**Создание графиков в Python**

В этой лабораторной работе мы с помощью Python научимся создавать графики, гистограммы, столбчатые диаграммы, точечные диаграммы и другие. Но для этого нам сперва нужно сказать пару слов о библиотеке matplotlib.

**Общие принципы matplotlib**

В Python существует множество библиотек для визуализации данных, однако базовой и одной из самых распространённых является библиотека matplotlib. С её помощью мы разберём базовые типы визуализаций.

Библиотека matplotlib устанавливается и импортируется так же, как и другие библиотеки:

|  |  |
| --- | --- |
|  | >>> **import** matplotlib.pyplot **as** plt |

Для иллюстрации работы библиотеки matplotlib мы будем использовать [датасет](https://www.kaggle.com/datasets/shivamb/disney-movies-and-tv-shows) с информацией о контенте стриминговой платформы Disney+.

Создание визуализаций можно представить как последовательное добавление различных элементов итогового графика:

* пустого графика с осями;
* элементов, отображающих переменные;
* подписей;
* цвета и стиля.

**Гистограммы**

Напомним, что гистограммы — такой вид визуализаций, где рассматривается распределение значений по одному параметру. Например, мы можем посмотреть, каким было распределение значений по году производства фильма или шоу.

|  |  |
| --- | --- |
|  | **import** pandas **as** pd |
|  | df = pd.read\_csv('disney\_plus\_titles.csv') |
|  | plt.hist(df['release\_year']) |
|  | plt.show() |

По оси X указан год производства, а по оси Y — количество фильмов, произведённых в этот год. Как видно из этого графика, производство фильмов начало расти после восьмидесятых, причём после начала нулевых производство фильмов резко увеличилось.

Столбцы на получившейся гистограмме довольно широкие, что позволяет нам довольно примерно оценить соотношение года и количества произведённых фильмов. Однако это регулируемый параметр. По умолчанию он равен 10, но давайте удвоим количество столбцов (иначе их называют корзинами):

|  |  |
| --- | --- |
|  | plt.hist(df['release\_year'], bins=20) |
|  | plt.show() |

Мы добавили параметр функции bins. Именно он отвечает за количество столбцов. Конечно, это не единственный параметр, с помощью которого мы настраиваем способ отображения данных, по ходу этого параграфа мы увидим и другие.

Помимо этого, мы можем добавить на график сразу несколько разных данных. Предположим, мы хотим одновременно посмотреть на гистограммы фильмов и ТВ-шоу. Для этого сделаем отдельные наборы данных:

|  |  |
| --- | --- |
|  | df\_film = df[df['type'] == 'Movie'] |
|  | df\_show = df[df['type'] == 'TV Show'] |

Теперь добавим интересующие нас данные на график:

|  |  |
| --- | --- |
|  | plt.hist(df\_film['release\_year']) |
|  | plt.hist(df\_show['release\_year']) |
|  | plt.show() |

Обратите внимание на интересную вещь. Мы рисуем две гистограммы, но видим их не отдельно, а вместе. Дело в том, что если мы определяем две гистограммы подряд, то пакет интерпретирует это как предложение нарисовать их *поверх* друг друга. Это может быть полезным.

С помощью такой визуализации мы можем увидеть, как менялись темпы производства фильмов и шоу. Однако эта визуализация будет не вполне понятна для аудитории: что именно изображено на графике? Что отражает график синего и оранжевого цвета? Эти вопросы стоит прояснить, поэтому мы добавим текстовые объяснения на график.

|  |  |
| --- | --- |
|  | plt.hist(df\_film['release\_year'], label = 'Films') |
|  | plt.hist (df\_show['release\_year'], label = 'Shows') |
|  | plt.legend() |
|  | plt.show() |

Давайте разберём, что мы добавили на график:

* label — с помощью этого параметра мы добавили подписи к гистограммам;
* legend — с помощью этого аргумента мы выводим на график подписи к данным к ним — это не только названия данных, но и, например, названия осей и графика в целом.

Посмотрим, как добавить эти названия тоже.

|  |  |
| --- | --- |
|  | plt.hist(df\_film['release\_year'], label = 'Films', bins = 20) |
|  | plt.hist(df\_show['release\_year'], label = 'Shows', bins = 20) |
|  | plt.xlabel("Year") |
|  | plt.ylabel("amount") |
|  | plt.title('Hist #1') |
|  | plt.legend() |
|  | plt.show() |

К нашему коду мы добавили строки, которые задают названия осей и графика в целом. Посмотрим теперь, как работать с категориальными переменными.

**Столбчатые диаграммы**

Для визуализации категориальных переменных мы будем использовать столбчатые диаграммы. Они могут внешне напоминать гистограммы, но сильно отличаются по смыслу, в первую очередь как раз тем, что используются при работе с категориальными шкалами.

Для начала посмотрим, какие возрастные категории фильмов и шоу были самыми распространёнными. В нашем наборе данных есть переменная Rating, которая отражает, какой именно возрастной рейтинг был присвоен каждому фильму и шоу. Для начала посмотрим на эти данные без визуализации. Напомним, что для этого можно использовать функцию value\_counts():

|  |  |
| --- | --- |
|  | >>> df['rating'].value\_counts() |
|  | TV-G 318 |
|  | TV-PG 301 |
|  | G 253 |
|  | PG 236 |
|  | TV-Y7 131 |
|  | TV-14 79 |
|  | PG-13 66 |
|  | TV-Y 50 |
|  | TV-Y7-FV 13 |
|  | Name: rating, dtype: int64 |

Так мы можем понять наиболее и наименее распространённые рейтинги. В кодбуке к набору данных (или просто в интернете) можно узнать, чему соответствует каждый рейтинг. Например, TV-G означает, что фильм подходит для всех возрастных категорий. Однако теперь попробуем визуализировать эти данные.

Для этого мы будем использовать следующую запись:

|  |  |
| --- | --- |
|  | df['rating'].value\_counts().plot(kind='bar', ylabel='frequency') |
|  | plt.show() |

Несколько пояснений по этому коду, поскольку такой вид записи отличается от тех, что мы рассматривали ранее. Раз мы используем преобразованные данные, то код устроен следующим образом:

* сначала мы проделываем работу с данными — это первая часть кода;
* затем используем функцию plot() — она указывает, что полученные данные необходимо визуализировать;
* параметры этой функции задают способ визуализации и дополнительные параметры, в нашем случае — тип графика и подпись к оси Y.

Вот ещё возможные значения параметра kind:

* line — линейный график;
* hist — гистограмма;
* box — бокс-плот;
* pie — круговая диаграмма;
* kde — линия плотности распределения;
* density — гибрид линейного графика и гистограммы;
* scatter — точечная диаграмма;

В итоге получаем количество фильмов и шоу по возрастным категориям:

Мы можем отдельно оценить количество фильмов и ТВ-шоу по категориям. Опять же, начнём с кода, который покажет нам такие данные:

|  |  |
| --- | --- |
|  | df.groupby('type')['rating'].value\_counts() |
|  | type rating |
|  | Movie G 253 |
|  | PG 235 |
|  | TV-G 233 |
|  | TV-PG 181 |
|  | PG-13 66 |
|  | TV-14 37 |
|  | TV-Y7 36 |
|  | TV-Y7-FV 7 |
|  | TV-Y 3 |
|  | TV Show TV-PG 120 |
|  | TV-Y7 95 |
|  | TV-G 85 |
|  | TV-Y 47 |
|  | TV-14 42 |
|  | TV-Y7-FV 6 |
|  | PG 1 |
|  | Name: rating, dtype: int64 |

Тут вас может смутить синтаксис, расскажем о нём чуть подробнее:

* на объекте df вызываем метод groupby, передав в качестве параметра аргумент type — этот метод группирует датафрейм по переменной type;
* он возвращает новый объект — сгруппированный датафрейм;
* у него выбираем все значения по переменной rating;
* и подсчитываем количество наблюдений для каждого уникального значения.

Полученные данные уже сложнее воспринимать. Рассмотрим код для визуализации этих данных:

|  |  |
| --- | --- |
|  | vis\_1 = df.groupby('type')['rating'].value\_counts() |
|  | vis\_1.unstack(0).plot(kind='bar', ylabel='frequency') |

Первой строкой мы сохранили данные в преобразованном виде.

Второй строкой мы задали параметры визуализации. Все его части нам знакомы, кроме функции unstack. Она нужна для того, чтобы на визуализации объединить под одной категорией и фильмы, и ТВ-шоу. Чтобы лучше понять принцип работы этой функции, вы можете удалить эту часть кода и попробовать визуализировать без неё. В текущем же виде мы получим вот такую визуализацию:

Глядя на эту визуализацию, можно обнаружить, что, хотя в большинстве категорий фильмы преобладают над ТВ-шоу, есть несколько категорий, где ситуация радикально иная.

**Линейные графики**

Линейные графики нужны нам, когда мы визуализируем количественные показатели и их связь между собой. В нашем наборе данных один из таких показателей — это год производства фильма. Однако все остальные показатели — категориальные, поэтому нам необходимо выполнить некоторые преобразования. Например, мы можем посчитать, какое количество фильмов было произведено в каждый год.

Для этого используем следующий код:

|  |  |
| --- | --- |
|  | >>> df['release\_year'].value\_counts() |
|  | 2021 125 |
|  | 2020 114 |
|  | 2019 99 |
|  | 2017 69 |
|  | 2018 65 |
|  | ... |
|  | 1944 1 |
|  | 1945 1 |
|  | 1970 1 |
|  | 1968 1 |
|  | 1928 1 |
|  | Name: release\_year, Length: 90, dtype: int64 |

Так мы узнали, сколько фильмов было произведено в каждый год. Теперь визуализируем это соотношение. Чтобы это сделать, нам необходимо ещё одно преобразование — связь года и количества произведённых фильмов необходимо сохранить как датафрейм.

|  |  |
| --- | --- |
|  | >>> film\_year = pd.DataFrame(df['release\_year'].value\_counts().reset\_index().values, columns=["Year", "amount"]) |
|  | >>> film\_year = film\_year.sort\_values('Year', ascending = True) |
|  | >>> print(film\_year) |
|  | Year amount |
|  | 89 1928 1 |
|  | 74 1932 3 |
|  | 73 1933 3 |
|  | 56 1934 4 |
|  | 57 1935 4 |
|  | .. ... ... |
|  | 3 2017 69 |
|  | 4 2018 65 |
|  | 2 2019 99 |
|  | 1 2020 114 |
|  | 0 2021 125 |
|  |  |
|  | [90 rows x 2 columns] |

Этим кодом мы создали первой строкой датафрейм с колонками “Year” и “amount”, а второй строкой отсортировали его по годам. Дальнейшая визуализация будет очень простой:

|  |  |
| --- | --- |
|  | plt.plot(film\_year['Year'], film\_year['amount']) |
|  | plt.show() |

Простой, но отчётливо наблюдаемый тренд — это увеличение производства фильмов и ТВ-шоу от года к году.

**Точечная диаграмма**

Точечная диаграмма нужна нам для визуализации двух переменных и, как мы уже видели в предыдущем параграфе, часто используется при работе с корреляциями.

В нашем наборе данных есть две подходящие переменные: год производства и длительность фильма. Однако длительность фильма изначально представлена как строка, поскольку содержит не только данные о количестве минут или сезонов, но и текст. Это не подходит для наших задач, поэтому мы преобразуем переменную. Мы не будем подробно разбирать, как именно это сделать в Python, однако код для этой операции выглядит следующим образом:

|  |  |
| --- | --- |
|  | df['duration'] = df['duration'].replace('min' **or** 'Season', '', regex=True).apply(pd.to\_numeric, errors='coerce') |
|  | df\_2 = df[df['duration'] > 15] |

Мы убрали текст из колонки и превратили строки в числа, а затем исключили значения меньше 15, чтобы убрать данные, касающиеся сезонов, а не фильмов.

|  |  |
| --- | --- |
|  | plt.scatter(df\_2['release\_year'], df\_2['duration'], alpha=0.5) |
|  | plt.show() |

Код выглядит очень просто. Мы указываем тип визуализации, переменные по шкале X и Y, а также для прозрачности показатель alpha, с которым мы уже сталкивались.

По этой визуализации сложно однозначно сказать, изменилась ли значимо средняя длина фильмов за указанный период, однако она позволяет нам увидеть сразу на одной картинке связь между годом и длительностью фильма. В ряде случаев такие визуализации оказываются полезны для отражения зависимостей различных переменных.

Задачи для самостоятельного решения.

* 1. **Постройте гистограмму распределения показателя атаки (Attack) у покемонов**

Как мы узнали из прошлой части, гистограмма показывает распределение численных значений. Давайте попробуем визуализировать диапазон атаки покемонов, чтобы узнать пропорции «сильных» и «слабых». А также понять, какова примерная мода значений. В используемой базе данных переменная именуется Attack.

CSV-файл доступен под названием "PokemonData.csv".

* 1. **Гистограмма: постройте две переменные на одном графике**

В предыдущем задании вы успешно построили распределение атаки. Давайте теперь попробуем отразить две шкалы на одном графике. Посмотрим, насколько сильно отличается распределения обычной атаки покемона и его специальной атаки.

Правда ли, что специальная атака мощнее обычной базовой атаки? Показатели специальной атаки записаны в переменной SpAtk.

* 1. **Гистограмма: прозрачность**

Отлично! Вы получили нужный результат. Но на такой график не очень удобно смотреть, правда?

Удобно подмечать, что не так с графиком, когда представляешь, что на него смотрит посторонний человек. Кратко пройдемся по ошибкам:

1. Его колонки имеют сплошную заливку. Так неудобно смотреть на некоторые значения.
2. Непонятно, каким цветом выделяется обычная атака, а каким – специальная.
3. Что на графике изображено в принципе?

Для полного понимания происходящего исправим наш график. Сначала настроим прозрачность переменных. В строках каждой переменных в методе plt.hist укажем параметр alpha и приравняем его к 0.5.

* 1. **Гистограмма: легенда**

В строках с переменными в функции plt.hist добавим аргументы label, чтобы присвоить переменной название, отражающееся в легенде. Пропишем, что label="Обычная атака" и в строке со специальной атакой label="Специальная атака". После этого добавим строчку plt.legend(), чтобы вывести эти названия в легенду.

* 1. **Гистограмма: подписи к осям**

Используем предыдущий график. Подпишем оси графика с помощью команд plt.xlabel('Название оси') и plt.ylabel('Название оси'). Для оси X укажите значение "Мощность атаки", а для Y - "Количество покемонов".

* 1. **Точечная диаграмма**

Взглянем на соотношение стандартной атаки и стандартной защиты покемонов. Правда ли, что чем покемон сильнее, тем больше его защита? Постройте точечную диаграмму, чтобы проверить это. Переменная с характеристикой атаки: 'Attack'. Переменная с характеристикой защиты: 'Defense'.

* 1. **Точечная диаграмма: прозрачность**

На прошлом шаге мы построили точечную диаграмму. Теперь давайте сделаем точки прозрачнее, чтобы была видна кучность результатов. Для этого прямо внутри функции plt.scatter нужно ввести параметр alpha. Сделаем его значение равным 0.3.

* 1. **Столбчатая диаграмма**

Визуализируем частоту наблюдений по типам покемонов. Информация о типаже покемона заложена в переменной 'Type1'. Отразим частоту наблюдений по категориям с помощью столбчатой диаграммы. Напомним, что синтаксис барчарта выглядит следующим образом: df['variable'].value\_counts().plot(kind = 'bar')/

* 1. **Столбчатая диаграмма: несколько переменных.**

Отлично! Теперь попытаемся отобразить пропорцию легендарных и нелегендарных покемонов в каждом виде. И сделаем мы это на одном барчарте.

Переменная легендарности обозначена в данных как Legendary. Сделайте так, чтобы в каждом типе покемона отображался столбик с количество легендарных и обычных покемонов. Помните, что для отражения данных по нескольким категориям нужно их сгруппировать перед визуализацией. Не забывайте о том, что нам понадобится функция .unstack(0), чтобы категории разделялись по цвету. Синтаксис, который может вам пригодиться:

df.groupby('grouping\_variable')['categorical\_variable'].value\_counts().unstack(0).plot(kind = 'bar')

* 1. **Столбчатая диаграмма: названия осей**

Остался последний штрих. Давайте внутри функции plot() переименуем оси и добавим название, чтобы было понятно, о чем наша визуализация. За это отвечают параметры xlabel, ylabel, title. Пусть название оси Х будет "Тип покемонов", оси Y — "Количество", а название –"Легендарные покемоны по типам в сравнении с обычными".