Курс «Базовая обработка данных на языке Python»

автор: Киреев В.С., к.т.н., доцент

Лабораторная работа № 2

Тема: «Работа с модулем Pandas в Python. Группировка и агрегация данных.»

Цель работы: изучить методы работы с библиотекой pandas.

Теоретическая справка

Pandas - это высокоуровневая библиотека Python для анализа данных. Она построена поверх более низкоуровневой библиотеки NumPy (написана на Си), что является большим плюсом в производительности. В экосистеме Python, pandas является наиболее продвинутой и быстроразвивающейся библиотекой для обработки и анализа данных.

DataFrame и Series

Чтобы эффективно работать с pandas, необходимо освоить самые главные структуры данных библиотеки: DataFrame и Series.

Series

Структура/объект Series представляет из себя объект, похожий на одномерный массив (питоновский список, например), но отличительной его чертой является наличие ассоциированных меток, т.н. индексов, вдоль каждого элемента из списка. Такая особенность превращает его в ассоциативный массив или словарь в Python.

```
import pandas as pd

# Создание объекта Series
my_series = pd.Series([5, 6, 7, 8, 9, 10])
```

В строковом представлении объекта Series, индекс находится слева, а сам элемент справа. Если индекс явно не задан, то pandas автоматически создаёт RangeIndex от 0 до N-1, где N общее количество элементов.

```
# Получение индекса и значений
my_series.index
my_series.values
```

Доступ к элементам объекта Series возможны по их индексу.

```
# Получение элемента по индексу
```

```
my_series[4]
```

Индексы можно задавать явно.

```
# Создание объекта Series с явно заданными индексами
my_series2 = pd.Series([5, 6, 7, 8, 9, 10], index=['a', 'b', 'c', 'd', 'e', 'f'])
```

Можно делать выборку по нескольким индексам и осуществлять групповое присваивание.

```
# Выборка по нескольким индексам
my_series2[['a', 'b', 'f']]

# Групповое присваивание
my_series2[['a', 'b', 'f']] = 0
```

Фильтровать Series как душе заблагорассудится, а также применять математические операции и многое другое.

```
# Фильтрация и математические операции
my_series2[my_series2 > 0]
my_series2[my_series2 > 0] * 2
```

Если Series напоминает нам словарь, где ключом является индекс, а значением сам элемент, то можно сделать так:

```
# Создание объекта Series из словаря
my_series3 = pd.Series({'a': 5, 'b': 6, 'c': 7, 'd': 8})
```

DataFrame

DataFrame - это двумерная структура данных, которую можно представить как таблицу с индексированными строками и столбцами. DataFrame можно создать из различных типов данных: словаря, двумерного массива, другого DataFrame и т.д.

```
# Создание DataFrame из словаря
df = pd.DataFrame({
    'A': pd.Series([1, 2, 3], index=['a', 'b', 'c']),
    'B': pd.Series([4, 5, 6, 7], index=['a', 'b', 'c', 'd'])
})
```

В DataFrame можно обращаться к столбцам как к атрибутам объекта или как к ключам словаря.

```
# Обращение к столбцу
df.A
df['A']
```

DataFrame поддерживает различные типы индексации.

```
# Индексация по строкам
df.loc['a']
```

```
# Индексация по позиции
df.iloc[0]
```

В DataFrame можно добавлять новые столбцы.

```
# Добавление нового столбца
df['C'] = df['A'] + df['B']
```

В DataFrame можно удалять строки и столбцы.

```
# Удаление столбца
df = df.drop('C', axis=1)
```

В DataFrame можно применять функции к каждому столбцу или строке.

```
# Применение функции к каждому столбцу
df.apply(np.sum)
```

В DataFrame можно сортировать данные.

```
# Сортировка по столбцу
df.sort_values(by='A')
```

В DataFrame можно группировать данные.

```
# Группировка данных по столбцу
df.groupby('A').sum()
```

Все эти возможности делают pandas мощным инструментом для анализа данных в Python.

SQLite - это библиотека C, предоставляющая облегченную дисковую базу данных, которая не требует отдельного серверного процесса и позволяет получать доступ к базе данных с использованием нестандартного варианта языка запросов SQL. Некоторые приложения могут использовать SQLite для внутреннего хранения данных. Также возможно создать прототип приложения с использованием SQLite, а затем перенести код в более крупную базу данных, такую как PostgreSQL или Oracle.

Для создания подключения к базе в Python необходимо использовать команды:

```
import sqlite3
con = sqlite3.connect("путь к базе на диске")
```

В сочетании с pandas можно обращаться к данным таких баз с помощью команд вида:

```
df=pd.read_sql("SELECT * FROM TABLE_NAME;", con=con)
```

Если в датафрейме содержится столбец с временной меткой (datetime), то данный датафрейм можно проиндексировать по столбцу со временной меткой и изменить гранулярность (периодичность) столбца с количественными данными, командов вида:

```
df.set_index('orderdate')['revenuedaily'].resample('1W').sum()
```

Самостоятельное задание

1. Сделать единую бд sqlite3 из 3х представленных файлов. Выгрузить полученную бд в датафрейм. Можно использовать метод **PD.READ_SQL.**
2. Создать столбец exp - содержащий среднее значение лет опыта. Создать столбец sal - содержащий среднее значение лет зп в рублях
3. Построить ящичковые диаграммы по зп в разрезе региона. 6. Построить ящичковые диаграммы по зп в разрезе навыков.
4. Проверить наличие зависимости зп от опыта. Проверить наличие зависимости занятости от региона. Здесь и далее, можно использовать библиотеку **SCIPY** и её модуль **STATS**, для проведения статистических тестов – Т-тестов (Стьюдента), теста Хи-квадрат (Пирсона) и т.п.
5. Выделить из названий вакансий две группы - Seniors (ведущий, старший) и Others. Можно использовать метод **DF.MAP** и строковые функции питона, такие как **REPLACE.**
6. Проверить наличие зависимости зп от групп.
7. С помощью **PANDAS** создайте датафрейм из 1 млн. строк, содержащий 4 столбца – пол, возраст, доход, профессия. В категориальной переменной профессия сделайте 3-4 категории, в переменной пол – 2, возраст и доход сделайте количественными, с типом INT. Для генерации этих столбцов используйте методы **NP.RANDOM.NORMAL, NP.RANDOM.CHOICE.**
8. Замерьте объем памяти датафрейма в памяти с помощью методов **DF.INFO()**, **DF.MEMORY_USAGE()**
9. Сохраните датафрейм в формате CSV, загрузите его в новый датафрейм и замерьте скорость записи и чтения, с помощью «магической» команды %%**TIMEIT**.
10. В качестве агрегации надо рассчитать средний возраст, максимальный доход, самый частый пол, самый редкий тип профессии. Замерьте время выполнения этих операций с помощью «магической» команды %%**TIMEIT**.