Методы очищения данных: устранение пропусков, исправление ошибок, стандартизация значений.

Обнаружение пропусков. Для обнаружения пропусков можно воспользоваться методами info(), isna(), или тепловой картой (рис.1).

```
[ ]  import seaborn as sns
     import matplotlib.pyplot as plt

     plt.figure(figsize=(10,6))
     sns.heatmap(df.isna().transpose());
```

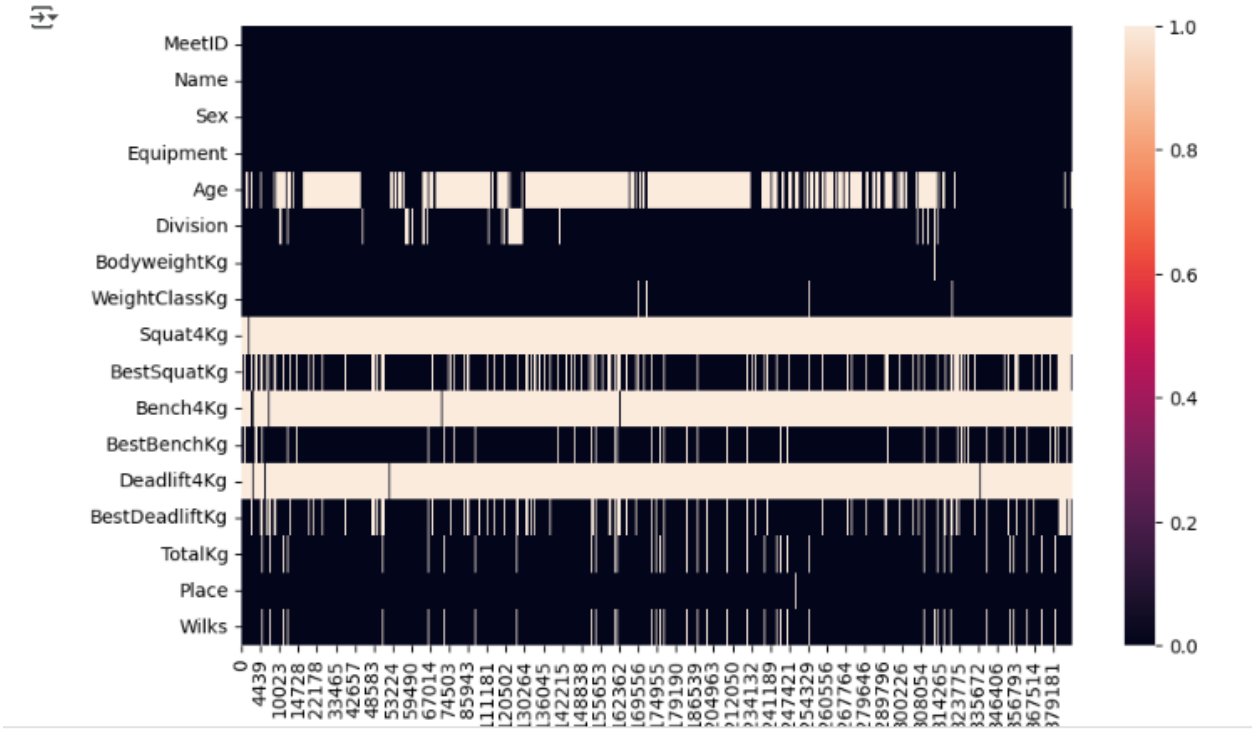

Рисунок 1 – Обнаружение пропуском при помощи тепловой карты

Если более 70% данных в колонке отсутствует, такой признак будет очень сложно восстановить и его лучше исключить из анализа (рис 2).

Удалить столбцы/строки с пропусками

```
[ ]  cols_to_drop = []

     for col in df.columns:
         if df[col].isna().sum() / df.shape[0] > 0.7:
             cols_to_drop.append(col)

     cols_to_drop
```

```
['Squat4Kg', 'Bench4Kg', 'Deadlift4Kg']
```

Рисунок 2 – Обнаружение признаков с большой долей отсутствующих значений

Устранение пропусков может быть достигнуто несколькими способами: удалить строки/столбцы с пропусками, заполнить пропуски, закодировать, интерполировать.

Метод **dropna()**

Метод dropna() в pandas используется для удаления пропущенных значений (NaN, None или подобных) из DataFrame или Series. Он позволяет убрать строки или столбцы, содержащие пропуски, согласно заданным параметрам.

Основные параметры метода dropna() для DataFrame:

axis: определяет, удалять ли строки (axis=0 или 'index', значение по умолчанию) или столбцы (axis=1 или 'columns') с пропущенными значениями.

how: указывает условие удаления:

'any' (по умолчанию) — удалить, если есть хотя бы один пропуск,

'all' — удалить, только если все значения пропущены.

thresh: целое число, указывающее минимальное количество непустых (не NaN) значений, чтобы сохранить строку или столбец.

subset: список колонок, по которым проверять пропуски. Если указан, проверка и удаление делается только по этим столбцам.

inplace: булево значение, если True, метод изменит исходный DataFrame без создания копии, иначе вернёт новый DataFrame (по умолчанию False).

Примеры использования:

df.dropna() — удаляет все строки, содержащие хотя бы один NaN.

df.dropna(axis=1, how='all') — удалит столбцы, в которых все значения пропущены.

df.dropna(thresh=2) — оставит строки, в которых как минимум 2 непустых значения.

df.dropna(subset=['A', 'B']) — удалит строки, в которых в столбцах 'A' или 'B' есть пропуски.

Метод возвращает изменённый DataFrame (если inplace=False) или None (если inplace=True).

Для Series метод dropna() работает аналогично, удаляя пропуски из одномерного массива.

Таким образом, dropna() гибко управляет удалением пропусков в данных с помощью перечисленных атрибутов для эффективной предобработки данных.

Метод fillna() в pandas используется для заполнения пропущенных значений (NA/NaN/None) в DataFrame или Series заданными значениями или методами заполнения.

Основные параметры метода fillna() для DataFrame:

value: скаляр, словарь, Series или DataFrame — значение (или набор значений), которыми надо заполнить пропуски. Например, число 0 или словарь с соответствием столбец:значение. Если передан словарь/Series, заполнение происходит по названиям индексов/столбцов.

method: один из {'ffill', 'pad', 'backfill', 'bfill'} или None (по умолчанию None). Метод заполнения:

'ffill' или 'pad' — заполнить пропуски предыдущим действительным значением по оси,

'backfill' или 'bfill' — заполнить пропуски следующим действительным значением.

axis: 0 или 'index' (по умолчанию), 1 или 'columns' — вдоль какой оси проводить заполнение.

inplace: булево, по умолчанию False. Если True, заменяет значения в исходном объекте без создания копии.

limit: целое число или None, ограничивает максимальное число заполняемых подряд пропусков.

downcast: словарь типов для попытки приведения типов после заполнения (редко используется).

Примеры:

df.fillna(0) — заменить все пропуски на 0.

df.fillna({'A': 0, 'B': 1}) — заполнить пропуски в столбце 'A' нулями, в столбце 'B' — единицами.

df.fillna(method='ffill') — заполнить пропуски предыдущими значениями по умолчанию по строкам.

df.fillna(method='bfill', limit=1) — заполнить пропуски следующим значением, не более одного подряд.

Метод возвращает новый DataFrame/Series с заполненными значениями, если inplace=False, иначе возвращает None и меняет данные на месте.

Для заполнения пропусков в **количественных признаках** медианными значениями в pandas можно использовать метод fillna() вместе с вычислением медианы столбца через метод median(). На рисунке 3 видно, что пропуск в признаке A заполнен медианным значением.

```python
import pandas as pd
import numpy as np

# Пример DataFrame с пропусками
df = pd.DataFrame({
    'A': [1, 2, np.nan, 4, 5],
    'B': [np.nan, 2, 3, np.nan, 5]
})

# Вычисляем медиану столбца 'A'
median_A = df['A'].median()

# Заполняем пропуски в столбце 'A' медианой
df['A'].fillna(median_A, inplace=True)

print(df)
```

```
     A    B
0  1.0  NaN
1  2.0  2.0
2  3.0  3.0
3  4.0  NaN
4  5.0  5.0
```

Рисунок 3 – Заполнение пропусков в количественном признаке медианным значением

Для заполнения пропуском в **качественном признаке** можно воспользоваться модальным значением, как показано на рисунке 4.

```python
import pandas as pd
import numpy as np

# Пример DataFrame с категориальным признаком и пропусками
df = pd.DataFrame({
    'Category': ['Red', 'Blue', np.nan, 'Green', 'Blue', np.nan, 'Red']
})

# Заполняем пропуски модой (самым частым значением)
mode_value = df['Category'].mode()[0]
df['Category'].fillna(mode_value, inplace=True)

print(df)
```

```
  Category
0      Red
1     Blue
2     Blue
3    Green
4     Blue
5     Blue
6      Red
```

Рисунок 4 – Заполнение пропусков в качественном признаке

Так же для работы с пропусками часто применяется создание индикатора – столбца, с булевыми значениями — 1 для строк, где есть пропуск, и 0 там, где пропуска нет. Такой столбец помогает моделям и анализу учитывать факт отсутствия данных как отдельную характеристику. Так, на рисунке 5 показано создание индикатора для пропуска в столбце A.

```python
import pandas as pd
import numpy as np

# Пример DataFrame с пропусками
df = pd.DataFrame({
    'A': [1, 2, np.nan, 4, 5],
    'B': [np.nan, 2, 3, np.nan, 5]
})

# Создаем индикатор пропусков для столбца 'A'
df['A_missing'] = 0
df.loc[df['A'].isna(), 'A_missing'] = 1

print(df)
```

```
     A    B  A_missing
0  1.0  NaN          0
1  2.0  2.0          0
2  NaN  3.0          1
3  4.0  NaN          0
4  5.0  5.0          0
```

Рисунок 5 – Создание индикатора

Интерполяция. Пусть имеется ряд, приведенный на рисунке 6.

```
x = np.linspace(-10, 10, num=50)
y = 2.5 * np.sin(x) + x

plt.figure(figsize=(7, 4))
plt.scatter(x, y);
```

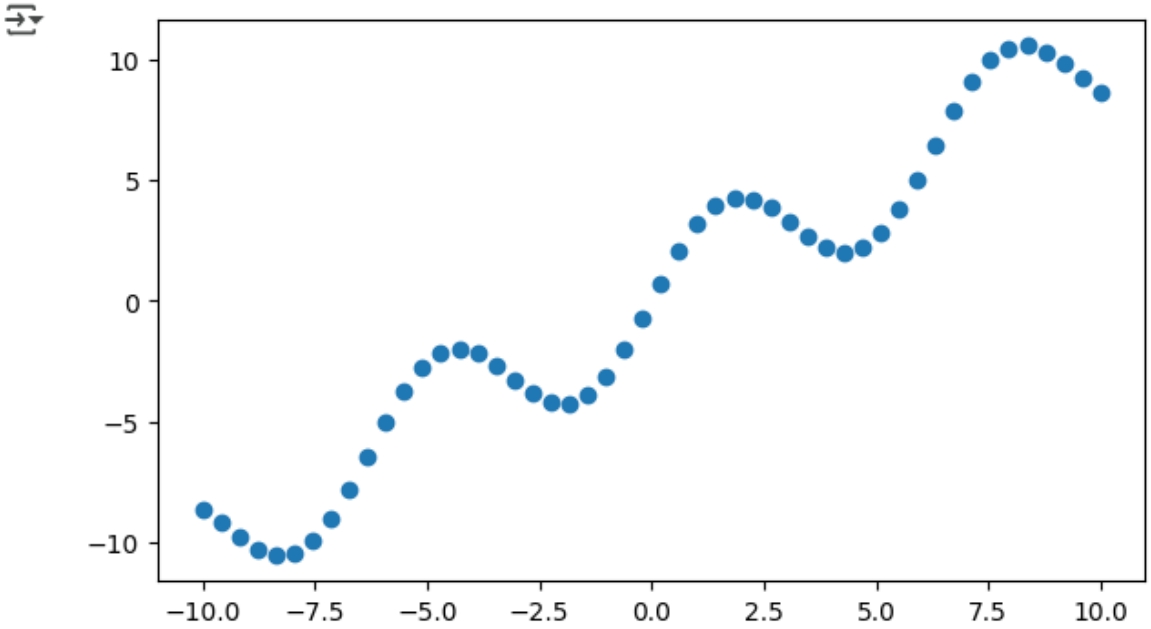

Рисунок 6 – Исходные данные для задачи

Исключим из ряда несколько точек, чтобы показать разные способы интерполяции. На рисунке 7 синим цветом показаны реальные данные, которые были потеряна и требуют восстановления.

```
plt.figure(figsize=(10, 5))
plt.scatter(nan_x, nan_y, label='nan')
plt.scatter(df['x'], df['y'])
plt.legend();
```

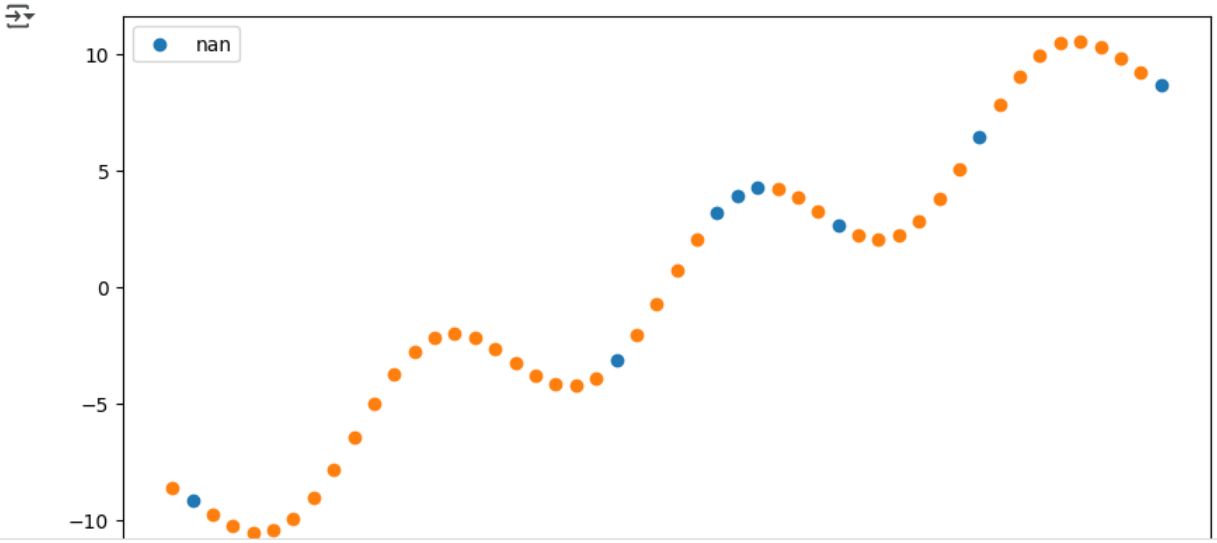

Рисунок 7 – Данные с пропусками

Линейная интерполяция. Метод interpolate() в pandas используется для заполнения пропущенных значений (NaN) в DataFrame или Series с помощью различных техник интерполяции — то есть оценки недостающих данных на основе окружающих значений.

По умолчанию действует линейная интерполяция (method='linear'), которая постепенно заполняет пропуски, рассчитывая промежуточные значения между известными точками.

Метод поддерживает разные варианты заполнения:

'linear' — линейная интерполяция (по умолчанию),

'time' — для временных рядов учитывает интервалы времени,

'index' — по числовым значениям индекса,

'values' — по значениям данных,

'nearest' — заполнение ближайшим известным значением,

'pad' (или 'ffill') — заполнение предыдущим известным значением,

'backfill' (или 'bfill') — заполнение следующим известным значением,

'polynomial' — полиномиальная интерполяция с заданным порядком (через параметр order),

и другие (например, 'spline', 'quadratic', 'cubic' — применяют методы из библиотеки SciPy).

Некоторые важные параметры:

axis — ось для интерполяции (0 — по строкам, 1 — по столбцам),

limit — максимальное число подряд заполняемых пропусков,

limit_direction — направление заполнения: 'forward', 'backward', 'both',

inplace — изменять исходный объект на месте (True) или вернуть новую копию (False, по умолчанию),

order — степень полинома при методе 'polynomial'.

Пример показан на рисунке 8.

```
df['y_lin'] = df['y'].interpolate()
plt.figure(figsize=(7, 4))
plt.scatter(nan_x, nan_y, label='nan')
plt.scatter(df['x'], df['y_lin'], alpha=0.5)
plt.legend();
```

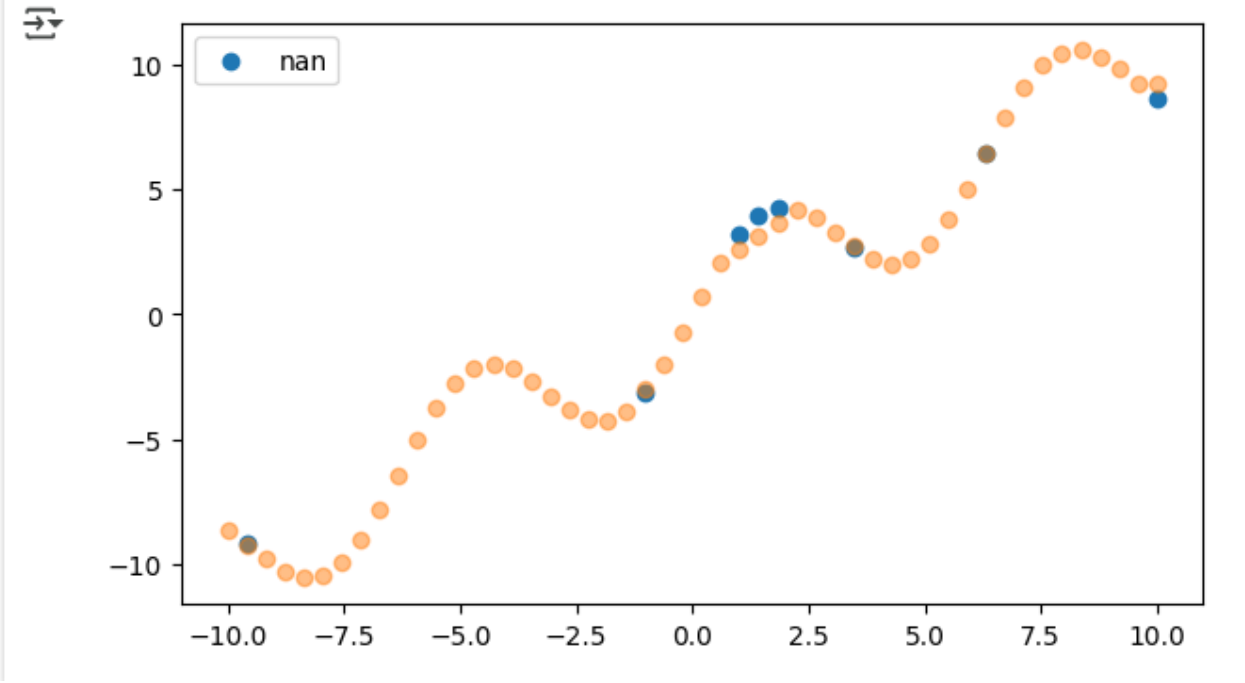

Рисунок 8 – Линейная интерполяция

На рисунке 8 видно, что данные восстановились не точно, т.к. линейная интерполяция не способна восстановить нелинейную зависимость.

Padding. Метод interpolate(method='pad', limit=2) в pandas используется для заполнения пропущенных значений (NaN) путём заполнения их ближайшим предыдущим существующим значением (аналогично заполнению методом "forward fill" или "pad").

Атрибуты:

method='pad' означает, что пропуски заменяются последним валидным (ненулевым) значением, которое встречается выше по индексу (для Series) или по выбранной оси (для DataFrame). Это эквивалентно методу fillna(method='pad') или fillna(method='ffill').

limit=2 ограничивает максимальное количество подряд идущих пропущенных значений, которые будут заполнены этим методом. То есть, если подряд идут более двух пропусков, будут заполнены только первые два, а дальше останется NaN.

По умолчанию limit_direction равен 'forward', что логично для pad — мы заполняем пропуски значениями, идущими сверху вниз.

Если не указать inplace=True, метод возвращает копию с заполненными значениями, не меняя исходный объект.

Таким образом, interpolate(method='pad', limit=2) позволяет аккуратно заполнить подряд идущие пропуски повторением предыдущего значения, но с контролем на максимальное количество замен подряд. Это полезно, если хотите избегать слишком долгого протягивания одного значения.

Этот метод полезен при обработке временных рядов или последовательных данных, где логично заполнить пропуски последними наблюденными значениями с ограничением по длине заполнения (рисунок 9).

```
df['y_pad'] = df['y'].interpolate(method='pad', limit=2)
plt.figure(figsize=(7, 4))
plt.scatter(nan_x, nan_y, label='nan')
plt.scatter(df['x'], df['y_pad'], alpha=0.5)
plt.legend();
```

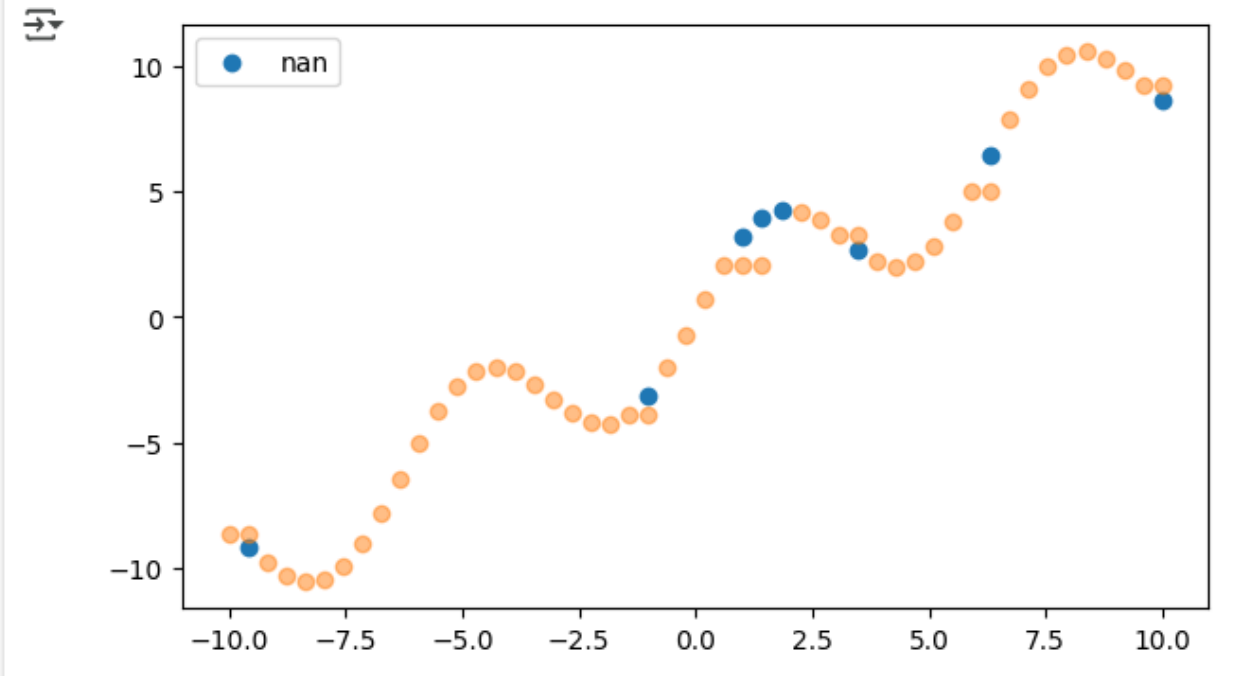

Рисунок 9 – Заполнение данных при помощи паддинга

bfill. В pandas параметр method='bfill' (или method='backfill') в методах заполнения пропущенных значений (fillna() или interpolate()) означает следующий принцип заполнения: пропущенные значения (NaN) заполняются следующими (следующими по индексу) известными ненулевыми значениями. Если в строке или столбце встречается NaN, он будет заменён ближайшим валидным значением, идущим вниз по оси (вперёд по индексу). Аналогом является fillna(method='backfill'). Пример показан на рисунке 10.

```
df['y_bfill'] = df['y_bfill'].fillna(method="bfill")
plt.figure(figsize=(7, 4))
plt.scatter(nan_x, nan_y, label='nan')
plt.scatter(df['x'], df['y_bfill'], alpha=0.5)
plt.legend();
```

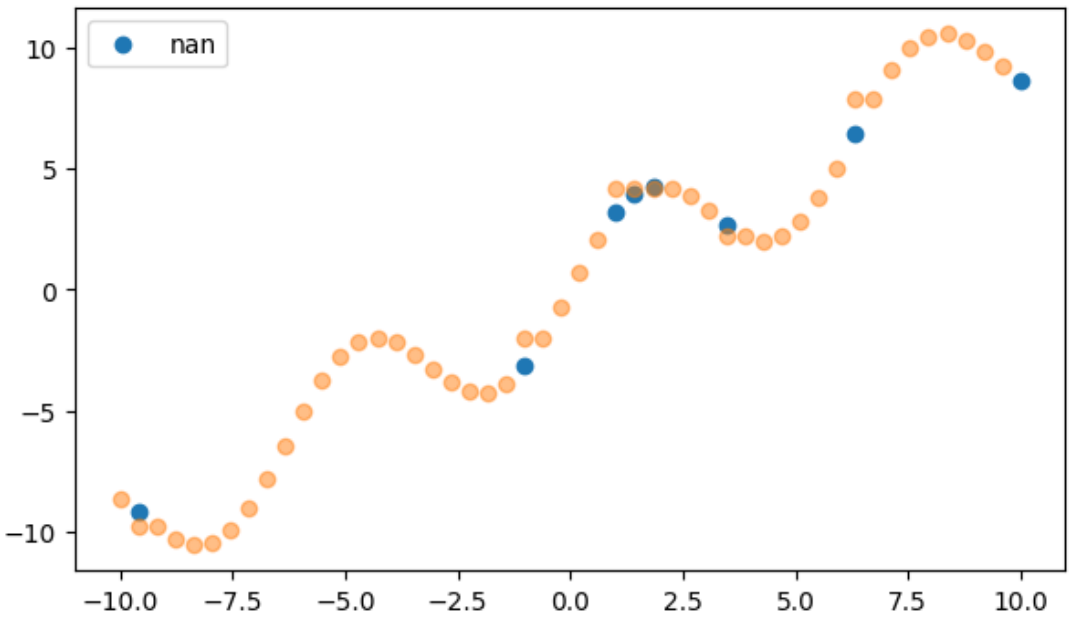

Рисунок 10 – Заполнение пропусков при помощи параметра bfill

Nearest. Метод interpolate() в pandas с параметром method='nearest' заполняет пропущенные значения (NaN) значениями из ближайших существующих (ненулевых) точек данных. То есть каждый пропуск заменяется значением, которое находится максимально близко к нему по индексу или оси — без вычисления новых значений (как при линейной или полиномиальной интерполяции). Метод не создаёт новых значений, а лишь "копирует" уже существующие. Он подходит для категориальных данных или числовых данных, где важно сохранить оригинальные значения. Если пропуск находится одинаково близко к двум значениям, выбирается одно из ближайших. Параметр limit ограничивает максимальное число подряд заполненных пропусков этим методом.

```
df['y_nearest'] = df['y'].interpolate(method='nearest')
plt.figure(figsize=(7, 4))
plt.scatter(nan_x, nan_y, label='nan')
plt.scatter(df['x'], df['y_nearest'], alpha=0.5)
plt.legend();
```

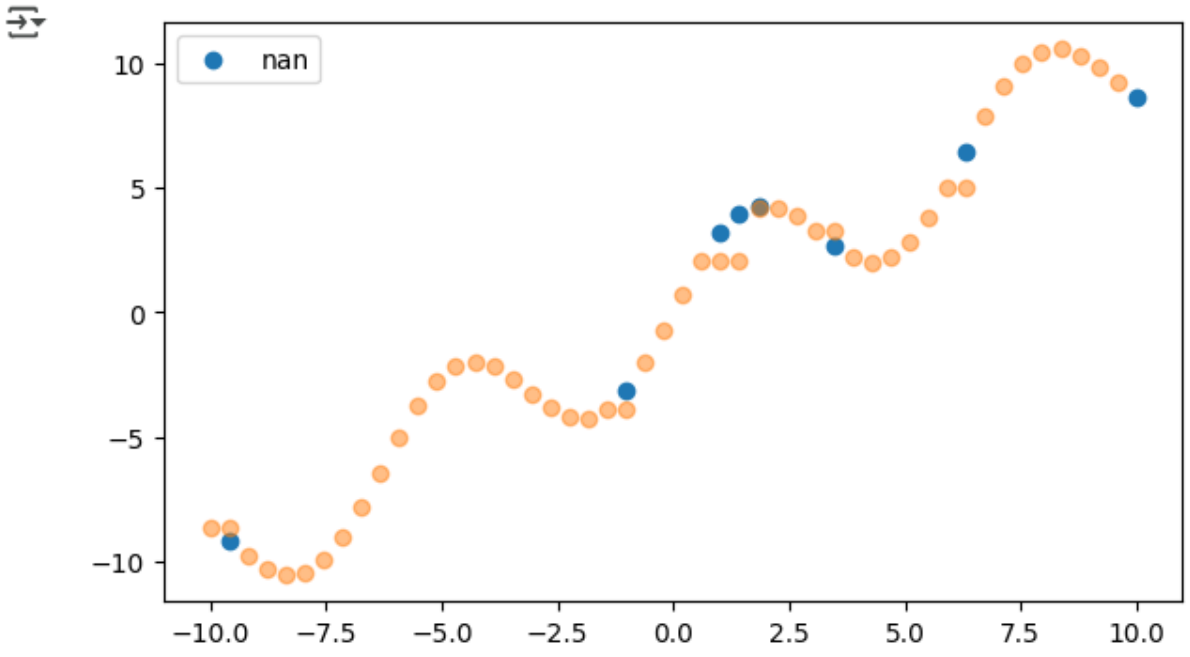

Рисунок 11 – Применение метода nearest

Polynomial. В pandas метод interpolate() с параметром method='polynomial' используется для заполнения пропущенных значений (NaN) с помощью полиномиальной интерполяции.

Метод 'polynomial' использует числовые значения индекса для построения полинома, который аппроксимирует (приближает) данные вокруг пропусков.

Для работы необходимо дополнительно указать параметр order — степень полинома (целое число). Например, order=2 для квадратичной, order=3 для кубической и т.д. Полиномиальная интерполяция может лучше подходить для сглаживания и заполнения в случаях, когда данные имеют нелинейный тренд.

```
df['y_poly'] = df['y'].interpolate(method='polynomial', order=2)
plt.figure(figsize=(7, 4))
plt.scatter(nan_x, nan_y, label='nan')
plt.scatter(df['x'], df['y_poly'], alpha=0.5)
plt.legend();
```

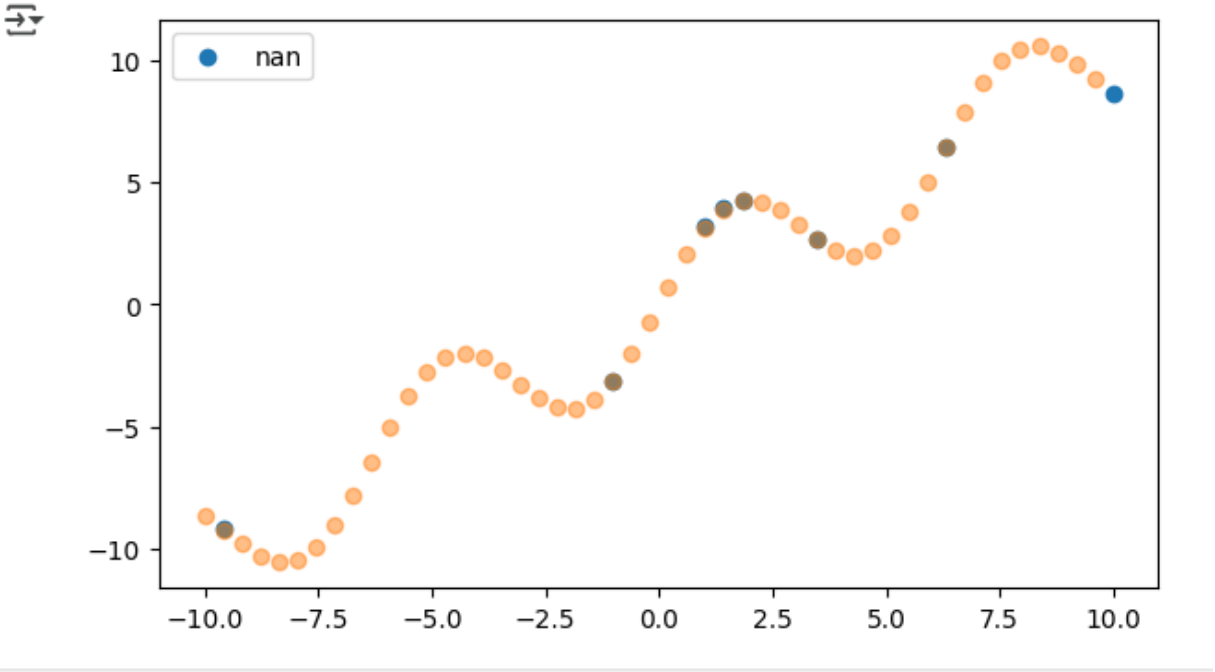

Рисунок 12 – Применение метода polynomial