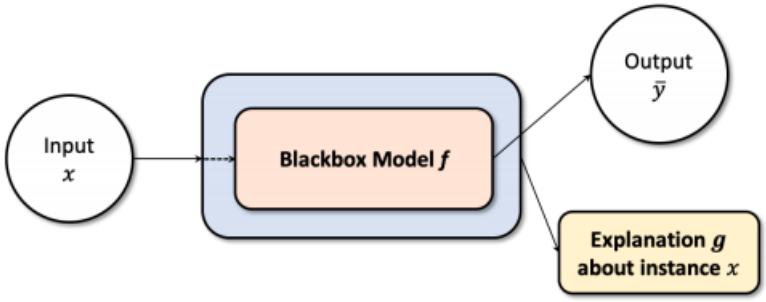

Рис. 17: Общая иллюстрация локально объяснимых моделей. Как правило, единственный входной экземпляр используется для объяснения.

Основополагающие исследования в области локальных объяснений используют байесовские методы и отмечают важность функций матрицы для понимания прогнозов на выходе модели. Пояснениями к выходным данным были всегда положительные вещественные матрицы или векторы. Новые исследования в локальных объяснимых моделях улучшают старые методы на основе графов и теории игр в которых мы получаем функциональную оценку положительных и отрицательных корреляции в выходной классификации. Здесь положительное значение приводит к тому, что конкретная функция улучшает вероятность выходного класса, а отрицательное значение означает, что функция уменьшила вероятность выходного класса.

В 2016 году Рибейро и др. представил локальные интерпретируемые модельно-агностические объяснения (LIME) [12]. Чтобы получить представление понятное для людей, LIME пытается найти важность смежных суперпикселей (фрагменты пикселей) в источнике изображение в выходном классе. Следовательно, LIME находит двоичный вектор $x_0 \in 0, 1$ для

обозначения наличия или отсутствия непрерывного пути или "суперпиксель обеспечивающий наивысшее представление к выходу класса. Это работает на уровне патча для одного ввода данных. Следовательно, метод подпадает под локальные объяснения. Существует также глобальная модель объяснения, основанная на LIME называется SP-LIME, описанный в глобальном объяснимом подразделе модели. Здесь мы сосредотачиваемся на местных объяснениях.

Algorithm 1 LIME algorithm for local explanations

Input: classifier $\boldsymbol{f}$, input sample $\boldsymbol{x}$, number of superpixels $\boldsymbol{n}$, number of features to pick $\boldsymbol{m}$
Output: explainable coefficients from the linear model

$\bar{\boldsymbol{y}} \leftarrow \boldsymbol{f}$.predict($\boldsymbol{x}$)
for i in $\boldsymbol{n}$ **do**
    $\boldsymbol{p_i} \leftarrow$ Permute($\boldsymbol{x}$) ▷ Randomly pick superpixels
    $\boldsymbol{obs_i} \leftarrow \boldsymbol{f}$.predict($\boldsymbol{p}$)
    $\boldsymbol{dist_i} \leftarrow |\bar{\boldsymbol{y}} - \boldsymbol{obs_i}|$
end for
$\boldsymbol{simscore} \leftarrow$ SimilarityScore($\boldsymbol{dist}$)
$\boldsymbol{x_{pick}} \leftarrow$ Pick($\boldsymbol{p}, \boldsymbol{simscore}, \boldsymbol{m}$)
$\boldsymbol{L} \leftarrow$ LinearModel.fit($\boldsymbol{p}, \boldsymbol{m}, \boldsymbol{simscore}$)
return $\boldsymbol{L}$.weights

Рис. 18: Алгоритм LIME

Рассмотрим $g \in G$, объяснение как образец из класса потенциально интерпретируемых моделей G. Здесь g может быть решено с помощью деревьев, линейной модели или других моделей различной интерпретируемости. Пусть сложность объяснения измеряется $\Omega(g)$. Если $\pi_x(z)$ является мерой близости между двумя экземплярами x и z вокруг x, а $\mathcal{L}(f, g, \pi_x)$ представляет точность g в приближении f в местности, определяемой π_x, то объяснение ξ для входных данных выборка данных x определяется уравнением LIME:

$$\xi(x) = argmin_{g \in G} \mathcal{L}(f, g, \pi_x) + \Omega(g) \tag{4}$$

Теперь в уравнении 4 целью оптимизации LIME необходимо минимизировать потерю с учетом локальности $\mathcal{L}(f, g, \pi_x)$ в модели используется агностический способ.

Рис. 19: Локальные объяснения предсказания классификации изображений, описанного с использованием LIME. Здесь три лучших класса - ”электрогитара” (p = 0,32), ”акустическая гитара” (p = 0,24) и ”лабрадор” (p = 0,21). Выбрав группу "суперпикселей’ из входного изображения, классификатор предоставляет визуальные объяснения для верхних предсказанных меток.

Пример визуализации алгоритма LIME на один экземпляр показан на рисунке 18. Алгоритм 1 показывает шаги для объяснения модели для одного входного образца и общую процедуру LIME. Здесь для входного экземпляра мы переставляем данные, находя суперпиксель. Затем мы вычисляем расстояние (оценку сходства) между перестановками и оригинальными наблюдениями. Теперь мы получаем разные баллы класса для исходного ввода и нового «фальшивых» данных. Затем мы можем делать прогнозы на основе новых «фальшивых» данных, используя модель f. Она зависит от количества суперпикселей, выбранных из исходных данных. Самая главная особенность, можно выбрать, что улучшило предсказание на переставленных данных. Если мы подбираем простую модель, часто локально взвешенную регрессионную с переставленными данными с m функциями и оценками сходства в качестве весов, мы можем использовать веса характеристик, или коэффициенты из простой модели для объяснения локального поведения сложной модели.

В последние годы набирает обороты много исследований, улучшающих и расширяющих LIME алгоритм для множества новых задач. Резюмируем некоторые из них:

• Саумитра Мишра и его коллеги [31] представили расширенный алгоритм LIME для музыки, контент-анализ по временной сегментации и частоте и частотно-временная сегментация входной мел-спектограммы. Их подход назывался Sound-LIME (SLIME) и был применялся для объяснения

предсказаний голоса.

• Томи Пелтола [32] описал LIME, основанный на расстоянии Кульбака-Лейблера, под названием KL-LIME для объяснения байесовских моделей прогноза. Подобно LIME, объяснения генерируются с использованием интерпретируемой модели, параметры которой находятся путем минимизации KL-расходимости и модели проноза. Таким образом, местные интерпретируемые объяснения генерируются путем проецирования информации из распределения прогноза в более простую интерпретируемую вероятностную модель объяснения.

•Рехман и его коллеги [33] использовали агломеративный иерархический алгоритм кластеризации (HC) и K-ближайшего соседа (KNN) для замены случайного возмущения LIME. Здесь авторы используют метод HC для группировки обучающих данных, по аналогии с кластерами, и KNN используется для поиска ближайших соседей к тестируемому экземпляру. Когда-то KNN выбирает кластер, этот кластер передает в качестве входных данных возмущение вместо случайного возмущения как в LIME. Авторы сообщают, что их подход создает модели объяснения, которые более стабильны, чем традиционный алгоритм LIME.

• Брэмхолл и его коллеги [34] скорректировали линейные отношения LIME для рассмотрения нелинейных отношений с использованием структуры квадратичной аппроксимации, называемой QuadraticLIME (QLIME). Они достигают результатов, учитывая линейные приближения в сложной функции. Результаты свидетельствуют о том, что среднеквадратичные потери (MSE) линейной зависимости LIME на местном уровне улучшаются при использовании QLIME.

• С. Ши и его колелеги [35] представили метод замены для выбора информации суперпикселей для данных изображения с использованием Modified Perturbed Sampling для LIME (MPS-LIME). Авторы преобразовали традиционную операцию выбора суперпикселей в задачу построения множества откликов путем преобразования суперпикселей в неориентированный граф. Операция откликов улучшает время работы за счет значительного сокращения количества возмущенных образцов в MPS-LIME методе. Авторы сравнили свой метод с LIME и использованием Mean Absolute

Error (MAE) и коэффициента определения R^2, в результате сообщили о лучших результатах с точки зрения понятности, точности и оперативности.

3.4 Результаты работы XAI методов

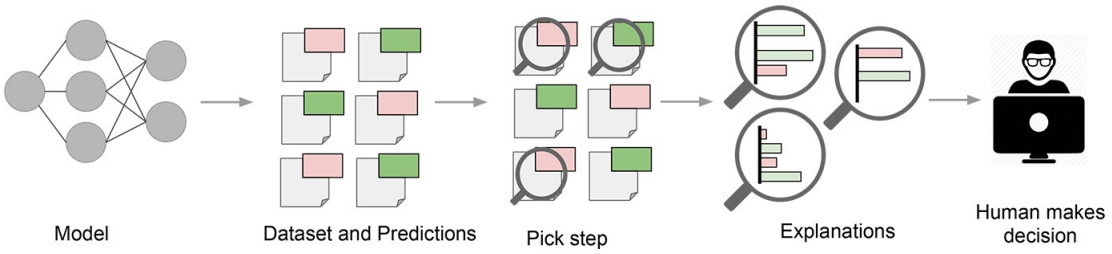

Рис. 20: LIME объясняет предсказания модели на уровне выборки данных. Это позволяет конечным пользователям интерпретировать эти прогнозы и предпринимать действия на их основе.

LIME можно применить к любой модели машинного обучения. Этот метод пытается понять модель, изменяя ввод выборок данных и понимая, как меняются прогнозы. Понимание модели происходит путем анализа внутренних компонентов и того, как они взаимодействуют.

LIME обеспечивает интерпретируемость локальной модели, изменяя одну выборку данных и настраивая значения функций, наблюдает, как это повлияет на результат.

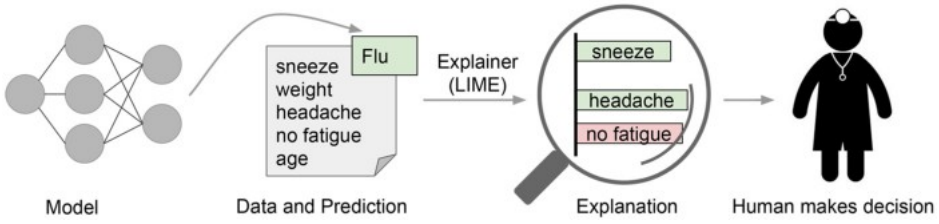

Рис. 21: LIME пытается играть роль «объяснителя», объясняя прогнозы для каждой выборки данных.

Вывод LIME представляет собой список объяснений, отражающих вклад каждой функции в прогнозирование выборки данных. Это обеспечивает локальную интерпретируемость, а также позволяет определить, какие изменения характеристик окажут наибольшее влияние на прогноз.

Объяснение создается путем аппроксимации базовой модели локально интерпретируемой. Интерпретируемые модели - это, например, линейные модели с сильной регуляризацией, деревья решений и т. д. Интерпретируемые модели обучаются небольшим возмущениям исходного экземпляра

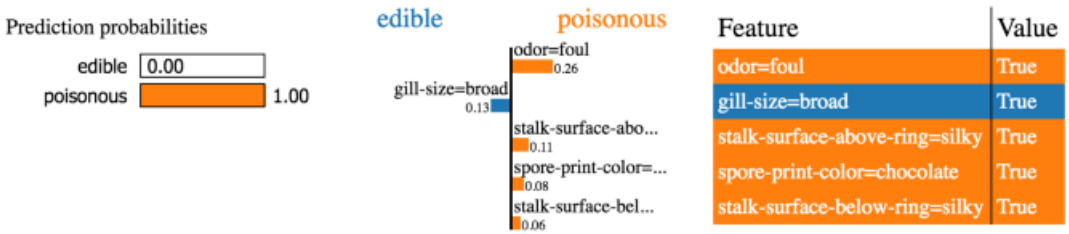

Рис. 22: Пример LIME, примененный к классической задаче классификации

и должны обеспечивать только хорошее локальное приближение. Набор данных создается, например, путем добавления шума к непрерывным объектам, удаления слов или скрытия частей изображения. Только аппроксимируя черный ящик локально (по соседству с выборкой данных), задача значительно упрощается.

LIME имеет несколько недостатков. Во-первых, в текущей реализации для аппроксимации локального поведения используются только линейные модели. В определенной степени это предположение верно, если смотреть на очень маленькую область вокруг выборки данных. Однако при расширении этой области линейная модель может оказаться недостаточно мощной, чтобы объяснить поведение исходной модели. Нелинейность в локальных регионах возникает для тех наборов данных, которые требуют сложных, не интерпретируемых моделей. Невозможность применить LIME в этих сценариях является серьезной ловушкой.

Во-вторых, тип изменений, которые необходимо выполнить для данных, чтобы получить правильные объяснения, обычно зависит от варианта использования. Можно привести следующий пример: модель, которая предсказывает, что изображения в тонах сепии являются ретро, не может быть объяснена наличием или отсутствием супер пикселей. Часто простых возмущений бывает недостаточно. В идеале возмущения были бы вызваны вариацией, наблюдаемой в наборе данных. Ручное управление возмущениями, вероятно, не лучшая идея, так как это, скорее всего, внесет систематическую ошибку в объяснения модели.

Проанализуем работу алгоритма LIME для распознавания речи. На предыдущем этапе с помощью model.predict были предсказаны эмоции для

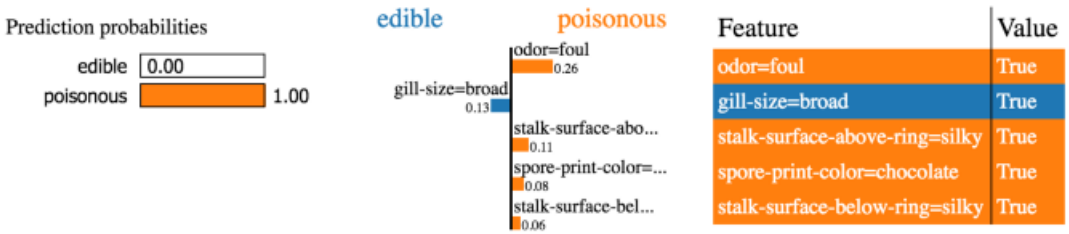

Рис. 23: Пример LIME, примененный к классической задаче классификации

имеющегося набора данных, для примера возьмем первые 15 результатов из них возьмем по одной эмоции и проанализируем их.

	Predicted Labels	Actual Labels
0	happy	happy
1	angry	angry
2	angry	angry
3	happy	happy
4	angry	angry
5	fear	fear
6	angry	angry
7	happy	happy
8	neutral	neutral
9	neutral	neutral
10	disgust	disgust
11	angry	angry
12	fear	fear
13	fear	fear
14	fear	fear

Рис. 24: Первые 15 результатов model.predict

Используя алгоритм LIME обьясняем полученный результат. На вход поступает 2820 элементов тестовой выборки и 162 признака. Можно отследить какие признаки по-мнению LIME повлияли на результат. Рассмотрим сэмпл номер 11, через model.predict она определяется как angry. Рассмотрим вклад каждого признака из категорий. Ответ с наибольшим значением принимается за основной. В данном наборе признаков с вероятностью 95% предсказание модели будет disgust.

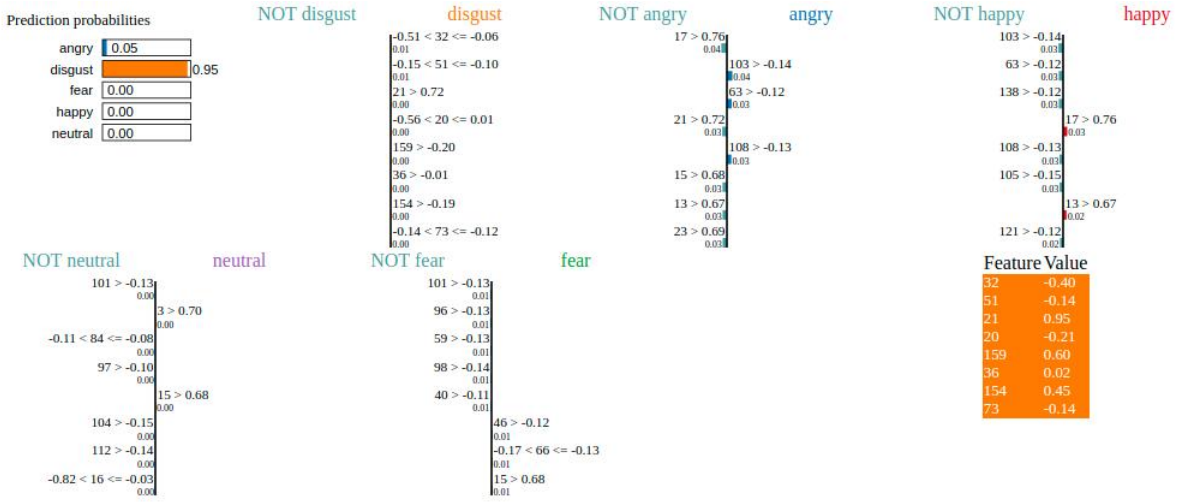

Рис. 25: LIME обьяснение эмоции angry

Рассмотрим сэмп номер 10, через model.predict она определяется как disgust. Рассмотрим вклад каждого признака из категорий. Ответ с наибольшим значением принимается за основной. В данном наборе признаков с вероятностью 95% предсказание модели будет disgust.

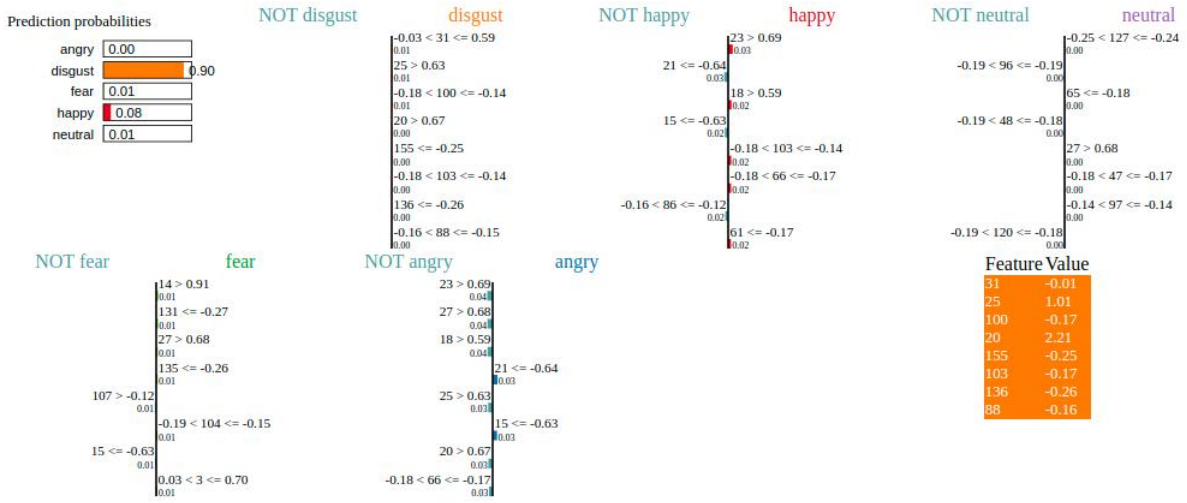

Рис. 26: LIME обьяснение эмоции disgust

Рассмотрим сэмпл номер 5, через model.predict она определяется как fear. Рассмотрим вклад каждого признака из категорий. Ответ с наибольшим значением принимается за основной. В данном наборе признаков с вероятностью 94% предсказание модели будет happy.

Рассмотрим сэмпл номер 7, через model.predict она определяется как

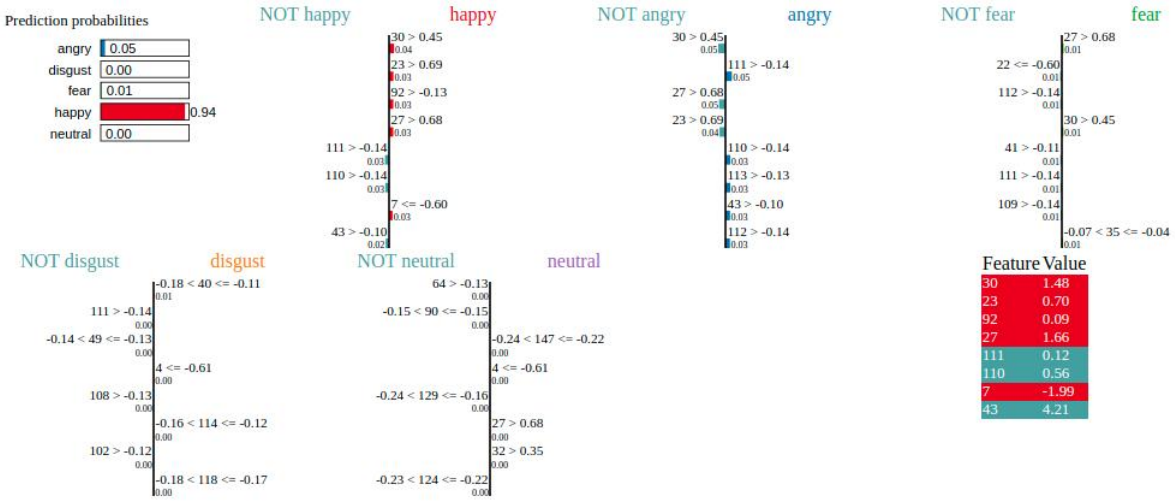

Рис. 27: LIME обьяснение эмоции fear

happy. Рассмотрим вклад каждого признака из категорий. Ответ с наибольшим значением принимается за основной. В данном наборе признаков с вероятностью 95% предсказание модели будет happy.

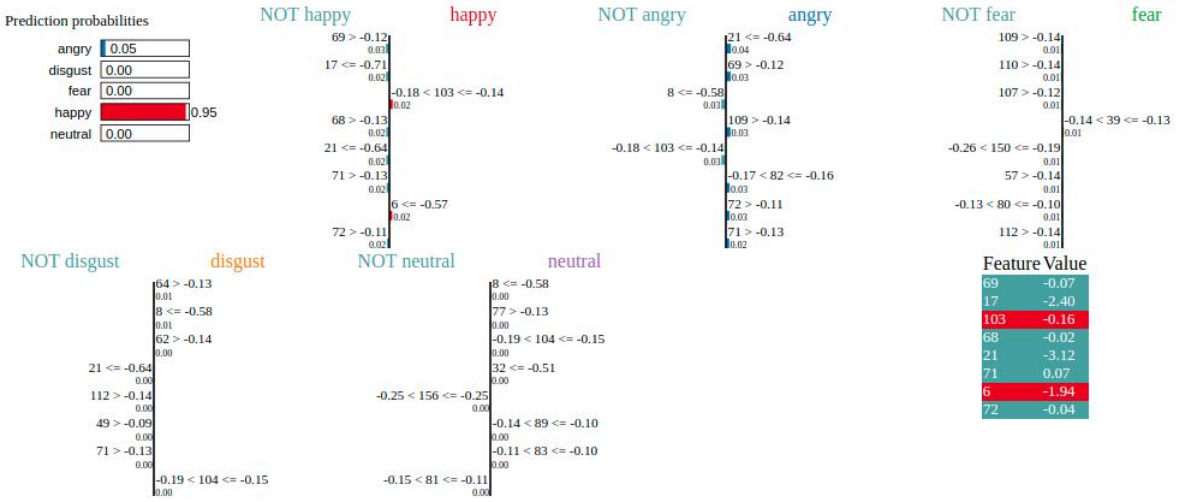

Рис. 28: LIME обьяснение эмоции happy

Рассмотрим сэмпл номер 9, через model.predict она определяется как neutral. Рассмотрим вклад каждого признака из категорий. Ответ с наибольшим значением принимается за основной. В данном наборе признаков с вероятностью 100% предсказание модели будет neutral.

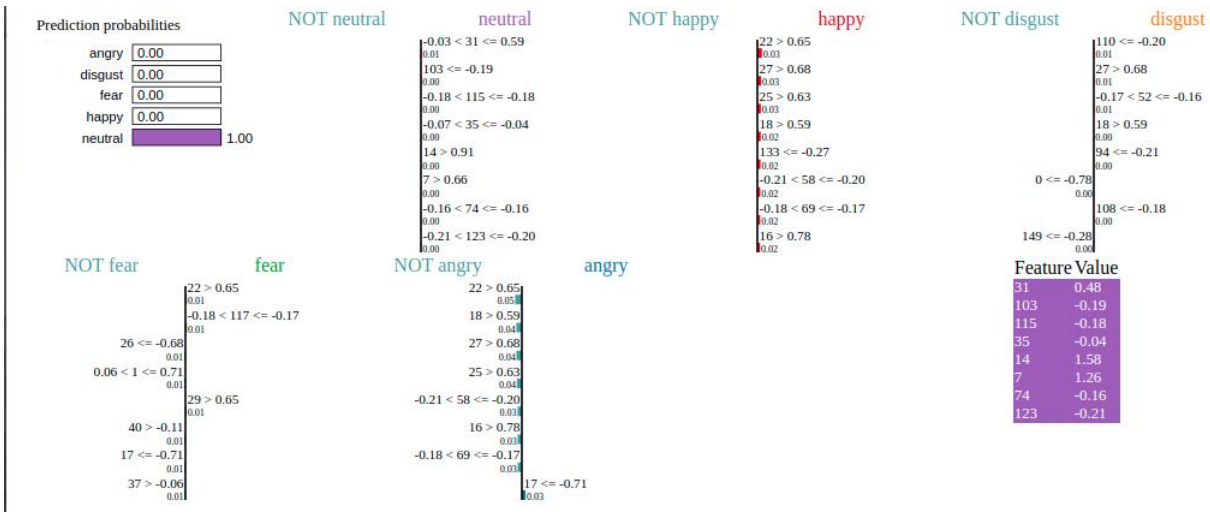

Рис. 29: LIME обьяснение эмоции neutral

SHAP - это теоретико-игровой подход, цель которого, объяснить прогноз для любого экземпляра x как сумму вкладов от его индивидуальных значений характеристик.

Предполагается, что индивидуальные значения характеристик относится к кооперативной игре, выплата которой является предсказанием. В этом параметре значения Shapley предоставляют средства для справедливого распределения выплаты между значениями функций. Обратите внимание, что «значение признака» здесь относится к числовому или категориальному значению признака для экземпляра x.

Точное вычисление значений SHAP является сложной вычислительной задачей. В документе SHAP описаны два метода аппроксимации, не зависящие от модели, один из которых уже известен (значения выборки Шепли), а другой является новым и основан на LIME (Kernel SHAP). В документе SHAP также описывается несколько методов аппроксимации для конкретных типов моделей, таких как Linear SHAP, Tree SHAP, Deep SHAP и т. д. Эти методы предполагают независимость характеристик и линейность модели, чтобы упростить вычисление значений SHAP.

Перейдем к представлению результатов SHAP для тех же сэплов, что были описаны в LIME.

Выбираем сэмпл номер 11, через model.predict она определяется как angry. Рассмотрим вклад каждого признака из категорий. Ответ с наиболь-

шим значением принимается за основной. Здесь основным компонентом выступают признаки эмоции angry, наибольшее значение которой 0.2 для Feature 45.

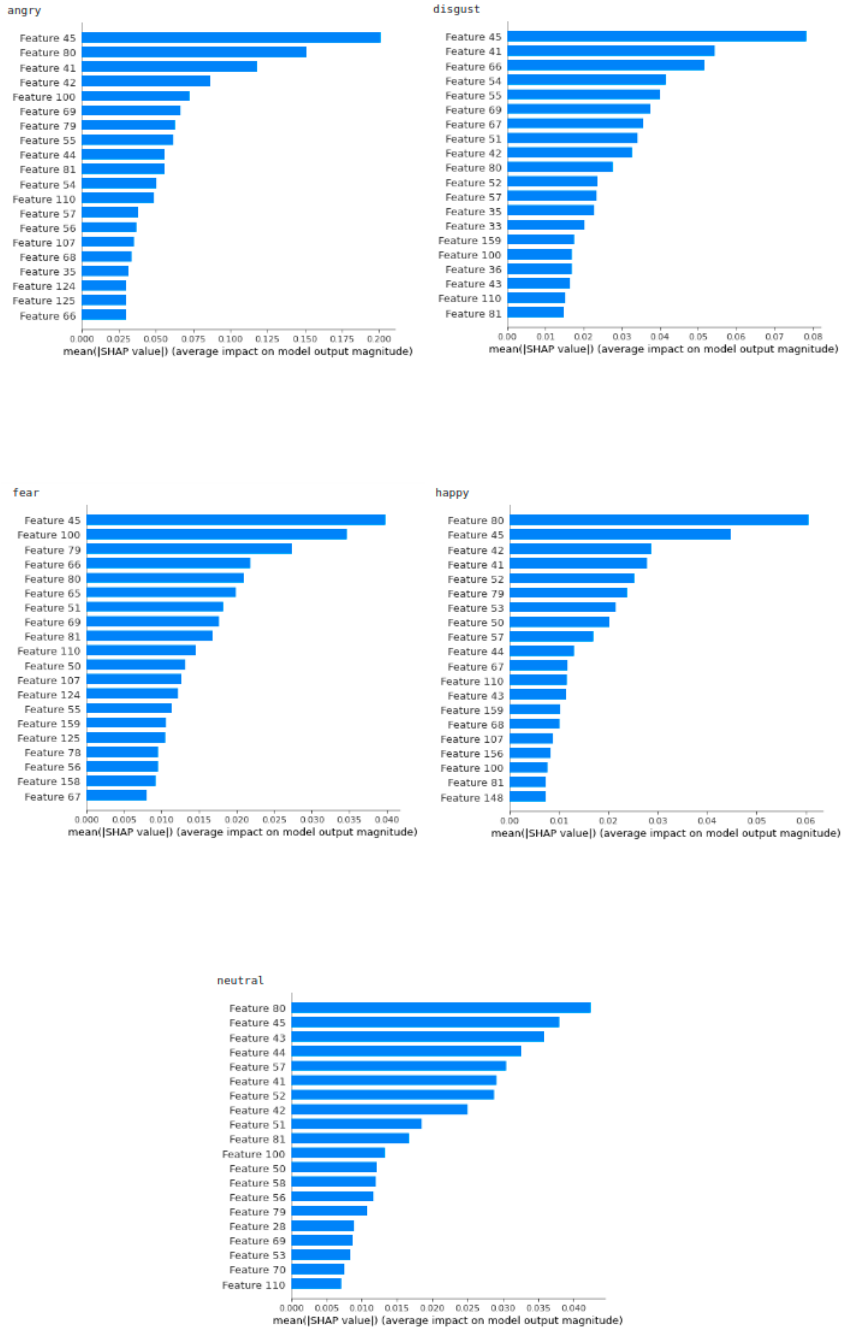

Рис. 30: SHAP обьяснение эмоции angry

Выбираем сэмпл номер 10, через model.predict она определяется как disgust. Рассмотрим вклад каждого признака из категорий. Ответ с наибольшим значением принимается за основной. Здесь основным компонентом выступают признаки эмоции fear, наибольшее значение которой больше 0.2 для Feature 19.

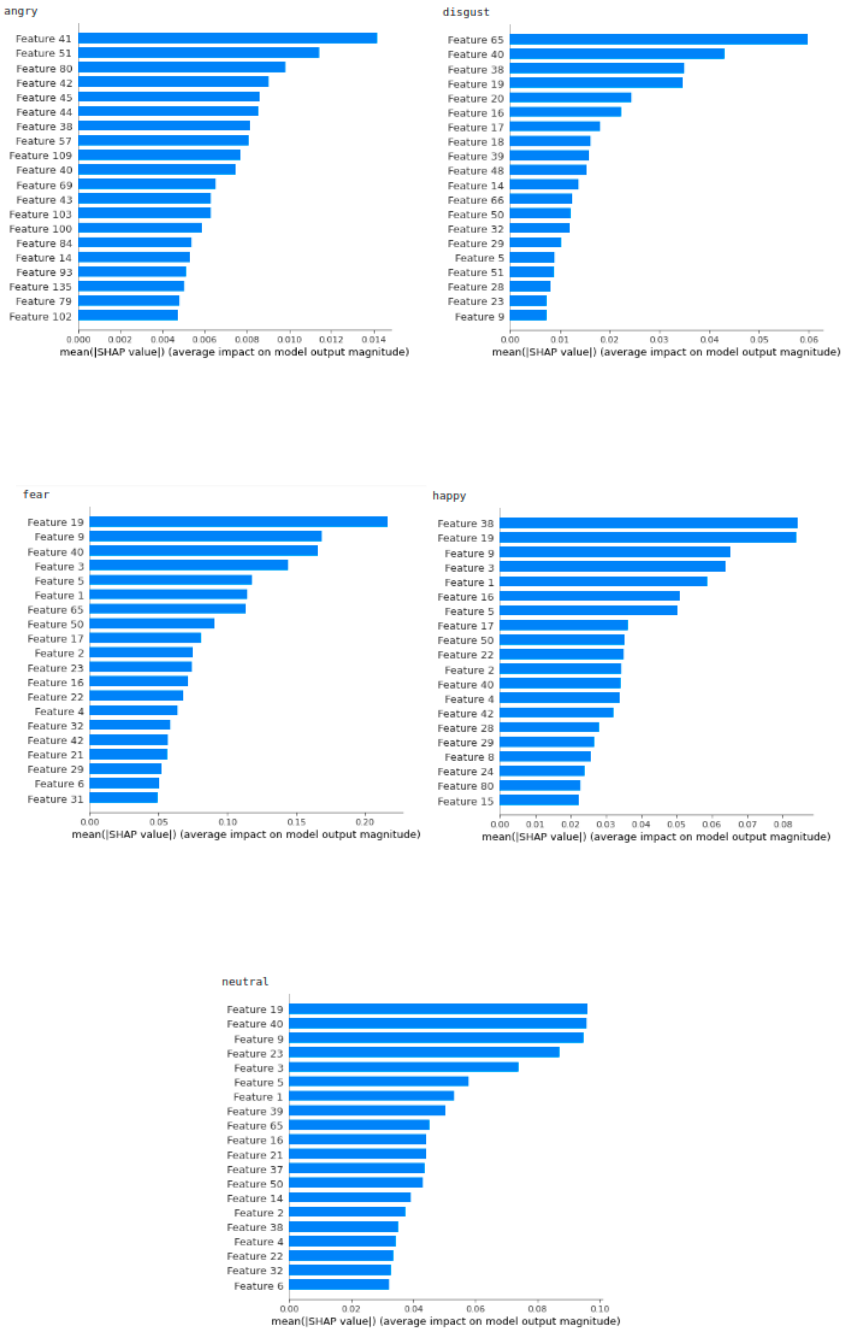

Рис. 31: SHAP обьяснение эмоции disgust

Выбираем сэмпл номер 5, через model.predict она определяется как fear. Рассмотрим вклад каждого признака из категорий. Ответ с наибольшим значением принимается за основной. Здесь основным компонентом выступают признаки эмоции fear, наибольшее значение которой больше 0.2 для Feature 65.

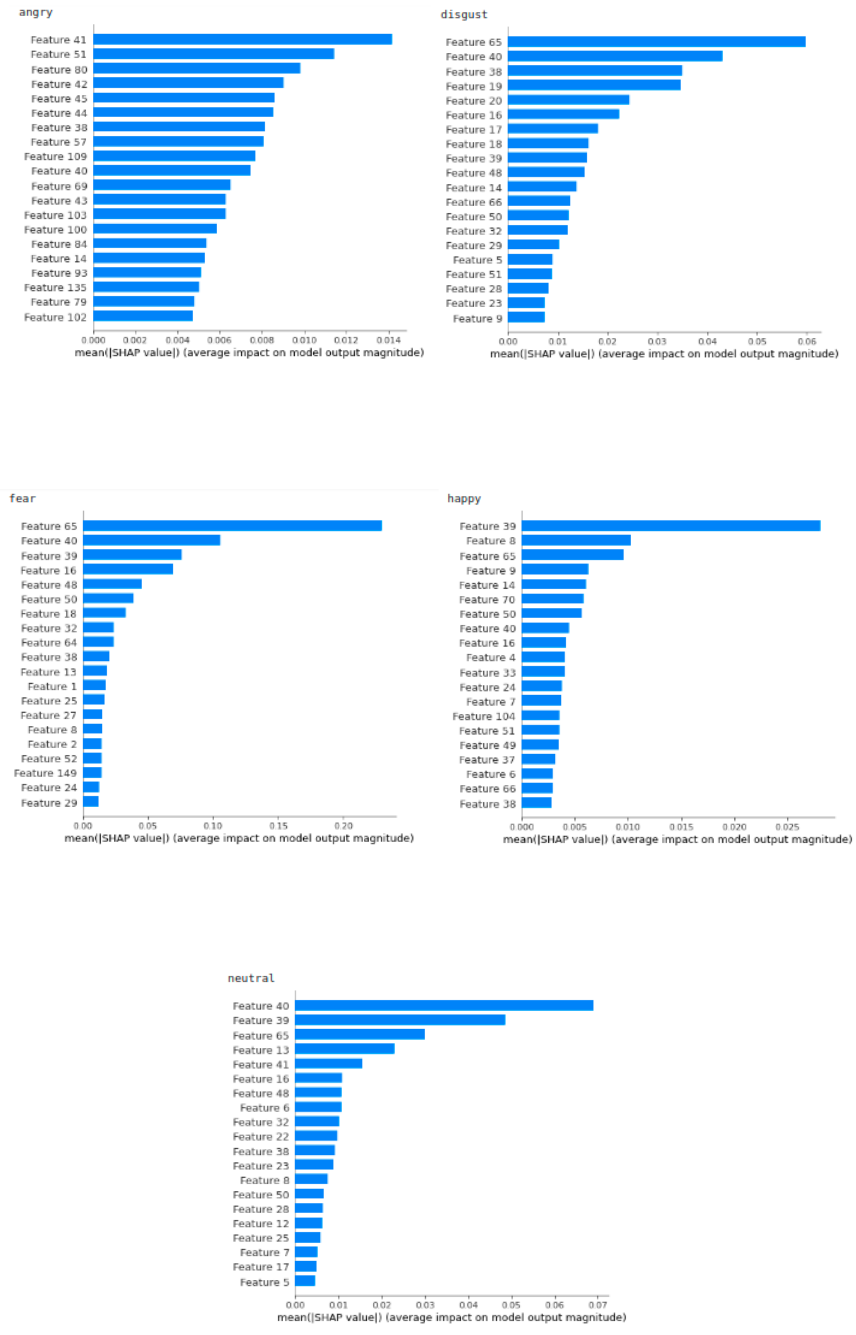

Рис. 32: SHAP обьяснение эмоции fear

Выбираем сэмпл номер 7, через model.predict она определяется как happy. Рассмотрим вклад каждого признака из категорий. Ответ с наибольшим значением принимается за основной. Здесь основным компонентом выступают признаки эмоции fear, наибольшее значение которой больше 0.14 для Feature 39.

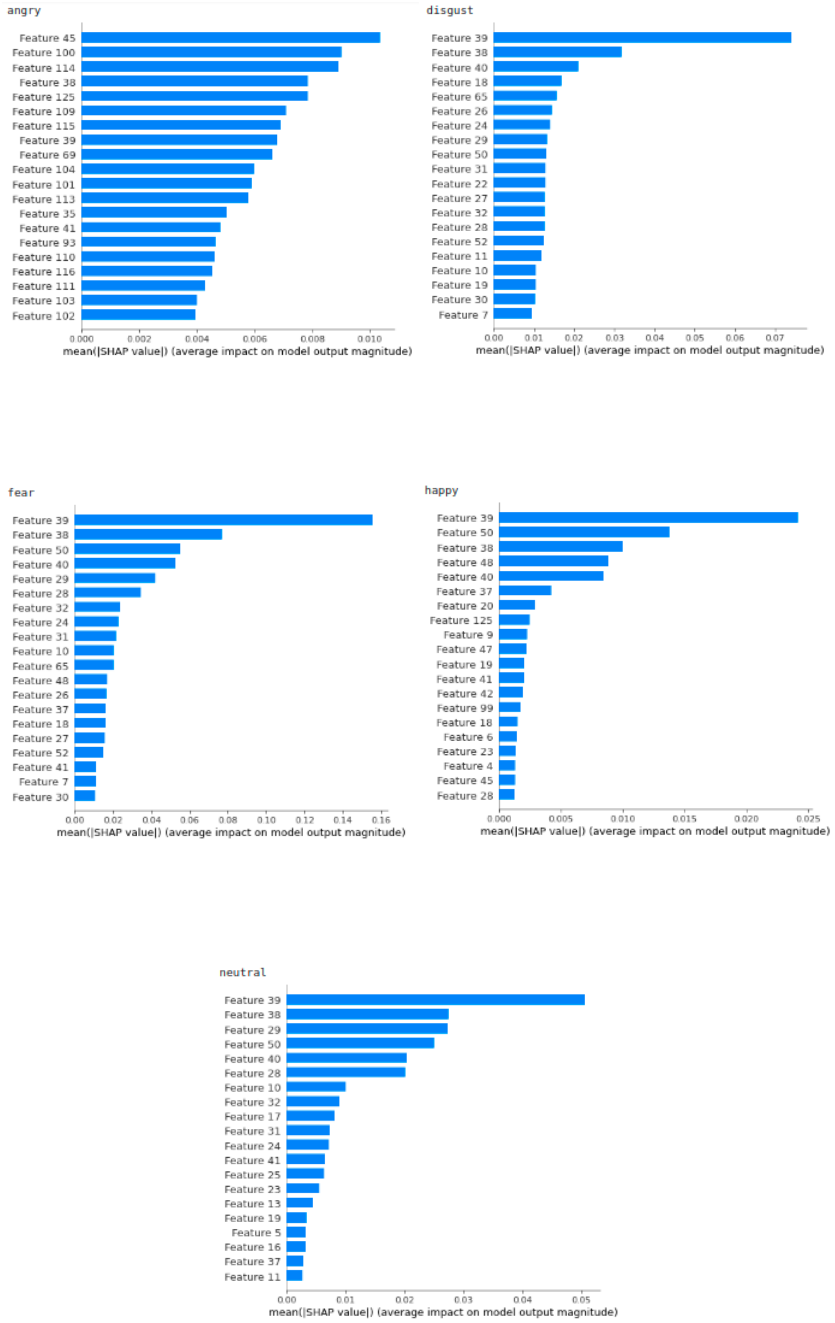

Рис. 33: SHAP обьяснение эмоции happy

Выбираем сэмпл номер 9, через model.predict она определяется как neutral. Рассмотрим вклад каждого признака из категорий. Ответ с наибольшим значением принимается за основной. Здесь основным компонентом выступают признаки эмоции disgust, наибольшее значение которой больше 0.04 для Feature 17.

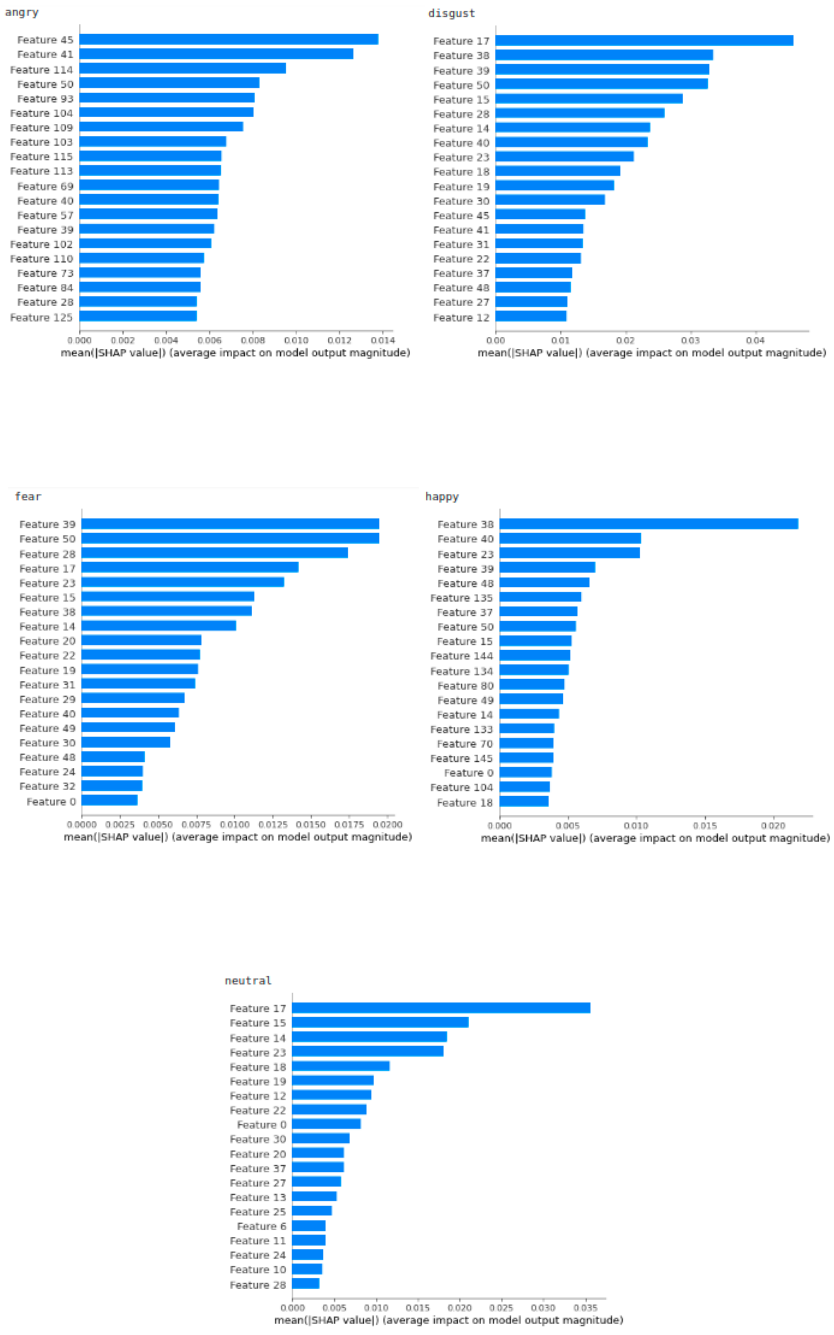

Рис. 34: SHAP обьяснение эмоции neutral

Сравним результаты работы алгоритмов между собой, в таблице будут отражены предсказания эмоций через model.predict и эмоции которые больше всего повлияли на результат по версии алгоритмов SHAP и LIME.

Emotion (model.predict)	LIME	SHAP
angry(11)	disgust	angry
disgust(10)	disgust	fear
fear(5)	happy	fear
happy(7)	happy	fear
neutral(9)	neutral	disgust

Рис. 35: Сравнение результатов алгоритмов

Желтым отмечены результаты которые совпали с предсказаниями model.predict. Сравнение показывает что лучше справляется с отображением признаков аудиовизуальных данных алгоритм LIME. В свою очередь алгоритм SHAP смог точно выявить признаки отвечающие за эмоции с которыми не справился LIME.

Таким образом можно определить какие Features отвечают за конкретную эмоцию, в той или иной степени в зависимости от веса признака:

Emotion (model.predict)	LIME	SHAP
angry(11)	-	Features: 45,80,41,42,100,69,79, 55, 44,81,54,110,57,56,107,68,35,124,125,66
disgust(10)	Features: 31,25,100,20,155,103,136,88	-
fear(5)	-	Features: 65,40,39,16,48,50, 18,32,64,38,13,1,25,27,8,2,52,149,24,29
happy(7)	Features: 103,6	-
neutral(9)	Features: 31,103,115,35,14,7,74,123	-

Рис. 36: Features для конкретных эмоций

На данном этапе исследования, можно заключить, что имеет смысл использовать алгоритмы не раздельно, а вместе для более точного анализа предсказаний.

Выводы

По результатам данного исследования, были проведены следующие этапы работы:

1)Найдены и объединены в единый набор аудиоданных, датасеты с голосами разных актеров, различной возратной группы и этнического происхождения.

2)Построена модель, которая включает в себя:

- выделение признаков из аудиофайлов;
- моделирование нейронной сети и ее обучение;
- использование model.predict для предсказания значений.

3)Работа модели использована для предсказания эмоций настоящих пациентов с психологическими отклонениями.

4) К аудиоданным применены алгоритмы XAI (LIME и SHAP). Описаны результаты применения алгоритма для каждой эмоции: angry, disgust, fear, happy, neutral. На основе выделенных признаков, провелен анализ и выявлены основные features для различных эмоций.

Заключение

Построение моделей и умение делать правильные предсказания являются важнейшими задачами современного мира. В долгосрочной перспективе нынешние инициативы по искусственному интеллекту состоят в том, чтобы внести свой вклад в разработку, проектрирование и внедрение ориентирвоанных на человека систем ИИ. Агенты которых сотрудничали бы с человеком интерпретирумым и объяснимым образом, чтобы обеспечить принятие справедливых решений, а также их прозрачность. Для достижения этой цели, одними из инструментов реализации предлагаются алгоритмы XAI, а именно SHAP и LIME. Предполагаемая предметная область для использования инструмента медицинская диагностика.

Первая глава посвящена обзору наборов аудиоданных и выделению признаков из них.

Вторая глава описывает общую структуру работы нейронной сети

Третья глава формализует методы XAI, далее демонстрируется практическая часть применения двух алгоритмов обьяснения SHAP и LIME.

LIME - отличный инструмент для объяснения того, что делают классификаторы (или модели) машинного обучения.Он не зависит от модели, использует простые и понятные идеи и не требует больших усилий для запуска.Как всегда, даже при использовании LIME важно правильно интерпретировать вывод.

SHAP - подходит для понимания любых моделей машинного обучения, а также имеет несколько других плюсов, например, может обьяснить как индивидуальные результаты, так и общее решение, а также имеет хорошую теоритическую основу из теории игр.

На пути к справедливым и прозрачным моделям, основанным на ИИ, остатется много открытых вопросов, например как оценка рисков, избегание алгоритмических и социальных предубеждений. Алгоритмы обьяснимого ИИ должны нести ответсвенность перед заинтересованными сторонами за свое решение, а также оно должно подчиняться анализу, в котором должна быть понятна степень прозрачности алгоритма.

Обьяснимые методы ИИ также могут быть использованы для выявле-

ния потенциальных рисков, например проверка модели на справедливость.

Будущее взаимодействие между людьми и машинами имеет важное значение, для адаптивных обьяснимых моделей важно участие опытных исследователей из предметной области. Коллективный опыт знания из различных областей (например, здравоохранение, финансы, медицина, безопасность, оборона) могут способствовать дальнейшему развитию исследований ИИ, ориентированных на человека. Таким образом, существует необходимость в растущем интересе к междисциплинарным исследованиям для продвижения ориентированного на человека искусственного интеллекта, а также XAI в критически важных исследованиях для различных областей.

Список литературы

[1] Arun Das, Paul Rad«Opportunities and Challenges in Explainable Artificial Intelligence (XAI): A Survey ». arXiv:2006.11371, 2020

[2] Katharina Weitz, Dominik Schiller, Ruben Schlagowski, Tobias Huber, Elisabeth Andre «"Let me explain!": exploring the potential of virtual agents in explainable AI interaction design ». Journal on Multimodal User Interfaces volume 15, pages 87–98 (2021)

[3] Alejandro Barredo Arrieta, Natalia Diaz-Rodriguez и др.«Explainable Artificial Intelligence (XAI): Concepts, Taxonomies, Opportunities and Challenges toward Responsible AI ». arXiv:1910.10045, 2019

[4] Cao H, Cooper DG, Keutmann MK, Gur RC, Nenkova A, Verma R. «CREMA-D: Crowd-sourced Emotional Multimodal Actors Dataset.».IEEE transactions on affective computing. 2014;5(4):377-390. doi:10.1109/TAFFC.2014.2336244.

[5] Livingstone S.R., Russo F.A. «The Ryerson Audio-Visual Database of Emotional Speech and Song (RAVDESS): A dynamic, multimodal set of facial and vocal expressions in North American English. ». PLoS ONE 13(5): e0196391. https://doi.org/10.1371/journal.pone.0196391.

[6] «Surrey Audio-Visual Expressed Emotion (SAVEE) Database». http://kahlan.eps.surrey.ac.uk/savee/

[7] Kate Dupuis, M. Kathleen Pichora-Fuller «Toronto emotional speech set (TESS)». University of Toronto, Psychology Department, 2010., https://tspace.library.utoronto.ca/handle/1807/24487

[8] The University of Southern California Institute for Creative Technologies «DAIC-WOZ Database Description». https://dcapswoz.ict.usc.edu/

[9] »IJCAI-PRICAI2020 ». https://ijcai20.org/

[10] B.G. Buchanan., E.H. Shortliffe «Rule-based expert systems: The MYCIN experiments of the stanford heuristic programming project»., 1984

[11] Erico Tjoa, Cuntai Guan«A Survey on Explainable Artificial Intelligence (XAI): Towards Medical XAI ». arXiv:1907.07374, 2020

[12] M. T. Ribeiro, S. Singh, and C. Guestrin«Why Should I Trust You?». arXiv:1602.04938, 2016

[13] «Librosa documentation.». https://librosa.org/doc/latest/index.html

[14] Theodoros Giannakopoulos, Aggelos Pikrakis «Introduction to Audio Analysis». 2014.

[15] Meinard Miller Stefan Balke «Short-Time Fourier Transform and Chroma Features ». Университет Эрлангена-Нюрнберга им. Фридриха-Александра., 2015.

[16] Kannan Venkataramanan, Haresh Rengaraj Rajamohan «Emotion Recognition from Speech ». arXiv:1912.10458, 2019.

[17] Mohammad Ahsan., Madhu Kumari «Physical Features Based Speech Emotion Recognition Using Predictive Classification». April 2016., International Journal of Computer Science and Information Technology 8(2):63-74 DOI:10.5121/ijcsit.2016.8205

[18] Roman A. Solovyev, Maxim Vakhrushev, Alexander Radionov, Vladimir Aliev, Alexey A. Shvets «Deep Learning Approaches for Understanding Simple Speech Commands». arXiv:1810.02364., 2018.

[19] «Classification Report documentation». https://www.scikit-yb.org/en/latest/api/classifier/classification_report.html

[20] Shane T. Mueller, Robert R. Hoffman, William Clancey, Abigail Emrey, Gary Klein «Explanation in Human-AI Systems: A Literature Meta-Review, Synopsis of Key Ideas and Publications, and Bibliography for Explainable AI ».arXiv:1902.01876, 2019.

[21] Jerome H. Friedman, Bogdan E. Popescu «Predictive learning via rule ensembles». The Annals of Applied Statistics, vol. 2, no. 3, pp. 916– 954, 2008.

[22] L. S. Shapley «A value for n-person games». Contributions to the Theory of Games, vol. 2, no. 28, pp. 307–317, 1953.

[23] Christoph Molnar «Interpretable Machine Learning: A Guide for Making Black Box Models Explainable.». Springer; 1st ed. 2018.

[24] S. M. Lundberg and S. I. Lee«A unified approach to interpreting model predictions »in Advances in Neural Information Processing Systems, 2017, pp. 4765–4774

[25] C. Molnar «AInterpretable Machine Learning »Lulu. com, 2020

[26] L. Antwarg, B. Shapira, and L. Rokach «Explaining anomalies detected by autoencoders using shap »arXiv:1903.02407, 2019.

[27] M. Sundararajan and A. Najmi«The many shapley values for model explanation»arXiv:1908.08474, 2019.

[28] K. Aas, M. Jullum, and A. Løland «Explaining individual predictions when features are dependent: More accurate approximations to shapley values »arXiv:1903.10464, 2019

[29] S. M. Lundberg, G. Erion, H. Chen, A. DeGrave, J. M. Prutkin, B. Nair, R. Katz, J. Himmelfarb, N. Bansal «From local explanations to global understanding with explainable ai for trees »Nature machine intelligence, vol. 2, no. 1, pp. 2522–5839, 2020.

[30] M. Vega Garcia and J. L. Aznarte «Shapley additive explanations for NO2 forecasting »Ecol. Inform., vol. 56, p. 101039, Mar 2020.

[31] S. Mishra, B. L. Sturm, and S. Dixon«Local Interpretable Model-Agnostic Explanations for Music Content Analysis». Proc. 18th Int. Soc. Music Inf. Retr. Conf. ISMIR 2017, 2017, pp. 537–543.

[32] T. Peltola«Local interpretable model-agnostic explanations of bayesian predictive models via kullback-leibler projections». arXiv:1810.02678, 2018.

[33] M. Rehman Zafar and N. Mefraz Khan«Dlime: A deterministic local interpretable model-agnostic explanations approach for computer-aided diagnosis systems,». arXiv:1906.10263, 2019.

[34] S. Bramhall, H. Horn, M. Tieu, and N. Lohia«Qlime-a quadratic local interpretable model-agnostic explanation approach». MU Data Science Review, vol. 3, no. 1, p. 4, 2020.

[35] S. Shi, X. Zhang, and W. Fan«A modified perturbed sampling method for local interpretable model-agnostic explanation». arXiv:2002.07434, 2020.

Приложение

```
CNN for speech recognition and XAI algorithms

import pandas as pd
import numpy as np

import os
import sys

import librosa
import librosa.display
import seaborn as sns
import matplotlib.pyplot as plt

from sklearn.preprocessing import StandardScaler, OneHotEncoder
from sklearn.metrics import confusion_matrix, classification_report
from sklearn.model_selection import train_test_split

from IPython.display import Audio

import keras
from keras.callbacks import ReduceLROnPlateau
from keras.models import Sequential
from keras.layers import Dense, Conv1D, MaxPooling1D, Flatten,
Dropout, BatchNormalization
from keras.utils import np_utils, to_categorical
from keras.callbacks import ModelCheckpoint

import warnings
if not sys.warnoptions:
    warnings.simplefilter("ignore")
warnings.filterwarnings("ignore", category=DeprecationWarning)
```

```
from google.colab import drive

Ravdess = "/content/drive/MyDrive/actors/"
Crema = "/content/drive/MyDrive/AudioWAV/"
Tess = "/content/drive/MyDrive/TESS Toronto emotional speech
set data/"
Savee = "/content/drive/MyDrive/ALL/"

ravdess_directory_list = os.listdir(Ravdess)

file_emotion = []
file_path = []
for dir in ravdess_directory_list:

    actor = os.listdir(Ravdess + dir)
    for file in actor:
        part = file.split('.')[0]
        part = part.split('-')

        try:
          file_emotion.append(int(part[2]))
          file_path.append(Ravdess + dir + '/' + file)
        except IndexError:
          pass

# dataframe for emotion of files
emotion_df = pd.DataFrame(file_emotion, columns=['Emotions'])

# dataframe for path of files.
path_df = pd.DataFrame(file_path, columns=['Path'])
Ravdess_df = pd.concat([emotion_df, path_df], axis=1)
```

```
# changing integers to actual emotions.
Ravdess_df.Emotions.replace({1:'neutral', 2:'calm', 3:'happy',
4:'sad', 5:'angry', 6:'fear', 7:'disgust', 8:'surprise'}
inplace=True)

crema_directory_list = os.listdir(Crema)

file_emotion = []
file_path = []

for file in crema_directory_list:
    # storing file paths
    file_path.append(Crema + file)
    # storing file emotions
    part=file.split('_')
    if part[2] == 'SAD':
        file_emotion.append('sad')
    elif part[2] == 'ANG':
        file_emotion.append('angry')
    elif part[2] == 'DIS':
        file_emotion.append('disgust')
    elif part[2] == 'FEA':
        file_emotion.append('fear')
    elif part[2] == 'HAP':
        file_emotion.append('happy')
    elif part[2] == 'NEU':
        file_emotion.append('neutral')
    else:
        file_emotion.append('Unknown')

# dataframe for emotion of files
emotion_df = pd.DataFrame(file_emotion, columns=['Emotions'])
```

```
# dataframe for path of files.
path_df = pd.DataFrame(file_path, columns=['Path'])
Crema_df = pd.concat([emotion_df, path_df], axis=1)

tess_directory_list = os.listdir(Tess)

file_emotion = []
file_path = []

for dir in tess_directory_list:
    directories = os.listdir(Tess + dir)
    for file in directories:
        part = file.split('.')[0]
        try:
          part = part.split('_')[2]
          if part=='ps':
              file_emotion.append('surprise')
          else:
              file_emotion.append(part)
          file_path.append(Tess + dir + '/' + file)
        except IndexError:
          pass

# dataframe for emotion of files
emotion_df = pd.DataFrame(file_emotion, columns=['Emotions'])

# dataframe for path of files.
path_df = pd.DataFrame(file_path, columns=['Path'])
Tess_df = pd.concat([emotion_df, path_df], axis=1)

savee_directory_list = os.listdir(Savee)

file_emotion = []
```

```
file_path = []

for file in savee_directory_list:
    file_path.append(Savee + file)
    part = file.split('_')[1]
    ele = part[:-6]
    if ele=='a':
        file_emotion.append('angry')
    elif ele=='d':
        file_emotion.append('disgust')
    elif ele=='f':
        file_emotion.append('fear')
    elif ele=='h':
        file_emotion.append('happy')
    elif ele=='n':
        file_emotion.append('neutral')
    elif ele=='sa':
        file_emotion.append('sad')
    else:
        file_emotion.append('surprise')

# dataframe for emotion of files
emotion_df = pd.DataFrame(file_emotion, columns=['Emotions'])

# dataframe for path of files.
path_df = pd.DataFrame(file_path, columns=['Path'])
Savee_df = pd.concat([emotion_df, path_df], axis=1)
Savee_df.head()

# creating Dataframe using all the 4 dataframes we created so far.
data_path = pd.concat([Ravdess_df, Crema_df, Tess_df, Savee_df],
axis = 0)
data_path.to_csv("data_path.csv",index=False)
```

```
data_path.head()

"""## Data Visualisation and Exploration"""

plt.title('Count of Emotions', size=16)
sns.countplot(data_path.Emotions)
plt.ylabel('Count', size=12)
plt.xlabel('Emotions', size=12)
sns.despine(top=True, right=True, left=False, bottom=False)
plt.show()

def create_waveplot(data, sr, e):
    plt.figure(figsize=(10, 3))
    plt.title('Waveplot for audio with {} emotion'.format(e),
    size=15)
    librosa.display.waveplot(data, sr=sr)
    plt.show()

def create_spectrogram(data, sr, e):
    X = librosa.stft(data)
    Xdb = librosa.amplitude_to_db(abs(X))
    plt.figure(figsize=(12, 3))
    plt.title('Spectrogram for audio with {}
    emotion'.format(e), size=15)
    librosa.display.specshow(Xdb, sr=sr, x_axis='time',
    y_axis='hz')
    #librosa.display.specshow(Xdb, sr=sr, x_axis='time',
    y_axis='log')
    plt.colorbar()

emotion='fear'
path = np.array(data_path.Path[data_path.Emotions==emotion])[1]
data, sampling_rate = librosa.load(path)
```

```
create_waveplot(data, sampling_rate, emotion)
create_spectrogram(data, sampling_rate, emotion)
Audio(path)

emotion='angry'
path = np.array(data_path.Path[data_path.Emotions==emotion])[1]
data, sampling_rate = librosa.load(path)
create_waveplot(data, sampling_rate, emotion)
create_spectrogram(data, sampling_rate, emotion)
Audio(path)

emotion='sad'
path = np.array(data_path.Path[data_path.Emotions==emotion])[1]
data, sampling_rate = librosa.load(path)
create_waveplot(data, sampling_rate, emotion)
create_spectrogram(data, sampling_rate, emotion)
Audio(path)

emotion='happy'
path = np.array(data_path.Path[data_path.Emotions==emotion])[1]
data, sampling_rate = librosa.load(path)
create_waveplot(data, sampling_rate, emotion)
create_spectrogram(data, sampling_rate, emotion)
Audio(path)

"""## Data Augmentation"""

def noise(data):
    noise_amp = 0.035*np.random.uniform()*np.amax(data)
    data = data + noise_amp*np.random.normal(size=data.shape[0])
    return data

def stretch(data, rate=0.8):
```

```
    return librosa.effects.time_stretch(data, rate)

def shift(data):
    shift_range = int(np.random.uniform(low=-5, high = 5)*1000)
    return np.roll(data, shift_range)

def pitch(data, sampling_rate, pitch_factor=0.7):
    return librosa.effects.pitch_shift(data, sampling_rate,
    pitch_factor)

# taking any example and checking for techniques.
path = np.array(data_path.Path)[1]
data, sample_rate = librosa.load(path)

"""#### 1. Simple Audio"""

plt.figure(figsize=(14,4))
librosa.display.waveplot(y=data, sr=sample_rate)
Audio(path)

"""#### 2. Noise Injection"""

x = noise(data)
plt.figure(figsize=(14,4))
librosa.display.waveplot(y=x, sr=sample_rate)
Audio(x, rate=sample_rate)

"""#### 3. Stretching"""

x = stretch(data)
plt.figure(figsize=(14,4))
librosa.display.waveplot(y=x, sr=sample_rate)
Audio(x, rate=sample_rate)
```

```
"""#### 4. Shifting"""

x = shift(data)
plt.figure(figsize=(14,4))
librosa.display.waveplot(y=x, sr=sample_rate)
Audio(x, rate=sample_rate)

"""#### 5. Pitch"""

x = pitch(data, sample_rate)
plt.figure(figsize=(14,4))
librosa.display.waveplot(y=x, sr=sample_rate)
Audio(x, rate=sample_rate)

"""## Feature Extraction

"""

def extract_features(data):
    # ZCR
    result = np.array([])
    zcr = np.mean(librosa.feature.zero_crossing_rate(y=data).T,
    axis=0)
    result=np.hstack((result, zcr)) # stacking horizontally

    # Chroma_stft
    stft = np.abs(librosa.stft(data))
    chroma_stft = np.mean(librosa.feature.chroma_stft(S=stft,
    sr=sample_rate).T, axis=0)
    result = np.hstack((result, chroma_stft))

    # MFCC
```

```
    mfcc = np.mean(librosa.feature.mfcc(y=data, sr=sample_rate).T,
    axis=0)
    result = np.hstack((result, mfcc)) # stacking horizontally

    # Root Mean Square Value
    rms = np.mean(librosa.feature.rms(y=data).T, axis=0)
    result = np.hstack((result, rms)) # stacking horizontally

    # MelSpectogram
    mel = np.mean(librosa.feature.melspectrogram(y=data,
    sr=sample_rate).T,
    axis=0)
    result = np.hstack((result, mel)) # stacking horizontally

    return result

def get_features(path):
    data, sample_rate = librosa.load(path, duration=2.5, offset=0.6)

    # without augmentation
    res1 = extract_features(data)
    result = np.array(res1)

    # data with noise

    noise_data = noise(data)
    res2 = extract_features(noise_data)
    result = np.vstack((result, res2)) # stacking vertically

    # data with stretching and pitching

    new_data = stretch(data)
    data_stretch_pitch = pitch(new_data, sample_rate)
```

```
    res3 = extract_features(data_stretch_pitch)
    result = np.vstack((result, res3)) # stacking vertically

    return result

X, Y = [], []
banned = ["sad", "sad (1)", "surprised", "surprise", "calm"]
for path, emotion in zip(data_path.Path[:100], data_path.Emotions):
    if emotion not in banned:
      try:
        # X.append(get_features(path))
        # Y.append(emotion)
        print(emotion)
        feature = get_features(path)
        for i, ele in enumerate(feature):
             X.append(ele)
             print(i, 'of', len(feature))
             augmentation techniques on each audio file.
             Y.append(emotion)
      except:
        pass

len(X), len(Y), data_path.Path.shape

Features = pd.DataFrame(X)
Features['labels'] = Y
Features.to_csv('features.csv', index=False)
Features.head()

"""## Data Preparation

X = Features.iloc[: ,:-1].values
Y = Features['labels'].values
```

```
encoder = OneHotEncoder()
Y_OLD = Y
Y = encoder.fit_transform(np.array(Y).reshape(-1,1)).toarray()

ys = zip(Y_OLD, list(Y))

mapping = {}
for emotion, code in ys:
  mapping[emotion] = code.argmax()

mapping

x_train, x_test, y_train, y_test = train_test_split
(X, Y,test_size=0.1, random_state=0, shuffle=True)
x_train.shape, y_train.shape, x_test.shape, y_test.shape

scaler = StandardScaler()
x_train = scaler.fit_transform(x_train)
x_test = scaler.transform(x_test)
x_train.shape, y_train.shape, x_test.shape, y_test.shape

x_train = np.expand_dims(x_train, axis=2)
x_test = np.expand_dims(x_test, axis=2)
x_train.shape, y_train.shape, x_test.shape, y_test.shape

from tensorflow.keras import layers
from tensorflow.keras import regularizers
from keras.layers.normalization import BatchNormalization

"""## Modelling"""

model=Sequential()
```

```
model.add(Conv1D(256, kernel_size=28, strides = 1, padding='same',
activation='relu', input_shape=(x_train.shape[1], 1)))
model.add(MaxPooling1D(pool_size=5, strides = 2, padding = 'same'))
model.add(Dropout(0.3))

model.add(Conv1D(256, kernel_size=28, strides=1, dilation_rate = 2,
padding='same', activation='tanh'))
model.add(MaxPooling1D(pool_size=5, strides = 2, padding = 'same'))
model.add(Dropout(0.3))

model.add(Conv1D(128, kernel_size=20, strides=1, dilation_rate = 2,
padding='same', activation='relu'))
model.add(MaxPooling1D(pool_size=5, strides = 2, padding = 'same'))
model.add(Dense(units=64,
kernel_regularizer=keras.regularizers.l1_l2(l1=0.0001, l2=0.0001),
activation='tanh'))
model.add(Dropout(0.3))

model.add(Conv1D(64, kernel_size=20, strides=1,
kernel_regularizer=keras.regularizers.l1_l2(l1=0.0001, l2=0.0001),
padding='same', activation='relu'))
model.add(MaxPooling1D(pool_size=5, strides = 2, padding = 'same'))
model.add(Dropout(0.3))

model.add(Flatten())
model.add(Dense(units=64,
kernel_regularizer=keras.regularizers.l1_l2(l1=0.0001, l2=0.0001),
activation='tanh'))
model.add(Dropout(0.3))

model.add(Dense(units=5, activation='softmax'))
model.compile(optimizer = 'adamax' ,
loss = 'categorical_crossentropy', metrics = ['accuracy'])
```

```
model.summary()

rlrp = ReduceLROnPlateau(monitor='loss', factor=0.4, verbose=0,
patience=2, min_lr=0.0000001)
history=model.fit(x_train, y_train, batch_size=64, epochs=200,
validation_data=(x_test, y_test), callbacks=[rlrp])

print("Accuracy of our model on test data : " ,
model.evaluate(x_test,y_test)[1]*100 , "%")

epochs = [i for i in range(200)]
fig , ax = plt.subplots(1,2)
train_acc = history.history['accuracy']
train_loss = history.history['loss']
test_acc = history.history['val_accuracy']
test_loss = history.history['val_loss']

fig.set_size_inches(20,6)
ax[0].plot(epochs , train_loss , label = 'Training Loss')
ax[0].plot(epochs , test_loss , label = 'Testing Loss')
ax[0].set_title('Training & Testing Loss')
ax[0].legend()
ax[0].set_xlabel("Epochs")

ax[1].plot(epochs , train_acc , label = 'Training Accuracy')
ax[1].plot(epochs , test_acc , label = 'Testing Accuracy')
ax[1].set_title('Training & Testing Accuracy')
ax[1].legend()
ax[1].set_xlabel("Epochs")
plt.show()

pred_test = model.predict(x_test)
```

```
y_pred = encoder.inverse_transform(pred_test)
Y_test = y_test
y_test = encoder.inverse_transform(y_test)

df = pd.DataFrame(columns=['Predicted Labels', 'Actual Labels'])
df['Predicted Labels'] = y_pred.flatten()
df['Actual Labels'] = y_test.flatten()

df.head(15)

cm = confusion_matrix(y_test, y_pred)
plt.figure(figsize = (12, 10))
cm = pd.DataFrame(cm , index = [i for i in encoder.categories_] ,
columns = [i for i in encoder.categories_])
sns.heatmap(cm, linecolor='white', cmap='Blues', linewidth=1,
annot=True, fmt='')
plt.title('Confusion Matrix', size=20)
plt.xlabel('Predicted Labels', size=14)
plt.ylabel('Actual Labels', size=14)
plt.show()

print(classification_report(y_test, y_pred))

"""##SHAP"""

import shap
import tensorflow as tf
tf.compat.v1.disable_v2_behavior()
import numpy as np

model = tf.keras.models.load_model('/content/drive/MyDrive
/saved_models/m.pb')
```

```
background = x_train[np.random.choice(x_train.shape[0], 100,
replace=False)]

e = shap.DeepExplainer(model, x_test)

shap_values = e.shap_values(x_test)

shap.initjs()

reversed_mapping = {value: key for key, value in mapping.items()}

for i in range(5):
  print(reversed_mapping[i])
  shap.summary_plot(shap_values[i][9].transpose(),
  x_test[9].transpose(), plot_size=(7, 7), plot_type='bar')

"""##LIME"""

import lime
import lime.lime_tabular

expl20 = lime.lime_tabular.LimeTabularExplainer
(x_test.squeeze(axis=2), class_names=['angry', 'disgust',
'fear', 'happy', 'neutral'])

exp21=expl20.explain_instance(x_test[9].squeeze(axis=1),
lambda x: model.predict(np.expand_dims(x, axis=2)),
num_features=8,top_labels=5)
exp21.show_in_notebook()
```