Базовые методы обработки данных с использованием Python

Лекция 8. Разведочный анализ данных (EDA). Обнаружение выбросов (аномалий). Библиотека PyOD. Основы визуализации данных.

Киреев Василий Сергеевич

к.т.н., доцент

Москва, 2024

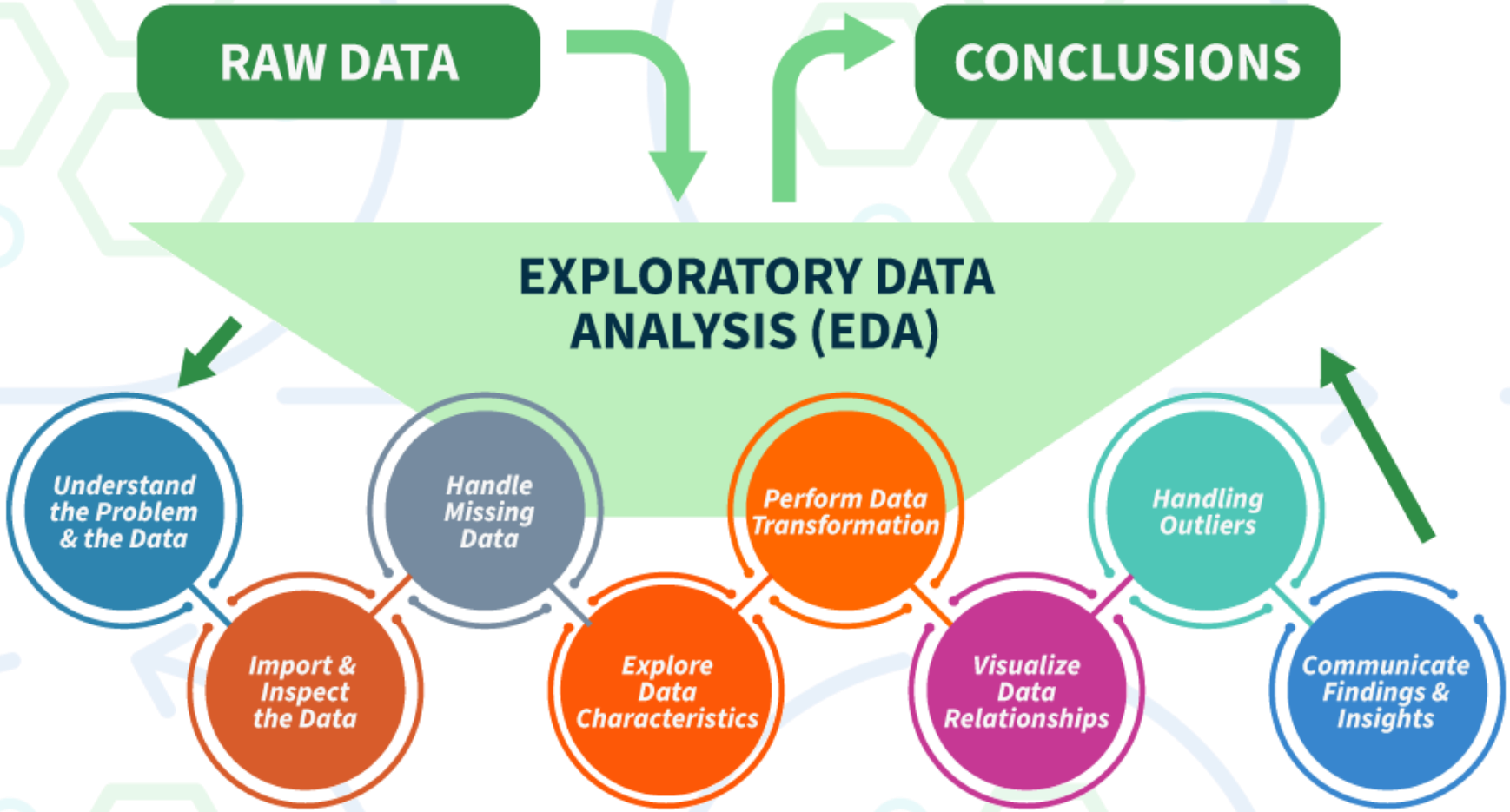
Разведочный анализ данных (EDA)

Исследовательский анализ данных (EDA) — это важный начальный шаг в проектах по науке о данных.

Он включает анализ и визуализацию данных для понимания их ключевых характеристик, выявления закономерностей и определения взаимосвязей между переменными, относится к методу изучения и исследования наборов записей для понимания их преобладающих черт, обнаружения закономерностей, поиска выбросов и определения взаимосвязей между переменными.

EDA обычно выполняется в качестве предварительного шага перед проведением дополнительных формальных статистических анализов или моделирования.

Разведочный анализ данных (EDA). Шаги



Разведочный анализ данных (EDA). Методы

Распределение данных : изучение распределения точек данных для понимания их диапазона, центральных тенденций (среднее значение, медиана) и дисперсии (дисперсия, стандартное отклонение).

Графические представления : использование диаграмм, таких как гистограммы, диаграммы размаха, диаграммы рассеяния и столбчатые диаграммы, для визуализации взаимосвязей внутри данных и распределений переменных.

Обнаружение выбросов : выявление необычных значений, которые отклоняются от других точек данных. Выбросы могут влиять на статистический анализ и могут указывать на ошибки ввода данных или уникальные случаи.

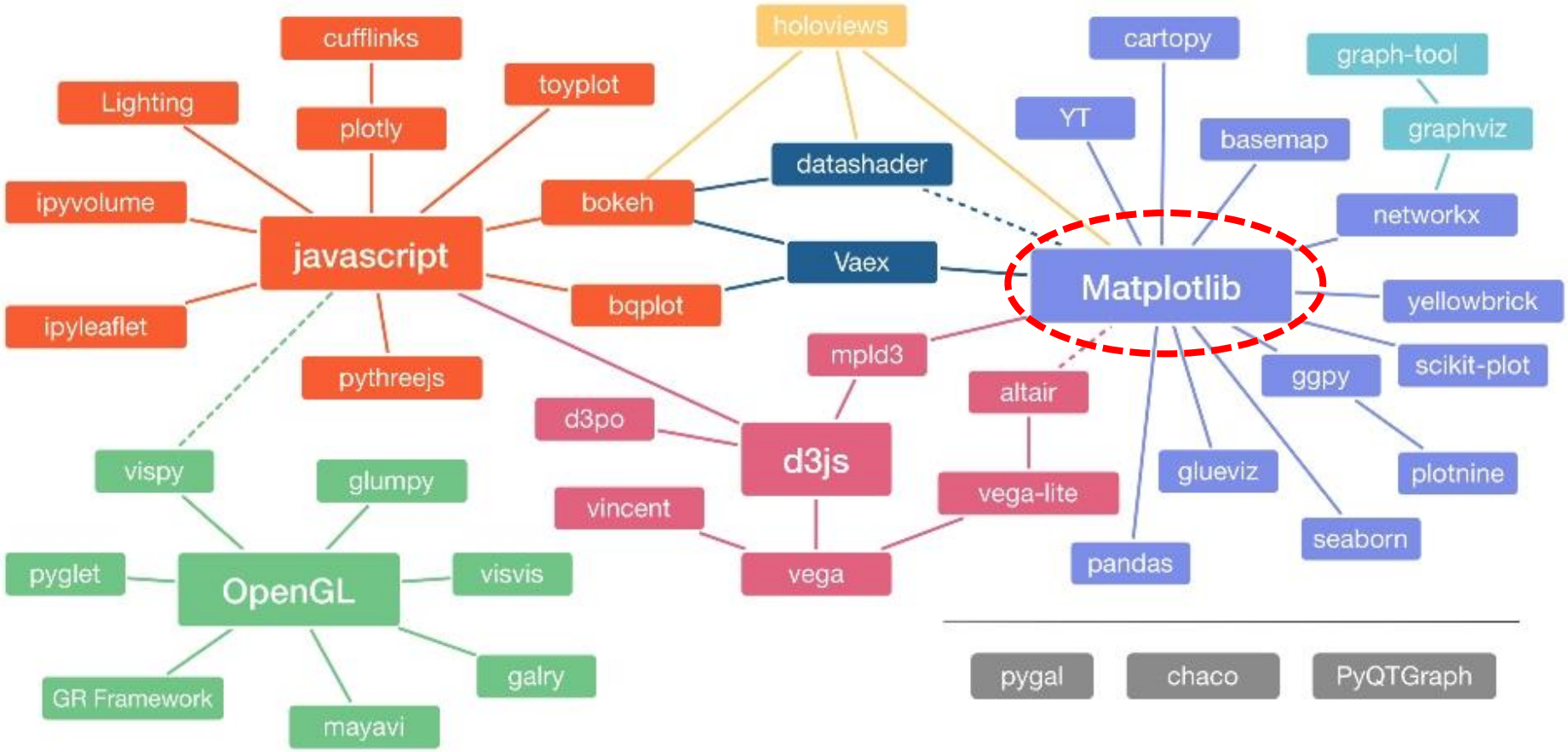
Анализ корреляции : проверка взаимосвязей между переменными для понимания того, как они могут влиять друг на друга. Это включает вычисление коэффициентов корреляции и создание матриц корреляции.

Обработка пропущенных значений : обнаружение и принятие решения о том, как устранить пропущенные точки данных, путем подстановки или удаления, в зависимости от их влияния и объема пропущенных данных.

Сводная статистика: расчет ключевых статистических данных, дающих представление о тенденциях и нюансах данных.

Тестирование предположений : многие статистические тесты и модели предполагают, что данные соответствуют определенным условиям (например, нормальности или гомоскедастичности). EDA помогает проверить эти предположения.

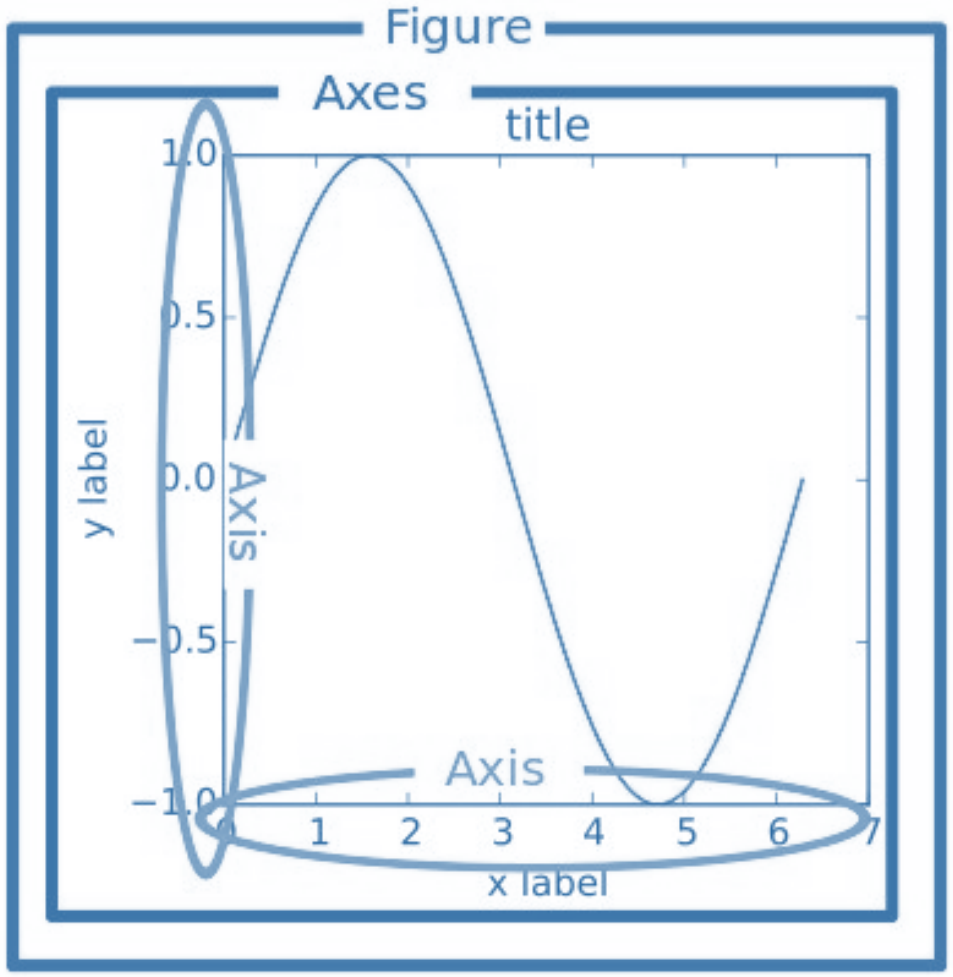
Важные фреймворки визуализации



Модуль matplotlib. Определения

Matplotlib - это основная библиотека для построения научных графиков в Python. Включает функции для создания высококачественных визуализаций: линейных диаграмм, гистограмм и т.д. Изначально matplotlib планировался как свободная альтернатива MATLAB, где в одной среде имелись бы средства как для рисования, так и для численного анализа. Любой рисунок в matplotlib имеет вложенную структуру.

Модуль matplotlib. Структура объектов



Модуль matplotlib. Рисунок

Рисунок - это объект самого верхнего уровня, на котором располагаются:

- области рисования (Axes);
- элементы рисунка Artists (заголовки, легенда и т.д.);
- основа-холст (Canvas).

Модуль matplotlib. Область рисования (Axes)

Объект среднего уровня. Это часть изображения с пространством данных. Каждая область рисования Axes содержит две (или три в случае трёхмерных данных) координатных оси (Axis объектов), которые упорядочивают отображение данных.

Модуль matplotlib. Координатная ось (Axis)

Координатная ось является объектом среднего уровня, которая определяет область изменения данных. На них наносятся:

- деления ticks;
- подписи к делениям ticklabels.

Расположение делений определяется объектом Locator, а подписи делений обрабатывает объект Formatter. Конфигурация координатных осей заключается в комбинировании различных свойств объектов Locator и Formatter.

Модуль matplotlib. Элементы рисунка (Artists)

Практически всё, что отображается на рисунке является элементом рисунка (Artist), даже объекты Figure, Axes и Axis. Элементы рисунка Artists включают в себя такие простые объекты как:

- текст (Text);
- плоская линия (Line2D);
- фигура (Patch) и другие.

Модуль matplotlib. Элементы рисунка (Artists)

Практически всё, что отображается на рисунке является элементом рисунка (Artist), даже объекты Figure, Axes и Axis. Элементы рисунка Artists включают в себя такие простые объекты как:

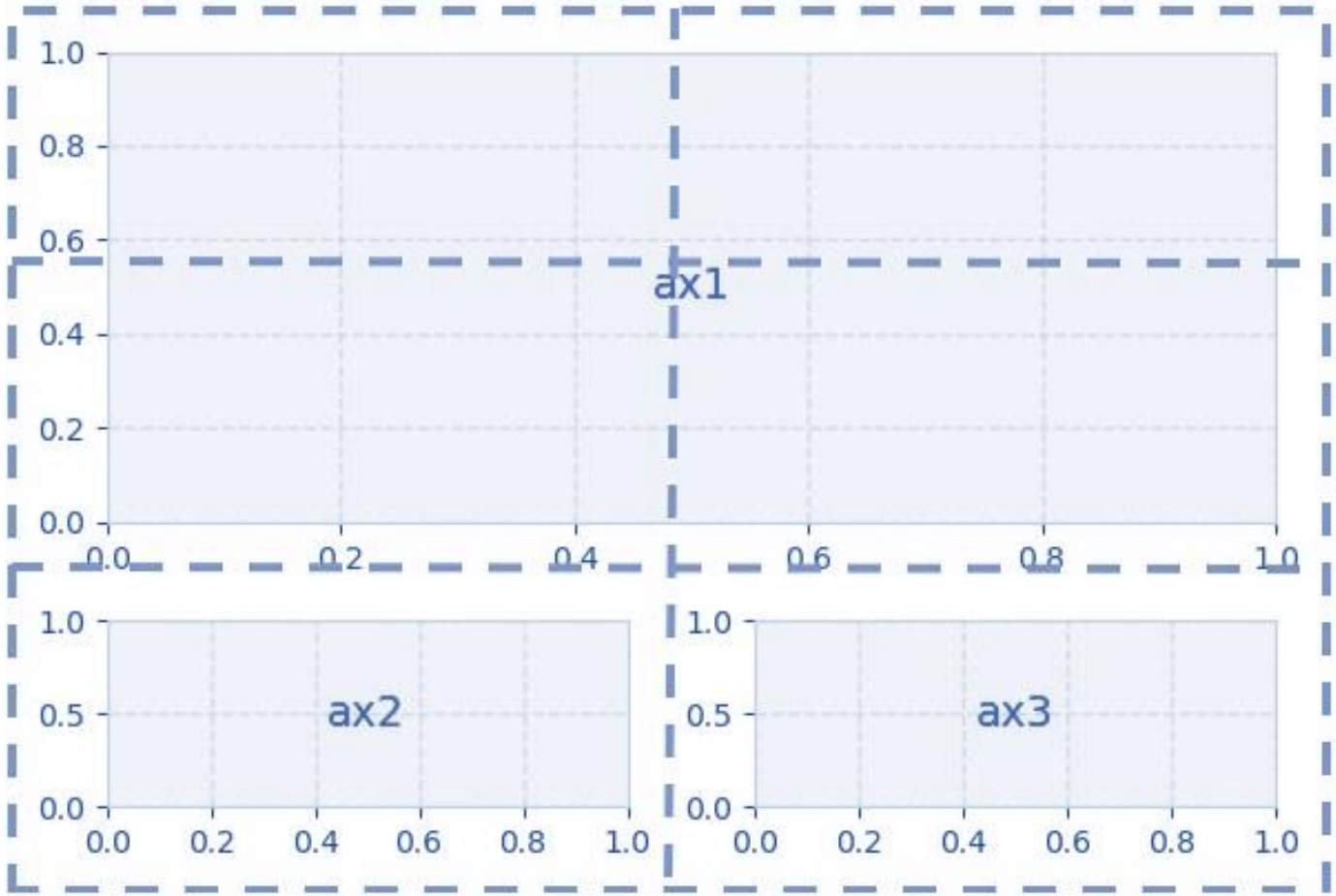
- текст (Text);
- плоская линия (Line2D);
- фигура (Patch) и другие.

Модуль matplotlib. Подписи осей

Для задания подписи оси x используется функция xlabel, оси y - ylabel. Для функций xlabel/ylabel основными являются следующие аргументы:

- xlabel (или ylabel): str
- labelpad: численное значение либо None;

Модуль matplotlib. Структура совокупности графиков



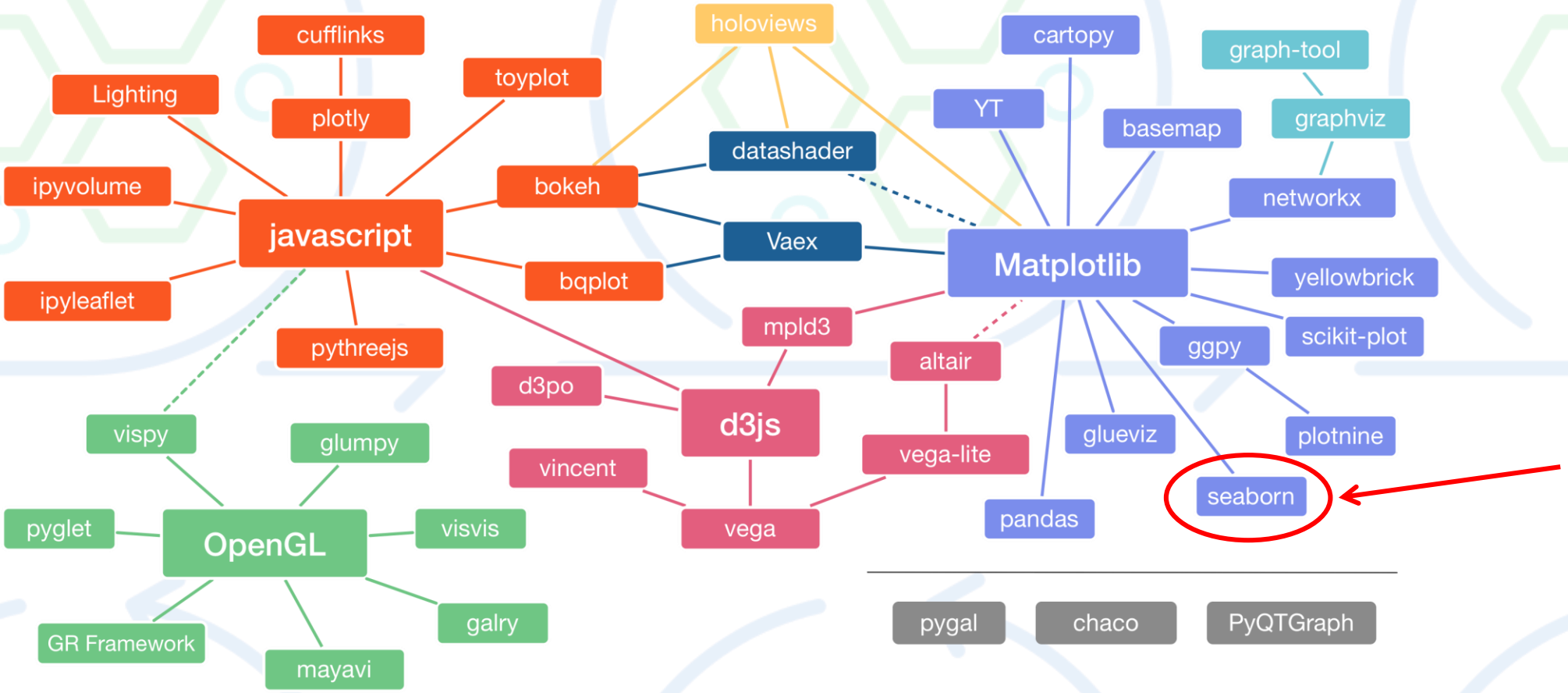
Визуализация в matplotlib. Столбиковая диаграмма

```python
import matplotlib.pyplot as plt

year = range(2010,2016,1)
freq = [50, 60, 75, 45, 70, 105]

plot = plt.bar(year, freq)
for value in plot:
    height = value.get_height()
    plt.text(value.get_x() + value.get_width()/2.,
             1.002*height,'%d' % int(height), ha='center', va='bottom')
plt.title("Столбиковая диаграмма")
plt.xlabel("год")
plt.ylabel("частота")
plt.show()
```

Место Seaborn среди библиотек визуализации



Библиотека seaborn

seaborn – это библиотека для создания статистической инфографики на Python. Она построен поверх matplotlib, а также поддерживает структуры данных numpy и pandas. Она также поддерживает статистические объекты из scipy.

Плюсы и минусы seaborn:

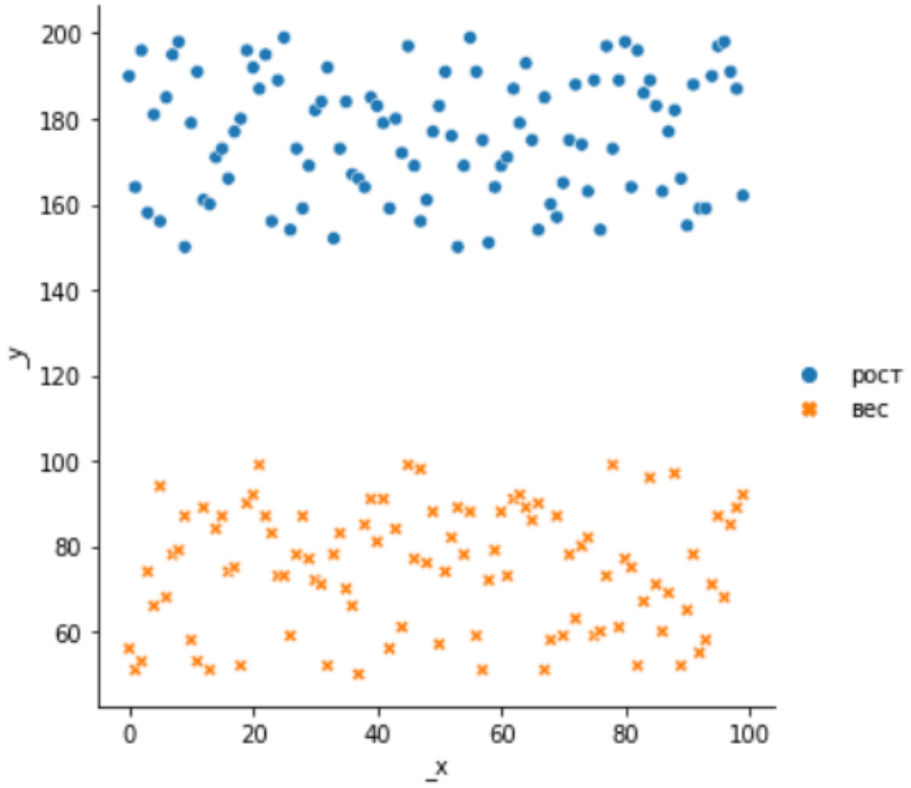
+ включает интерфейсы и настройки более высокого уровня, чем matplotlib

+ интегрирована в использовании с датафреймами pandas

- при построении множественных графиков могут быть проблемы с утечкой памяти, что в меньшей степени характерно для matplotlib

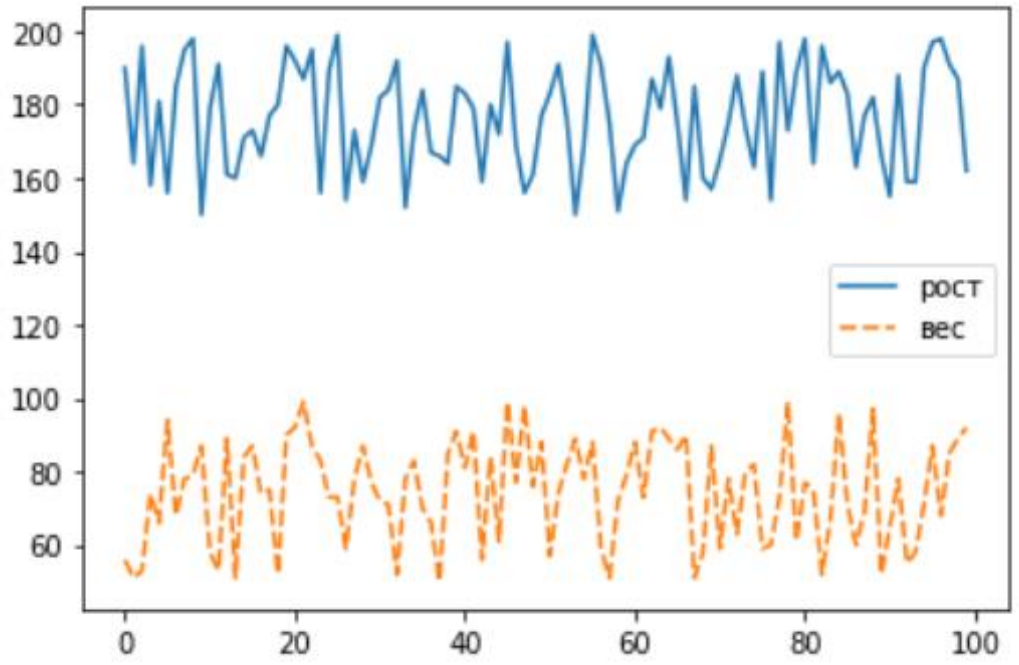
Визуализация в seaborn. Диаграмма отношения

```
1 import seaborn as sns
2 sns.relplot(data=df);
```



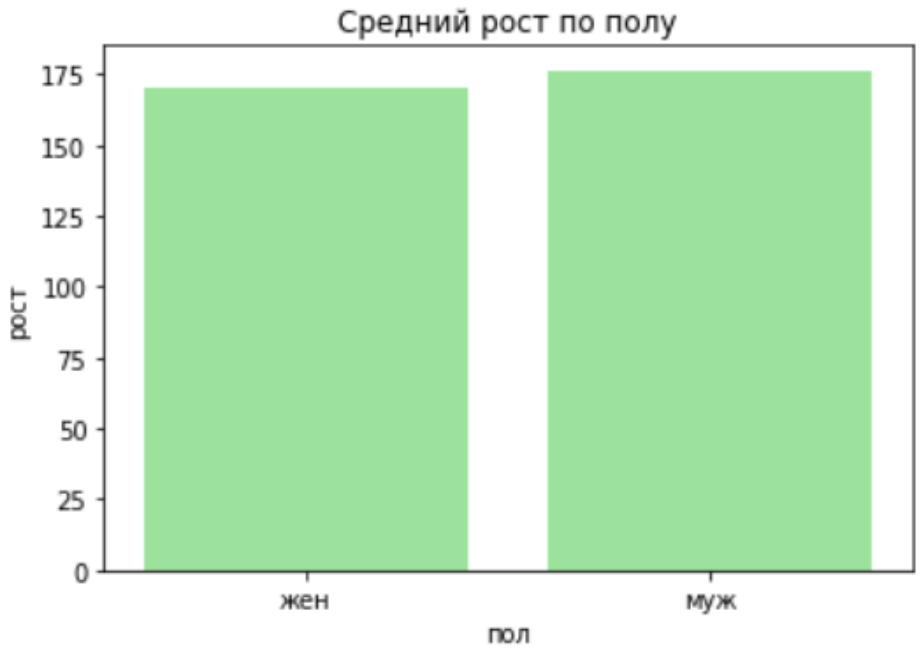
Визуализация в seaborn. Линейный график

```
1 import seaborn as sns
2 sns.lineplot(data=df);
```



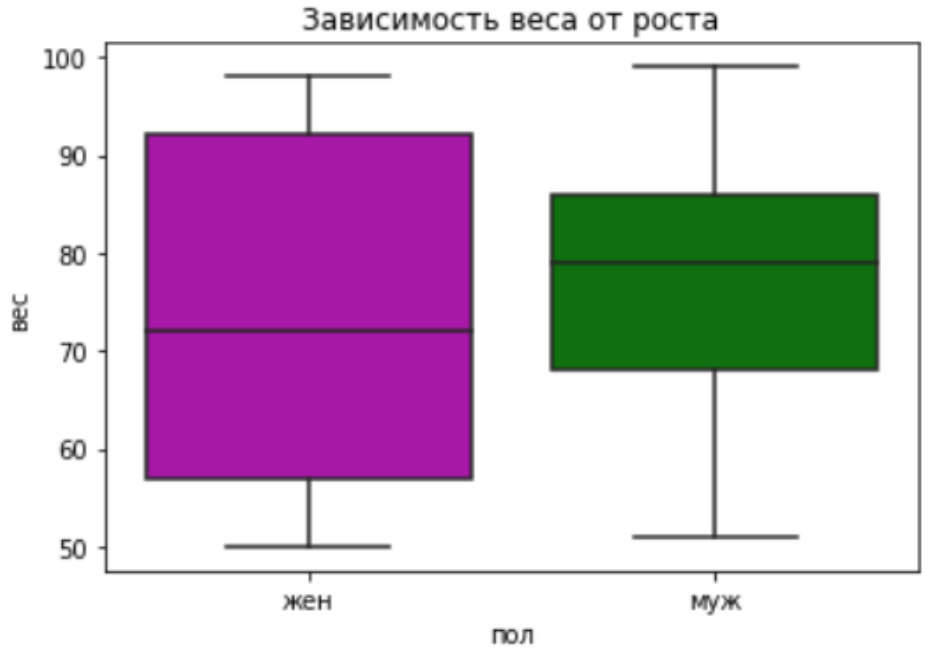
Визуализация в seaborn. Столбиковая диаграмма

```
1 import seaborn as sns
2
3 df1=df.groupby('пол')['рост'].mean().reset_index()
4 sns.barplot(x="пол", y="рост", data=df1, color='lightgreen').\
5 set(title='Средний рост по полу');
```



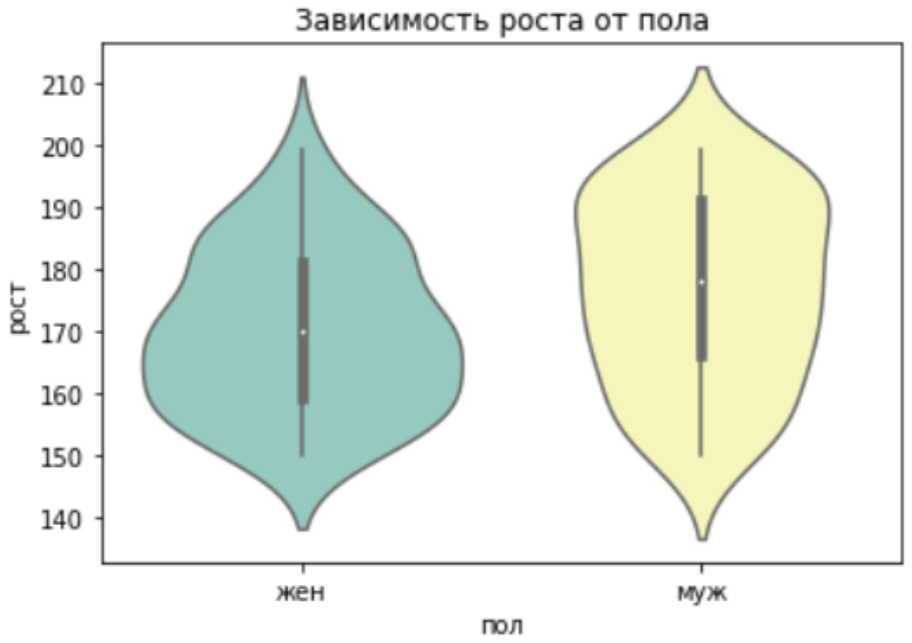
Визуализация в seaborn. Ящичковая диаграмма

```
1 import seaborn as sns
2
3 sns.boxplot(x="пол", y="вес", data=df,
4             palette={'жен':'m','муж':'g'}).\
5 set(title='Зависимость веса от пола');
```



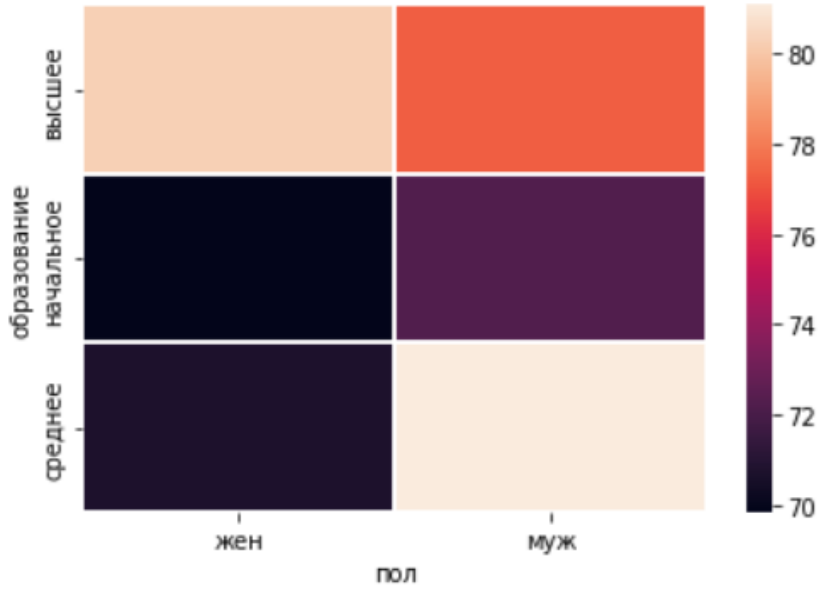
Визуализация в seaborn. Скрипичная диаграмма

```
1 import seaborn as sns
2
3 sns.violinplot(x="пол", y="рост", data=df,
4                 palette="Set3").\
5 set(title='Зависимость роста от пола');
```



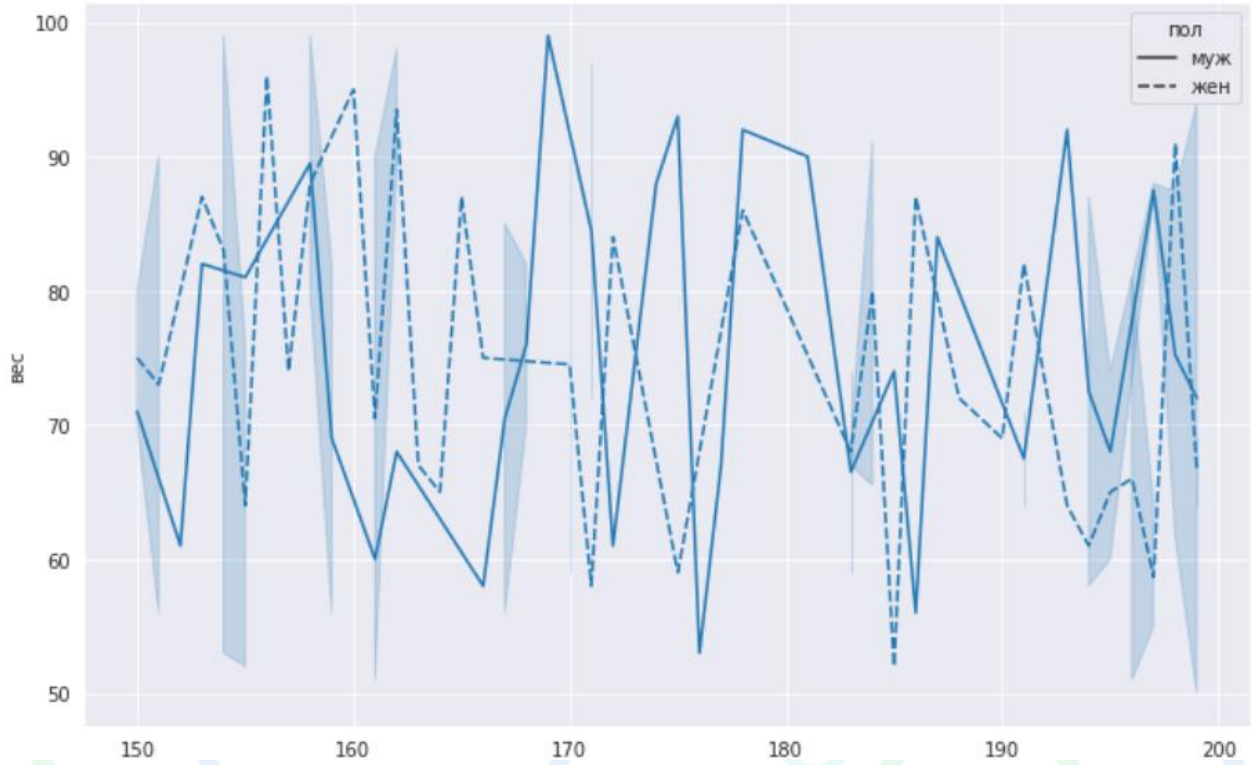
Визуализация в seaborn. Тепловая карта

```
1 import seaborn as sns
2
3 df1=df.groupby(['пол','образование'])['вес'].mean().reset_index()
4 sns.heatmap(data=df1.pivot(index='образование',columns='пол',values='вес'),linewidths=2);
```



Визуализация в seaborn. Изменение размера графика

```
1 import seaborn as sns
2 import matplotlib.pyplot as plt
3
4 fig,ax=plt.subplots(figsize=(11,7))
5 ax=sns.lineplot(data=df,x='рост',y='вес', style="пол");
```



Понятие выброса (аномалии)

Выброс по сути является статистической аномалией, точкой данных, которая значительно отклоняется от других наблюдений в наборе данных. Выбросы могут возникать из-за ошибок измерения, естественной вариации или редких событий, и они могут оказывать непропорциональное влияние на статистический анализ и модели машинного обучения, если с ними не обращаться должным образом.

Типы выбросов

Тип	Описание
Одномерные выбросы	выбросы, которые возникают в одной переменной или признаке
Многомерные выбросы	Эти выбросы возникают при рассмотрении нескольких переменных одновременно. Точка данных может не быть выбросом в каком-либо одном измерении, но может быть выбросом при рассмотрении нескольких измерений.
Глобальные выбросы	также известные как точечные аномалии, эти точки данных значительно отличаются от остального набора данных.
Контекстные выбросы	точки данных, которые считаются выбросами в определенном контексте. Например, высокая температура может быть нормой летом, но выбросом зимой.
Коллективные выбросы	совокупность точек данных, которые значительно отклоняются от остального набора данных, даже если отдельные точки в совокупности не являются выбросами.

Влияние выбросов на модели машинного обучения

Смещенные модели: выбросы могут смещать модель машинного обучения в сторону значений выброса, что приводит к плохой производительности на остальных данных. Это может быть особенно проблематично для алгоритмов, чувствительных к выбросам, таких как линейная регрессия.

Снижение точности: выбросы могут вносить шум в данные, затрудняя изучение моделью машинного обучения истинных базовых закономерностей. Это может привести к снижению точности и производительности.

Увеличение дисперсии: выбросы могут увеличить дисперсию модели машинного обучения, делая ее более чувствительной к небольшим изменениям в данных. Это может затруднить обучение стабильной и надежной модели.

Сниженная интерпретируемость: выбросы могут затруднить понимание того, чему модель машинного обучения научилась на основе данных. Это может затруднить доверие к прогнозам модели и помешать усилиям по улучшению ее производительности.

Методы обработки выбросов

Влияние выбросов на модели машинного обучения

М-оценки — широкий класс статистических оценок, доставляющих минимум суммы каких-либо функций от данных:

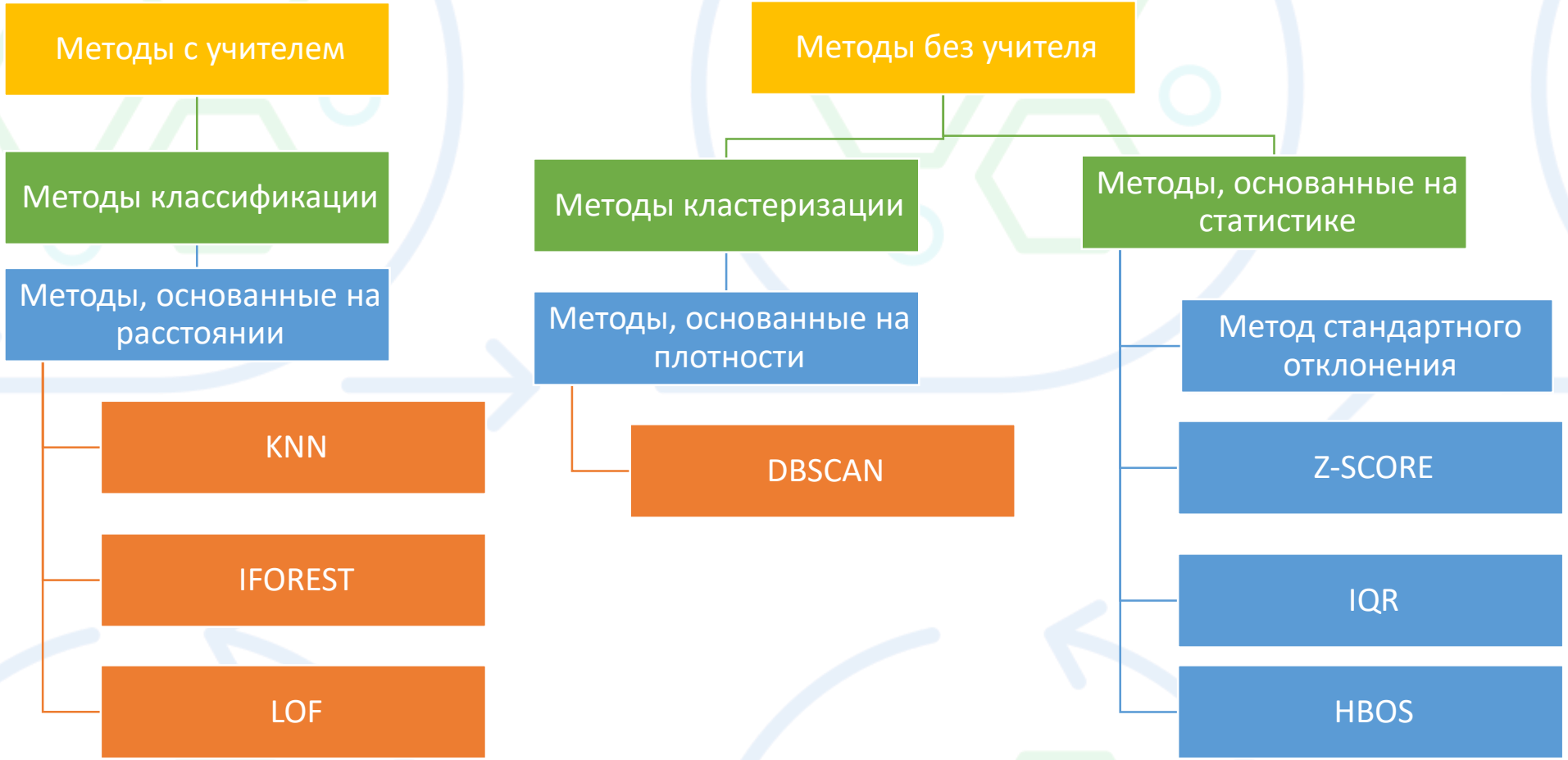
$$\widehat{\theta} = \operatorname{argmin}_{\theta} \sum_{i=1}^{n} \rho(x_i, \theta)$$

М-оценками являются, в частности, оценки наименьших квадратов, а также многие оценки максимального правдоподобия. Функция выбирается таким образом, чтобы обеспечить желаемые свойства оценки (несмещённость и эффективность) в условиях, когда данные взяты из известного распределения, и достаточную устойчивость к отклонениям от этого распределения.

Задачи обнаружения выбросов

- **Повышение точности**: удаление или правильная обработка выбросов повышает производительность и предсказуемость моделей данных.
- **Обнаружение мошенничества**: выбросы могут быть симптомом мошеннической деятельности, особенно в финансовых или транзакционных данных.
- **Качество данных**: регулярное обнаружение выбросов имеет решающее значение для поддержания целостности и качества данных, что, в свою очередь, влияет на процессы принятия решений на основе этих данных.
- **Производительность модели**: выбросы могут существенно влиять на производительность статистических моделей, алгоритмов машинного обучения и других аналитических методов. Выявляя и обрабатывая выбросы соответствующим образом, мы можем повысить надежность и точность этих моделей.
- **Генерация инсайтов** : выбросы могут представлять уникальные или интересные явления в данных. Выявление и анализ выбросов может привести к ценным инсайтам, таким как обнаружение новых тенденций, понимание редких событий или раскрытие потенциальных возможностей или угроз.

Методы обнаружения выбросов



Библиотека PyOD

PyOD – это Python-библиотека с более чем 30 современными алгоритмами обнаружения редких и подозрительных данных или событий. PyOD обладает рядом преимуществ и имеет множество полезных функций:

- исходный код с подробной документацией и примерами по различным алгоритмам
- Поддержка продвинутых моделей, включая нейронные сети, глубокое обучение и ансамбли выбросов
- Оптимизированная производительность благодаря JIT (Just in Time) и распараллеливанию с помощью numba и joblib

Метод стандартного отклонения

Метод стандартного отклонения основан на предположении, что данные распределены по нормальному закону. Выбросы определяются как наблюдения, которые находятся за пределами указанного числа стандартных отклонений от среднего значения. Обычно выбросами считаются точки данных, находящиеся за пределами трех стандартных отклонений от среднего значения. Этот метод эффективен для данных, которые близко следуют гауссовскому распределению.

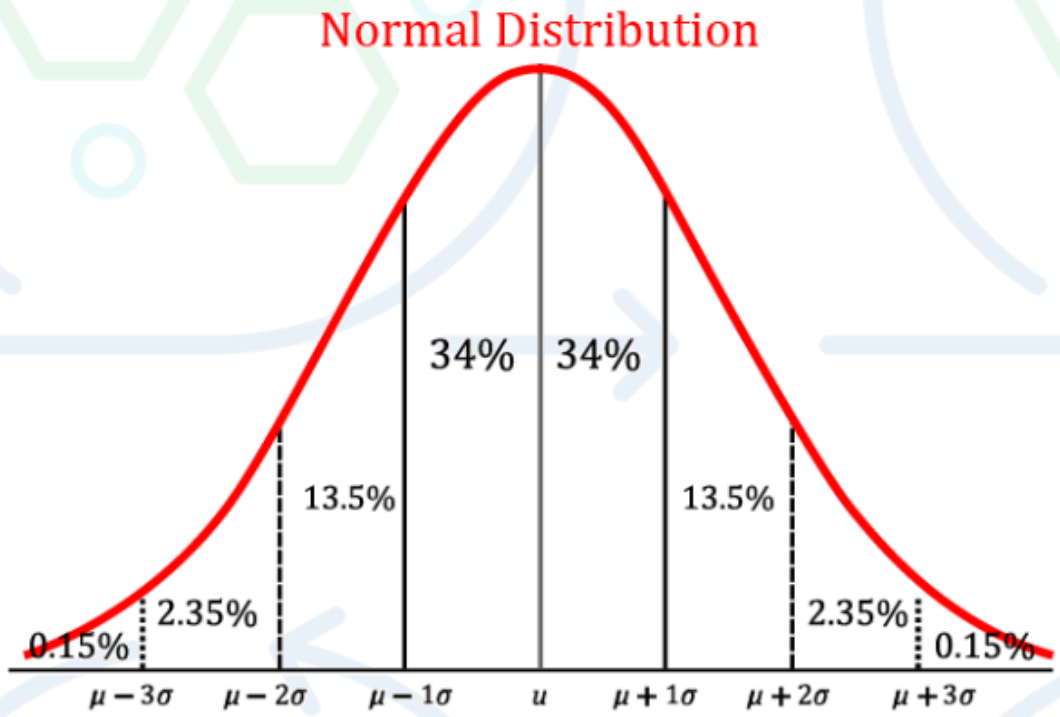
Обычно используется для одномерного анализа данных, где распределение можно считать приблизительно нормальным. Подходит для наборов данных с симметричным распределением, где экстремальные значения можно определить на основе их отклонения от среднего.

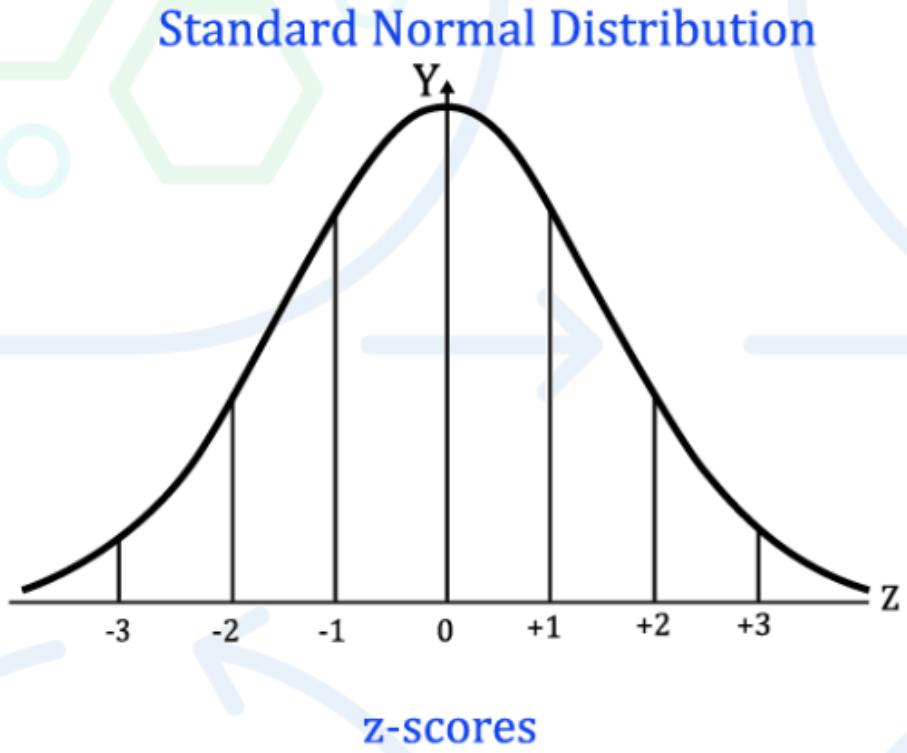
Метод Z-оценки (Z-score)

Метод Z-оценки вычисляет количество стандартных отклонений каждой точки данных от среднего значения. Устанавливается пороговое значение Z-оценки, обычно 3, и любая точка данных с Z-оценкой, превышающей это пороговое значение, считается выбросом. Этот метод предполагает нормальное распределение и чувствителен к экстремальным значениям в небольших наборах данных. Подходит для наборов данных с большими размерами выборки, где базовое распределение данных можно разумно аппроксимировать нормальным распределением.

$$Z = \frac{x - \mu}{\sigma}$$

Метод Z-оценки (Z-score)

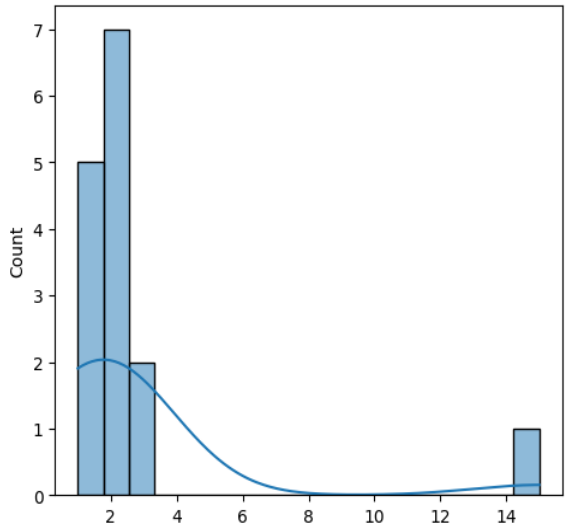


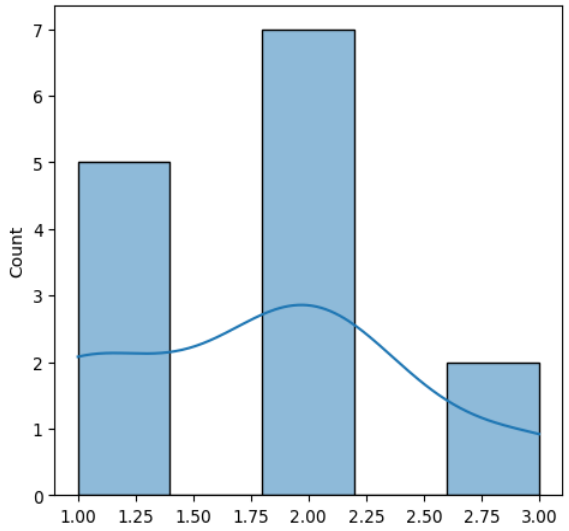


Метод Z-оценки (Z-score). Пример

Если z-оценка точки данных больше 3, это означает, что точка данных сильно отличается от других точек данных. Такая точка данных может быть выбросом.

```
import scipy.stats as st
data = np.array([1, 2, 2, 2, 3, 1, 1, 15, 2, 2, 2, 3,
1, 1, 2])
zr = st.zscore(data)
plt.figure(figsize=(12,5))
plt.subplot(1,2,1)
sns.histplot(data,kde=True)
plt.subplot(1,2,2)
sns.histplot(data[np.where(zr<3)[0]],kde=True)
```



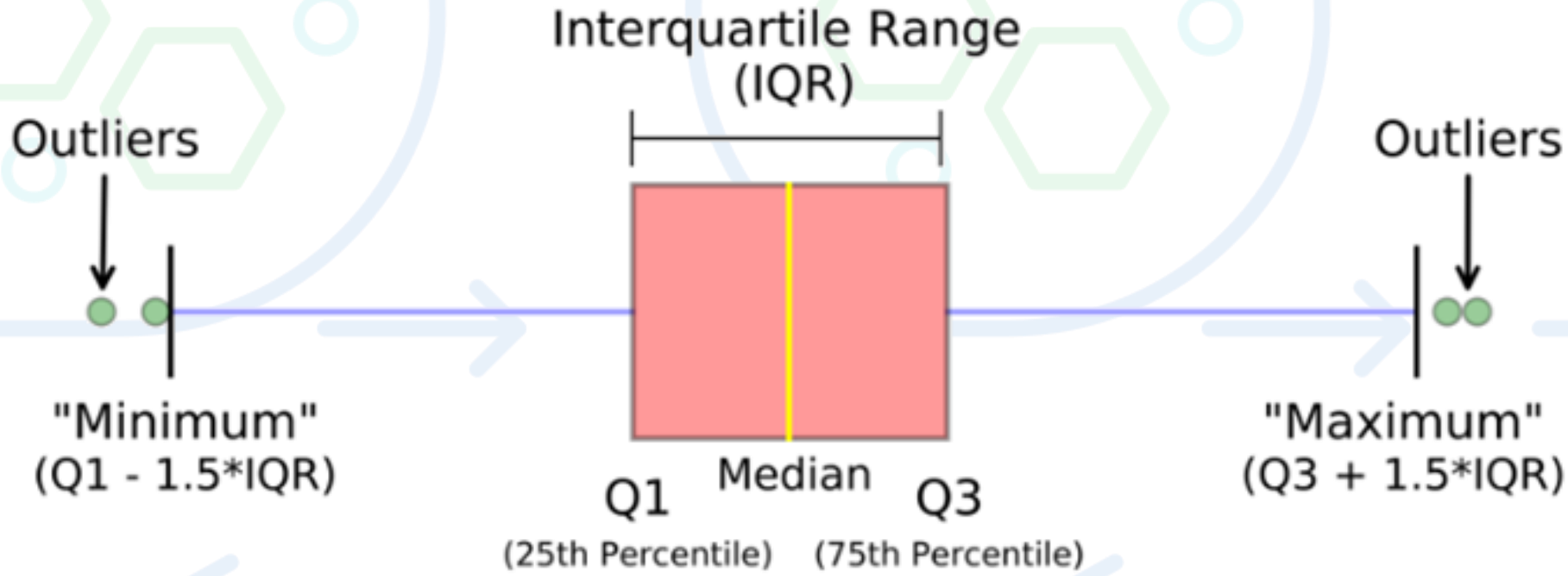


Метод межквартильного размаха (IQR)

Метод межквартильного размаха (IQR) фокусируется на разбросе средних 50% данных. Он вычисляет IQR как разницу между 75-м и 25-м процентилями данных и определяет выбросы как те точки, которые попадают ниже 1,5-кратного IQR ниже 25-го процентиля или выше 1,5-кратного IQR выше 75-го процентиля. Этот метод устойчив к выбросам и не предполагает нормального распределения.

Подходит для наборов данных с перекошенным или ненормальным распределением. Полезно для выявления выбросов в наборах данных, где разброс средних 50% данных более важен, чем среднее значение и стандартное отклонение.

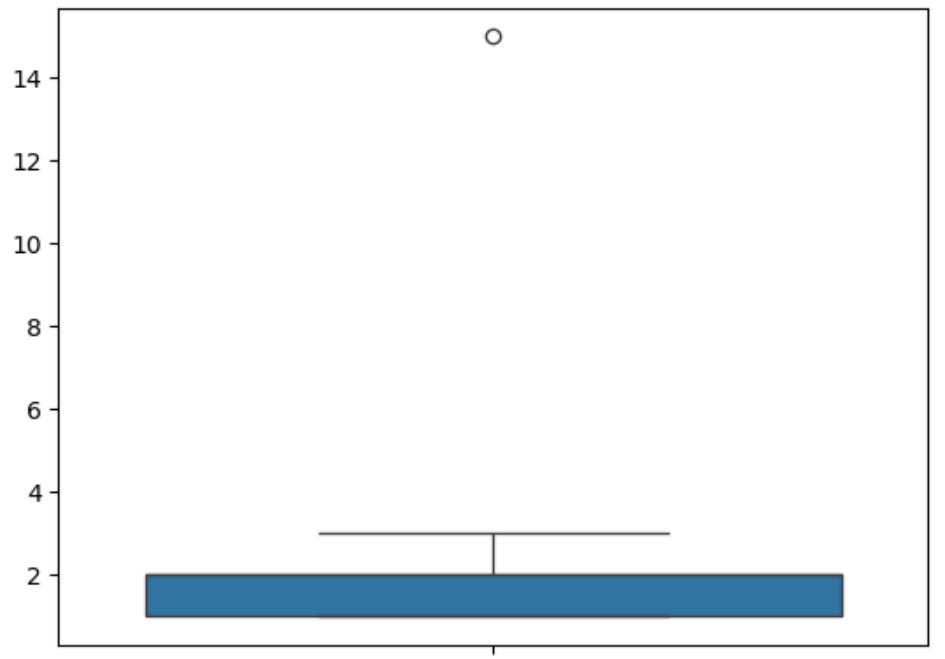
Метод межквартильного размаха (IQR)



Метод межквартильного размаха (IQR) . Пример

Для изучения набора данных и визуализации наличия выбросов можно использовать бокс-график или бокс-график с усами. Точки, лежащие за пределами усов, определяются как выбросы.

```
data = np.array([1, 2, 2, 2, 3, 1, 1, 15, 2, 2, 2, 3,
1, 1, 2])
sns.boxplot(data)
```



Метод ближайших соседей (KNN)

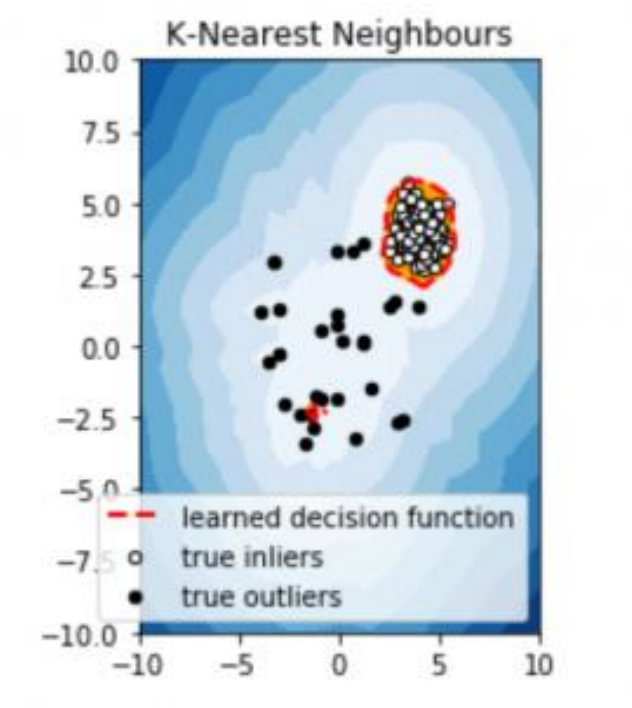
Алгоритм K-NN работает путем поиска K ближайших соседей для заданной точки данных на основе метрики расстояния, например евклидова расстояния. Затем класс или значение точки данных определяется большинством голосов или средним значением K соседей. Такой подход позволяет алгоритму адаптироваться к различным шаблонам и делать прогнозы на основе локальной структуры данных.

KNN. Пример

Для изучения набора данных и визуализации наличия выбросов можно использовать бокс-график или бокс-график с усами. Точки, лежащие за пределами усов, определяются как выбросы.

```
from pyod.models.knn import KNN
clf = KNN(contamination = outlier_fraction)
clf.fit(X_train, y_train)
 scores_pred = clf.decision_function(X_train)*-1
y_pred = clf.predict(X_train)
threshold = stats.scoreatpercentile(scores_pred, 100 * outlier_fraction)
Z = clf.decision_function(np.c_[xx.ravel(), yy.ravel()]) * -1
Z = Z.reshape(xx.shape)
  subplot = plt.subplot(1, 2, 1)
subplot.contourf(xx, yy, Z, levels = np.linspace(Z.min(),
                threshold, 10), cmap = plt.cm.Blues_r)
a = subplot.contour(xx, yy, Z, levels =[threshold],
                    linewidths = 2, colors ='red')
subplot.contourf(xx, yy, Z, levels =[threshold, Z.max()], colors ='orange')
# scatter plot of inliers with white dots
b = subplot.scatter(X_train[:-n_outliers, 0], X_train[:-n_outliers, 1],
                                c ='white', s = 20, edgecolor ='k')

# scatter plot of outliers with black dots
c = subplot.scatter(X_train[-n_outliers:, 0], X_train[-n_outliers:, 1],
                                c ='black', s = 20, edgecolor ='k')
subplot.axis('tight')
subplot.legend(
    [a.collections[0], b, c],
    ['learned decision function', 'true inliers', 'true outliers'],
    prop = matplotlib.font_manager.FontProperties(size = 10),
    loc ='lower right')
subplot.set_title('K-Nearest Neighbours')
subplot.set_xlim((-10, 10))
subplot.set_ylim((-10, 10))
plt.show()
```



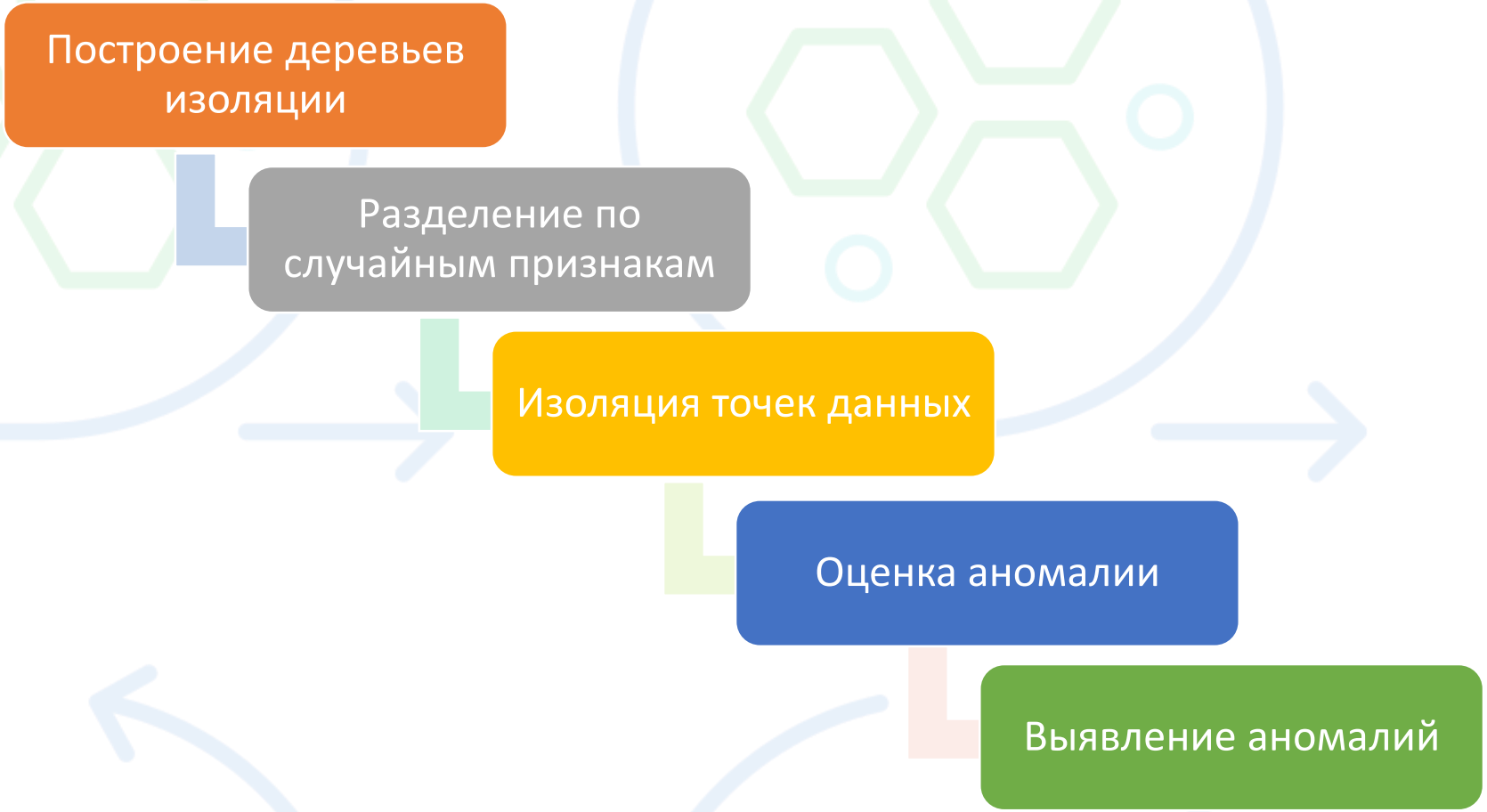
Isolation Forest

В отличие от других методов, Isolation Forest явно изолирует аномалии вместо профилирования нормальных точек данных. Он работает по принципу, что выбросы меньше и они различны, и поэтому их легче изолировать.

Алгоритм случайным образом выбирает признак и разделяет данные между максимальным и минимальным значениями выбранного признака. Это разделение продолжается рекурсивно до тех пор, пока точки не будут изолированы. Точки, требующие меньшего количества разделений, считаются выбросами.

Подходит для обнаружения выбросов в многомерных наборах данных, обнаружения аномалий в кибербезопасности и выявления аномалий в наборах данных с неоднородным распределением.

Isolation Forest. Алгоритм



Isolation Forest. Алгоритм. Построение деревьев изоляции

Алгоритм начинается с создания набора деревьев изоляции, обычно сотен или даже тысяч. Эти деревья похожи на традиционные деревья решений, но с ключевым отличием: они не построены для классификации точек данных по определенным категориям.

Вместо этого деревья изоляции нацелены на изоляцию отдельных точек данных путем многократного разделения данных на основе случайно выбранных признаков и разделенных значений.

Isolation Forest. Алгоритм. Разделение по случайным признакам

Деревья изоляции вносят случайность в каждый узел дерева, выбирается случайный признак из набора данных. Затем выбирается случайное значение разделения в диапазоне значений этого конкретного признака. Эта случайность помогает гарантировать, что аномалии, которые, как правило, отличаются от большинства точек данных, не будут скрыты в определенных ветвях дерева.

Isolation Forest. Алгоритм. Изоляция точек данных

Точки данных направляются вниз по ветвям дерева изоляции на основе значений их характеристик.

Если значение точки данных для выбранного признака падает ниже значения разделения, оно переходит в левую ветвь. В противном случае оно переходит в правую ветвь.

Этот процесс продолжается рекурсивно до тех пор, пока точка данных не достигнет конечного узла, который просто представляет собой изолированную точку данных.

Isolation Forest. Алгоритм. Оценка аномалии

Ключевая концепция изоляционных лесов заключается в длине пути точки данных через изоляционное дерево.

Аномалии, в силу их отличия от большинства, как правило, легче изолировать. Для достижения листового узла им требуется меньше случайных разделений, поскольку они, скорее всего, выйдут за пределы типичного диапазона значений для выбранных признаков.

И наоборот, нормальным точкам данных, которые имеют больше сходства друг с другом, может потребоваться больше разветвлений на их пути вниз по дереву, прежде чем они будут изолированы.

Isolation Forest. Алгоритм. Расчет показателя аномалии

Каждая точка данных оценивается по всем деревьям изоляции в лесу.

Для каждого дерева регистрируется длина пути (количество разбиений), необходимая для изоляции точки данных.

Затем для каждой точки данных рассчитывается показатель аномалии путем усреднения длин путей по всем изолированным деревьям в лесу.

Isolation Forest. Алгоритм. Выявление аномалий

Точки данных с более короткими средними длинами пути считаются более вероятными аномалиями. Это связано с тем, что их было легче изолировать, что предполагает, что они значительно отклоняются от основной массы данных.

Устанавливается пороговое значение для определения оценки аномалии, которая отделяет нормальные точки данных от аномалий. Это пороговое значение может быть определено на основе знаний предметной области, экспериментов или установленных статистических принципов.

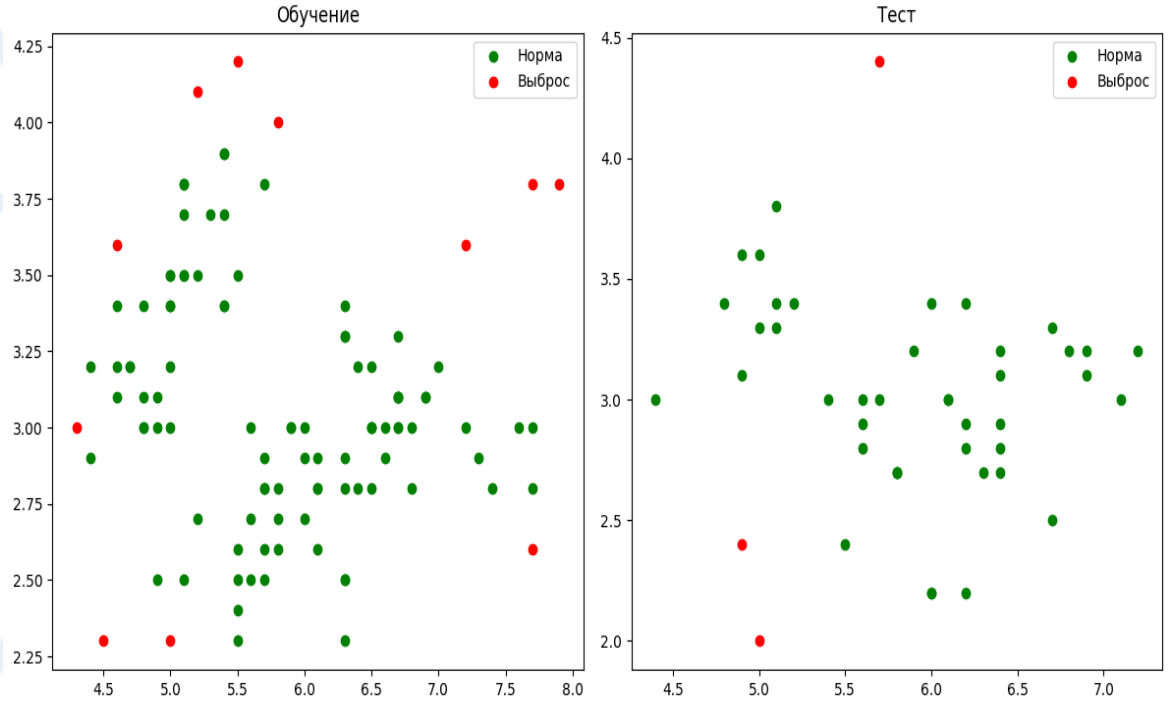
Isolation Forest. Пример

В качестве примера использовался датасет Ирисы Фишера, и реализация метода IFOREST из библиотеки sklearn.

```
from sklearn.ensemble import IsolationForest
from sklearn.model_selection import train_test_split
from sklearn.datasets import load_iris
import matplotlib.pyplot as plt

iris = load_iris()
X = iris.data
y = iris.target
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.3, random_state=1000)
clf = IsolationForest(contamination=0.1)
clf.fit(X_train)
y_pred_train = clf.predict(X_train)
y_pred_test = clf.predict(X_test)

def create_scatter_plots(X1, y1, title1, X2, y2, title2):
    fig, axes = plt.subplots(1, 2, figsize=(12, 6))
    axes[0].scatter(X1[y1==1, 0], X1[y1==1, 1], color='green', label='Норма')
    axes[0].scatter(X1[y1==-1, 0], X1[y1==-1, 1], color='red', label='Выброс')
    axes[0].set_title(title1)
    axes[0].legend()
    axes[1].scatter(X2[y2==1, 0], X2[y2==1, 1], color='green', label='Норма')
    axes[1].scatter(X2[y2==-1, 0], X2[y2==-1, 1], color='red', label='Выброс')
    axes[1].set_title(title2)
    axes[1].legend()
    plt.tight_layout()
    plt.show()
create_scatter_plots(X_train, y_pred_train, 'Обучение', X_test, y_pred_test, 'Тест')
```



Local Outlier Finder (LOF)

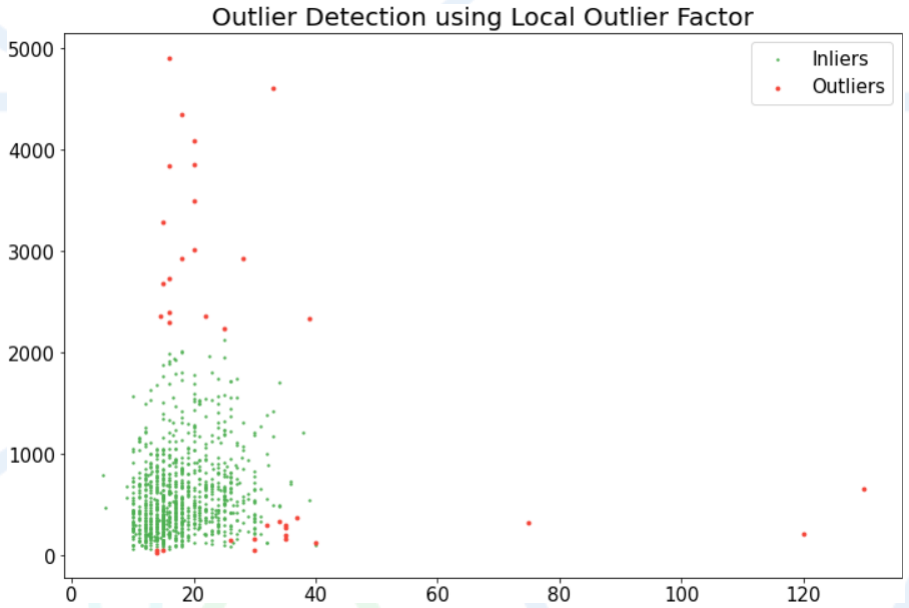
Local Outlier Finder - это неконтролируемый метод машинного обучения для обнаружения выбросов, основанный на плотности ближайших соседей точек данных и хорошо работающий, когда разброс набора данных (плотность) не одинаков. LOF в основном учитывает K-расстояние (расстояние между точками) и K-соседей (множество точек лежит в круге K-расстояния (радиуса)).

Local Outlier Finder (LOF). Пример

LOF учитывает два основных параметра:

- n_neighbors: Число соседей, которое по умолчанию равно 20
- Загрязнение: доля промахов в данном наборе данных, которая может быть задана как «авто» или плавающие значения (0, 0.02, 0.005).

```
from sklearn.neighbors import LocalOutlierFactor
d2 = df.valueslof = LocalOutlierFactor(n_neighbors=20,
contamination='auto')
good = lof.fit_predict(d2) == 1
plt.figure(figsize=(10,5))
plt.scatter(d2[good, 1], d2[good, 0], s=2, label="Inliers",
color="#4CAF50")
plt.scatter(d2[~good, 1], d2[~good, 0], s=8, label="Outliers",
color="#F44336")
plt.title('Outlier Detection using Local Outlier Factor',
fontsize=20)
plt.legend (fontsize=15, title_fontsize=15)
```



Методы кластеризации

Алгоритмы кластеризации, такие как DBSCAN, группируют данные в кластеры или группы на основе сходства данных. Точки, которые не принадлежат ни к одному кластеру, часто считаются выбросами. Алгоритм группирует точки, которые плотно упакованы вместе, отмечая как выбросы точки, которые лежат отдельно в областях с низкой плотностью.

Методы кластеризации полезны, когда данные включают пространственные отношения или когда выбросы определяются как точки, не принадлежащие ни одному кластеру. Они эффективны для выявления выбросов в наборах данных со сложной структурой и нелинейными отношениями. Подходят для пространственного анализа данных, обнаружения аномалий в сетевом трафике и выявления выбросов в наборах данных с кластерами или группами.

Пространственная кластеризация приложений с шумом на основе плотности (DBSCAN)

DBSCAN расшифровывается как пространственная кластеризация приложений с шумом на основе плотности. Он способен находить кластеры произвольной формы и кластеры с шумом (т.е. выбросы). Кластеры - это плотные области в пространстве данных, разделенные областями с меньшей плотностью точек.

Алгоритм DBSCAN основан на этом интуитивном понятии “кластеров” и “шума”. Ключевая идея заключается в том, что для каждой точки кластера окрестности заданного радиуса должны содержать по крайней мере минимальное количество точек.

Основные параметры метода DBSCAN

eps: Он определяет окрестности вокруг точки данных, т.е. если расстояние между двумя точками меньше или равно "eps", то они считаются соседями. Если значение eps выбрано слишком малым, то большая часть данных будет рассматриваться как выброс. Если он выбран очень большим, то кластеры объединятся, и большинство точек данных будут находиться в одних и тех же кластерах.

minPts: Минимальное количество соседей (точек данных) в радиусе eps. Чем больше набор данных, тем большее значение minPts должно быть выбрано. Как правило, минимальное значение minPts может быть получено из числа измерений D в наборе данных следующим образом: minPts >= D+1. Минимальное значение minPts должно быть выбрано не менее 3.

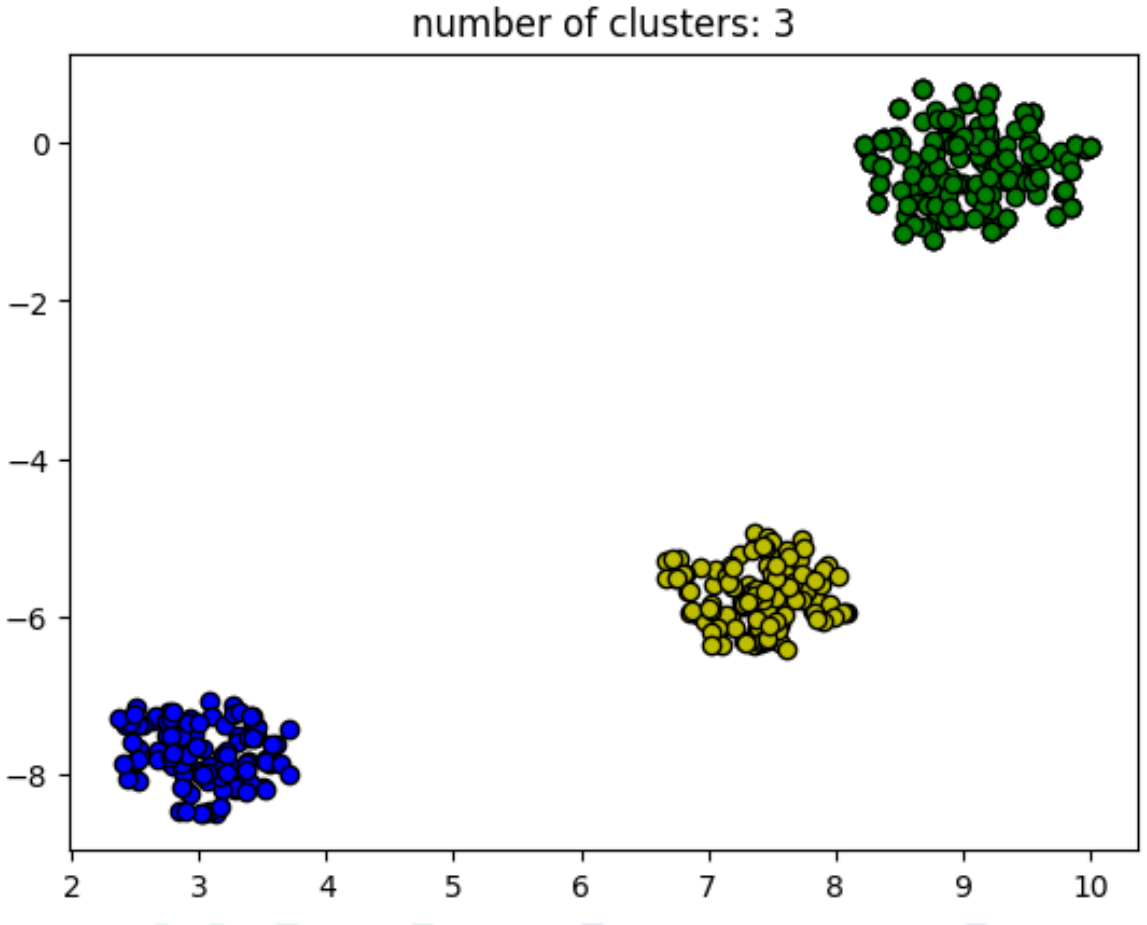
DBSCAN. Пример

```python
from sklearn.cluster import DBSCAN
db = DBSCAN(eps=0.3, min_samples=10).fit(X)
core_mask = np.zeros_like(db.labels_, dtype=bool)
core_mask[db.core_sample_indices_] = True
labels = db.labels_
n_clusters_ = len(set(labels)) - (1 if -1 in labels else 0)
```

```python
unique_labels = set(labels)
colors = ['y', 'b', 'g']
print(colors)
for k, col in zip(unique_labels, colors):
    if k == -1:
        col = 'k'
    mask = (labels == k)
    xy = X[mask & core_mask]
    plt.plot(xy[:, 0], xy[:, 1], 'o', markerfacecolor=col,
             markeredgecolor='k',
             markersize=6)

    xy = X[class_member_mask & ~core_mask]
    plt.plot(xy[:, 0], xy[:, 1], 'o', markerfacecolor=col,
             markeredgecolor='k',
             markersize=6)

plt.title('number of clusters: %d' % n_clusters_)
plt.show()
```



Спасибо за внимание!