Analyzing Tabular Data with Pandas

Class 5

18/2/2025

Acknowledgement

The series of the IT & Japanese language course is Supported by AOTS and OEC.

Ministry of Economy, Trade and Industry

Overseas Employment Corporation

What you have Learnt Last Week

We were focused on following points.

- Lower, upper, length (len), random and split function
- Random module
- Usage of control and loop flow statement
- Usage of lambda function
- How to define and use functions
- Local and Global Variables
- Understanding of 1D, and 2D NumPy arrays
- Array indexing and slicing

What you will Learn Today

We will focus on following points.

- Performing Linear Algebra in Numpy
- Concatenation, and Stacking in NumPy
- Import and export data effortlessly between different file formats.
- Inspecting and Understanding Data
- Basics of creating, loading, and exploring DataFrames,
- Upload code on Github
- Quiz
- Q&A Session

Linear Algebra in NumPy

Vector Operation

[Dot Product (np.dot())]

```
import numpy as np

a = np.array([1, 2, 3])
b = np.array([4, 5, 6])

dot_product = np.dot(a, b)
print(dot_product)  # (1*4 + 2*5 + 3*6) = 32
```

[Cross Product (np.cross())]

```
a = np.array([1, 0, 0])
b = np.array([0, 1, 0])

cross_product = np.cross(a, b)
print(cross_product)  # [0, 0, 1]
```

NumPy’s linalg module

It provides functions for linear algebra operations like matrix multiplication, solving linear equations, eigenvalues

[Norm of a Vector (np.linalg.norm())]

```
# The norm of a vector is its magnitude (length).
a = np.array([3, 4])

norm = np.linalg.norm(a)
print(norm)  # sqrt(3² + 4²) = 5.0
```

NumPy’s linalg module

Various Matrix Operations

[Matrix Multiplication]

```
A = np.array([[1, 2],
              [3, 4]])
B = np.array([[5, 6],
              [7, 8]])

# Using @ operator
result1 = A @ B
# Using np.matmul()
result2 = np.matmul(A, B)
print(result1)
print(result2)
```

[Transpose of a Matrix]

```
A = np.array([[1, 2, 3],
              [4, 5, 6]])
# Swaps rows and columns
print(A.T)
```

[Determinant of a Matrix]

```
A = np.array([[1, 2],
              [3, 4]])

det = np.linalg.det(A)
print(det)  # -2.0
```

NumPy’s linalg module

Various Matrix Operations

[Inverse of a Matrix]

```
A = np.array([[1, 2],
              [3, 4]])

inverse = np.linalg.inv(A)
print(inverse)
```

[Eigenvalues & Eigenvectors]

```
A = np.array([[2, -1],
              [-1, 2]])

eigenvalues, eigenvectors = np.linalg.eig(A)
print("Eigenvalues:", eigenvalues)
print("Eigenvectors😐”:¥n", eigenvectors)
```

[Solving Linear Equations]

```
A = np.array([[2, 1],
              [1, -1]])
B = np.array([4, 1])

x = np.linalg.solve(A, B)
print(x)  # Solution to Ax = B
```

Concatenation and Stacking

Concatenation of Arrays

[np.concatenate()]

```
a = np.array([1, 2, 3])
b = np.array([4, 5, 6])

concat = np.concatenate((a, b))
print(concat)  # [1 2 3 4 5 6]
```

[Concatenating along Different Axes (axis=0, axis=1)]

```
A = np.array([[1, 2],
              [3, 4]])
B = np.array([[5, 6],
              [7, 8]])

concat_axis0 = np.concatenate((A, B), axis=0)  # Row-wise
concat_axis1 = np.concatenate((A, B), axis=1)  # Column-wise

print(concat_axis0)
print(concat_axis1)
```

Concatenation and Stacking

Stacking Arrays

[Vertical Stacking (np.vstack())]

```
A = np.array([1, 2, 3])
B = np.array([4, 5, 6])

vstack = np.vstack((A, B))
print(vstack)
```

[Horizontal Stacking (np.hstack()]

```
A = np.array([[1], [2], [3]])
B = np.array([[4], [5], [6]])

hstack = np.hstack((A, B))
print(hstack)
```

[Depth Stacking (np.dstack()]

```
A = np.array([[1, 2], [3, 4]])
B = np.array([[5, 6], [7, 8]])

dstack = np.dstack((A, B))
print(dstack)
```

Concatenation and Stacking

Splitting Arrays

[np.split()]

```
arr = np.array([1, 2, 3, 4, 5, 6])
split = np.split(arr, 3)

print(split)
# [array([1, 2]),
    array([3, 4]),
    array([5, 6])]
```

[np.vsplit()]

```
A = np.array([[1, 2],
              [3, 4],
              [5, 6]])

vsplit = np.vsplit(A, 3)
print(vsplit)
```

[np.hsplit()]

```
A = np.array([[1, 2, 3],
              [4, 5, 6]])

hsplit = np.hsplit(A, 3)
print(hsplit)
```

Concatenation and Stacking

Tiling and Repeating

[np.tile()]

#Repeats an array in a structured way

```
A = np.array([1, 2, 3])

tile = np.tile(A, (2, 3))
print(tile)
```

[np.repeat()]

#Repeats individual elements

```
A = np.array([1, 2, 3])

repeat = np.repeat(A, 3)
print(repeat)  # [1 1 1 2 2 2 3 3 3]
```

Understanding Different File Formats

Overview of File Formats

- **CSV (Comma-Separated Values)**: Text-based, stores tabular data.
- **Excel (XLSX)**: Spreadsheet format, supports multiple sheets.
- **JSON (JavaScript Object Notation)**: Lightweight format for structured data.
- **Parquet**: Columnar storage, optimized for big data.
- **SQL Databases**: Structured storage, used for relational data.

Understanding Different File Formats

When to Use Which Format?

- **CSV:** When you need a simple, widely supported text format.
- **Excel:** When working with formatted spreadsheets and multiple sheets.
- **JSON:** When exchanging data between applications (e.g., APIs).
- **Parquet:** When working with large datasets and performance matters.
- **SQL:** When dealing with structured relational data.

Importing Data

Importing various files

[Importing CSV Files]

```
import pandas as pd
# Basic CSV import
df = pd.read_csv("data.csv")
# Handling delimiters and encoding
df = pd.read_csv
      ("data.csv", delimiter=';', encoding='utf-8')
```

[Importing Excel Files]

```
# Read single sheet
df = pd.read_excel("data.xlsx", sheet_name="Sheet1")
# Read multiple sheets
dfs = pd.read_excel("data.xlsx", sheet_name=None)
# Returns a dictionary
```

Importing Data

Importing various files

[Importing JSON Files]

```
Import json
df = pd.read_json("data.json")
```

[Importing Data from SQL Databases]

```
import sqlite3
import pandas as pd

conn = sqlite3.connect("Car_Database.db")
df = pd.read_sql("SELECT * FROM Car_Parts", conn)
print(df)
df.to_csv("output1.csv", index=False)
```

Exporting Data

Exporting various files

- **Exporting to CSV**
- `df.to_csv("output.csv", index=False, compression='gzip')`
- **Exporting to Excel**
- `df.to_excel("output.xlsx", sheet_name="Sheet1", index=False)`
- **Exporting to JSON**
- `df.to_json("output.json", orient="records", indent=4)`
- **Exporting to Databases**
- `df.to_sql("table_name", conn, if_exists="replace", index=False)`

Handling Large Datasets

Chunk-wise Reading and Optimizing Memory Usage

[Chunk-wise Reading]

```
import pandas as pd

chunksize = 1000

for chunk in pd.read_csv("output.csv", chunksize=chunksize):

# Replace with your processing function

  print(chunk.head())
```

[Optimizing Memory Usage]

```
# Convert data types to reduce memory usage

df['First name'] = df['First name'].astype('category')
```

Automating Data Import/Export

Using Python Libraries for Automation

[Example]

```
import os, glob, shutil

# Move all CSV files to a backup folder
csv_files = glob.glob("*.csv")
for file in csv_files:
    shutil.move(file, "backup_folder/")
```

[Example]

```
def export_data(df, filename, format):
    if format == "csv":
        df.to_csv(filename, index=False)
    elif format == "json":
        df.to_json(filename, orient="records")
    print(f"Exported {filename} successfully!")
```

Make sure you have a folder named backup_folder and a pandas DataFrame to work with.

Overview of Data Inspection Methods

Importance of Inspecting Data Before Analysis

- Ensures data quality and consistency
- Identifies missing values, duplicates, and outliers
- Helps understand the dataset structure before analysis

[Example]

```
import pandas as pd

# Sample DataFrame
data = {'Name': ['Alice', 'Bob', 'Charlie',
None], 'Age': [25, 30, 35, None], 'Salary':
[50000, 60000, None, 70000]}

df = pd.DataFrame(data)

# Preview dataset
print(df.head())
```

Understanding Data Structure

Checking dataset structure and data type

[Checking dataset structure]

- print(df.info())

[Checking data type]

- print(df.dtypes)

```
import pandas as pd

# Sample DataFrame
data = {
   'Name': ['Alice', 'Bob', 'Charlie'],
   'Age': [25, 30, 35],
   'Salary': [50000, 60000, 70000]
}

# Create DataFrame
df = pd.DataFrame(data)

# Checking dataset structure
print("Dataset Info:")
print(df.info())

# Checking dataset data types
print("\nDataset Data Types:")
print(df.dtypes)
```

Viewing Data

Displaying first, last rows and random sample

[Displaying first and last rows]

- print(df.head()) # First 5 rows
- print(df.tail(2)) # Last 2 rows

[Displaying a random sample]

- print(df.sample(2))

```
import pandas as pd

# Sample DataFrame
data = {
   'Name': ['Alice', 'Bob', 'Charlie', 'David', 'Eva'],
   'Age': [25, 30, 35, 40, 45],
   'Salary': [50000, 60000, 70000, 80000, 90000]
}

# Create DataFrame
df = pd.DataFrame(data)

# Displaying first 5 rows
print("First 5 rows:")
print(df.head())  # Default is first 5 rows

# Displaying last 2 rows
print("\nLast 2 rows:")
print(df.tail(2))  # Last 2 rows

# Displaying a random sample of 2 rows
print("\nRandom Sample (2 rows):")
print(df.sample(2))  # Random sample of 2 rows
```

Summary Statistics

numerical insights and find missing values

[numerical insights]

- print(df.describe())

[Checking unique values]

- print(df.nunique())

[Finding missing values]

- print(df.isnull().sum())

```
import pandas as pd

# Sample DataFrame
data = {
   'Name': ['Alice', 'Bob', 'Charlie', 'David', 'Eva'],
   'Age': [25, 30, 35, 40, 45],
   'Salary': [50000, 60000, 70000, 80000, None]  #
Notice None as a missing value
}

# Create DataFrame
df = pd.DataFrame(data)

# Numerical insights (descriptive statistics)
print("Numerical Insights:")
print(df.describe())

# Checking unique values for each column
print("\nUnique Values in Each Column:")
print(df.nunique())

# Finding missing values in each column
print("\nMissing Values Count in Each Column:")
print(df.isnull().sum())
```

Identifying Data Issues

Detecting missing values and duplicate rows

[Detecting missing values]

- print(df.isnull().sum())

[Detecting duplicate rows]

- print(df.duplicated().sum())

[Checking outliers]

```
import matplotlib.pyplot as plt

df[['Age', 'Salary']].boxplot()
plt.show()
```

```
import pandas as pd
import matplotlib.pyplot as plt

# Sample DataFrame
data = {
   'Name': ['Alice', 'Bob', 'Charlie', 'David', 'Eva', 'Alice'],
   'Age': [25, 30, 35, 40, 45, 25],  # Duplicate age for
Alice
   'Salary': [50000, 60000, 70000, 80000, None,
50000]  # Missing value for Salary
}

# Create DataFrame
df = pd.DataFrame(data)

# Detecting missing values
print("Missing Values Count in Each Column:")
print(df.isnull().sum())

# Detecting duplicate rows
print("\nDuplicate Rows Count:")
print(df.duplicated().sum())

# Checking for outliers using a boxplot
print("\nBoxplot for 'Age' and 'Salary':")
df[['Age', 'Salary']].boxplot()
plt.show()
```

Data Cleaning Basics

Dropping or filling missing values and removing duplicates

[Dropping or filling missing]

```
# Remove missing values
df_cleaned = df.dropna()
# Fill missing values with mean
df_filled = df.fillna(df.mean())
```

[Removing duplicates]

- df_unique = df.drop_duplicates()

[Changing data types]

- df['Age'] = df['Age'].astype('Int64')

```
import pandas as pd

# Sample DataFrame with missing values and duplicates
data = {
   'Name': ['Alice', 'Bob', 'Charlie', 'David', 'Eva', 'Alice'],
   'Age': [25, 30, 35, 40, 45, None],  # Notice missing value for Age
   'Salary': [50000, 60000, 70000, 80000, None, 50000]  # Notice missing value
for Salary
}

# Create DataFrame
df = pd.DataFrame(data)

# Display original DataFrame
print("Original DataFrame:")
print(df)

# Remove missing values (drop rows with any NaN values)
df_cleaned = df.dropna()
print("\nDataFrame After Dropping Missing Values:")
print(df_cleaned)

# Fill missing values with the mean of the column (only for numeric columns)
df_filled = df.copy()
df_filled['Salary'] = df_filled['Salary'].fillna(df_filled['Salary'].mean())
df_filled['Age'] = df_filled['Age'].fillna(df_filled['Age'].mean())
print("\nDataFrame After Filling Missing Values with Mean:")
print(df_filled)

# Remove duplicate rows (based on all columns)
df_unique = df.drop_duplicates()
print("\nDataFrame After Removing Duplicates:")
print(df_unique)

# Change the data type of 'Age' column to Int64 (nullable integer type)
df['Age'] = df['Age'].astype('Int64')  # Nullable integer type
print("\nDataFrame After Changing Data Type of 'Age' to Int64:")
print(df)
```

Data Relationships & Correlations

Checking correlations, Grouping and summarizing data

[Checking correlations]

- print(df.corr())

[Grouping and summarizing data]

- print(df.groupby('Name').mean())

```
import pandas as pd

# Sample DataFrame
data = {
   'Name': ['Alice', 'Bob', 'Charlie', 'David', 'Eva', 'Alice'],
   'Age': [25, 30, 35, 40, 45, 25],  # Alice appears twice
   'Salary': [50000, 60000, 70000, 80000, 75000, 50000]
}

# Create DataFrame
df = pd.DataFrame(data)

# Checking Correlations between numeric columns
(excluding 'Name')
print("Correlation Matrix:")
print(df[['Age', 'Salary']].corr())  # Only pass numeric
columns for correlation calculation

# Grouping by 'Name' and calculating the mean of other
columns
print("\nGrouped Data (Mean of Age and Salary by
Name):")
print(df.groupby('Name', as_index=False).mean())
```

Quiz Section

Quiz

Everyone student should click on submit button before time ends otherwise MCQs will not be submitted

[Guidelines of MCQs]

1. There are 20 MCQs
2. Time duration will be 10 minutes
3. This link will be share on 6:10pm (Pakistan time)
4. MCQs will start from 6:15pm (Pakistan time)
5. This is exact time and this will not change
6. Everyone student should click on submit button otherwise MCQs will not be submitted after time will finish
7. Every student should submit Github profile and LinkedIn post link for every class. It include in your performance

Assignment

Assignment should be submit before the next class

[Assignments Requirements]

1. Create a post of today's lecture and post on LinkedIn.
2. Make sure to tag @Plus W @Pak-Japan Centre and instructors LinkedIn profile
3. Upload your code of assignment and lecture on GitHub and share your GitHub profile in respective your region group WhatsApp group
4. If you have any query regarding assignment, please share on your region WhatsApp group.
5. Students who already done assignment, please support other students

Q&A Session

ありがとうございます。
Thank you.
شکریا

For the World with Diverse Individualities