Анализ данных и машинное обучение, ч. 2

Лекция 2. Трансформация данных. Квантование. Кодирование. Анализ аномалий. Понятие и классификация аномалий. LOF, Isolation Forest. ZSCORE, IQR. HBOS. Восстановление пропущенных значений. Винсоризация

Киреев В.С.,
к.т.н., доцент

Москва, 2025

Трансформация данных

Трансформация данных – комплекс методов и алгоритмов, направленных на оптимизацию представления и форматов данных с точки зрения решаемых задач и целей анализа. Трансформация данных не ставит целью изменить информационное содержание данных. Её задача представить эту информацию в таком виде, чтобы она могла быть использована наиболее эффективно.

Методы трансформации данных

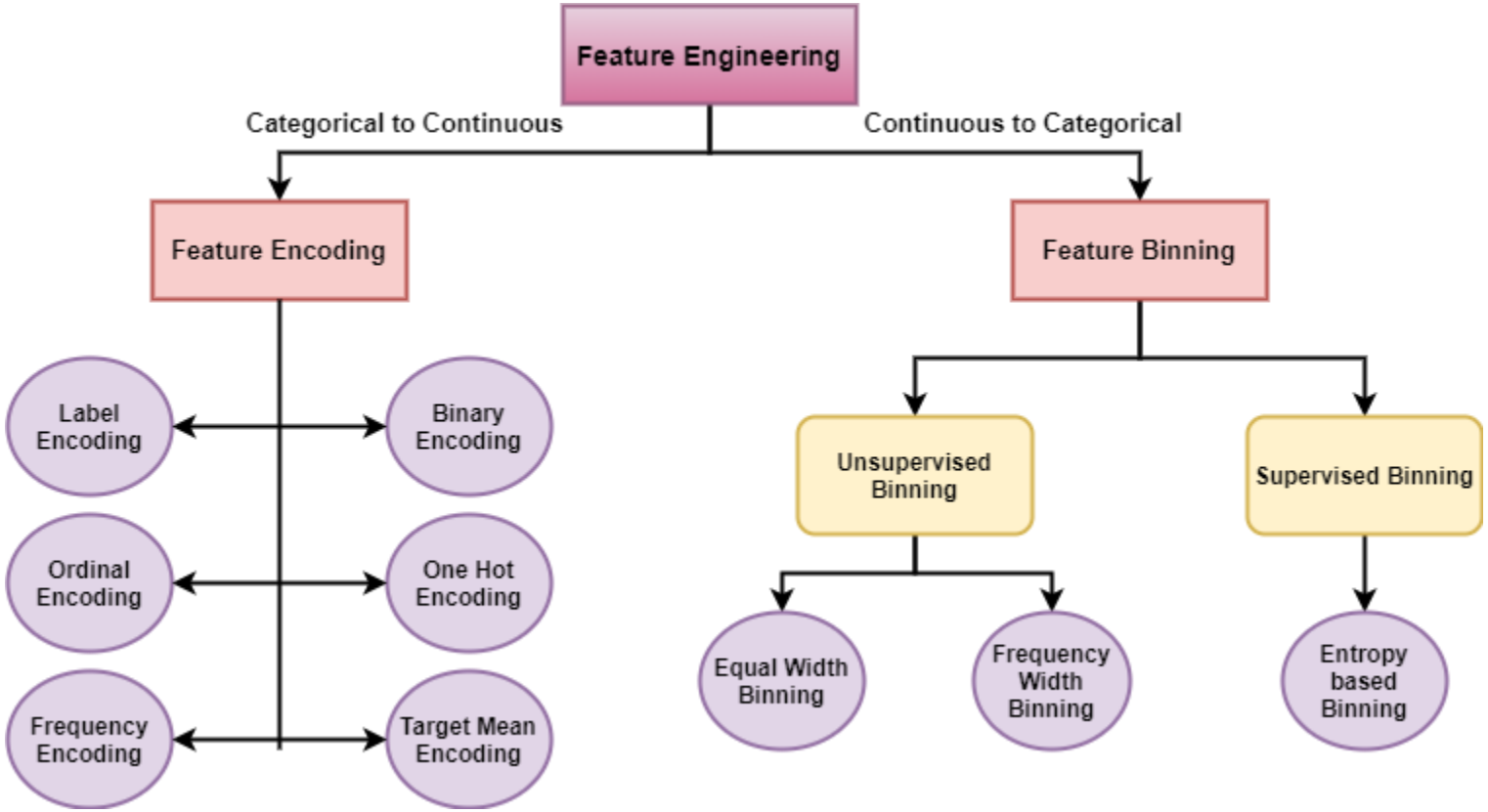
Квантование

Кодирование

Нормализация

Шкалы измерения признаков

Классификация методов трансформации данных



Квантование

Квантование – процедура преобразования данных, состоящая из 2-х шагов. На первом шаге диапазон значений переменной разбивается на заданное число интервалов, каждому из которых присваивается некоторый номер (уровень квантования). На втором шаге каждое значение заменяется номером интервала квантования.

Квантование. Виды

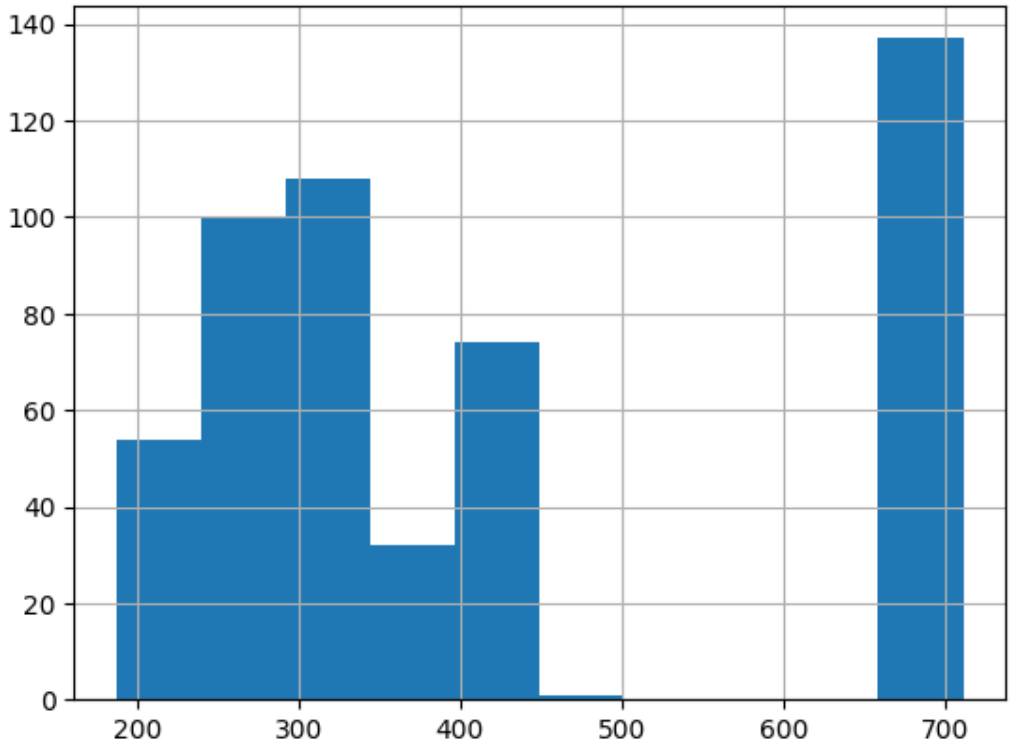
Равномерное (однородное) квантование

Неравномерное (неоднородное) квантование

Равномерное квантование

Равномерное (однородное) квантование – преобразование, при котором диапазон значений переменной разбивается на интервалы одинаковой длины. Имеет смысл, если значения распределены равномерно по всему диапазону значений.

Равномерное квантование. Пример



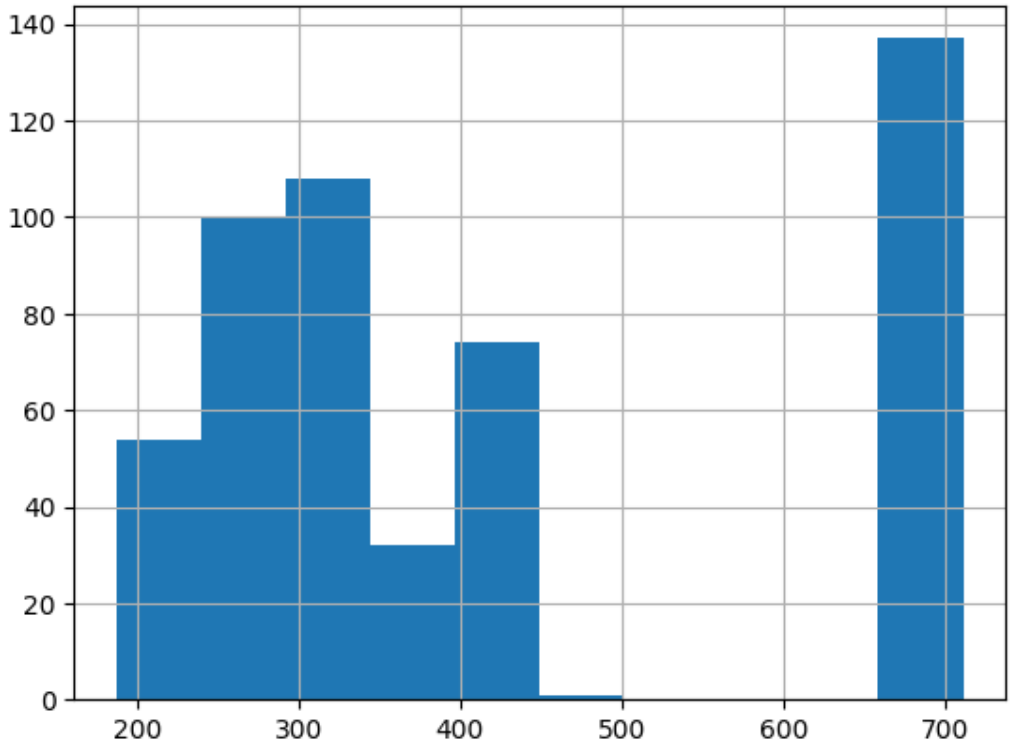

```
1 min_value = boston.TAX.min()
2 max_value = boston.TAX.max()
3
4 bins = np.linspace(min_value, max_value, 4)
5 bins
```

```
array([187.        , 361.66666667, 536.33333333, 711.        ])
```

Неравномерное квантование

Неравномерное (однородное) квантование – преобразование, при котором диапазон значений переменной разбивается на интервалы различной длины (асимметричные). Имеет смысл, если в значениях нет пропусков или сгустков.

Неравномерное квантование. Пример



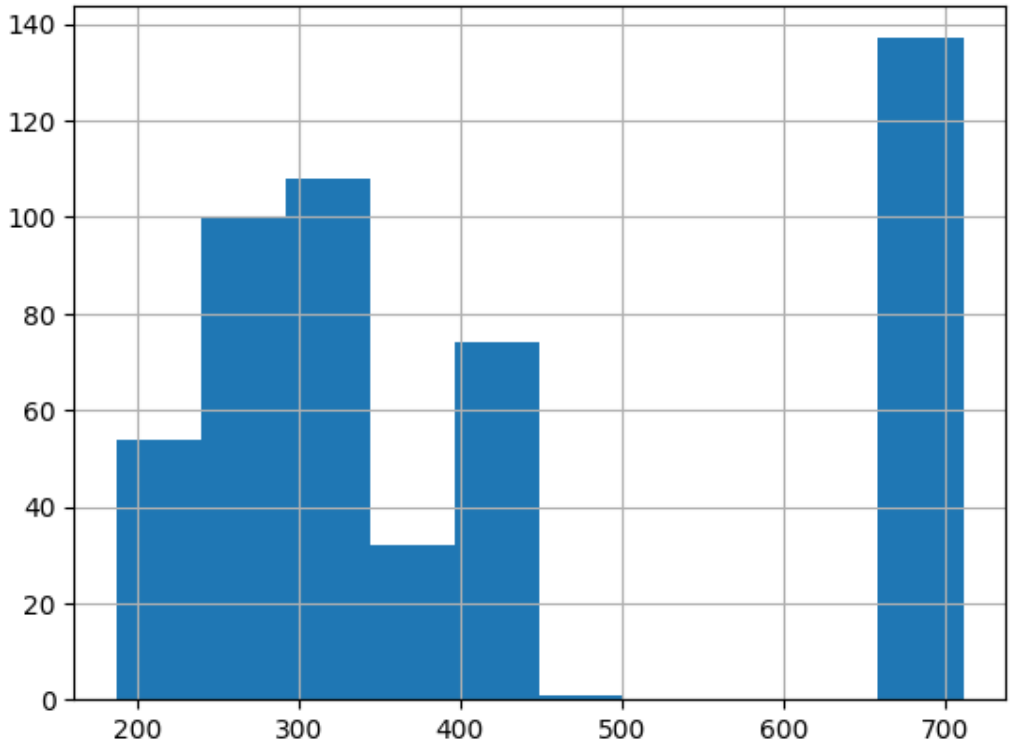

```
1 boston['TAX_qbinned'], boundaries = pd.qcut(boston.TAX,
2                                             q = 3,
3                                             precision = 1,
4                                             retbins = True)
5
6 boundaries
```

```
array([187., 300., 403., 711.])
```

Алгоритм Дженкса

Алгоритм естественных границ Дженкса (Jenks natural breaks optimization) делит данные на группы (кластеры) таким образом, чтобы минимизировать отклонение наблюдений от среднего каждого класса (дисперсию внутри классов) и максимизировать отклонение среднего каждого класса от среднего других классов (дисперсию между классами).

Алгоритм Дженкса. Пример




```
!pip install jenkspy
```

```
import jenkspy

breaks = jenkspy.jenks_breaks(boston.TAX, n_classes = 3)
breaks
```

```
[187.0, 337.0, 469.0, 711.0]
```

Особенности категориальных признаков

- С категориальными признаками можно столкнуться с проблемами с редкими метками, категориями /группами, которые крайне редки в наборе данных. Эта проблема часто связана с функциями, имеющими высокую мощность — другими словами, с множеством различных категорий.
- Наличие слишком большого количества категорий, и особенно редких категорий, приводит к зашумленному набору данных. Алгоритму ML может быть трудно пробиться сквозь этот шум и извлечь уроки из более значимых сигналов в данных.
- Высокая кардинальность также может усугубить проклятие размерности, если необходимо однократно закодировать свои категориальные характеристики.

Кодирование категориальных признаков. One Hot Encoding

Горячее кодирование (One Hot Encoding) используется для преобразования категориальных переменных в формат, который может быть легко использован алгоритмами машинного обучения . Основная идея горячего кодирования заключается в создании новых переменных, которые принимают значения 0 и 1 для представления исходных категориальных значений.

Кодирование категориальных признаков. One Hot Encoding. Пример

```python
from sklearn.preprocessing import OneHotEncoder
from seaborn import load_dataset

df = load_dataset('penguins')
ohe = OneHotEncoder()
transformed = ohe.fit_transform(df[['island']])
print(transformed.toarray())
```

```
[[0. 0. 1.]
 [0. 0. 1.]
 [0. 0. 1.]
 ...
 [1. 0. 0.]
 [1. 0. 0.]
 [1. 0. 0.]]
```

Кодирование категориальных признаков. Label Encoding

Label Encoder (кодирование меткой) очень прост и включает преобразование каждого значения признака в число. Label-Encoder, стоит применять когда категориальный признак является порядковым (например, низкий, средний, высокий) , и количество категорий довольно велико.

Кодирование категориальных признаков. Label Encoding. Пример

```python
from sklearn.preprocessing import LabelEncoder
from seaborn import load_dataset

df = load_dataset('penguins')
ohe = LabelEncoder()
transformed = ohe.fit_transform(df[['island']])
print(transformed)
```

```
2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 0 0 0 0 0 0 0 0 0 1 1 1 1 1 1 1
1 1 1 1 1 1 1 1 1 1 1 1 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 2 2 2 2 2 2
2 2 2 2 2 2 2 2 2 2 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 1 1 1 1 1 1 1 1 1 1 1 1 1 1
1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
```

Нормализация

Нормализация необходима, так как многие алгоритмы чувствительны к выбросам, а так же распределению данных в выборке. Z-преобразование центрирует данные, удаляет среднее значение для каждого объекта, а затем масштабирует, деля на среднее отклонение. Минимаксное шкалирование вычитает минимум из значения выборки и делит на размах.

Нормализация. Пример

```
1 from sklearn.preprocessing import StandardScaler
2 data = [[0, 0], [0, 0], [1, 1], [1, 1]]
3 scaler = StandardScaler()
4
5 print(scaler.fit(data))
6 print(scaler.transform(data))
```

```
StandardScaler()
[[-1. -1.]
 [-1. -1.]
 [ 1.  1.]
 [ 1.  1.]]
```

Обнаружение аномальных значений

Обнаружение аномальных значений (иначе - анализ выбросов или отклонений) - это этап интеллектуального анализа данных, который идентифицирует точки данных, события и/или наблюдения, которые отклоняются от нормального поведения набора данных.

Поиск и идентификация отклонений помогает предотвратить мошенничество, атаки злоумышленников и сетевые вторжения/ критические инциденты. Аномальные данные могут указывать на, потенциальные возможности, например, изменение поведения потребителей.

Понятие аномалии

Аномалия (выброс) — это наблюдение, которое существенно отличается от остальных данных и не соответствует ожидаемому поведению системы.

Применения:

- Обнаружение мошенничества
- Мониторинг кибербезопасности
- Диагностика оборудования
- Медицинская диагностика

Цель анализа аномалий: выявить редкие события, которые могут быть критичными, но неочевидными при стандартном анализе.

Типы аномалий:

- Точечные (point anomalies)
- Контекстуальные (contextual)
- Коллективные (collective)

Аномальные значения. Типизация

Выбросы	короткие/небольшие аномальные паттерны, которые проявляются несистематическим образом при сборе данных.
Изменение событий	систематическое или внезапное изменение по сравнению с предыдущим нормальным поведением.
Дрейфы	медленное, ненаправленное, долгосрочное изменение данных.

Классификация методов

Методы делятся на три основные категории:

- Статистические методы — основаны на предположении о распределении данных (нормальное, экспоненциальное и т.д.)
    - Z-score, Grubbs’ test и др.
- Методы, основанные на плотности и расстоянии — используют локальную структуру данных
    - LOF, kNN и др.
- Методы, основанные на деревьях и ансамблях — строят модели для изоляции или классификации аномалий
    - Isolation Forest, HBOS и др.
- Машинное обучение с учителем и без
    - автоэнкодеры, SVM, GAN и др.

Метод межквартильного размаха

Метод межквартильного размаха (Interquartile Range, IQR) используется для измерения изменчивости путем деления набора данных на квартили.

IQR – это диапазон между первым и третьим квартилями, а именно Q1 и Q3: IQR = Q3 - Q1. Точки данных, которые находятся ниже Q1 – 1,5 IQR или выше Q3 + 1,5 IQR, являются выбросами.

Аномальные значения. Обнаружение. Метод межквартильного размаха. Пример

```python
import numpy as np

X=[[6], [2], [3], [4], [5], [1], [100]]
q3, q1 = np.percentile(X, [75, 25])
print(q3, q1)
print(len(X))
IQR = q3 - q1

upper_bound = q3 + 1.5 * IQR
lower_bound = q1 - 1.5 * IQR

(X < upper_bound)&(X>lower_bound)
```

```
5.5 2.5
7
array([[ True],
       [ True],
       [ True],
       [ True],
       [ True],
       [ True],
       [False]])
```

ZSCORE

Алгоритм:

Для одномерной выборки $X = \{x_1, x_2, ..., x_n\}$ вычисляется:

$$z_i = \frac{x_i - \mu}{\sigma}$$

где:

- $\mu = \frac{1}{n} \sum_{i-1}^{n} x_i$ — среднее,
- $\sigma = \sqrt{\frac{1}{n} \sum_{i-1}^{n} (x_i - \mu)^2}$ — стандартное отклонение.

Правило: если $|z_i| > \tau$ (обычно $\tau = 3$), то x_i — аномалия.

ZSCORE. Характеристики

Плюсы:

- Простота и интерпретируемость
- Быстрый расчет

Минусы:

- Чувствителен к выбросам (влияют на μ и σ)
- Предполагает нормальное распределение
- Не работает в многомерном случае без модификаций
-

HBOS

Идея: оценка аномальности по гистограммам признаков. Предполагается, что признаки независимы.

Алгоритм:

1. Для каждого признака j строится гистограмма с k бинами.
2. Для точки $\mathbf{x} = (x_1, \ldots, x_d)$ вычисляется:

$$HBOS(\mathbf{x}) = \sum_{j=1}^{d} \log\left(\frac{1}{\text{hist}_j(x_j) + \epsilon}\right)$$

где $\text{hist}_j(x_j)$ — высота бина, в который попало x_j , ϵ — малая константа для избежания деления на ноль.

HBOS. Характеристики

Плюсы:

- Очень быстрый (линейная сложность)
- Хорошо масштабируется
- Не требует обучения

Минусы:

- Игнорирует корреляции между признаками
- Чувствителен к выбору числа бинов
- Плохо работает на малых выборках
-

LOF (Local Outlier Factor)

Идея:

Аномальность точки определяется не абсолютной плотностью, а **относительно плотности её соседей**. Если точка находится в "разреженной" области по сравнению с соседями — она аномальна.

Алгоритм:

1. Для каждой точки p находятся k -ближайшие соседи — обозначим как $N_k(p)$.
2. Определяется **достижимое расстояние (reachability distance)** от p до соседа o :

$$\text{reach-dist}_k(p, o) = \max\{\text{k-distance}(o),\ d(p, o)\}$$

где:

- $d(p, o)$ — евклидово расстояние между p и o ,
- $\text{k-distance}(o)$ — расстояние от o до её k -го ближайшего соседа (включая саму o ? — нет, только других точек).

3. Вычисляется **локальная достижимая плотность (local reachability density, lrd)** точки p :

$$lrd_k(p) = \frac{1}{\frac{1}{|N_k(p)|}\sum_{o \in N_k(p)} \text{reach-dist}_k(p, o)}$$

LOF (Local Outlier Factor). Характеристики

Плюсы:

- Учитывает локальную структуру данных — работает даже при неоднородной плотности.
- Эффективен на нелинейных многообразиях и сложных формах кластеров.
- Не требует предположений о распределении данных.

Минусы:

- Вычислительно затратен — $O(n2)$ для поиска всех kNN (можно ускорить через KD-Tree или Ball Tree — $O(n\log n)$ в среднем).
- Чувствителен к выбору k — слишком маленькое k → шум воспринимается как аномалия; слишком большое → “размывает” локальные особенности.
- Трудно интерпретировать абсолютные значения — LOF не даёт вероятности, только относительную аномальность.
- Не масштабируется на очень большие данные без приближённых методов/

Аномальные значения. Обнаружение. Фактор локального выброса. Пример

```
1 from sklearn.neighbors import LocalOutlierFactor
2 X=[[6], [2], [3], [4], [5], [1], [100]]
3 clf = LocalOutlierFactor(n_neighbors=2)
4 clf.fit_predict(X)
```

```
array([ 1,  1,  1,  1,  1,  1, -1])
```

Isolation Forest

Идея: аномалии легче изолировать (разделить) случайными разбиениями, чем нормальные точки.

Алгоритм:

1. Строится множество деревьев (лес). Каждое дерево строится рекурсивно:
    1. Выбирается случайный признак и случайное значение разбиения в диапазоне признака.
    2. Данные разделяются на подмножества до тех пор, пока не останется одна точка или не будет достигнута максимальная глубина.
2. Для точки *x* вычисляется средняя длина пути $E(h(x))$ до листа по всем деревьям.

3. Аномальность:

$$s(x, n) = 2^{-\frac{E(h(x))}{c(n)}}$$

где $c(n) = 2H_{n-1} - 2\frac{n-1}{n}$, H_k — k -е гармоническое число.

Значение $s \in [0, 1]$: близко к 1 — аномалия, близко к 0.5 — норма.

Isolation Forest. Характеристики

Плюсы:

- Линейная сложность по времени и памяти
- Не требует вычисления расстояний
- Устойчив к выбросам

Минусы:

- Плохо работает при наличии категориальных признаков
- Не учитывает корреляции
- Требует настройки глубины и числа деревьев

Аномальные значения. Обнаружение. Изолирующий лес. Пример

```
1 from sklearn.ensemble import IsolationForest
2 import numpy as np
3
4 X=[[6], [2], [3], [4], [5], [1], [100]]
5 clf = IsolationForest(random_state=0).fit(X)
6 clf.predict([[0.1], [0], [100]])
```

```
array([-1, -1, -1])
```

Сравнительный анализ методов

Метод	Скорость	Масштабируемость	Учет корреляций	Интерпретируемость	Чувствительность к шуму
Z-score	Очень высокая	Низкая (1D)	Нет	Высокая	Высокая
HBOS	Очень высокая	Высокая	Нет	Средняя	Средняя
LOF	Низкая	Низкая	Да	Низкая	Низкая
Isolation Forest	Высокая	Высокая	Нет	Средняя	Низкая

Расширения и современные подходы

Гибридные методы:

- Комбинация iForest + LOF для улучшения качества
- Ансамбли детекторов (например, Feature Bagging)

Глубокое обучение:

- Autoencoders — аномалии имеют высокую ошибку реконструкции
- GANomaly, Deep SVDD — специализированные архитектуры

Контрастивное обучение и self-supervised методы — обучение без меток через создание искусственных аномалий.

Активные направления:

- Обнаружение аномалий во временных рядах (LSTM-AE, USAD)
- Интерпретируемость (SHAP, LIME для аномалий)

Аномальные значения. Обнаружение. Робастная ковариация

Для независимых от гаусса объектов можно использовать простые статистические методы для обнаружения аномалий в наборе данных. Для гауссовского/нормального распределения точки данных, лежащие в стороне от 3-го отклонения, можно рассматривать как аномалии.

Для набора данных, имеющего все признаки гауссовой природы, статистический подход может быть обобщен путем определения эллиптической гиперсферы, которая охватывает большинство обычных точек данных, а точки данных, лежащие вдали от гиперсферы, можно рассматривать как аномалии.

Аномальные значения. Обнаружение. Робастная ковариация. Пример

```
1 from sklearn.covariance import EllipticEnvelope
2
3 X=[[6], [2], [3], [4], [5], [1], [100]]
4 clf = EllipticEnvelope()
5 clf.fit_predict(X)
```

```
array([ 1,  1,  1,  1,  1,  1, -1])
```

Аномальные значения. Обнаружение. Метод SVM

Базовый алгоритм SVM пытается найти гиперплоскость, которая наилучшим образом разделяет два класса точек данных. Для SVM одного класса, где у нас есть один класс точек данных, и задача состоит в том, чтобы предсказать гиперсферу, которая отделяет кластер точек данных от аномалий.

Аномальные значения. Обнаружение. Метод SVM. Пример

```
1 from sklearn.svm import OneClassSVM
2
3 X=[[6], [2], [3], [4], [5], [1], [100]]
4 clf = OneClassSVM(gamma='auto').fit(X)
5 clf.predict(X)
```

```
array([-1, -1, -1,  1, -1, -1,  1])
```

Пропущенные значения

Пропущенные значения могут возникать, когда не предоставляется информация по одному или нескольким элементам или по целому подразделению. Пропущенные значения - очень большая проблема в реальных сценариях.

- В Pandas Пропущенные значения представлены двумя значениями: None - это одноэлементный объект Python, который часто используется для обозначения отсутствующих данных в коде Python.
- NaN (аббревиатура от "Не число") - это специальное значение с плавающей запятой, распознаваемое всеми системами, использующими стандартное представление IEEE с плавающей запятой

Пропущенные значения. Обнаружение в Pandas

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

dict = {'1':[100, 90, np.nan, 95],
        '2': [30, 45, 56, np.nan],
        'T3':[np.nan, 40, 80, 98]}

df = pd.DataFrame(dict)
df.isnull()
```

	1	2	T3
0	False	False	True
1	False	False	False
2	True	False	False
3	False	True	False

Пропущенные значения. Восстановление в Pandas

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

dict = {'1':[100, 90, np.nan, 95],
        '2': [30, 45, 56, np.nan],
        '3':[np.nan, 40, 80, 98]}

df = pd.DataFrame(dict)
df.fillna(0)
```

	1	2	3
0	100.0	30.0	0.0
1	90.0	45.0	40.0
2	0.0	56.0	80.0
3	95.0	0.0	98.0

Спасибо за внимание!