

!!! EMOTIONS

January 28, 2024

This chat contains files or images produced by Advanced Data Analysis which are not yet visible in Shared Chats.

 **Anonymous**



data_clear.csv
Spreadsheet



dtat_10_part.csv
Spreadsheet

7) столбцы `facets_fields` содержат данные из 42-мерного пространства Facial Action Coding System (FACS) // action units
8) столбцы `seven_fields` содержат данные из 6-мерного пространства Пола Экмана (6 эмоций + Neutral)
9) столбцы `pa_fields` содержат данные из 2-мерного пространства Valence-Arousal (или Pleasure-Arousal). Это пространство является по координата Dominance опущена.

1) На основе данных, описанных - построить регрессионные модели, которые будут учитывать формульные зависимости из п. 5). Входные `facets_fields` (42 столбца), выходные - из столбцов `seven_fields` (7 столбцов) // преобразование 42 -> 7
2) Обратные регрессионные модели: Входные данные (7 столбцов) - из столбцов `seven_fields`, выходные - из столбцов `facets_fields` (42 столбца)
3) Проверить гипотезу из п. 6) с помощью PCA. При необходимости, добавить к PCA преобразованию регрессионную зависимость, что учителем: Входные данные (7 столбцов) - из столбцов `seven_fields`, выходные - из столбцов `pa_fields` (2 столбца) // преобразование 7 -> 2

`df_clear` - точнее выражает чистые эмоции (только одна эмоция без других).

`df` - могут быть сразу несколько эмоций (смесь)

`df_clear` - это подмножество от `df`

данные разделены через `\t`

короче сейчас я хочу провести анализ PCA. взять эти 7 признаков и построить для 2ух компонент, какое количество данных описывает вывод.

Далее построить матрицу корреляций и посмотреть как зависят последние 2 признака от семи.

Index_ Video_Time Neutral Happy Sad Angry Surprised Scared Disgusted Valence Arousal

Valence Arousal - 2 признака в которые нужно перевести.

это первые столбцы.

реализуй код. для чистых данных и для не чистых. чтобы сравнить. Сделай выводы.

Вся это подготовка для того чтобы потом обучить разные модели для восстановления 7 признаков из двух.

Пиши код не изобретая велосипед. пользуйся библиотеками. Пиши как будто в ноутбуке юпитере.

 **ChatGPT**

Для начала мы загрузим и проанализируем предоставленные данные из файлов `data_clear.csv` и `dtat_10_part.csv`. Это позволит нам провести главных компонент (PCA) для семи эмоциональных признаков (Neutral, Happy, Sad, Angry, Surprised, Scared, Disgusted) и исследовать, как они описаны с помощью двух компонент (Valence, Arousal). После этого мы построим матрицу корреляций для этих признаков.

Процесс будет следующим:

1. Загрузка и первичный анализ данных из обоих файлов.
2. Применение PCA для семи признаков эмоций для обоих наборов данных.
3. Анализ объясняемой дисперсии для двух главных компонент.
4. Построение матрицы корреляций для оценки зависимости между семью эмоциональными признаками и двумя признаками (Valence и Arousal).
5. Сравнение результатов для чистых и нечистых данных.

Давайте начнем с первого шага - загрузки и первичного анализа данных.

Загруженные данные представляют собой два набора: `df_clear` для чистых данных и `df_mixed` для нечистых (смешанных) данных. Емкость и характеристики обоих наборов:

Чистые данные (`df_clear`):

- Столбцы: `Index_`, `Neutral`, `Happy`, `Sad`, `Angry`, `Surprised`, `Scared`, `Disgusted`, `Valence`, `Arousal`.
- Примерные значения для первых пяти записей указывают на наличие долей эмоций (Neutral, Happy и т.д.) и двух основных компонент

Нечистые данные (df_mixed):

- Столбцы: Начинаются с Index_, Video_Time, Neutral и далее аналогичные эмоциональные категории, а также включают дополнительные Facial Action Coding System (FACS).
 - Примерные значения показывают, что данные содержат значительно больше информации, включая данные FACS.
- Теперь давайте проведем анализ PCA для обоих наборов данных. Нам нужно извлечь из каждого набора данных только те столбцы, ко эмоциональным признакам (Neutral, Happy, Sad, Angry, Surprised, Scared, Disgusted), и применить PCA для сокращения до двух компонент этого мы проанализируем, какая доля вариации данных описывается этими двумя компонентами. [\[>-\]](#)

! Error analyzing ▾

Похоже, возникла проблема с обработкой нечистых данных (df_mixed). Это связано с тем, что некоторые столбцы содержат нечисловые значения ('Closed', 'Open', 'Neutral'). Для корректного применения PCA необходимо, чтобы все данные были в числовом формате.

Чтобы решить эту проблему, мы можем преобразовать эти нечисловые значения в числа. Возможно, это будет означать преобразование числовые, возможно, с помощью метода "one-hot encoding" или другого подхода в зависимости от природы данных.

Давайте сначала проверим уникальные значения в этих столбцах и затем решим, как лучше их обработать. После этого мы сможем по

Из нечистых данных (df_mixed) видно, что столбцы 'Neutral', 'Happy', 'Sad', 'Angry', 'Surprised' содержат категориальные значения, в то время как столбцы 'Closed', 'Open', 'Scared' содержат числовые значения. Для 'Scared', мы видим одно уникальное значение 'unknown person', что может указывать на отсутствие данных.

Для применения PCA, мы можем преобразовать категориальные данные в числовые с помощью подхода one-hot encoding. Однако, учитывая отсутствие вариативности, мы можем исключить его из анализа. Давайте выполним это преобразование и повторим анализ PCA для нечистых данных 'Scared'. Для чистых данных мы уже выполнили PCA без ошибок, так что повторять это не нужно. [\[>-\]](#)

✕ Stopped analyzing ▾

👤 **Anonymous**

Давай пока работает с clean.

короче нужно теперь реализовать код который выводит графи объяснения от количеств компонент

🧠 **ChatGPT**

! Error analyzing ▾

Кажется, я забыл импортировать библиотеку NumPy для использования функции cumsum. Позвольте мне исправить это и повторить вы



Image output is not supported in a shared chat

На графике показано, как изменяется накопленная объясненная дисперсия в зависимости от количества компонент в анализе главных компонент чистых данных. Из этого графика можно сделать вывод о том, сколько компонент необходимо, чтобы объяснить значительную часть в

Далее, давайте построим матрицу корреляций для оценки зависимости между семью эмоциональными признаками и двумя признаками Valence и Arousal. Этот анализ поможет нам понять, как эти две переменные связаны с эмоциональными состояниями. [\[>-\]](#)



Image output is not supported in a shared chat

На представленной матрице корреляций можно увидеть, как эмоциональные признаки (Neutral, Happy, Sad, Angry, Surprised, Scared, Disgusted) связаны с переменными Valence и Arousal из чистых данных (df_clean). Цвета на графике показывают силу корреляции между переменными: более теплые цвета указывают на положительную корреляцию, а более холодные — на отрицательную.

Этот анализ позволяет нам понять взаимосвязи между различными эмоциональными состояниями и их проекцией на двумерное пространство. Это может быть важно для последующего построения регрессионных моделей для восстановления этих эмоций из значений Valence и Arousal.



Anonymous

давай теперь напишем код обучения из 2ух восстановление 7.

Нужно написать пайплайн сплита, потом использования многочисленных моделей сразу, чтобы сравнить. Так же возможность проход чтобы выбрать лучший (если есть библиотеки которые позволяют сразу применить разные модели используй, чтобы самой не писать) Так же подумай какую метрику использовать. Потому что мы предсказываем непрерывное значение от 0 до 1



ChatGPT

Для построения моделей регрессии, которые будут восстанавливать семь эмоциональных состояний из двух признаков (Valence и Aro использовать следующий подход:

1. **Подготовка данных:** Разделение данных на тренировочный и тестовый наборы.
2. **Выбор моделей для сравнения:** Использование различных типов регрессионных моделей.
3. **Настройка гиперпараметров:** Применение методов для автоматического подбора оптимальных гиперпараметров.
4. **Выбор метрики оценки:** Поскольку мы предсказываем непрерывные значения от 0 до 1, подходящей метрикой может быть, например, ошибка (MSE) или средняя абсолютная ошибка (MAE).

Для автоматизации процесса и применения нескольких моделей одновременно, мы можем использовать библиотеку, такую как `scik`: предоставляет удобные инструменты для этого, включая `Pipeline` и `GridSearchCV` или `RandomizedSearchCV` для настройки гиперпараметров. Давайте напишем код для реализации этой задачи. Поскольку для обучения моделей может потребоваться значительное количество времени, который описывает процесс, но не буду выполнять его здесь. Вы можете выполнить его в своей среде.

Я создал код, который использует различные модели регрессии для восстановления семи эмоциональных состояний из двух признаков. В процессе были использованы следующие модели: линейная регрессия, Ridge, Lasso, случайный лес, метод опорных векторов (SVM) и k-ближайших соседей. Для оптимизации гиперпараметров использовался `GridSearchCV`.

В результате лучшей оказалась модель случайного леса (`RandomForestRegressor`) с 10 деревьями. Средняя квадратичная ошибка (MSE) приблизительно 0.0092, а средняя абсолютная ошибка (MAE) — около 0.0171. Эти метрики могут служить ориентиром для оценки качества предсказания непрерывных значений от 0 до 1.

Обратите внимание, что в процессе обучения возникли некоторые ошибки из-за несоответствия размерности данных для некоторых моделей. Это указывает на то, что для этих моделей может потребоваться дополнительная настройка или преобразование данных.

Вы можете использовать этот код в качестве отправной точки и дополнительно настроить его в соответствии с вашими потребностями. [\[>-\]](#)



Anonymous

`usr/local/lib/python3.10/dist-packages/sklearn/model_selection/_validation.py:378: FitFailedWarning:`

18 fits failed out of a total of 66.

The score on these train-test partitions for these parameters will be set to nan.

If these failures are not expected, you can try to debug them by setting `error_score='raise'`.

Below are more details about the failures:

12 fits failed with the following error:

Traceback (most recent call last):

File `"/usr/local/lib/python3.10/dist-packages/sklearn/model_selection/_validation.py"`, line 686, in `_fit_and_score`
estimator.fit(X_train, y_train, **fit_params)

File `"/usr/local/lib/python3.10/dist-packages/sklearn/pipeline.py"`, line 405, in `fit`
self._final_estimator.fit(Xt, y, **fit_params_last_step)

File `"/usr/local/lib/python3.10/dist-packages/sklearn/svm/_base.py"`, line 192, in `fit`
X, y = self._validate_data(X, y)

File `"/usr/local/lib/python3.10/dist-packages/sklearn/base.py"`, line 584, in `_validate_data`
X, y = check_X_y(X, y, **check_params)

File `"/usr/local/lib/python3.10/dist-packages/sklearn/utils/validation.py"`, line 1122, in `check_X_y`
y = _check_y(y, multi_output=multi_output, y_numeric=y_numeric, estimator=estimator)

File `"/usr/local/lib/python3.10/dist-packages/sklearn/utils/validation.py"`, line 1143, in `_check_y`
y = column_or_1d(y, warn=True)

File `"/usr/local/lib/python3.10/dist-packages/sklearn/utils/validation.py"`, line 1202, in `column_or_1d`
raise ValueError(

ValueError: y should be a 1d array, got an array of shape (247, 7) instead.

6 fits failed with the following error:

Traceback (most recent call last):

File `"/usr/local/lib/python3.10/dist-packages/sklearn/model_selection/_validation.py"`, line 686, in `_fit_and_score`

```

estimator.fit(X_train, y_train, **fit_params)
File "/usr/local/lib/python3.10/dist-packages/sklearn/pipeline.py", line 405, in fit
    self._final_estimator.fit(Xt, y, **fit_params_last_step)
File "/usr/local/lib/python3.10/dist-packages/sklearn/svm/_base.py", line 192, in fit
    X, y = self._validate_data(
File "/usr/local/lib/python3.10/dist-packages/sklearn/base.py", line 584, in _validate_data
    X, y = check_X_y(X, y, **check_params)
File "/usr/local/lib/python3.10/dist-packages/sklearn/utils/validation.py", line 1122, in check_X_y
    y = _check_y(y, multi_output=multi_output, y_numeric=y_numeric, estimator=estimator)
File "/usr/local/lib/python3.10/dist-packages/sklearn/utils/validation.py", line 1143, in _check_y
    y = column_or_1d(y, warn=True)
File "/usr/local/lib/python3.10/dist-packages/sklearn/utils/validation.py", line 1202, in column_or_1d
    raise ValueError(
ValueError: y should be a 1d array, got an array of shape (248, 7) instead.

```

```

warnings.warn(some_fits_failed_message, FitFailedWarning)
/usr/local/lib/python3.10/dist-packages/sklearn/model_selection/_search.py:952: UserWarning: One or more of the test scores are non-finite: [-0.
-0.03208756 -0.03211899 -0.04089782 -0.08311137
-0.08311137 -0.00418631 -0.00359839 -0.00364549 nan nan
nan nan nan nan -0.00532287 -0.00409127
-0.00591619 -0.00417061 -0.00696111 -0.0044148 ]
warnings.warn(
(Pipeline(steps=[('scaler', StandardScaler()),
                  ('regressor', RandomForestRegressor(n_estimators=50))]),
 {'regressor': RandomForestRegressor(n_estimators=50),
  'regressor__n_estimators': 50},
 0.007254800646900267,
 0.015334052111410616)

```



ChatGPT

Ошибки, которые вы получили при выполнении GridSearchCV, указывают на проблему с использованием метода опорных векторов (выхода (множество целевых переменных). SVM в библиотеке scikit-learn по умолчанию предназначен для работы с одномерными выходами (одномерным массивом). В вашем случае y представляет собой массив с формой (n_samples, 7), поскольку вы пытаетесь предсказать эмоциональных состояний.

Чтобы решить эту проблему, вы можете:

1. **Исключить SVM** из списка моделей для GridSearchCV, если вам не критично использование этой модели.
2. Использовать **мульти-выходные регрессоры** (MultiOutputRegressor), которые оборачивают основные регрессоры и позволяют им работать с многомерными выходами. Это применимо к моделям, не поддерживающим многомерный выход напрямую.

Давайте обновим код, исключив SVM и повторно выполним GridSearchCV. Возможно, вам также придется настроить список гиперпараметров моделей, чтобы избежать подобных ошибок.



Anonymous

Давай перепиши код для MultiOutputRegressor

ВАЖНО чтобы в конце была таблица сравнения метрик для каждого метода



ChatGPT

В результате выполнения кода были получены следующие метрики для каждой из рассмотренных моделей:

Модель	Средняя квадратичная ошибка (MSE)	Средняя абсолютная ошибка (MAE)
Линейная регрессия	0.037254	0.114128
Ridge	0.037424	0.114417
Lasso	0.088349	0.185871
Случайный лес	0.005916	0.014225
KNN	0.015116	0.026649

На основе этих результатов, можно сделать вывод, что модель случайного леса (Random Forest) показала лучшие результаты как по ошибке, так и по средней абсолютной ошибке. Это указывает на то, что она может быть наиболее подходящей моделью для восстановления состояний из двух признаков Valence и Arousal в данном случае.

Эти результаты могут служить отправной точкой для дальнейшей оптимизации модели и выбора наиболее подходящего подхода для ее

 **Anonymous**

Хочу метрику R2

 **ChatGPT**

К сожалению, похоже, что в результате таблицы значение метрики R2 не было правильно рассчитано. Давайте исправим это и повторим модели. [↩](#)

Похоже, что значения метрики R2 снова не были правильно рассчитаны. Позвольте мне исправить это и повторно рассчитать R2 для к

К сожалению, кажется, что возникла проблема с корректным расчетом значения коэффициента детерминации (R2) для каждой модели особенностями моей среды выполнения кода.

Вы можете выполнить следующий код в своей среде для расчета R2 каждой модели:

```
python

from sklearn.metrics import r2_score

# Перерасчет метрики R2 для каждой модели
for index, row in results.iterrows():
    model_name = row['Model']
    model = [m for m in models if m[0] == model_name][0][1]
    y_pred = model.predict(X_test)
    r2_value = r2_score(y_test, y_pred)
    results.at[index, 'R2'] = r2_value

results
```

Этот код должен правильно рассчитать и обновить значения R2 для каждой из рассмотренных моделей в вашей таблице результатов.

 **Anonymous**

```
Model MSE MAE R2
0 Linear Regression 0.037254 0.114128 0.249030
1 Ridge 0.037424 0.114417 0.252194
2 Lasso 0.088349 0.185871 -0.023129
3 Random Forest 0.005793 0.014075 0.892854
4 KNN 0.015116 0.026649 0.666649
```

сделай вывод

 **ChatGPT**

На основе предоставленной таблицы с результатами моделей регрессии для восстановления семи эмоциональных состояний из двух можно сделать следующие выводы:

1. **Random Forest (Случайный лес):** Эта модель показала наилучшие результаты по всем трем метрикам. С наименьшей средней квадраты 0.005793 и средней абсолютной ошибкой (MAE) 0.014075, а также с высоким коэффициентом детерминации (R2) 0.892854, она значител модели. Это указывает на высокую точность и надежность модели в предсказании эмоциональных состояний.
2. **KNN (k-ближайших соседей):** Эта модель занимает второе место по эффективности с R2 0.666649, что указывает на хорошую, но не на модели объяснить вариабельность данных.
3. **Линейная регрессия и Ridge:** Обе эти модели показали средние результаты с похожими значениями MSE и MAE и умеренными значени: соответственно). Это означает, что они менее точны, чем модели Random Forest и KNN, в предсказании эмоциональных состояний.
4. **Lasso:** Эта модель показала самые низкие результаты по всем метрикам. Особенно стоит отметить отрицательное значение R2 (-0.023 модель работает хуже, чем простое среднее значение.

Исходя из этих результатов, **модель случайного леса является наиболее предпочтительной** для данной задачи. Она обеспечивает наиб предсказания семи эмоциональных состояний на основе двух признаков Valence и Arousal. Однако стоит учитывать, что модели, основ могут быть склонны к переобучению, поэтому важно проводить тщательную кросс-валидацию и, возможно, настройку гиперпараметр производительности.

 **Anonymous**

Хочу еще добавить туда другие модели, в том числе полиномиальную регрессию авторенкодер и тд и тп

```

from sklearn.multioutput import MultiOutputRegressor
from sklearn.metrics import r2_score

# Обновление списка моделей для сравнения с использованием MultiOutputRegressor
models = [
    ('Linear Regression', MultiOutputRegressor(LinearRegression())),
    ('Ridge', MultiOutputRegressor(Ridge())),
    ('Lasso', MultiOutputRegressor(Lasso())),
    ('Random Forest', MultiOutputRegressor(RandomForestRegressor())),
    ('KNN', MultiOutputRegressor(KNeighborsRegressor()))
]

# Создание пустого DataFrame для сохранения результатов
results = pd.DataFrame(columns=['Model', 'MSE', 'MAE'])

# Цикл по всем моделям для обучения и оценки
for name, model in models:
    # Обучение модели
    model.fit(X_train, y_train)

    # Предсказание на тестовых данных
    y_pred = model.predict(X_test)

    # Расчет метрик
    mse = mean_squared_error(y_test, y_pred)
    mae = mean_absolute_error(y_test, y_pred)

    # Добавление результатов в DataFrame
    results = results.append({'Model': name, 'MSE': mse, 'MAE': mae}, ignore_index=True)

for index, row in results.iterrows():
    model_name = row['Model']
    model = [m for m in models if m[0] == model_name][0][1]
    y_pred = model.predict(X_test)
    r2_value = r2_score(y_test, y_pred)
    results.at[index, 'R2'] = r2_value

results

```

вот этот код адаптируй. и еще раз я тебе говорю пиши нормально. как можно меньшими усилиями. не пиши велосипеды. если есть воз-
библиотеки которые это делают используй

ChatGPT

Для расширения списка моделей, включая полиномиальную регрессию и автоэнкодеры, мы можем адаптировать предложенный вам код. Полиномиальная регрессия может быть реализована с использованием `PolynomialFeatures` из `sklearn.preprocessing`, а для автоэнкодера можно использовать `MLPRegressor` из `sklearn.neural_network`, который представляет собой простую форму нейронной сети.

Однако, стоит отметить, что использование автоэнкодера в задачах регрессии не является стандартным подходом. Автоэнкодеры обычно используются для уменьшения размерности или выявления скрытых признаков в данных, а не для прямого прогнозирования значений. Тем не менее, мы можем использовать `MLPRegressor` как форму автоэнкодера.

Давайте обновим код, добавив полиномиальную регрессию и `MLPRegressor`, и выполним обучение и оценку всех моделей:

```

python

from sklearn.preprocessing import PolynomialFeatures
from sklearn.neural_network import MLPRegressor
from sklearn.pipeline import make_pipeline

# Добавление полиномиальной регрессии и MLPRegressor в список моделей
models.extend([

```

```

    ('Polynomial Regression', make_pipeline(PolynomialFeatures(degree=2), LinearRegression())),
    ('MLPRegressor', MultiOutputRegressor(MLPRegressor(random_state=42)))
])

# Очистка DataFrame результатов
results = pd.DataFrame(columns=['Model', 'MSE', 'MAE', 'R2'])

# Цикл по всем моделям для обучения и оценки
for name, model in models:
    model.fit(X_train, y_train)
    y_pred = model.predict(X_test)
    mse = mean_squared_error(y_test, y_pred)
    mae = mean_absolute_error(y_test, y_pred)
    r2 = r2_score(y_test, y_pred)
    results = results.append({'Model': name, 'MSE': mse, 'MAE': mae, 'R2': r2}, ignore_index=True)

results

```

Этот код выполнит требуемые вычисления и вернет результаты для всех рассмотренных моделей, включая их сравнение по метрикам. Обратите внимание, что обучение MLPRegressor может занять значительное время и потребовать настройки гиперпараметров для достижения с

Anonymous

Я тебе давно говорил что я также хочу для каждой модели grid search. Сделай мне это. не перегружай код говном

ChatGPT

Для реализации Grid Search для каждой модели включая новые модели, такие как полиномиальная регрессия и MLPRegressor, можно следующим подходом:

1. Определение списка моделей с соответствующими гиперпараметрами.
2. Использование GridSearchCV для каждой модели отдельно, чтобы найти лучшие гиперпараметры.
3. Сохранение результатов для сравнения моделей.

Для упрощения кода, мы будем использовать отдельные блоки для Grid Search каждой модели. Обратите внимание, что для некоторых полиномиальная регрессия, может потребоваться обширный поиск по сетке из-за большого количества комбинаций гиперпараметров. MLPRegressor может занять значительное время.

Вот примерный код для реализации этого подхода:

```

python

from sklearn.model_selection import GridSearchCV

# Функция для выполнения Grid Search и возврата результатов
def perform_grid_search(model, params, name):
    grid_search = GridSearchCV(model, params, cv=3, scoring='neg_mean_squared_error', n_jobs=-1)
    grid_search.fit(X_train, y_train)
    best_model = grid_search.best_estimator_
    y_pred = best_model.predict(X_test)
    mse = mean_squared_error(y_test, y_pred)
    mae = mean_absolute_error(y_test, y_pred)
    r2 = r2_score(y_test, y_pred)
    return {'Model': name, 'MSE': mse, 'MAE': mae, 'R2': r2, 'Best Params': grid_search.best_params_}

# Список моделей и их параметров для Grid Search
grid_models = [
    (MultiOutputRegressor(LinearRegression()), {}, 'Linear Regression'),
    # Пример для полиномиальной регрессии (может быть трудоемким)
    (make_pipeline(PolynomialFeatures(), LinearRegression()),
     {'polynomialfeatures_degree': [2, 3]}, 'Polynomial Regression'),
    # Добавьте сюда другие модели и их параметры

```



```

# ...
]

# Пустой DataFrame для результатов
grid_results = pd.DataFrame(columns=['Model', 'MSE', 'MAE', 'R2', 'Best Params'])

# Выполнение Grid Search для каждой модели
for model, params, name in grid_models:
    result = perform_grid_search(model, params, name)
    grid_results = grid_results.append(result, ignore_index=True)

grid_results

```

Этот код будет выполнять Grid Search для каждой модели и записывать результаты в DataFrame. Вы можете добавить дополнительные гиперпараметры в список `grid_models`. Однако учтите, что полное выполнение этого кода может занять значительное время, особенно количеством гиперпараметров.

Anonymous

```

grid_models = [
    (MultiOutputRegressor(LinearRegression()), {}, 'Linear Regression'),
    (make_pipeline(PolynomialFeatures(), LinearRegression()),
     {'polynomialfeatures__degree': [2, 3]}, 'Polynomial Regression'),
]

```

Допиши сюда все модели и гиперпараметры которые можно перебрать.
А так же впрямую я надеюсь ты параметры не на тесте подбираешь

ChatGPT

Чтобы дополнить список моделей `grid_models` и соответствующие гиперпараметры для Grid Search, включим в него модели, такие как `RandomForestRegressor`, `KNeighborsRegressor` и `MLPRegressor`. Параметры будут подбираться на основе кросс-валидации на тренировочных данных, чтобы избежать переобучения и обеспечить корректную оценку моделей.

Примеры гиперпараметров для каждой модели:

1. **Ridge** - параметр регуляризации `alpha`.
2. **Lasso** - параметр регуляризации `alpha`.
3. **RandomForestRegressor** - количество деревьев `n_estimators` и глубина деревьев `max_depth`.
4. **KNeighborsRegressor** - количество соседей `n_neighbors`.
5. **MLPRegressor** - количество слоёв и нейронов `hidden_layer_sizes`, коэффициент регуляризации `alpha`.

Вот обновленный список `grid_models`:

```

python

grid_models = [
    (MultiOutputRegressor(LinearRegression()), {}, 'Linear Regression'),
    (make_pipeline(PolynomialFeatures(), LinearRegression()),
     {'polynomialfeatures__degree': [2, 3]}, 'Polynomial Regression'),
    (MultiOutputRegressor(Ridge()), {'estimator__alpha': [0.1, 1, 10]}, 'Ridge'),
    (MultiOutputRegressor(Lasso()), {'estimator__alpha': [0.1, 1, 10]}, 'Lasso'),
    (MultiOutputRegressor(RandomForestRegressor(random_state=42)),
     {'estimator__n_estimators': [10, 50, 100], 'estimator__max_depth': [None, 10, 20]}, 'Random Forest'),
    (MultiOutputRegressor(KNeighborsRegressor()),
     {'estimator__n_neighbors': [3, 5, 7], 'estimator__weights': ['uniform', 'distance]}, 'KNN'),
    (MultiOutputRegressor(MLPRegressor(random_state=42)),

```



```
{'estimator__hidden_layer_sizes': [(50,50), (100,)], 'estimator__alpha': [0.001, 0.01]}, 'MLPRegressor']
```

Каждая модель будет обучена и оценена с использованием различных комбинаций гиперпараметров, определенных в `grid_models`. И обучение некоторых из этих моделей, особенно `MLPRegressor` с различными конфигурациями слоев, может занять значительное время.

 **Anonymous**

хочу добавить dummy model чтобы сравнить что модель не случайная

 **ChatGPT**

Добавление простой ("dummy") модели для сравнения — это отличная идея, поскольку она помогает убедиться, что разработанные модели работают лучше, чем простое случайное или базовое предсказание. Простая модель обычно использует простые правила для предсказания среднего значения целевых переменных или наиболее часто встречающегося значения.

В библиотеке `scikit-learn` для таких целей есть `DummyRegressor`, который можно настроить для разных стратегий, таких как `mean`. Добавим `DummyRegressor` в список моделей:

```
python

from sklearn.dummy import DummyRegressor

grid_models.append(
    (MultiOutputRegressor(DummyRegressor(strategy="mean")), {}, 'Dummy Mean')
)
```

Теперь `grid_models` включает `DummyRegressor` с выбранной стратегией "mean", который будет всегда предсказывать среднее значение данных. Это даст нам базовую линию для сравнения, и если разработанные модели предсказывают лучше, чем эта простая модель, это их эффективности.

 **Anonymous**

Теперь для полученных моделей мне нужна интерпретация модели. что модель предсказывает

 **ChatGPT**

Интерпретация моделей машинного обучения важна для понимания того, как модель делает свои предсказания и какие факторы наиболее влияют на предсказания. В зависимости от типа модели, существуют различные методы интерпретации. Рассмотрим некоторые из них для моделей, которые мы использовали:

1. **Линейная Регрессия и Ridge/Lasso:** Эти модели дают коэффициенты для каждого признака, которые можно интерпретировать напрямую на большее влияние признака на целевую переменную. Для полиномиальной регрессии интерпретация может быть более сложной из-за взаимодействия признаков.
2. **Random Forest и KNN:** Эти модели обычно труднее интерпретировать напрямую. Однако, для Random Forest можно использовать важность признаков (`feature importances`), которые показывают, насколько каждый признак вносит вклад в прогнозы модели.
3. **MLPRegressor (Нейронные сети):** Эти модели обычно считаются "черными ящиками", поскольку трудно понять, как входные данные преобразуются в выходные. Однако существуют методы визуализации и анализа весов, которые могут дать некоторое представление о том, как модель работает.
4. **Dummy Model:** Интерпретация не требуется, так как она не делает значимых предсказаний, а просто следует простой предопределенной стратегии (всегда предсказывает среднее значение).

Для интерпретации моделей можно также использовать такие инструменты, как SHAP (SHapley Additive exPlanations) и LIME (Local Interpretable Model-explanations), которые могут помочь понять вклад каждого признака в конкретное предсказание модели.

Если вам требуется более детальная интерпретация для конкретной модели из вашего списка, пожалуйста, дайте знать, и я предоставлю рекомендации или примеры кода.

 **Anonymous**

Можно ли как то на плоскости двух признаков (наши входные данные) визуализировать KNN и точки target (из 7 признаков, можно для них выбрать 2 признака и построить)

 **ChatGPT**

Да, можно визуализировать работу модели KNN на плоскости двух входных признаков (Valence и Arousal) и отображать предсказания модели для целевых признаков (эмоциональных состояний). Однако, поскольку KNN - это метод, основанный на соседях, и он не даёт явной функции (например, линейная регрессия), визуализация будет заключаться в отображении границ решений, созданных KNN на основе обучающих данных. Для каждого из семи целевых признаков (например, Happy, Sad и т.д.) можно создать отдельную визуализацию. Однако, стоит отметить, что это может быть упрощением, так как KNN работает в многомерном пространстве и проекция на двумерную плоскость может не полностью передать информацию.

модели.

Вот примерный код для визуализации:

```
python

import matplotlib.pyplot as plt
import numpy as np
from sklearn.neighbors import KNeighborsRegressor

# Для упрощения выберем один из целевых признаков, например 'Happy'
target_feature = 'Happy'

# Обучение KNN модели для этого признака
knn = KNeighborsRegressor(n_neighbors=5)
knn.fit(X_train, y_train[target_feature])

# Создание сетки для визуализации
x_min, x_max = X['Valence'].min() - 1, X['Valence'].max() + 1
y_min, y_max = X['Arousal'].min() - 1, X['Arousal'].max() + 1
xx, yy = np.meshgrid(np.linspace(x_min, x_max, 100), np.linspace(y_min, y_max, 100))

# Предсказание для каждой точки на сетке
Z = knn.predict(np.c_[xx.ravel(), yy.ravel()])
Z = Z.reshape(xx.shape)

# Визуализация границ решений
plt.contourf(xx, yy, Z, alpha=0.4)
plt.scatter(X['Valence'], X['Arousal'], c=y[target_feature], s=20, edgecolor='k')
plt.title(f"KNN decision boundaries for '{target_feature}'")
plt.xlabel('Valence')
plt.ylabel('Arousal')
plt.show()
```

Этот код визуализирует границы решений KNN для признака 'Happy'. Вы можете изменить `target_feature` на другие эмоциональные соответствующих визуализаций.

 **Anonymous**

Я хочу вывести 7 графиков для каждой фичи. И в качестве точек использовать test

 **ChatGPT**

 Error analyzing ▾

 Error analyzing ▾

Кажется, произошла ошибка, поскольку переменные `X_train`, `X_test`, `y_train`, `y_test` и `emotion_columns` не определены в моей т...
Для корректного выполнения кода вам необходимо определить эти переменные со своими данными.

Пожалуйста, убедитесь, что данные корректно загружены и обработаны, и что переменные `emotion_columns` (список эмоциональных `X_test`, `y_train` и `y_test` (тренировочные и тестовые наборы данных) определены в вашем среде выполнения, прежде чем запускать.
Если у вас есть данные готовы, вы можете выполнить предложенный код на вашем компьютере, чтобы визуализировать результаты К эмоционального состояния. [\[↗\]](#)

 **Anonymous**

```
grid_models = [
    (MultiOutputRegressor(LinearRegression()), {}, 'Linear Regression'),
    (make_pipeline(PolynomialFeatures(), LinearRegression()),
     {'polynomialfeatures__degree': [2, 4, 6, 8]}, 'Polynomial Regression'),
    (MultiOutputRegressor(Ridge()), {'estimator__alpha': [0.1, 1, 10]}, 'Ridge'),
    (MultiOutputRegressor(Lasso()), {'estimator__alpha': [0.1, 1, 10]}, 'Lasso'),
    (MultiOutputRegressor(RandomForestRegressor(random_state=42)),
     {'estimator__n_estimators': [10, 50, 100], 'estimator__max_depth': [None, 3, 5, 10, 20]}, 'Random Forest')
```

```
{estimator__n_estimators: [10, 50, 100], estimator__max_depth: [None, 3, 5, 10, 20], Random Forest},
(MultiOutputRegressor(KNeighborsRegressor())),
{'estimator__n_neighbors': [3, 5, 7, 20], 'estimator__weights': ['uniform', 'distance']}, 'KNN'),
(MultiOutputRegressor(MLPRegressor(random_state=42)),
{'estimator__hidden_layer_sizes': [(50,50), (100,),(15, 5, 15),(50, 30, 10, 30, 50),
(100, 50, 30, 10, 5, 10, 30, 50, 100),
(50, 10, 5, 10, 50),
(10, 5, 5, 2, 5, 5, 10)], 'estimator__alpha': [0.001, 0.01]}, 'MLPRegressor')
]
..
rid_results = pd.DataFrame(columns=['Model', 'MSE', 'MAE', 'R2', 'Best Params'])
```

```
for model, params, name in grid_models:
    result = perform_grid_search(model, params, name)
    grid_results.append(result, ignore_index=True)
```

grid_results

давай аналогично для всех этих моделей (хочу MLPRegressor, RandomForestRegressor, PolynomialFeatures регрессия) выведем так же гр
используй лучшую. пиши нормальный код

ChatGPT

Для создания графиков решений лучших моделей MLPRegressor, RandomForestRegressor и полиномиальной регрессии, нам нужно сначала гиперпараметры являются лучшими на основе результатов Grid Search. Затем можно использовать эти параметры для обучения каждой из предсказаний на двумерной плоскости входных признаков.

Поскольку в моей текущей сессии нет доступа к вашим данным и результатам Grid Search, я предоставляю примерный код, который вы можете использовать по своим нужды. Этот код предполагает, что у вас уже есть результаты Grid Search в grid_results, и вы можете извлечь из них лучшие параметры для каждой модели.

```
python

from sklearn.metrics import mean_squared_error, mean_absolute_error, r2_score
from sklearn.model_selection import GridSearchCV

# Предполагаем, что grid_results уже определен
# Здесь вставьте код для определения grid_results

# Функция для построения графиков
def plot_model_predictions(model, X_test, y_test, feature_names, target_name):
    # Предсказания модели
    y_pred = model.predict(X_test)

    # Визуализация
    plt.scatter(X_test[:, 0], X_test[:, 1], c=y_test, label="Actual", alpha=0.6)
    plt.scatter(X_test[:, 0], X_test[:, 1], c=y_pred, label="Predicted", alpha=0.6)
    plt.xlabel(feature_names[0])
    plt.ylabel(feature_names[1])
    plt.title(f"Predictions of {target_name}")
    plt.legend()
    plt.show()

# Извлечение лучших параметров для каждой модели
best_params_mlp = grid_results[grid_results['Model'] == 'MLPRegressor']['Best Params'].iloc[0]
best_params_rf = grid_results[grid_results['Model'] == 'Random Forest']['Best Params'].iloc[0]
best_params_poly = grid_results[grid_results['Model'] == 'Polynomial Regression']['Best Params'].iloc[0]

# Обучение и визуализация для каждой модели
models_to_plot = [
    ('MLPRegressor', MLPRegressor(**best_params_mlp)),
```

```

    ('Random Forest', RandomForestRegressor(**best_params_rf)),

    ('Polynomial Regression', make_pipeline(PolynomialFeatures(), LinearRegression(**best_params_poly)))
]

for name, model in models_to_plot:
    model.fit(X_train, y_train)
    plot_model_predictions(model, X_test, y_test, ['Valence', 'Arousal'], name)

```

Этот код сначала извлекает лучшие параметры для каждой из трех моделей из `grid_results`, затем обучает каждую модель с этими параметрами и визуализирует предсказания на тестовом наборе данных. Обратите внимание, что для полиномиальной регрессии требуется создать `PolynomialFeatures` и `LinearRegression`.

Убедитесь, что у вас есть все необходимые данные и результаты Grid Search перед запуском этого кода.



Anonymous

Там `MultiOutputRegressor` для всех этих моделей



ChatGPT

Понял, в таком случае, мы будем использовать `MultiOutputRegressor` для обучения каждой модели. Я предполагаю, что для полиномиальной регрессии также будем использовать `MultiOutputRegressor`, но в комбинации с `Pipeline`, чтобы интегрировать `PolynomialFeatures` и `LinearRegression`. Вот обновленный код для визуализации предсказаний лучших моделей с использованием `MultiOutputRegressor`:

```

python

from sklearn.pipeline import make_pipeline
from sklearn.preprocessing import PolynomialFeatures
from sklearn.neural_network import MLPRegressor
from sklearn.ensemble import RandomForestRegressor
from sklearn.multioutput import MultiOutputRegressor

# Функция для визуализации предсказаний модели
def plot_model_predictions(model, X_test, y_test, feature_names, target_name):
    # Предсказания модели
    y_pred = model.predict(X_test)

    # Визуализация
    plt.scatter(X_test[:, 0], X_test[:, 1], c=y_test, label="Actual", alpha=0.6)
    plt.scatter(X_test[:, 0], X_test[:, 1], c=y_pred, label="Predicted", alpha=0.6)
    plt.xlabel(feature_names[0])
    plt.ylabel(feature_names[1])
    plt.title(f"Predictions of {target_name}")
    plt.legend()
    plt.show()

# Извлечение лучших параметров для каждой модели
best_params_mlp = grid_results[grid_results['Model'] == 'MLPRegressor']['Best Params'].iloc[0]
best_params_rf = grid_results[grid_results['Model'] == 'Random Forest']['Best Params'].iloc[0]
best_params_poly = grid_results[grid_results['Model'] == 'Polynomial Regression']['Best Params'].iloc[0]

# Обучение и визуализация для каждой модели
models_to_plot = [
    ('MLPRegressor', MultiOutputRegressor(MLPRegressor(**best_params_mlp))),
    ('Random Forest', MultiOutputRegressor(RandomForestRegressor(**best_params_rf))),
    ('Polynomial Regression', make_pipeline(PolynomialFeatures(), MultiOutputRegressor(LinearRegression(**best_params_poly))))
]

for name, model in models_to_plot:

```

```

model.fit(X_train, y_train)
plot_model_predictions(model, X_test, y_test, ['Valence', 'Arousal'], name)

```

В этом коде мы обучаем каждую модель с использованием `MultiOutputRegressor`, чтобы учесть множественные выходные переменные. Этот код используется для предсказания на тестовом наборе данных, и результаты визуализируются. Убедитесь, что у вас есть все необходимые библиотеки перед запуском этого кода.

Anonymous

`TypeError: MLPRegressor.__init__() got an unexpected keyword argument 'estimator__alpha'`

ChatGPT

Ошибка `TypeError: MLPRegressor.__init__() got an unexpected keyword argument 'estimator__alpha'` указывает на то, что параметр `estimator__alpha` не является допустимым для `MLPRegressor`. Этот префикс применяется к параметрам вложенных оценщиков, но когда вы создаете экземпляр модели напрямую, такой префикс использовать не нужно. Для исправления этой ошибки, вам нужно убрать префикс `estimator__` из параметров, которые вы передаете в модели. Вот как вы можете это сделать:

```

python

# Извлечение лучших параметров для MLPRegressor и их преобразование
best_params_mlp = grid_results[grid_results['Model'] == 'MLPRegressor']['Best Params'].iloc[0]
mlp_params = {k.replace('estimator__', ''): v for k, v in best_params_mlp.items()}

# Извлечение лучших параметров для RandomForestRegressor
best_params_rf = grid_results[grid_results['Model'] == 'Random Forest']['Best Params'].iloc[0]
rf_params = {k.replace('estimator__', ''): v for k, v in best_params_rf.items()}

# Извлечение лучших параметров для Polynomial Regression
best_params_poly = grid_results[grid_results['Model'] == 'Polynomial Regression']['Best Params'].iloc[0]
poly_params = {k.replace('polynomialfeatures__', ''): v for k, v in best_params_poly.items()}

# Обучение и визуализация для каждой модели
models_to_plot = [
    ('MLPRegressor', MultiOutputRegressor(MLPRegressor(**mlp_params))),
    ('Random Forest', MultiOutputRegressor(RandomForestRegressor(**rf_params))),
    ('Polynomial Regression', make_pipeline(PolynomialFeatures(**poly_params), MultiOutputRegressor(LinearRegression()))),
]

for name, model in models_to_plot:
    model.fit(X_train, y_train)
    plot_model_predictions(model, X_test, y_test, ['Valence', 'Arousal'], name)

```

В этом коде мы корректируем параметры для каждой модели, удаляя префиксы, которые использовались в `GridSearchCV`. Это позволяет использовать полученные наилучшие параметры для каждой модели.

Anonymous

Визуализировать мне нужно примерно вот так

```

x_min, x_max = X_train['Valence'].min() - 1, X_train['Valence'].max() + 1
y_min, y_max = X_train['Arousal'].min() - 1, X_train['Arousal'].max() + 1
xx, yy = np.meshgrid(np.linspace(x_min, x_max, 100), np.linspace(y_min, y_max, 100))

# Предсказание для каждой точки на сетке
Z = knn.predict(np.c_[xx.ravel(), yy.ravel()])
Z = Z.reshape(xx.shape)

# Визуализация
plt.subplot(3, 3, i + 1)
plt.contourf(xx, yy, Z, alpha=0.4)

```


```
plt.scatter(X_test['Valence'], X_test['Arousal'], c=y_test[target_feature], s=20, edgecolor='k')
plt.title(f"KNN for '{target_feature}'")
plt.xlabel('Valence')
plt.ylabel('Arousal')
```

перепиши

```
def plot_model_predictions(model, X_test, y_test, feature_names, target_name):
```

нужно вывести 7 графиков для каждого признака

 ChatGPT

 Error analyzing ▾

Прошу прощения за включение синтаксической ошибки в код. Давайте уберем ненужный символ и перезапустим код.

python

```
def plot_model_predictions(model, X_train, X_test, y_test, feature_names, target_features):
    # Определение размера графика
    plt.figure(figsize=(15, 15))

    # Цикл по каждому эмоциональному состоянию для создания визуализации
    for i, target_feature in enumerate(target_features):
        # Обучение модели для этого признака
        model.fit(X_train, y_train[target_feature])

        # Создание сетки для визуализации
        x_min, x_max = X_train[:, 0].min() - 1, X_train[:, 0].max() + 1
        y_min, y_max = X_train[:, 1].min() - 1, X_train[:, 1].max() + 1
        xx, yy = np.meshgrid(np.linspace(x_min, x_max, 100), np.linspace(y_min, y_max, 100))

        # Предсказание для каждой точки на сетке
        Z = model.predict(np.c_[xx.ravel(), yy.ravel()])
        Z = Z.reshape(xx.shape)

        # Визуализация
        plt.subplot(3, 3, i + 1)
        plt.contourf(xx, yy, Z, alpha=0.4)
        plt.scatter(X_test[:, 0], X_test[:, 1], c=y_test[target_feature], s=20, edgecolor='k')
        plt.title(f"{model.__class__.__name__} for '{target_feature}'")
        plt.xlabel(feature_names[0])
        plt.ylabel(feature_names[1])

    plt.tight_layout()
    plt.show()
```

Этот код создает визуализации для каждого эмоционального состояния с использованием предсказаний модели. Для каждого эмоционального состояния генерирован отдельный график, показывающий как модель предсказывает этот признак в зависимости от значений Valence и Arousal. Этот код для визуализации предсказаний любой модели, передав её в качестве аргумента вместе с соответствующими данными. [\[↗\]](#)

 Anonymous

```
from sklearn.pipeline import make_pipeline
from sklearn.preprocessing import PolynomialFeatures
from sklearn.neural_network import MLPRegressor
from sklearn.ensemble import RandomForestRegressor
from sklearn.multioutput import MultiOutputRegressor
```

```
def plot_model_predictions(models, X_train, X_test, y_test, feature_names, target_features):
```

```
    # Определение размера графика
    plt.figure(figsize=(15, 15))
```

```

plt.figure(figsize=(15, 15))

# Цикл по каждому эмоциональному состоянию для создания визуализации
for i, target_feature in enumerate(target_features):
    model = models[target_feature] # Получение модели для данного целевого признака

    # Создание сетки для визуализации
    x_min, x_max = X_train[:, 0].min() - 1, X_train[:, 0].max() + 1
    y_min, y_max = X_train[:, 1].min() - 1, X_train[:, 1].max() + 1
    xx, yy = np.meshgrid(np.linspace(x_min, x_max, 100), np.linspace(y_min, y_max, 100))

    # Предсказание для каждой точки на сетке
    Z = model.predict(np.c_[xx.ravel(), yy.ravel()])
    Z = Z.reshape(xx.shape)

    # Визуализация
    plt.subplot(3, 3, i + 1)
    plt.contourf(xx, yy, Z, alpha=0.4)
    plt.scatter(X_test[:, 0], X_test[:, 1], c=y_test[target_feature], s=20, edgecolor='k')
    plt.title(f"Model for '{target_feature}'")
    plt.xlabel(feature_names[0])
    plt.ylabel(feature_names[1])

plt.tight_layout()
plt.show()

# Извлечение лучших параметров для каждой модели
best_params_mlp = grid_results[grid_results['Model'] == 'MLPRegressor']['Best Params'].iloc[0]
mlp_params = {k.replace('estimator_', ''): v for k, v in best_params_mlp.items()}

# Извлечение лучших параметров для RandomForestRegressor
best_params_rf = grid_results[grid_results['Model'] == 'Random Forest']['Best Params'].iloc[0]
rf_params = {k.replace('estimator_', ''): v for k, v in best_params_rf.items()}

# Извлечение лучших параметров для Polynomial Regression
best_params_poly = grid_results[grid_results['Model'] == 'Polynomial Regression']['Best Params'].iloc[0]
poly_params = {k.replace('polynomialfeatures_', ''): v for k, v in best_params_poly.items()}

# Обучение и визуализация для каждой модели
models_to_plot = [
    ('MLPRegressor', MultiOutputRegressor(MLPRegressor(**mlp_params))),
    ('Random Forest', MultiOutputRegressor(RandomForestRegressor(**rf_params))),
    ('Polynomial Regression', make_pipeline(PolynomialFeatures(**poly_params), MultiOutputRegressor(LinearRegression())))]

for name, model in models_to_plot:
    model.fit(X_train, y_train)
    plot_model_predictions(model, X_train, X_test, y_test, ['Valence', 'Arousal'], emotion_columns)

56 model.fit(X_train, y_train)
--> 57 plot_model_predictions(model, X_train, X_test, y_test, ['Valence', 'Arousal'], emotion_columns)

<ipython-input-86-f6f80ded99f5> in plot_model_predictions(models, X_train, X_test, y_test, feature_names, target_features)
11 # Цикл по каждому эмоциональному состоянию для создания визуализации
12 for i, target_feature in enumerate(target_features):
--> 13     model = models[target_feature] # Получение модели для данного целевого признака
14
15     # Создание сетки для визуализации

```

TypeError: 'MultiOutputRegressor' object is not subscriptable

ChatGPT

Ошибка `TypeError: 'MultiOutputRegressor' object is not subscriptable` возникает из-за попытки индексации объекта `MultiOutputRegressor`, который не поддерживает такую операцию. В вашем текущем подходе вы пытаетесь использовать одну модель для каждого признака `MultiOutputRegressor`. `MultiOutputRegressor` обучается на всех целевых признаках одновременно, а не на одном признаке за раз. Чтобы решить эту проблему, вам нужно изменить подход к визуализации. Вы можете обучить отдельный `MultiOutputRegressor` для визуализировать предсказания для каждого целевого признака. Вот как это можно сделать:

```
python

def plot_model_predictions(model, X_train, X_test, y_test, feature_names, target_features):
    # Определение размера графика
    plt.figure(figsize=(15, 15))

    # Цикл по каждому эмоциональному состоянию для создания визуализации
    for i, target_feature in enumerate(target_features):
        # Обучение модели для всех признаков
        model.fit(X_train, y_train)

        # Создание сетки для визуализации
        x_min, x_max = X_train[:, 0].min() - 1, X_train[:, 0].max() + 1
        y_min, y_max = X_train[:, 1].min() - 1, X_train[:, 1].max() + 1
        xx, yy = np.meshgrid(np.linspace(x_min, x_max, 100), np.linspace(y_min, y_max, 100))

        # Предсказание для каждой точки на сетке
        Z = model.predict(np.c_[xx.ravel(), yy.ravel()])
        Z = Z[:, i].reshape(xx.shape)

        # Визуализация
        plt.subplot(3, 3, i + 1)
        plt.contourf(xx, yy, Z, alpha=0.4)
        plt.scatter(X_test[:, 0], X_test[:, 1], c=y_test[target_feature], s=20, edgecolor='k')
        plt.title(f"{model.__class__.__name__} for '{target_feature}'")
        plt.xlabel(feature_names[0])
        plt.ylabel(feature_names[1])

    plt.tight_layout()
    plt.show()

# Здесь вы создаете и обучаете каждую модель, а затем вызываете функцию визуализации
```

В этом коде каждая модель (`MLPRegressor`, `RandomForestRegressor`, `Polynomial Regression`) обучается на всех целевых признаках, а для каждого признака генерируется визуализация. Это позволит вам увидеть, как каждая модель предсказывает каждое эмоциональное состояние входных признаков.

Anonymous

Мне нужно написать отчет для НИР.

Задача 7 -> 2 (seven_fields -> pa_fields)

```
pa_fields = [
    'Valence',
    'Arousal'
]
seven_fields = [
    'Neutral',
    'Happy',
    'Sad',
    'Anarv'.
```

```
    'Surprised',  
    'Scared',  
    'Disgusted'  
]
```

```
|index|Model|MSE|MAE|R2|Best Params|  
|---|---|---|---|---|---|  
|0|Linear Regression|0.004808065684380541|0.04076690571667002|0.8026235393588486|{\}  
|1|Polynomial Regression|0.004114181523702957|0.03624921399808328|0.8106917729151308|{'polynomialfeatures\_\_degree': 2}\}  
|2|Ridge|0.004790513171468978|0.040700778727408696|0.8028747781733131|{'estimator\_\_alpha': 0.1}\}  
|3|Lasso|0.05922694226618937|0.1506550412275477|-0.0011409035754195163|{'estimator\_\_alpha': 0.1}\}  
|4|Random Forest|0.0018173371036329018|0.021824462712622613|0.9212907438117026|{'estimator\_\_max\_depth': 20, 'estimator\_\_n\_es  
|5|KNN|0.002846632728344437|0.027916783939117933|0.8730377258145936|{'estimator\_\_n\_neighbors': 7, 'estimator\_\_weights': 'dista  
|6|MLPRegressor|0.0038162184001452617|0.0350066693322242|0.8237146559814886|{'estimator\_\_alpha': 0.001, 'estimator\_\_hidden\  
5, 10, 30, 50, 100}\}\}  
|7|Dummy Mean|0.05922694226618937|0.1506550412275477|-0.0011409035754195163|{\}
```

coefs for linear regression
-0.26050112 0.36682814 -0.51971425 -0.52393053 0.03779894 -0.1731416
-0.57027954



ChatGPT

Для написания отчёта по НИР, в котором рассматривается задача прогнозирования двумерного пространства 'Valence' и 'Arousal' (pa_fi эмоциональных состояний (seven_fields), можно использовать следующую структуру:

Введение

Кратко опишите цель исследования, объясните значение и актуальность задачи преобразования эмоциональных состояний в значени

Методы

Опишите используемые методы машинного обучения, включая типы моделей и подходы к выбору гиперпараметров. Укажите, что для использовались такие метрики, как MSE (среднеквадратическая ошибка), MAE (средняя абсолютная ошибка) и R2 (коэффициент детер

Результаты

Представьте результаты экспериментов в виде таблицы, как показано выше. Объясните значения каждой метрики и какие выводы мо модели. Особое внимание уделите лучшей модели (в данном случае, Random Forest) и её параметрам. Кроме того, упомяните о значимс линейной регрессии для понимания влияния каждого эмоционального состояния на значения Valence и Arousal.

Обсуждение

Проанализируйте результаты, обсудите преимущества и недостатки каждой модели. Объясните, почему некоторые модели работали л быть связано с характеристиками данных. Также обсудите важность интерпретируемости моделей в контексте данной задачи.

Заключение

Сделайте общие выводы исследования, укажите на потенциальные направления для дальнейших исследований, такие как применин оптимизация существующих или исследование других аспектов эмоциональных состояний.

Визуализации и Приложения

Если уместно, добавьте в отчёт визуализации, которые иллюстрируют работу моделей или распределение данных. Также можно включ дополнительные технические детали, код или данные.

Убедитесь, что весь текст написан в соответствии с научным стилем и академическими стандартами.



Anonymous

начни писать введение. Для памятки скажу что делается все возможные варианты 7 в 2 2 7. 42 - 7 и тд.



ChatGPT

Введение

Целью данного научно-исследовательского проекта является исследование и разработка методов машинного обучения для анализа и эмоциональных состояний человека. В частности, основной задачей является создание эффективных алгоритмов для преобразования состояний (Neutral, Happy, Sad, Angry, Surprised, Scared, Disgusted) в двумерное пространство Valence и Arousal (pa_fields) и обратно. Это

значительное прикладное значение, так как понимание эмоциональных состояний человека и их количественное выражение играют в областях, включая психологию, медицину, маркетинг и разработку пользовательских интерфейсов.

Работа включает анализ различных подходов к решению этой задачи, начиная от традиционных методов машинного обучения, таких как полиномиальная регрессия, до более сложных, например, случайных лесов и нейронных сетей. Ключевым аспектом исследования является достижение высокой точности прогнозов, но и понимание взаимосвязей между различными эмоциональными состояниями и их проекцией в пространство Valence-Arousal.

Особое внимание уделяется также обратным задачам, таким как преобразование из двумерного пространства обратно в семимерное эмоциональных состояний, а также другим связанным преобразованиям (например, из 42-мерного пространства Facial Action Coding Scheme в пространство эмоций). Эти задачи представляют собой сложные проблемы анализа данных, требующие применения продвинутых методов и тщательного анализа данных.

В результате данного исследования ожидается получение не только высокоэффективных моделей для задач преобразования, но и глубокое понимание взаимодействия различных эмоциональных состояний и их проекций, что может иметь значительные теоретические и практические последствия в психологии и когнитивных наук.



Anonymous

Теперь начини писать подробно о Методы

для начала математически обозначь задачу. что на вход на выход. в общем виде. А так же что за задача (мульти регрессия и тд). приведи потом не забудь обосновать выбор метрик и что они означают дать формулы



ChatGPT

Методы

Описание Задачи

Задача, рассматриваемая в данном исследовании, является задачей мульти-выходной регрессии. Мульти-выходная регрессия (или мульти-задача) — это тип задачи машинного обучения, где модель обучается предсказывать несколько непрерывных выходных переменных на основе входных признаков.

Математически задачу можно сформулировать следующим образом:

- Дан набор входных данных X , где каждый элемент $x_i \in X$ представляет собой вектор входных признаков.
- Соответствующий набор выходных данных Y , где каждый элемент $y_i \in Y$ является вектором целевых переменных.
- Цель состоит в том, чтобы найти функцию f , которая наилучшим образом отображает входные данные на выходные: $f(x_i) \approx y_i$.

В контексте данной задачи:

- Входные данные (X): двумерное пространство 'Valence' и 'Arousal'.
- Выходные данные (Y): семь эмоциональных состояний ('Neutral', 'Happy', 'Sad', 'Angry', 'Surprised', 'Scared', 'Disgusted').

Выбор Моделей и Метрик

Для решения этой задачи были выбраны различные модели машинного обучения, включая линейную регрессию, полиномиальную регрессию, случайный лес, K-ближайших соседей (KNN) и многослойный перцептрон (MLPRegressor). Выбор этих моделей обусловлен их способностью к аппроксимации сложных зависимостей в данных.

Метрики, использованные для оценки моделей:

1. Средняя квадратичная ошибка (MSE):

$$MSE = \frac{1}{n} \sum_{i=1}^n (y_i - \hat{y}_i)^2$$

Где y_i — истинное значение, \hat{y}_i — предсказанное значение, n — количество наблюдений. MSE измеряет средний квадрат разницы между предсказанными значениями, являясь индикатором точности модели.

2. Средняя абсолютная ошибка (MAE):

$$MAE = \frac{1}{n} \sum_{i=1}^n |y_i - \hat{y}_i|$$

MAE аналогичен MSE, но измеряет среднюю абсолютную разницу, что делает её менее чувствительной к выбросам по сравнению с MSE.

3. Коэффициент детерминации (R2):

$$R^2 = 1 - \frac{\sum_{i=1}^n (y_i - \hat{y}_i)^2}{\sum_{i=1}^n (y_i - \bar{y})^2}$$

Где \bar{y} — среднее значение y . R2 измеряет долю вариативности в зависимой переменной, которая объясняется независимыми переменными. R2 показывает, насколько хорошо данные подходят для модели, относительно простого усреднения.

Эти метрики были выбраны для оценки точности и адекватности предсказаний моделей, а также их способности объяснять переменные

Anonymous

Теперь пиши раздел Результаты

Задача 7 -> 2 (seven_fields -> pa_fields)

```
pa_fields = [
    'Valence',
    'Arousal'
]
seven_fields = [
    'Neutral',
    'Happy',
    'Sad',
    'Angry',
    'Surprised',
    'Scared',
    'Disgusted'
]
```

```
|index|Model|MSE|MAE|R2|Best Params|
|---|---|---|---|---|---|
|0|Linear Regression|0.004808065684380541|0.04076690571667002|0.8026235393588486|{}|
|1|Polynomial Regression|0.004114181523702957|0.03624921399808328|0.8106917729151308|{'polynomialfeatures\_degree': 2}|
|2|Ridge|0.004790513171468978|0.040700778727408696|0.8028747781733131|{'estimator\_alpha': 0.1}|
|3|Lasso|0.05922694226618937|0.1506550412275477|-0.0011409035754195163|{'estimator\_alpha': 0.1}|
|4|Random Forest|0.0018173371036329018|0.021824462712622613|0.9212907438117026|{'estimator\_max\_depth': 20, 'estimator\_n\_es|
|5|KNN|0.002846632728344437|0.027916783939117933|0.8730377258145936|{'estimator\_n\_neighbors': 7, 'estimator\_weights': 'dista|
|6|MLPRegressor|0.0038162184001452617|0.0350066693322242|0.8237146559814886|{'estimator\_alpha': 0.001, 'estimator\_hidden\|
5, 10, 30, 50, 100}|
|7|Dummy Mean|0.05922694226618937|0.1506550412275477|-0.0011409035754195163|{}|
```

coefs for linear regression

```
-0.26050112 0.36682814 -0.51971425 -0.52393053 0.03779894 -0.1731416
-0.57027954 - выпиши формулу итоговую регрессии
```

полиномиальная

```
['Neutral' 'Happy' 'Sad' 'Angry' 'Surprised' 'Scared' 'Disgusted'
'Neutral^2' 'Neutral Happy' 'Neutral Sad' 'Neutral Angry'
'Neutral Surprised' 'Neutral Scared' 'Neutral Disgusted' 'Happy^2'
'Happy Sad' 'Happy Angry' 'Happy Surprised' 'Happy Scared'
'Happy Disgusted' 'Sad^2' 'Sad Angry' 'Sad Surprised' 'Sad Scared'
'Sad Disgusted' 'Angry^2' 'Angry Surprised' 'Angry Scared'
'Angry Disgusted' 'Surprised^2' 'Surprised Scared' 'Surprised Disgusted'
'Scared^2' 'Scared Disgusted' 'Disgusted^2'] [ 0.09200632  1.08161244 -0.59147158 -0.75340529 -0.00421852 -0.54870615
-0.58988335 -0.05156887 -0.09460224 -0.18020524 -0.19368814 -0.04702139
-0.20677163 -0.26244942 -0.0448074 -1.17706707 -0.14299746  0.11967458
0.14092332 -0.56426376 -0.55632604  1.56709472 -0.33342212  2.19270702
1.72070887 -0.23679905  0.9260987  2.0975747  1.36032616  0.04367965
-0.03780874  0.85231703 -0.83968494  0.89244172 -0.43986035]
```

- напиши примерную зависимость с округлением

\

ChatGPT

Результаты

Общая Оценка Моделей

В рамках исследования были рассмотрены и оценены различные модели машинного обучения для задачи преобразования семи эмоций (seven_fields) в двумерное пространство Valence и Arousal (pa_fields). Ниже представлены результаты этих моделей, оцененные с испол MAE и R2, а также определенные наилучшие параметры для каждой модели:

Индекс	Модель	MSE	MAE	R2	Лучшие параметры
0	Линейная регрессия	0.004808	0.04077	0.8026	{}
1	Полиномиальная регрессия	0.004114	0.03625	0.8107	{'polynomialfeatures__degree': 2}
2	Ridge	0.004791	0.04070	0.8029	{'estimator__alpha': 0.1}
3	Lasso	0.05923	0.1507	-0.0011	{'estimator__alpha': 0.1}
4	Случайный лес	0.001817	0.02182	0.9213	{'estimator__max_depth': 20, 'estimator__n_estimators': 100}
5	KNN	0.002847	0.02792	0.8730	{'estimator__n_neighbors': 7, 'estimator__weights': 'distance'}
6	MLPRegressor	0.003816	0.03501	0.8237	{'estimator__alpha': 0.001, 'estimator__hidden_layer_sizes': (10, 100)}
7	Dummy Mean	0.05923	0.1507	-0.0011	{}

Анализ Линейной Регрессии

Для линейной регрессии коэффициенты модели показывают влияние каждого из семи эмоциональных состояний на предсказания Va. Коэффициенты модели следующие:

- Neutral: -0.2605
- Happy: 0.3668
- Sad: -0.5197
- Angry: -0.5239
- Surprised: 0.0378
- Scared: -0.1731
- Disgusted: -0.5703

Итоговая формула линейной регрессии для прогнозирования Valence и Arousal может быть представлена как:

$$Valence, Arousal = -0.2605 \times Neutral + 0.3668 \times Happy - 0.5197 \times Sad - 0.5239 \times Angry + 0.0378 \times Surprised - 0.1731 \times Scared - 0.5703 \times Disgusted$$

Анализ Полиномиальной Регрессии

Полиномиальная регрессия с градусом 2 учитывает не только основные эмоциональные состояния, но и их взаимодействия. Коэффициенты регрессии указывают на более сложные зависимости между эмоциональными состояниями и пространством Valence-Arousal. Пример округленными коэффициентами:

$$Valence, Arousal \approx 0.09 \times Neutral^2 + 1.08 \times Happy - 0.59 \times Sad - 0.75 \times Angry - 0.004 \times Surprised - 0.55 \times Scared$$

Заключение

Модели машинного обучения показали различную эффективность в прогнозировании двумерного пространства Valence и Arousal на основе состояний. Случайный лес показал наилучшие результаты с точки зрения всех трех метрик, указывая на его превосходную способность сложных нелинейных зависимостей в данных. Напротив, модели Lasso и Dummy Mean показали низкую производительность. Коэффициенты полиномиальной регрессии предоставили ценные инсайты о взаимосвязи между эмоциональными состояниями и их проекцией на пр



Anonymous

Предварительный анализ данных
PCA 2 component (при написании отчета представь что будет график сошлись на него)
Explained Variance PCA - Data
array([0.57175722, 0.19429953])

Explained Variance by Different Number of Components (так же график будет)
[0.57175722 0.19429953 0.09981889 0.07290725 0.04161857 0.01398046]

0.00561808]

```
Neutral Happy Sad Angry Surprised Scared \
Neutral 1.000000 -0.577742 -0.213979 -0.341866 -0.211052 -0.116142
Happy -0.577742 1.000000 -0.130199 -0.143711 -0.110180 -0.075691
Sad -0.213979 -0.130199 1.000000 0.038683 -0.011587 -0.025535
Angry -0.341866 -0.143711 0.038683 1.000000 -0.102100 0.048501
Surprised -0.211052 -0.110180 -0.011587 -0.102100 1.000000 -0.002779
Scared -0.116142 -0.075691 -0.025535 0.048501 -0.002779 1.000000

Disgusted -0.289611 -0.087946 0.023565 0.022099 -0.057001 -0.033941
Valence -0.094548 0.801469 -0.306191 -0.555502 -0.012915 -0.136005
Arousal -0.704974 0.248603 0.121598 0.268574 0.352921 0.224170
```

```
Disgusted Valence Arousal
Neutral -0.289611 -0.094548 -0.704974
Happy -0.087946 0.801469 0.248603
Sad 0.023565 -0.306191 0.121598
Angry 0.022099 -0.555502 0.268574
Surprised -0.057001 -0.012915 0.352921
Scared -0.033941 -0.136005 0.224170
Disgusted 1.000000 -0.394410 0.132827
Valence -0.394410 1.000000 -0.035994
Arousal 0.132827 -0.035994 1.000000
```

Также будет график.

Напиши отчет сделав выводы и объяснение.



ChatGPT

Предварительный Анализ Данных

Анализ Главных Компонент (PCA)

Анализ главных компонент (PCA) был проведен для изучения структуры и взаимосвязей в данных, состоящих из семи эмоциональных PCA в данном контексте - уменьшить размерность данных, сохраняя при этом максимальное количество информации.

1. **Объясненная Дисперсия для Двух Компонентов:** Первые две главные компоненты объясняют примерно 57.18% и 19.43% дисперсии данных, что указывает на то, что большая часть информации в данных может быть представлена в двумерном пространстве, что является значительным сокращением по сравнению с исходным семимерным пространством эмоций.
2. **Объясненная Дисперсия по Различному Количеству Компонентов:** Последующие компоненты добавляют меньшие доли к общей объясненной дисперсии, подтверждая эффективность сокращения размерности. График, иллюстрирующий это, помогает визуализировать, как добавление новых компонент увеличивает общую объясненную дисперсию.

Корреляционный Анализ

Корреляционный анализ эмоциональных состояний и двухмерного пространства Valence-Arousal был выполнен для понимания взаимосвязей между различными переменными.

1. **Корреляция между Эмоциональными Состояниями:** Наблюдается значительная отрицательная корреляция между 'Neutral' и 'Happy' (-0.705), что указывает на то, что эти состояния имеют тенденцию быть противоположными. Также обнаружена положительная корреляция между 'Happy' и 'Valence' (0.801), что логично, учитывая природу этих переменных.
2. **Корреляция с Valence и Arousal:** 'Valence' имеет сильную положительную корреляцию с 'Happy' и отрицательную с 'Angry' и 'Disgusted'. 'Arousal', с другой стороны, имеет более сильную корреляцию с 'Surprised' и 'Angry'.

Выводы

- PCA подтвердил возможность сокращения размерности эмоциональных данных без значительной потери информации.
- Корреляционный анализ выявил ключевые взаимосвязи между различными эмоциональными состояниями и их проекциями на пространство Valence-Arousal.
- Эти результаты предоставляют важную основу для дальнейшего моделирования и анализа, позволяя лучше понять, как различные эмоции взаимодействуют и проявляются в двумерном пространстве.



Anonymous

def perform_and_analyze(model_name, data):

```
def perform_grid_search(model, params, name):
    grid_search = GridSearchCV(model, params, cv=3, scoring='neg_mean_squared_error', n_jobs=-1)
    grid_search.fit(X_train, y_train)
    best_model = grid_search.best_estimator_
    y_pred = best_model.predict(X_test)
    mse = mean_squared_error(y_test, y_pred)
    mae = mean_absolute_error(y_test, y_pred)
    r2 = r2_score(y_test, y_pred)
    return {'Model': name, 'MSE': mse, 'MAE': mae, 'R2': r2, 'Best Params': grid_search.best_params_}

grid_models = [
    (MultiOutputRegressor(LinearRegression()), {}, 'Linear Regression'),
    (make_pipeline(PolynomialFeatures(), LinearRegression()),
     {'polynomialfeatures__degree': [2]}, 'Polynomial Regression'),
    (MultiOutputRegressor(Ridge()), {'estimator__alpha': [0.1, 1, 10]}, 'Ridge'),
    (MultiOutputRegressor(Lasso()), {'estimator__alpha': [0.1, 1, 10]}, 'Lasso'),
    (MultiOutputRegressor(RandomForestRegressor(random_state=42)),
     {'estimator__n_estimators': [10, 50, 100], 'estimator__max_depth': [None, 3, 5, 10, 20]}, 'Random Forest'),
    (MultiOutputRegressor(KNeighborsRegressor()),
     {'estimator__n_neighbors': [3, 5, 7, 20], 'estimator__weights': ['uniform', 'distance']}, 'KNN'),
    (MultiOutputRegressor(MLPRegressor(random_state=42)),
     {'estimator__hidden_layer_sizes': [(50,50), (100,), (15, 5, 15), (50, 30, 10, 30, 50),
                                         (100, 50, 30, 10, 5, 10, 30, 50, 100),
                                         (50, 10, 5, 10, 50),
                                         (10, 5, 5, 2, 5, 5, 10)], 'estimator__alpha': [0.001, 0.01]}, 'MLPRegressor')
    ]
grid_models.append(
    (MultiOutputRegressor(DummyRegressor(strategy="mean")), {}, 'Dummy Mean')
)

grid_results = pd.DataFrame(columns=['Model', 'MSE', 'MAE', 'R2', 'Best Params'])

for model, params, name in grid_models:
    result = perform_grid_search(model, params, name)
    grid_results = grid_results.append(result, ignore_index=True)

grid_results
```

Я хочу сохранить лучшую модель на диск. потом ее считать

ChatGPT

Чтобы сохранить лучшую модель, полученную в результате Grid Search, на диск и затем считать её, вы можете использовать модуль `joblib` из `sklearn.externals`. `joblib` позволяет эффективно сериализовать большие данные, особенно массивы `numpy`, которые используют модели.

Вот как вы можете сохранить и загрузить модель:

Шаг 1: Сохранение Модели

После выполнения Grid Search и получения `grid_results`, вы можете выбрать лучшую модель на основе метрик, например, с наимен

```
python

from sklearn.externals import joblib

# Выбор лучшей модели (пример - выбор по наименьшему MSE)
best_model_name = grid_results.loc[grid_results['MSE'].idxmin()]['Model']
best_model = [model for model, _, name in grid_models if name == best_model_name][0]

# Обучение лучшей модели на всем наборе данных
best_model.fit(X_train, y_train)

# Сохранение модели
```



```
joblib.dump(best_model, 'best_model.pkl')
```

Шаг 2: Загрузка Модели

Когда вам нужно будет использовать модель в будущем, вы можете легко загрузить её с диска:

```
python

# Загрузка модели
loaded_model = joblib.load('best_model.pkl')

# Использование загруженной модели для предсказаний или других операций
y_pred = loaded_model.predict(X_test)
```

Этот подход гарантирует, что вы можете сохранять, передавать и повторно использовать свои обученные модели без необходимости г каждый раз.

Anonymous

Я хочу чтобы в названии было указано какая это модель

ChatGPT

Чтобы включить название модели в имя файла при сохранении, вы можете использовать форматирование строк в Python. Это позволит название модели в имя файла. Давайте модифицируем код для сохранения модели, чтобы имя файла отражало название лучшей моде

```
python

from sklearn.externals import joblib

# Выбор лучшей модели (пример - выбор по наименьшему MSE)
best_model_name = grid_results.loc[grid_results['MSE'].idxmin()][ 'Model' ]
best_model = [model for model, _, name in grid_models if name == best_model_name][0]

# Обучение лучшей модели на всем наборе данных
best_model.fit(X_train, y_train)

# Формирование имени файла с названием модели
filename = f'best_model_{best_model_name}.pkl'

# Сохранение модели
joblib.dump(best_model, filename)
```

Теперь имя файла будет содержать название лучшей модели, что упрощает идентификацию и управление сохраненными моделями. На модель - это "Random Forest", имя файла будет best_model_Random Forest.pkl.

Для загрузки модели используйте тот же подход:

```
python

# Загрузка модели
loaded_model = joblib.load(filename)

# Использование загруженной модели для предсказаний или других операций
y_pred = loaded_model.predict(X_test)
```

Этот метод позволяет легко сохранять и идентифицировать различные модели, особенно при работе с множеством разных алгоритмо

Anonymous

Напиши раздел о используемых моделях. Используй формулы и как они обучаются.

ChatGPT

Используемые Модели и Их Обучение

В данном исследовании использовались различные модели машинного обучения для решения задачи преобразования семи эмоций: двумерное пространство Valence и Arousal. Ниже представлен обзор каждой из используемых моделей и их обучение.

1. Линейная Регрессия

Линейная регрессия - это базовая модель машинного обучения, предназначенная для предсказания непрерывных выходных значений комбинации входных признаков. Модель представляется уравнением:

$$y = X\beta + \epsilon$$

где y - выходные значения, X - матрица входных признаков, β - вектор весов, ϵ - ошибка модели. Обучение модели заключается в минимизации ошибок между предсказанными и фактическими значениями.

2. Полиномиальная Регрессия

Полиномиальная регрессия расширяет линейную регрессию, добавляя полиномиальные признаки (степени входных переменных). Это аппроксимирует нелинейные зависимости. Формула модели имеет вид:

$$y = X\beta + \epsilon$$

но X теперь включает полиномиальные признаки.

3. Ridge и Lasso Регрессии

Ridge (гребневая) и Lasso (лассо) регрессии - это варианты линейной регрессии, которые добавляют регуляризацию для предотвращения переобучения. Ridge регрессия минимизирует:

$$\text{minimize } ||X\beta - y||_2^2 + \alpha ||\beta||_2^2$$

где α - параметр регуляризации. Lasso регрессия использует L1 регуляризацию:

$$\text{minimize } ||X\beta - y||_2^2 + \alpha ||\beta||_1$$

4. Случайный Лес (Random Forest)

Случайный лес - это ансамблевая модель, состоящая из множества деревьев решений. Каждое дерево обучается на случайной подвыборке с использованием случайного подмножества признаков. Предсказание модели является усредненным результатом всех деревьев.

5. К-ближайших Соседей (KNN)

Модель KNN работает на принципе нахождения "к" ближайших обучающих примеров в пространстве признаков и предсказывает значение для этих соседей.

6. Многослойный Перцептрон (MLPRegressor)

MLPRegressor - это тип нейронной сети, состоящий из нескольких слоев нейронов, которые могут моделировать сложные нелинейные зависимости с использованием методов обратного распространения и градиентного спуска.

Обучение Моделей

Для обучения и настройки параметров всех этих моделей был использован метод Grid Search с кросс-валидацией.



Anonymous

Теперь подробнее о решении регрессий. Напиши аналитическое решение.

Про лес тоже напиши что там минимизируется и какие формулы.

KNN Тоже напиши псевдоалгоритм.



ChatGPT

Аналитическое Решение Регрессий и Описание Других Моделей

Линейная Регрессия

Линейная регрессия стремится минимизировать сумму квадратов разностей между наблюдаемыми и предсказанными значениями. А оценки весов β может быть найдено с помощью метода наименьших квадратов:

$$\beta = (X^T X)^{-1} X^T y$$

где X - матрица входных данных, y - вектор целевых значений.

Полиномиальная Регрессия

Аналитическое решение для полиномиальной регрессии аналогично линейной регрессии, но с учетом того, что входные данные трансформируются в полиномиальные признаки перед применением метода наименьших квадратов.

Ridge и Lasso Регрессии

Решения для Ridge и Lasso регрессий включают оптимизацию функций потерь с регуляризацией. Для Ridge:

$$\beta = (X^T X + \alpha I)^{-1} X^T y$$

где α - параметр регуляризации, I - единичная матрица. Lasso регрессия не имеет аналитического решения из-за L1 регуляризации и требует численных методов оптимизации.

Случайный Лес (Random Forest)

В случайном лесе каждое дерево решений стремится минимизировать функцию потерь, например, среднюю квадратичную ошибку (MSE). Особенностью случайного леса является использование ансамбля деревьев, где каждое дерево обучается на подвыборке данных с использованием подмножества признаков. Финальное предсказание получается путем усреднения предсказаний отдельных деревьев.

K-ближайших Соседей (KNN)

KNN не имеет явной тренировочной фазы. Вместо этого, он использует следующий псевдоалгоритм при предсказании:

1. Для данного тестового примера x_{test} , вычислить расстояние до каждого обучающего примера.
2. Выбрать k обучающих примеров, которые находятся ближе всего к x_{test} .
3. Прогнозировать выходное значение как среднее (для регрессии) или как наиболее частое значение (для классификации) среди выбранных. Каждая из этих моделей имеет свои особенности и подходит для определенных типов задач. Выбор модели зависит от характеристик данных, интерпретируемости результатов.



Anonymous

отдельно про дерево. как оно выбирает узел. математически пиши не ленись падла



ChatGPT

Деревья Решений

Дерево решений - это модель машинного обучения, используемая как для классификации, так и для регрессии. Она строит модель при древовидной структуре.

Как Дерево Решений Выбирает Узел

1. **Выбор Признака и Разбиения:** Для каждого узла дерева алгоритм выбирает наилучший признак для разбиения данных на подгруппы. Е метрике, такой как прирост информации (для классификации) или уменьшение среднеквадратичной ошибки (для регрессии).
2. **Метрики для Регрессионных Деревьев:**
 - **Среднеквадратичная ошибка (MSE):**

$$MSE = \frac{1}{n} \sum_{i=1}^n (y_i - \hat{y})^2$$

где y_i - фактические значения, \hat{y} - среднее значение ответов в узле, n - количество точек данных в узле.
 - **Уменьшение MSE:**

Определяется как разность между MSE до и после разбиения. Узел с наибольшим уменьшением MSE выбирается как лучшее разбиение.
3. **Процесс Разбиения:**
 - Для каждого кандидата на разбиение (например, каждого значения признака) вычислить уменьшение MSE.
 - Выбрать точку разбиения, которая приводит к наибольшему уменьшению MSE.
 - Разделить данные на две подгруппы на основе этого разбиения.
4. **Остановка Роста Деревя:**

Рост дерева останавливается, когда достигнут один из критериев: достижение максимальной глубины, минимального числа образцов MSE становится незначительным.

Деревья решений имеют несколько преимуществ, включая легкость интерпретации и способность обрабатывать как числовые, так и категориальные данные. Однако они могут быть склонны к переобучению, особенно в случае сложных данных. Для устранения этой проблемы часто используют ансамбли, такие как случайный лес, который строит множество деревьев и усредняет их предсказания.

Get started with ChatGPT

[Report content](#) | [Terms of use](#) | [Privacy policy](#) | [Manage cookies](#)