Machine Learning

Pandas Application & Regression Preview

Dept. SW and Communication Engineering

Prof. Giseop Noh (kafa46@hongik.ac.kr)

Lecture Goals

- **Pandas Application**
  - Grouping, Combining
  - Pivoting, Reshaping
  - Custom Functions
- **Working with Database using Pandas**
  - SQLite
  - Create/Load/Handing DB
- **Regression Preview**
  - Using Numpy
  - Using Scikit-learn

Pandas Applications

Recap Pandas Basics

- **One of the most powerful and widely used Python libraries for data analysis.**
- **Beyond basic tasks like loading CSV files or selecting rows and columns.**
- **Offers advanced data manipulation tools**
  - Cleaning and transforming large datasets
  - Structuring data for machine learning
  - Handling real-world data challenges efficiently

Grouping

Grouping

- **Grouping Data and Computing Summary Statistics**

- **In many analysis tasks, we group data by categories**
  - E.g., genre, country, customer

- **Pandas provides the groupby() method to split data into groups.**

- **Combine with agg() or simple functions (mean, sum, std) to compute:**
  - Examples
    - Average sales by country
    - Total revenue by artist
    - Standard deviation of ratings by genre

DataFrame.groupby

DataFrame.groupby( )

- Simply groups data based on one or more keys
- By itself, it does **not perform any calculations.** Returns a **"grouped" object**

  <pandas.core.groupby.generic.DataFrameGroupBy object at 0x...>
- Using groupby() Alone
  - Iterating Through Groups (English Explanation)

```
for name, group in df.groupby("Genre"):
    print("Group:", name)
    print(group.head(2))
```

```
team: A
  team  score  hours
0    A     85      5
1    A     90      6
team: B
  team  score  hours
2    B     78      4
3    B     88      5
team: C
  team  score  hours
4    C     95      7
5    C     92      6
```

`groupby` with Multiple Columns

- `groupby(["col1", "col2", ...])`
  - Groups data based on combinations of multiple columns
  - Each unique combination of values forms one group.
  - By default, the result uses a MultiIndex
    - But you can set as_index=False to return a flat DataFrame.

```
import pandas as pd
data = {
    "Country": ["USA", "USA", "Canada", "USA", "Canada", "Canada"],
    "Genre":   ["Rock","Pop", "Rock", "Rock","Jazz",  "Rock"],
    "Year":    [2023, 2023,   2023, 2024,   2023,     2024],
    "Revenue": [10,   15,     8,    12,     7,       6]
}
df = pd.DataFrame(data)
out = df.groupby(["Country", "Genre"])["Revenue"].sum()
print(out)
print("---" * 10)
out_flat = df.groupby(
         ["Country", "Genre"],
         as_index=False)["Revenue"].sum()
print(out_flat)
```

```
Country  Genre
Canada   Jazz      7
         Rock     14
USA      Pop      15
         Rock     22
Name: Revenue, dtype:
int64
--------------------------
  Country Genre  Revenue
0  Canada  Jazz        7
1  Canada  Rock       14
2     USA   Pop       15
3     USA  Rock       22
```

DataFrame.agg (1/2)

- **agg() (aggregate) function**
  - Summarize data by applying one or more statistical functions to each group.

- **Basic Usage**

```
df.groupby("Genre")["Revenue"].agg(["sum", "mean", "std"])
```

- **Different Functions for Different Columns**

```
df.groupby("Genre").agg({
    "Revenue": ["sum", "mean"],
    "Price": "max"
})
```

DataFrame.agg (2/2)

Without groupby( )

- Use agg() directly on a DataFrame

```
df.agg(["sum", "mean", "std"])
```

  - Computes the statistics across all numeric columns.

Custom Aggregation

- Define custom functions using lambda

```
df.agg({
    "Sales": lambda x: x.max() - x.min(),
    "Profit": "mean"
})
```

Exercise: groupby + agg

```
import pandas as pd

# Create a DataFrame with teams and performance data
data = {
    "team": ["A","A","B","B","C","C","A","B","C","A"],
    "score": [85,90,78,88,95,92,89,84,91,87],
    "hours": [5,6,4,5,7,6,8,3,4,7]
}
df = pd.DataFrame(data)

# Group data by team and calculate mean and standard deviation
result = df.groupby("team").agg({"score":"mean", "hours":"std"})
print(result)
```

```
        score     hours
team
A     87.750000  1.290994
B     83.333333  1.000000
C     92.666667  1.527525
```

Combining

Combining DataFrames

- **Real-world datasets often come from multiple sources.**

- **Pandas offers several ways to combine DataFrames:**
  - merge: SQL-style join on keys
  - concat: Concatenate DataFrames along rows or columns
  - join: Merge DataFrames by index

- **Combining Exercise**
  - Source Codes: https://www.deepshark.org/courses/machine_learning_1/w/06_pandas_apps_and_regression_preview#**grouping_exercise**

Pivoiting

DataFrame.pivot_table

- `pivot_table` reshapes and **summarizes** data, similar to Excel Pivot Tables.
  - It turns long/transactional data into a compact matrix for quick comparison in dataset

- Core Syntax

```
df.pivot_table(
    values=None,
    index=None,
    columns=None,
    aggfunc="mean",
    fill_value=None,
    margins=False,
    margins_name="All",
    dropna=True,
    observed=False
)
```

Parameters (Signatures)

values: Column(s) to aggregate (e.g., "Revenue"). If None, all numeric columns may be aggregated.

index: Row groups (one or more columns).

columns: Column groups (one or more columns).

aggfunc: Aggregation function(s), e.g., "sum", "mean", "count", np.sum, or a list/dict for multiple metrics.

fill_value: Replace NaN in the result with a value (commonly 0).

margins: Add row/column totals (True/False).

margins_name: Label for totals (default "All").

dropna: Drop columns in the result that are all-NaN.

observed: For categorical groupers, include only observed combinations (perf/size optimization).

pivot_table Exercise

- **Source Codes for pivot_table**
  - https://www.deepshark.org/courses/machine_learning_1/w/06_pandas_apps_and_regression_preview#pivot_exercise

Working with DB

Why use Databases with Pandas?

- **Why use Databases with Pandas?**
  - Beyond CSV or Excel, Pandas integrates seamlessly with SQL databases like SQLite.
  - Ideal for structured datasets that fit well into tables with relationships.

- **SQL?**
  - SQL is a standard language used to manage and query relational databases.
  - It allows to store, retrieve, and manipulate structured data efficiently.

- **Key Advantages**
  - Store, query, and manipulate large datasets efficiently.
  - Use SQL queries directly with Pandas DataFrames.
  - Combine data analytics (Pandas) with data persistence (SQLite).

Key SQL Operations

Operation	Keyword	Description
Create	CREATE TABLE	Define a new table
Insert	INSERT INTO	Add new records
Select	SELECT ... FROM	Retrieve data from tabl es
Update	UPDATE ... SET	Modify existing records
Delete	DELETE FROM	Remove records
Join	JOIN	Combine data from m ultiple tables

SQLite

■ Pandas integrates seamlessly with SQLite through the `sqlite3` module

- Lightweight, serverless relational database engine.
- Not require a separate server process.
- Stores the entire database in a single file (e.g., `sample.db`).
- Convenient for learning, prototyping, and handling
- Use familiar SQL commands such as

  `SELECT`, `INSERT`, and `UPDATE`.

■ Capability of SQLIte

- Maximum DB size: 281 TB
- Columns in a Table: 2,000
- Rows in a Table: $2^{64} \approx 1.8 \times 10^{19}$ (약 18경 개)

Note for SQLite

- **Pandas can also connect to other relational databases such as MySQL and PostgreSQL.**
  - Instead of sqlite3, you would typically use connectors such as pymysql (for MySQL) or psycopg2 (for PostgreSQL).
  - Combined with SQLAlchemy, Pandas can use the same read_sql() and to_sql() methods to read from or write to those databases as well.
  - Pandas a very versatile tool for integrating with various database systems.

Typical Workflows

- **The workflow**
  - Use Pandas to create a DataFrame with 10 rows of student data (ID, name, score).
  - Open a connection to sample.db using the sqlite3 module.
  - Save the DataFrame into the database as a table named students using Pandas' to_sql() method.
  - If the table already exists, we can choose to replace or append data.
  - Close the connection to finalize the operation.

Pandas + DB Excercise

- **Source Codes**
  - https://www.deepshark.org/courses/machine_learning_1/w/06_pandas_apps_and_regression_preview#sqlite_exercise

Actual Practice using Real Data

- **Chinook (음원 스토어 데이터베이스)**
  - A sample relational database designed for learning database management and practicing SQL queries. It models a fictional digital media store and was developed as an alternative to the Northwind database.
  - Useful Links: https://www.deepshark.org/courses/machine_learning_1/w/06_pandas_apps_and_regression_preview#chinook_links
- **Key features**
  - Digital media store model: Contains information about albums, artists, media tracks, customers, employees, and invoices.
  - Based on real data: The media-related data is derived from an actual iTunes library.
  - Multiple formats supported: Available for SQLite, MySQL, SQL Server, PostgreSQL, Oracle, and other database management systems.

- **Codes for Exercise:**
  https://www.deepshark.org/courses/machine_learning_1/w/06_pandas_apps_and_regression_preview#chinook_exercise

Regression Preview

What is Regression?

- **Regression is one of the simplest machine learning algorithms.**

- **Its goal is to predict a continuous output variable (y) from one or more input variables (X).**

- **Common examples include:**

  - Study hours → Exam scores

  - Advertising expenses → Sales revenue

Mathematical form of a simple linear regression

$$y = \beta_0 + \beta_1 x + \epsilon$$

- y : dependent variable (target to predict)
- x : independent variable (input feature)
- β_0 : intercept (constant term)
- β_1 : slope (effect of x on y)
- ϵ : error term

Regression Toy Practice

- **Source Codes for Toy Practice**

  https://www.deepshark.org/courses/machine_learning_1/w/06_pandas_apps_and_regression_preview#regression_exercise

- **Evaluating Prediction Performance**

  - Measure how well the regression line fits the data using Mean Squared Error (MSE).

```
# Predictions from the regression line
pred = model.predict(df[["hours"]])

# Calculate Mean Squared Error
mse = np.mean((df["scores"] - pred)**2)
print("MSE:", mse)
```

Homeworks

- https://www.deepshark.org/courses/machine_learning_1/w/06_pandas_apps_and_regression_preview#homeworks

Thank you!