AIG150- Week 5

Working With Functions & Grouping

Reading Text: Ch 05,08 Pandas for everyone

Agenda

- Apply Functions
  - Apply over a series and DataFrame
  - Column-wise operations
  - Row-wise operations
  - Vectorized functions
  - Lambda functions
- GroupBy operations
  - Understand Split–Apply–Combine concept
  - Use .groupby() for aggregation, transformation, and filtering
  - Write custom and multiple aggregation functions
  - Work with MultiIndex and flatten results using .reset_index()

Apply Functions

- apply() method takes a function and applies it across each row or column of a DataFrame
- Easy to group and reuse Python code
- First definite a function and then call it using

  apply() See the sample code

- Function factory: apply() takes a reference to the function

```
def my_sq(x):
        return x ** 2
def avg_2(x, y):
        return (x + y) / 2

def my_exp(x, e):
        return x ** e
df['col_a_cube'] = df['col_a'].apply(my_exp, e=3)
```

Vectorized Functions

- Vectorize with NumPy: Pass *np.vectorize()* to the function
- Vectorize with Numpy
- Lambda functions

The line `function_vec = np.vectorize(function)` takes the regular Python function `function` and creates a new function `function_vec` that can operate on entire NumPy arrays element by element.

Without `np.vectorize`, you would typically need to use a loop to apply `avg_2_mod` to each pair of elements from two arrays.

`np.vectorize` provides a convenient way to apply a function designed for single elements to arrays.

Panda_Apply_Example.ipynb

Groupby Operations

- To aggregate, transform and filter data

- Split-apply-combine
  1. Data is split into separate parts based on key(s)
  2. A function is applied to each part of the data
  3. The results from each part are combined to create a new data set

Groupby() Method

- Aggregation can be done by using conditional subsetting on a dataframe
- Transformation can be done by passing a column into

  a separate function Filtering can be done with

  conditional subsetting

- Built-in aggregation methods
- User defined aggregation methods

Grouping

Split	Divide data into groups based on a key column	Split by `continent`
Apply	Apply a function to each group	`mean()`, `std()`, custom func
Combine	Combine results into a single DataFrame	Summarized results

GroupBy_SplitApplyCombine.ipynb

Use Dict to perform multiple aggregations

```
# use a dictionary on a dataframe to agg different
columns # for each year,
calculate the # average lifeExp, median pop, and median
gdpPercap
gdf_dict = df.groupby("year").agg( { "lifeExp": "mean",
"pop": "median", "gdpPercap": "median" } )
print(gdf_dict)
```

Check the book to see how to specify dict on a Data Series

Transform

- One-to one transformation
- Check the following:
  - Z score example
  - Missing value example fill the missing values with the mean

Filter & Grouping

- Split the data by keys, and then perform some kind of Boolean subsetting on the data
- The *.aggregate()*, *.transform()*, and *.filter()* methods are commonly used ways of working with grouped objects in Pandas
- Grouped Objects
- Grouping on multiple columns

Summary of Apply

Function	A reusable block of code defined using def.
.apply()	Runs a function across rows or columns of a Series or DataFrame.
Axis Parameter	axis=0 → apply column-wise; axis=1 → apply row-wise.
Lambda Function	Anonymous inline function: lambda x: x*2
Vectorization	Speeds up operations using NumPy or Numba to apply functions element-wise without explicit loops.

Summary of Grouping methods

`.agg()`	Aggregate (reduces data size)
`.transform()`	Transform (keeps same size)
`.filter()`	Keep or remove groups based on condition
`.reset_index()`	Flatten MultiIndex results
`.get_group()`	Extract a specific group