Конспект по теме «Поиск аномалий»

Аномалии

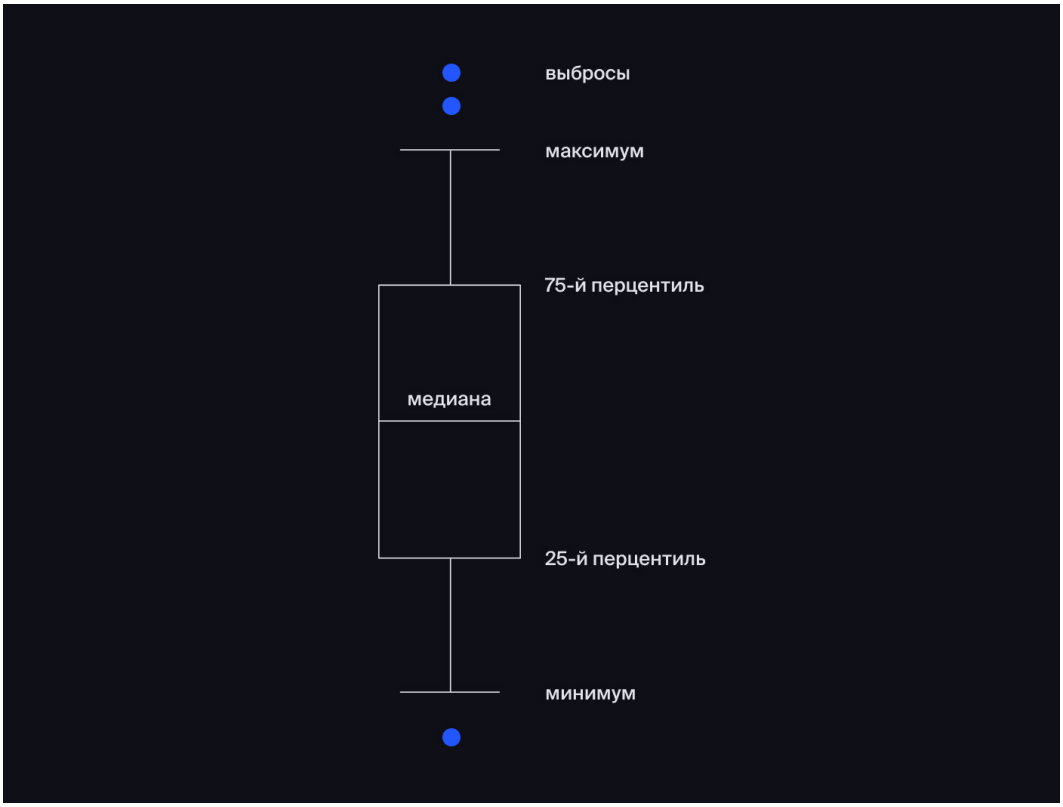
Аномалии, или **выбросы** (*anomalies; outliers*) — это объекты с «ненормальным» поведением, то есть отклоняющимся от общего тренда. Выбросы указывают на проблему в данных или на что-то нестандартное. Например, обнаружение подозрительных банковских операций: Иван из Москвы, радующий себя только шавермой по выходным, неожиданно «купил» очень дорогой смартфон в Ханое. Или прогнозирование природных аномалий: если можно было предсказать тёплую зиму, то Наташа сэкономила бы деньги на покупке нового пуховика. («Лучше бы пальто взяла или сразу босоножки!»)

Поскольку выбросы непредсказуемы, примеров аномальных объектов при обучении мало или вовсе нет.

Диаграмма размаха

Представим, что значения признака — это мешок чисел. Нужно найти числа, сильно отличающиеся от остальных. Для этого сравним их с медианой на **диаграмме размаха** (*boxplot*), или «ящике с усами».

Повторим обозначения. Верхняя и нижняя границы ящика — третья и первая квартиль (75% и 25% значений). Посередине обозначена медиана (50% значений). «Усы» простираются вверх и вниз от границ ящика на расстояние, равное 1.5 **межквартильным размахам** (IQR, *interquartile range*). Выбросы указаны за пределами усов — максимумом и минимумом.



Межквартильный размах IQR вычисляется так:

$$IQR = Q_3 - Q_1$$

Формула нижней границы ящика *L* такая:

$$L = Q_1 - k \times IQR$$

Формула верхней границы *R*:

$$R = Q_3 + k \times IQR$$

Чем больше коэффициент *k*, тем меньше объектов будут считаться выбросами. Обычно его указывают равным 1.5.

Построенная диаграмма даёт информацию обо всех выбросах. Она хранится в записи `"fliers"` внутри объекта *boxplot*. Вызовом функции `get_data()` из объектов получим числа. Нужные значения отделены индексами.

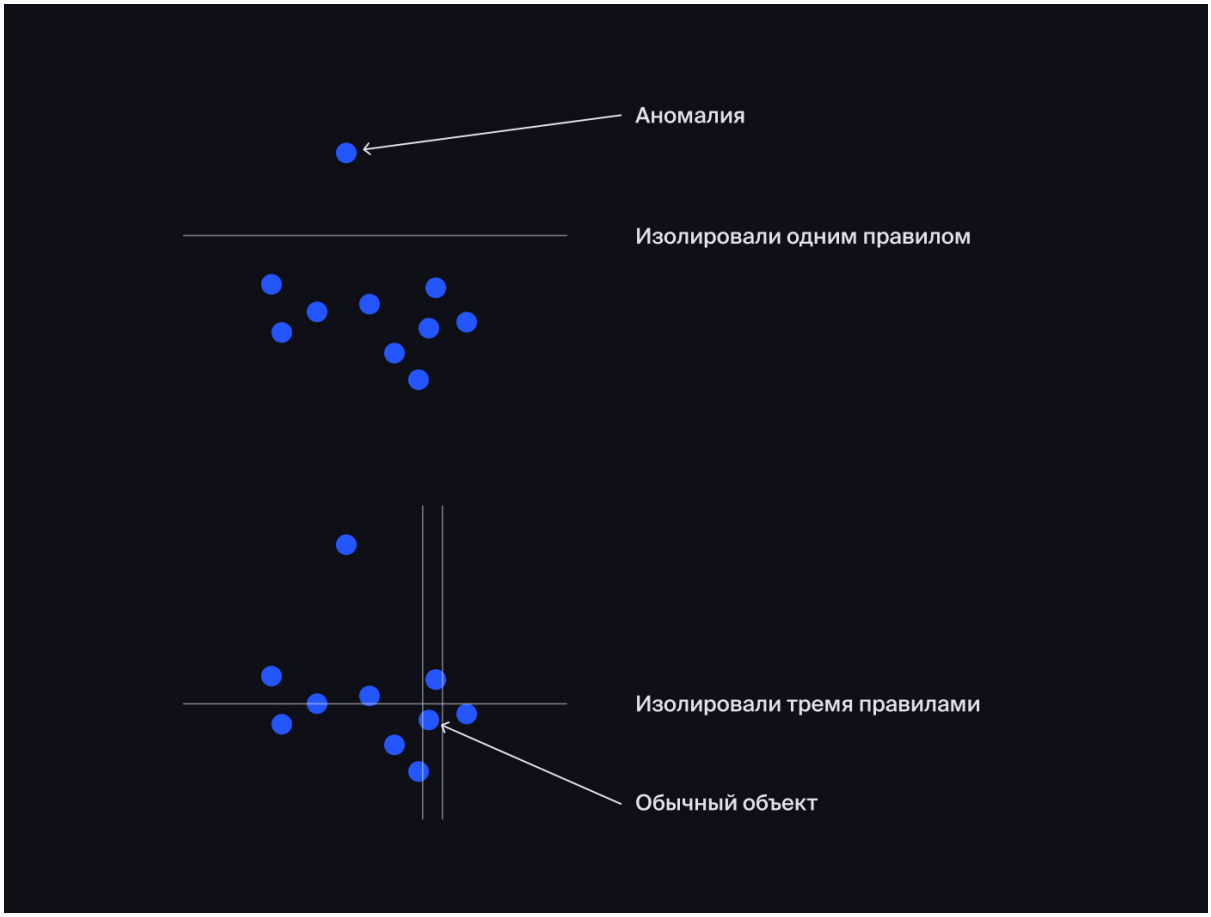
```
boxplot = plt.boxplot(df['column'].values)

# список выбросов
outliers = list(boxplot["fliers"][0].get_data()[1])
print("Выбросов: ", len(outliers))
```

Изоляционный лес

Познакомимся с другим алгоритмом поиска аномалий — **изоляционным лесом** (англ. *isolation forest*). Это ансамблевый метод. Поэтому, как и в случайном лесу, его оценки строятся на усреднённых оценках множества деревьев решений. В узлах деревьев находятся **решающие правила**. Они определяют, к какой ветви отнести объект.

Изоляционный лес основан на том, что аномальные объекты можно изолировать от остальных небольшим количеством решающих правил.



Изоляционное дерево строится как решающее дерево. А вот решающие правила в нём выбираются случайным образом. Объекты на маленькой глубине, которые можно легко изолировать, считаются аномальными, а остальные — нормальными. Оценки аномальности собираются от всех деревьев и усредняются.

Посмотрим, как изоляционный лес можно обучить в библиотеке *sklearn*. Импортируем класс *IsolationForest()* из модуля *sklearn.ensemble:*

```
from sklearn.ensemble import IsolationForest
```

Создадим модель. Пропишем количество деревьев в параметре *n_estimators*. Чем их больше, тем точнее результаты:

```
isolation_forest = IsolationForest(n_estimators=100)
```

Для поиска аномалий в одномерных данных преобразуем *df['column']* в двумерный массив:

```
sales = df['column'].values.reshape(-1, 1)
```

Выбор аномалий по одному признаку не даст представления обо всём датасете. Изоляционный лес найдёт выбросы по нескольким признакам:

```
data = df[['column1', 'column2']]
```

Дальнейшее обучение модели одинаково и для одномерных, и для многомерных данных. Функцией *fit()* обучим модель на данных о продажах и прибыли:

```
isolation_forest.fit(data)
```

Вызовом функции **decision_function()** узнаем, как модель оценила объекты:

```
anomaly_scores = isolation_forest.decision_function(data)
```

Оценки аномальности находятся в промежутке от -0.5 до 0.5. Чем ниже оценка, тем выше вероятность того, что перед вами аномалия.

Чтобы посчитать количество аномалий, вызовом функции *predict()* классифицируем объекты на нормальные и выбросы. Если объект получит класс «1», объект нормальный; если «-1» — перед вами выброс.

```
estimator = isolation_forest.predict(data)
```

Если оценки аномальности не нужны, функцией *fit_predict()* можно сразу обучить модель и получить классификацию:

```
estimator = isolation_forest.fit_predict(data)
```

KNN для поиска аномалий

Найти аномалии в многомерных данных можно и другим способом — методом ближайших соседей. **Метод ближайших соседей** (*k-Nearest Neighbors, KNN)* работает так: каждый объект датасета принимает за вектор и выбросы ищет в многомерном пространстве. Чем дальше объект от своих соседей, тем выше вероятность его аномальности.

Класс *KNN()* находится в библиотеке **PyOD**, содержащей различные методы поиска аномалий. Импортируем его из модуля *pyod.models.knn*:

```
from pyod.models.knn import KNN
```

Вызовом функции *fit()* обучим модель на выборке:

```
model = KNN()
model.fit(data)
```

Когда модель обучена, можно перейти к поиску аномалий в наборе данных. Вызовем функцию *predict():*

```
predictions = model.predict(data)
```

Функция *predict()* вернёт список, где «1» означает аномалию, а «0» — её отсутствие.