Санкт–Петербургский государственный университет

Методические указания для статистической обработки данных в Python и R

Подготовили:
Писарева С.М.
Поляков И.М.

Санкт-Петербург
2022 г.

Содержание

1 Основы работы в Python

1.1 Вывод данных

Для вывода данных на экран используется команда *print()*. Внутри скобок пишем, что хотим вывести на экран. Если это текст, то обязательно указываем его внутри кавычек. Кавычки могут быть одинарными *('_ ')* или двойными *("_ ")*. Соответственно, следующие две строчки выведут одно и то же:

print('Python')
print("Python")

То, что мы пишем в круглых скобках у команды *print()*, называется аргументами или параметрами команды.

Команда *print()* позволяет указывать несколько аргументов через запятую или же не указывать ничего. В первом случае она просто выведет все элементы через пробел:

print(1, 2, 3, 4)
Output: 1 2 3 4

Во втором случае просто выведет пустую строку.

1.2 Типы данных

В Python есть несколько стандартных типов данных:

- Числа *(int, float)* – неизменяемый;
- Строки *(str)* – неизменяемый, упорядоченный;
- Списки *(list)* – изменяемый, упорядоченный;
- Словари *(dict)* – изменяемый, упорядоченный;
- Кортежи *(tuple)* – неизменяемый, упорядоченный;
- Множества *set* – изменяемый, неупорядоченный;
- Логический тип данных *bool*.

1.2.1 Преобразование типов

В Python имеется возможность преобразовывать данные из одного типа в другой. Например:

Строка в число:
int("100")
Output: 100

Число в строку:
str(52.7)
Output: '52.7'

Список в множество:
set([1, 2, 3, 3, 2, 1, 1])

Output: {1, 2, 3}

Строку в список:
list('string')
Output: ['s', 't', 'r', 'i', 'n', 'g']

1.2.2 Основные математические операции

\+ - сложение;

– - вычитание;

* - умножение;

/ - деление;

// - целочисленное деление;

% - остаток от деления;

** - возведение в степень.

1.2.3 Числовые типы данных

Целые числа представлены в Python типом данных *int*. Для преобразования в целочисленный тип данных используется команда *int()*.

Отличительной особенностью языка Python является неограниченность целочисленного типа данных. По факту, размер числа зависит только от наличия свободной памяти на компьютере. Это отличает Python от тех языков, где переменные целого типа имеют ограничение. Например, в языке C# диапазон целых чисел находится от -2^{63} до $2^{64} - 1$. То есть,

Команда
*print(1024 ** 25)*
Вполне естественно выведет:
Output: 1809251394333065553493296640760748560207343510400633813116524750123642650624

1.2.4 Числа с плавающей точкой

Наравне с целочисленным типом данных в Python есть возможность работы с дробными (вещественными) числами. Например, числа π, $\sqrt{5}$ и $\frac{5}{4}$ не являются целыми, и типа int недостаточно для их представления. Для преобразования в вещественный тип данных используется команда *float()*.

print(float('25.1'))
Output: 25.1

print(float(3))
Output: 3.0

1.2.5 Строковый тип данных

Строковый тип данных, как и числовой, очень часто используется в программировании. Для создания строковой переменной достаточно заключить необходимый текст в кавычки.

```
s1 = 'Python' # объявление строки
s2 = ''       # пустая строка
s3 = ' '      # строка, содержащая пробел
```

Строки можно складывать и умножать на число:

```
s1 = '12'
s2 = '23'
s3 = '34'
print(s1 + s2 + s3)
print(s1 * 3)
```

```
Output: 122334
Output: 121212
```

Очень часто бывает необходимо обратиться к конкретному символу в строке. Для этого в Python используются квадратные скобки [], в которых указывается индекс (номер) нужного символа в строке.

Пусть `s = 'Python'`. Ниже показано, как работает индексация.

```
s[0]    # 'P' - первый символ строки
s[1]    # 'y' - второй символ строки
s[2]    # 't' - третий символ строки
s[3]    # 'h' - чётвертый символ строки
s[4]    # 'o' - пятый символ строки
s[5]    # 'n' - шестой символ строки
```

Также строки имеют встренную функцию `len()`, которая показывает длину строки, то есть количество символов в ней.

```
s = 'Python'
print(len(s))
```

```
Output: 6
```

1.3 Условный оператор

Программы должны уметь выполнять разные действия в зависимости от введенных данных. Для принятия решения программа проверяет, истинно или ложно определенное условие. В Python существует несколько способов проверки, и в каждом случае возможны два исхода: истина *(True)* или ложь *(False)*.

Проверка условий и принятие решений по результатам этой проверки называется ветвлением (branching). Программа таким способом выбирает, по какой из возможных ветвей ей двигаться дальше.

В Python проверка условия осуществляется при помощи ключевого слова `if`.

Общий вид использования условного оператора:

```
if *логическое условие*:
        *блок кода, если условие истинно*
else:
        *блок кода, если условие ложно*
```

Двоеточие (:) в конце строки с инструкцией *if* сообщает интерпретатору Python, что дальше находится блок команд. В блок команд входят все строки с отступом под строкой с инструкцией *if*, вплоть до следующей строки без отступа. Если условие истинно, выполняется весь расположенный ниже блок. Если условие ложно, то интерпретатор игнорирует блок кода под *if* и сразу переходит к блоку кода под *else*, если он существует. Важно отметить, что отступы перед блоками обязательны.

Пример:

```
x = 5
if x <= 10:
        print(x + 10)
else:
        print(x * 10)
```

Результатом выполнения данной программы будет:

```
Output: 15
```

1.3.1 Операторы сравнения и логические операции

В Python существуют 6 основных операторов сравнения:

```
if x > 5        # если x больше 5
if x < 5        # если x меньше 5
if x >= 5       # если x больше или равен 5
if x <= 5       # если x меньше или равен 5
if x == 5       # если x равно 5
if x != 5       # если x не равно 5
```

Существуют ситуации, когда у нас есть несколько условий, и нам необходимо создать сложное условие. Для это в Python есть 3 логических оператора:

- *and*
- *or*
- *not*

Таблицы истинности совпадают с таблицами истинности логического умножения, логического сложения и отрицания соответственно.

Примеры использования:

```
a = 10
b = 5
if a <= 5 and a + b == 15:
        print(True)
Output:


if not(a <= 5) or a + b == 0:
```

print(True)

Output: True

1.4 Циклы

1.4.1 For

1.4.2 While

1.5 Функции

2 Элементы анализа данных

Математическая статистика – это раздел математики, посвященный методам сбора, анализа и обработки статистических данных для научных и практических целей.

Статистические данные – такие данный, которые получены в результате обследования большого числа объектов или явлений, то есть, математическая статистика имеет дело с массовыми явлениями.

Статистический анализ данных включает в себя:

- *Описательная статистика (Descriptive Statistics)*

  Сюда входят методы описания статистических данных, представления их в форме таблиц и распределений и пр.

- *Индуктивная статистика (Infernal Statistics)*

  По-другому это может называться как *"аналитическая статистика"* или *"теория статистических выводов"*. В основном здесь реализуются обработка данных, которые были получены в ходе эксперимента, и формулировка выводов, имеющих прикладное значение для конкретной области человеческой деятельности. Теория статистических выводов базируется на математическом аппарате теории вероятности.

Предметом изучения в статистике являются изменяющиеся (варьирующие) признаки, которые называются статистическими признаками.

Наличие общего признака является основой для образования статистической совокупности. То есть, статистическая совокупность – это результаты описания или измерения общих признаков объектов исследования.

Признаки и переменные – это измеряемые явления. Значения признака определяются при помощи специальных шкал наблюдения.

Данные – результаты некоторого количества измерений какой-либо переменной (признака) или переменных (признаков) *(вес, длина тела, пол, температура...)*.

Данные бывают:

- Категориальные *(Качественные)*
  - Номинальные *(nominal)*

    Взаимоисключающие и неупорядоченные категории (нельзя выстроить в последовательность);
  - Порядковые *(ordinal)*

    Взаимоисключающие и упорядоченные категории (могут быть упорядочены, но размер интервалов на шкале может быть не одинаков);
- Количественные *(Числовые)*
  - Дискретные *(discrete)*

    Как правило, целочисленные значения, типичные для счёта;
  - Непрерывные *(continuous)*

    Любые значения в определенном интервале.

Три основных концепции в анализе данных:

1. Распределение переменной и его описание;
2. Распределение выборочных средних и связь её с распределением переменной;
3. Статистика Критерия.

2.1 Распределение переменной

Частотное распределение переменной *(frequency distribution)* – это соответствие между значениями переменной и их вероятностями (или количеством таких значений в выборке).

Вариационный ряд – ряд, в котором сопоставлены (по степени возрастания или убывания) варианты и соответствующие им частоты.

Частота (f_i) – число, показывающее сколько раз повторяется значение (x_i) признака в вариационном ряду.

Частостью или относительной частотой (w_i) называется отношение частоты к объему выборки (n):

$$w_i = \frac{f_i}{n} \tag{1}$$

Наиболее популярными и употребительными графиками для изображения вариационных рядов, то есть соотношений между значениями признака и соответствующими частотами или относительными частотами, являются *гистограмма* и *полигон частот*.

Гистограмма – графическое представление частотного распределения, разбитого по интервалам, где высота столбика отражает частоту.

Полигон частот используют для дискретных рядов. На оси абсцисс откладываются значения аргумента, а на оси ординат – значения частот или относительных частот.

Генеральная совокупность – множество всех объектов в рассматриваемой конкретной задаче.

Выборочная совокупность или выборка – часть генеральной совокупности, которая охватывается тем или иным экспериментом (наблюдением, опросом). Чтобы свойства выборки отражали свойства генеральной совокупности *(репрезентативная выборка)*, сама выборка должна быть случайной, то есть, все объекты в генеральной совокупности должны иметь одинаковые шансы в неё, и попадание в выборку одного объекта никак не должно влиять на попадание другого.

Основные характеристики частотного распределения переменной:

2.1.1 "Середина" распределения

- Среднее значение *(mean)*
- Медиана *(median)*
- Мода *(mode)*

Это величины могут служить оценками генеральной совокупности, а в выборке – это наиболее эффективная и несмещённая оценка.

Среднее значение

Среднее значение – сумма всех значений переменной, делённая на количество значений.

Среднее для выборки:

$$\overline{X} = \frac{\sum\limits_i X_i}{n} \tag{2}$$

Среднее для генеральной совокупности:

$$\overline{\mu} = \frac{\sum X}{N} \tag{3}$$

Для вычисления среднего используются все значения набора данных, само значение определяется математически выполнимым алгебраическим выражением, а также известно выборочное распределение. К недостаткам стоит отнести сильную зависимость от выбросов и асимметричных данных.

Медиана

Медиана - значение, которое делит распределение пополам. То есть, половина значений больше медианы, другая половина – не больше.

Если распределение не симметричное, медиана лучше характеризует центр распределения. Она содержит меньше информации, чем среднее, но зато она не чувствительна к выбросам и асиметричным данным и может применяться даже в случае, если не для всех значений измерения точные. Стоит отметить, что значение этой характеристики не определяется алгебраически, что может затруднить дальнейшее описание.

Распределение также можно поделить не на две части, а на четыре *(квартили)*, восемь *(октили)*, сто *(процентили)*, N *(квантили)*.

Квартили *(quartiles)* делят распределение на четыре части так, что в каждой из них оказывается поровну значений.

- 1-ая квартиль == 25% процентиль;
- 2-ая квартиль == медиана;
- 3-ая квартиль == 75% процентиль.

Интерквартильный размах – разница между третьей и первой квартилями.

Мода

Мода – наиболее часто встречающееся значение.

Данная характеристика легко определяется для категориальных данных, но она игнорирует большую часть информации, не определяется алгебраически, а также ничего не известно про выборочное распределение.

2.1.2 "Ширина" распределения

- Размах *(range)*
- Стандартное отклонение *(standart deviation)*
- Дисперсия *(variance)*

Размах

Размах – разность между максимальным и минимальным значениями:

$$range = X_n - X_1 \tag{4}$$

Размах хорош тем, что легко считается. Плох тем, что зависит только от 2-х точек из распределения. Недооценивает истинный размах генеральной совокупности. Для описания "Ширины"распределения помимо размаха стоит привести ещё какую-нибудь характеристику разброса.

Стандартное отклонение

3 Методы математической статистики

3.1 Выборка. Описательная статистика

3.2 Точечные оценки. Свойства и методы построения

3.3 Доверительные интервалы

3.4 Статистические гипотезы. Параметрические критерии

3.4.1 Критерии о параметрах нормального распределения

3.4.2 Критерии о параметрах нормального и биномиального распределений

3.5 Критерии однородности

3.5.1 Параметрические критерии однородности

3.5.2 Непараметрические критерии однородности

3.5.3 Однофакторный дисперсионный анализ

3.6 Критерии согласия. Таблицы сопряженности

3.6.1 Критерий согласия хи-квадрат и Колмогорова

3.6.2 Критерий нормальности

3.6.3 Таблицы сопряженности

3.7 Регрессионный анализ

3.7.1 Множественная линейная регрессия

3.7.2 Корреляционный анализ

4 Обзор библиотек Python для проведения анализа данных и моделирования

Уже довольно давно Python очаровывает учёных, занимающихся данными. Чем больше идёт взаимодействие с ресурсами, литературой, курсами, тренингами и людьми в науке о данных, тем более глубокие знания о Python приобретаются.

Специалисты в области Data Science точно знают о библиотеках Python, которые можно использовать в науке о данных, но когда в интервью просят назвать их или указать их функции, они часто могут не вспомнить даже и 5 библиотек.

Ниже представлен список и краткое изложение библиотек Python, которые помогают в области Data Science.

4.1 Pandas

Pandas – это пакет Python с открытым исходным кодом, который предоставляет высокоэффективные, просты в использовании структуры данных и инструменты анализа. Pandas – это инструмент, предназначенный для быстрой и простой обработки данных, чтения, агрегирования и визуализации.

Pandas берёт данные из файлов CSV, TSV или базы данных SQL и создает объект Python со строками и столбцами, который называется *фреймом данных*. Фрейм данных очень похож на табоицу в статистическом программном обеспечении (например, в Excel или SPSS).

Что можно делать с помощью Pandas?

1. Индексирование, манипулирование, переименование, сортировка, объединение фрейма данных;
2. Обновить, добавить, удалить столбцы из фрейма данных;
3. Восстановить недостающие файлы, обработать недостающие данные или NAN;
4. Построить гистограмму или прямоугольную диаграмму.

Это делает Pandas фундаментальной библиотекой в изучении Python для Data Science.

4.2 NumPy

NumPy – один из самых основных пакетов в Python, универсальный пакет для обработки массивов. Он предоставляет высокопроизводительные объекты многомерных массивов и инструменты для работы с ними. NumPy – эффективный контейнер универсальных многомерных данных. Основной объект NumPy – это однородный многомерный массив.

NumPy используется для обработки массивов, в которых хранятся значения одного и того же типа данных. Этот пакет облегчает математические операции над массивами и их векторизацию, что значительно повышает производительно, и, соответственно, ускоряет время выполнения.

Что можно делать с помощью NumPy?

1. Основные операции с массивами: добавление, умножение, срез, выравнивание, изменение формы, индексирование массивов;
2. Расширенные операции с массивами: стековые массивы, разбиение на секции, широковещательные массивы;
3. Работа с DateTime или линейной алгеброй;
4. Основные нарезки и расширенное индексирование в NumPy Python.

4.3 SciPy

Библиотека *SciPy* является одним из ключевых пакетов, которые составляют стек SciPy. Он основывается на обхекте массива NumPy и является частью стека, который включает в себя такие инструменты, как Matplotlib, Pandas и SymPy с дополнительными инструментами.

Библиотека Scipy содержит модули для эффективных математических процедур, таких как линейная алгебра, интерполяция, оптимизация, интеграция и статистика. Основной функционал SciPy построен на NumPy и его массивах.

SciPy использует массивы в качестве базовой структуры данных. Он имеет различные модули для выполнения общих задач научного программирования, таких как линейная алгреба, интеграция, математический анализ, обыкновенные дифференциальные уравнения и обработка сигналов.

Что можно делать с помощью SciPy?

1. Математические, научные, инженерные вычисления;
2. Процедуры численной интеграции и оптимизации;
3. Поиск минимумов и максимумов функций;
4. Вычисление интегралов функции;
5. Поддержка специальных функций;
6. Работа с генетическими алгоритмами;
7. Решение обыкновенных дифференциальных уравнений.

4.4 Matplotlib

Matplotlib – основная библиотека Python, предназначенная для визуализации данных и предоставляющая API для встраивания графиков в приложения. Очень напоминает MATLAB, встроенный в язык программирования Python.

Что можно делать с помощью Matplotlib?

1. Линейные диаграммы;
2. Точечные диаграммы;
3. Диаграммы с областями;
4. Столбцовые диаграммы и гистограммы;
5. Круговые диаграммы;
6. Диаграммы «стебель-листья»;
7. Контурные графики;
8. Поля векторов;
9. Спектрограммы.

4.5 Seaborn

Seaborn – это библиотека визуализации данных на основе Matplotlib, предоставляющая высокоуровенвый интерфейс для изображения интересных и информативных статистических графиков. Проще говоря, Seaboen – это расширение Matplotlib с дополнительными возможностями. Разница между ними состоит в том, что Matplotlib используется для основного построения столбцовых, круговых, линейных, точечных диаграмм, в то время как Seaborn предоставляет множество шаблонов визуализации с меньши количеством синтаксических правил.

Что можно делать с помощью Seaborn?

1. Определять отношения между несколькими переменными (корреляция);
2. Соблюдать качественные переменные для агрегированных статистических данных;
3. Анализировать одномерные или двумерные распределения и сравнивать их между различными подмножествами данных;
4. Построить модели линейной регрессии для зависимых переменных;
5. Обеспечить многоуровневые абстракции, многосюжетные сетки.

4.6 Scikit Learn

Scikit Learn – представляет собой надёжную библиотеку машинного обучения для Python. он включает в себя алгоритмы Machine Learning, такие как SVM, Random Forests, K-Means кластеризацию, спектральную кластеризацию, сдвиг среднего значения, перекрёстную проверку и другие.

Scikit Learn предоставляет ряд контролируемых и неконтролируемых алгоритмов обучения через согласованный интерфейс в Python. Данный пакет будет руководством для того, чтобы модели контролируемого обучения, например, Naive Bayes, группировали непомеченные данные, такие как K-Means.

Что можно сделать с помощью Scikit Learn?

1. Классификация: обнаружение спама, распознавание изображений;
2. Кластеризация: воздействия лекарственных препаратов, цена акций;
3. Регрессия: сегментация клиентов, группировка результатов эксперимента;
4. Уменьшение размерности: визуализация, повышенная эффективность;
5. Выбор модели: повышенная точность благодаря настройке параметров;
6. предварительная обработка: подготовка входных данных в виде текста для обработки с помощью алгоритмов машинного обучения.

4.7 TensorFlow

TensorFlow – это библиотека искуственного интеллекта (AI), которая помогает разработчикам создавать крупномасштабные нейронные сети со многими слоями, используя графики потоков данных. TensorFlow также облегчает построение моделей глубокого обучения, продвигает современную технологию ML/AI и позволяет легко развертывать приложения на базе ML.

TensorFlow достаточно эффективен, когда дело доходит до классификации, восприятия, понимания, обнаружения, прогнозирования и создания данных.

Что можно делать с помощью TensorFlow?

1. Распознование голоса/звука, интернет вещей, автомобильная промышленность, безопасность, UX/UI, телекоммуникации;
2. Анализ настроений – в основном для CRM или CX;
3. Текстовые приложения – обнаружение угроз, Google Translate, Gmail Smart Reply;
4. Распознавание лиц – Facebook's Deep Face, Photo tagging, Smart Unlock;
5. Временные ряды – рекомендации от Amazon, Google и Netflix;
6. Обнаружение видео – обнаружение движения, обнаружение угроз в реальном времени в играх, безопасности, аэропортах.

4.8 Keras

Keras – это высокоуровневый API TensorFlow для создания и обучения кода глубоких нейронных сетей. Это библиотека нейронных сетей с открытым исходным кодом на Python. С Keras статистическое моделирование, работа с изображениями и текстом намного легче с упрощённым для глубокого обучения. Keras создан для Python и делает его более удобным, модульным и компонуемым, чем TensorFlow.

Что можно делать с помощью Keras?

1. Определить процентную точность;
2. Функция вычисления потерь;
3. Создать пользовательские функциональные слои;
4. Встроенные функции обработки данных и изображений;
5. Функции с повторяющимися блоками кода: глубиной 20, 50, 100 слоев.

4.9 Statsmodels

Statsmodels – это универсальный пакет Python, который обеспечивает простые вычисления для описательной статистики и оценки и формирования статистических моделей. Данный пакет упрощает проведение статистических тестов и исследование статистических данных.

Что можно делать с помощью Statsmodels?

1. Линейная регрессия;
2. Корреляция;
3. Метод наименьшего квадрата (OLS);
4. Анализ выживания;
5. Обобщённые линейные модели и байесовская модель;
6. Однофакторный и двухфакторный анализ, проверка гипотез.

4.10 Plotly

Plotly – это графическая библиотека для Python. Пользователи могут импортировать, копировать, вставлять или передавать данные, которые должны быть проанализированы и визуализированы. Данный пакет можно использовать, когда необходимо создовать, отображать и обновлять фигуры, наводить курсор на текст для получения подробной информации. PLotly также имеет дополнительную функцию отправки данных на облачные серверы.

Что можно делать с помощью Plotly?

1. Основные диаграммы:
    - Линейные;
    - Круговые;
    - Точечные;
    - Пузырьковые;
    - Графики Ганта;
    - Санбёрст;
    - Древовидые;
    - Санкей;
    - Графики с областями;
2. Статистические стили и стили Seaborn:
    - Ошибки;
    - Гистограммы;
    - Диаграммы Facet и Trellis;
    - Деревообразные графики;
    - Графики-скрипки;
    - Линия тренда.
3. Научные карты:
    - Контур;
    - Троичный сюжет;
    - Логарифмический график;
    - Поля векторов;
    - Ковровый график;
    - Радарчарт;
    - Тепловые карты;
    - Роза ветров;
    - Полярный сюжет.
4. Финансовые графики;
5. Карты;
6. Подграфики;
7. Трансформации;
8. Взаимодействие Jupyter Widgets.

5 Основы R

6 Проведение анализа в R