**Практическая работа №2**

**Тема: «Предобработка данных в Pandas»**

**Техническое задание**

1. Файл Notebook Jupiter назвать по образцу: Practice2\_ SpirinIgor.ipynb
2. Содержимое вашего файла Practice2\_ XXXXXX.ipynb в Notebook Jupiter оформить по шаблону файла Practice2\_temple.ipynb
3. Скачать dataset Datasets, должен содержать более 10 000 записей содержать дубликаты и пустые значения
4. Задание сохранить на GitHub в ваш репозиторий

**datasets от kaggle**

[**https://www.kaggle.com/datasets**](https://www.kaggle.com/datasets)

**100 datasets для скачивания**

[**https://vc.ru/u/1389654-machine-learning/596807-100-datasetov-dlya-skachivaniya**](https://vc.ru/u/1389654-machine-learning/596807-100-datasetov-dlya-skachivaniya)

**Подборка datasets для машинного обучения**

[**https://habr.com/ru/articles/452392/**](https://habr.com/ru/articles/452392/)

**Теория**

**Введение в библиотеку pandas**

**Библиотека pandas** - это мощный инструмент для работы с данными в языке программирования Python. Она предоставляет простые и эффективные структуры данных, такие как **DataFrame** и **Series**, а также множество функций для манипуляций с данными. Pandas разработана для облегчения анализа и преобразования данных, а также их подготовки к визуализации или машинному обучению.

**Основные особенности pandas:**

* **DataFrame**: Табличная структура данных, похожая на таблицу в реляционных базах данных. Позволяет хранить и анализировать данные в удобной форме.
* **Series**: Одномерная структура данных, аналогичная массиву или списку. Используется для хранения столбцов в DataFrame.
* **Мощные операции с данными**: pandas предоставляет широкий набор методов для фильтрации, сортировки, агрегации и преобразования данных.
* **Работа с разными источниками данных**: Можно легко импортировать данные из файлов (CSV, Excel), баз данных, а также веб-ресурсов.
* **Обработка пропущенных значений**: pandas предоставляет инструменты для обнаружения и обработки пропущенных данных.

Pandas приносит значительные преимущества при работе с данными:

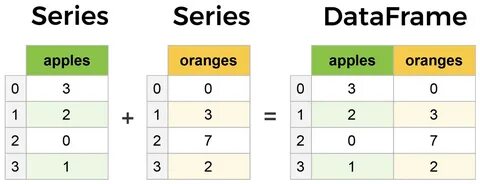
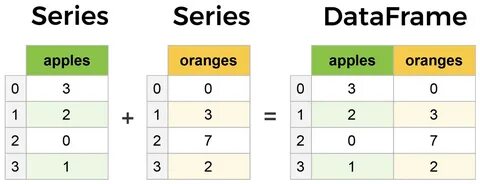
1. **Анализ данных**: Вы можете быстро провести анализ больших объемов данных и извлечь полезную информацию. Pandas предоставляет инструменты для исследования данных, вычисления статистик и выделения закономерностей.
2. **Подготовка данных**: Pandas позволяет легко очищать и преобразовывать данные для последующего анализа или использования в машинном обучении. Вы можете удалять пропущенные значения, масштабировать признаки и выполнять другие манипуляции с данными.
3. **Интеграция данных**: Библиотека Pandas позволяет объединять данные из разных источников, таких как CSV-файлы, базы данных и веб-сервисы. Это делает процесс интеграции данных более простым и эффективным.
4. **Визуализация данных**: Pandas интегрируется хорошо с библиотеками для визуализации данных, такими как Matplotlib и Seaborn. Вы можете легко создавать графики и диаграммы для наглядного представления данных.
5. **Поддержка временных рядов**: Pandas обладает мощными средствами для анализа и визуализации временных рядов. Это особенно полезно при работе с данными о финансах, погоде, или любыми данными, в которых временной аспект играет важную роль.
6. **Интеграция с NumPy**: Pandas построен поверх библиотеки NumPy, что означает, что вы можете легко выполнять вычисления с данными, представленными в формате Pandas DataFrame или Series, используя множество функций и методов NumPy.

**Список основных методов Pandas**

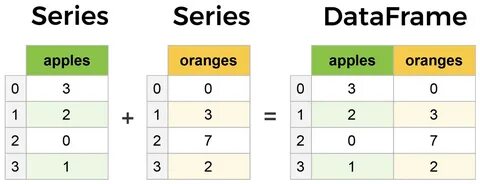
1. **pd.DataFrame(data, columns**): Создает новый DataFrame из данных. data может быть списком, массивом, словарем и другими структурами данных, а columns - список с именами столбцов.
2. **df.head(n):** Возвращает первые n строк DataFrame.
3. **df.tail(n):** Возвращает последние n строк DataFrame.
4. **df.info():** Выводит информацию о DataFrame, включая количество непропущенных значений, типы данных и использование памяти.
5. **df.describe():** Возвращает статистический анализ числовых столбцов, включая среднее, стандартное отклонение, минимум и максимум.
6. **df.shape:** Возвращает кортеж с количеством строк и столбцов в DataFrame.
7. **df.columns:** Возвращает список с названиями столбцов.
8. **df.index:** Возвращает индекс DataFrame.
9. **df.values:** Возвращает данные DataFrame в виде массива NumPy.
10. **df.set\_index(keys):** Устанавливает индекс DataFrame, где keys - столбец или список столбцов, которые будут индексами.
11. **df.reset\_index():** Сбрасывает индекс, восстанавливая его в числовой.
12. **df.sort\_values(by):** Сортирует DataFrame по указанному столбцу или столбцам.
13. **df.groupby(by):** Группирует данные по указанному столбцу или столбцам для проведения агрегации.
14. **df.apply(func):** Применяет функцию func к каждому столбцу или строке в DataFrame.
15. **df.isna():** Возвращает DataFrame с булевыми значениями, указывающими на пропущенные значения.
16. **df.dropna():** Удаляет строки с пропущенными значениями.
17. **df.fillna(value):** Заменяет пропущенные значения значением value.
18. **df.pivot\_table(values, index, columns, aggfunc):** Создает сводную таблицу, где values - агрегируемые значения, index - строки, columns - столбцы, и aggfunc - функция агрегации.
19. **df.merge(other, on):** Объединяет два DataFrame по общему столбцу или столбцам.
20. **df.plot(kind):** Создает график из данных, где kind - тип графика, такой как 'line', 'bar', 'hist', 'scatter' и другие.
21. **df.to\_csv(file\_path):** Сохраняет DataFrame в CSV-файл.
22. **df.to\_excel(file\_path):** Сохраняет DataFrame в Excel-файл.
23. **df.to\_sql(table\_name, connection):** Записывает DataFrame в SQL-базу данных.
24. **df.loc[row, column]:** Индексация DataFrame по меткам, позволяет выбирать конкретные строки и столбцы.
25. **df.iloc[row, column]:** Индексация DataFrame по целочисленным индексам, позволяет выбирать конкретные строки и столбцы.
26. **df.rename(columns):** Переименовывает столбцы DataFrame в соответствии с указанными именами в словаре columns.
27. **df.drop(columns):** Удаляет указанные столбцы из DataFrame.
28. **df.duplicated():** Возвращает булевы значения, указывающие на дубликаты строк.
29. **df.drop\_duplicates():** Удаляет дубликаты строк из DataFrame.
30. **df.corr():** Возвращает корреляционную матрицу для числовых столбцов.
31. **df.merge():** Объединяет два DataFrame по заданным столбцам и ключам.
32. **df.sample(n):** Возвращает случайные n строк из DataFrame.
33. **df.nlargest(n, column):** Возвращает n строк с наибольшими значениями в указанном столбце.
34. **df.nsmallest(n, column):** Возвращает n строк с наименьшими значениями в указанном столбце.
35. **df.drop(columns):** Удаляет указанные столбцы из DataFrame.
36. **df.astype(types):** Изменяет типы данных столбцов DataFrame в соответствии с указанными типами в словаре types.
37. **df.melt(id\_vars):** Переводит DataFrame в "расплавленный" формат, где столбцы преобразуются в два столбца: "переменная" и "значение".
38. **df.replace(to\_replace, value):** Заменяет значения в DataFrame на указанные значения.
39. **df.str.replace(old, new):** Производит замену текстовых значений в столбцах с использованием регулярных выражений.
40. **df.to\_datetime(column):** Преобразует столбец в формат даты и времени.
41. **df.resample(rule):** Применяет ресемплинг временных данных с указанным правилом (например, 'D' для дней, 'M' для месяцев).
42. **df.diff(periods):** Вычисляет разницу между текущим и предыдущим значением в столбце.
43. **df.rolling(window):** Выполняет скользящее окно для вычисления статистик на скользящих интервалах времени.
44. **df.nunique():** Возвращает количество уникальных значений в каждом столбце.
45. **df.value\_counts():** Подсчитывает количество вхождений каждого уникального значения в столбце.
46. **df.to\_dict():** Преобразует DataFrame в словарь, где ключами могут быть столбцы.
47. **df.to\_numpy():** Преобразует DataFrame в массив NumPy.
48. **df.dtypes:** Возвращает типы данных для каждого столбца в DataFrame.
49. **df.duplicated():** Возвращает булевы значения, указывающие на дубликаты строк.
50. **df.drop\_duplicates():** Удаляет дубликаты строк из DataFrame.
51. **df.isin(values):** Создает маску, указывающую на вхождение значений из values в DataFrame.
52. **df.where(condition, other):** Возвращает DataFrame, заменяя значения, которые не удовлетворяют условию, на other.
53. **df.droplevel(level):** Удаляет один или несколько уровней иерархического индекса.
54. **df.unstack(level):** Поворачивает иерархический индекс, преобразуя уровни в столбцы.
55. **df.stack():** Обратная операция unstack, преобразует столбцы в иерархический индекс.
56. **df.resample(rule):** Применяет ресемплинг временных данных с указанным правилом (например, 'D' для дней, 'M' для месяцев).
57. **df.diff(periods):** Вычисляет разницу между текущим и предыдущим значением в столбце.
58. **df.rolling(window):** Выполняет скользящее окно для вычисления статистик на скользящих интервалах времени.
59. **df.nunique():** Возвращает количество уникальных значений в каждом столбце.
60. **df.value\_counts():** Подсчитывает количество вхождений каждого уникального значения в столбце.
61. **df.to\_string():** Преобразует DataFrame в строку, удобную для отображения.
62. **df.to\_markdown():** Генерирует таблицу Markdown из DataFrame.
63. **df.style:** Позволяет настраивать стили отображения DataFrame, включая форматирование, цветовую подсветку и многое другое.
64. **df.to\_clipboard():** Копирует данные DataFrame в буфер обмена.
65. **df.to\_html():** Преобразует DataFrame в HTML-таблицу.

**Задания**

**Задание 1. Создать Series s1 и s2 по картинке**



**Задание 2. Создать DataFrame** **df1 по картинке**



**Задание 3. Загрузить ваш DataSet**

**Задание 4. Показать первые 10 записей вашего DataFrame**

**Задание 5. Показать последние 10 записей вашего DataFrame**

**Задание 6. Показать последние 10 записей вашего DataFrame**

**Задание 7. Показать случайные 10 строк из вашего DataFrame**

**Задание 8. Показать типы данных для каждого столбца в DataFrame**

**Задание 9. Показать информацию о DataFrame, включая количество непропущенных значений, типы данных**

**Задание 10. Показать статистический анализ числовых столбцов, включая среднее, стандартное отклонение, минимум и максимум о DataFrame**

**Задание 11. Показать Возвращает DataFrame с булевыми значениями, указывающими на пропущенные значения**

**Задание 12. Заменить пропущенные значения значением value в DataFrame**

**Задание 13. Заменить пропущенные значения значением среднего в DataFrame**

**Задание 14. Заменить пропущенные значения значением медианны в DataFrame**

**Задание 15. Заменить пропущенные значения значением максимальным в DataFrame**

**Задание 16. Заменить пропущенные значения значением минимальным в DataFrame**

**Задание 17. Показать количество уникальных значений в каждом столбце в DataFrame**

**Задание 18. Подсчитать количество вхождений каждого уникального значения в столбце в DataFrame**

**Задание 19. Удалить строки с пропущенными значениями в DataFrame**

**Задание 20. Показать булевы значения, указывающие на дубликаты строк в DataFrame**

**Задание 21. Удалить дубликаты строк из DataFrame**

**Задание 22. Заменить определенное значения в DataFrame на указанные значения**

**Задание 23. Сохранить DataFrame в формат Excel с именем** Practice2\_dataset.xlsx

**Задание 24. Сохранить DataFrame в формат CSV с именем** Practice2\_ dataset.CSV

**Задание 25. Сохранить DataFrame в формат JSON с именем** Practice2\_ dataset.json