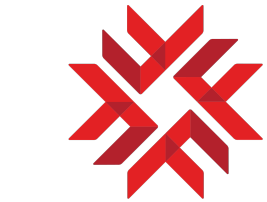
Working with Data Files 3

Course:

INFO-6145 Data Science and Machine Learning



Revised by:
Mohammad Noorchenarboo

September 26, 2024

Contents

1. Data Analysis
   - Why Data Analysis?
   - 4 (or 5) V's of Data
   - Types of Data Analysis
   - Stages of Data Analysis
2. Working with Data File
   - Terminology
   - Pandas
   - Jupyter Notebook
3. Coding
   - Installing Pandas
   - Example: CSV Import and Dataframe Exploration
   - Example: Exploring a Dataframe
   - Dataset Verification
4. Useful DataFrame Methods

Current Section

1. Data Analysis
   - Why Data Analysis?
   - 4 (or 5) V's of Data
   - Types of Data Analysis
   - Stages of Data Analysis
2. Working with Data File
   - Terminology
   - Pandas
   - Jupyter Notebook
3. Coding
   - Installing Pandas
   - Example: CSV Import and Dataframe Exploration
   - Example: Exploring a Dataframe
   - Dataset Verification
4. Useful DataFrame Methods

Why Data Analysis?

Data analysis is essential for:

Validate Assumptions

Definition: Use data to confirm or challenge initial hypotheses or beliefs.
Example: A company analyzes customer purchase data to confirm if discounts lead to increased sales.

Find Root Causes of Issues

Definition: Identify underlying problems by examining data patterns.
Example: A hospital analyzes patient feedback to find that long waiting times are the main reason for dissatisfaction.

Why Data Analysis?

Predict Future Outcomes

Definition: Use historical data to make future predictions.
Example: A retailer uses sales data from the past years to predict demand for the upcoming holiday season.

4 (or 5) V's of Data

Data characteristics:

Volume

Definition: The amount of data. **Example:** Social media platforms process terabytes of data daily.

Velocity

Definition: The speed at which data is generated and processed.
Example: Stock market data is analyzed in real-time to make trading decisions.

Variety

Definition: The different types of data.
Example: Data can come in forms like text, images, or videos.

4 (or 5) V's of Data

Variability

Definition: The inconsistency of data flows.
Example: Web traffic peaks during major news events but remains low during off-hours.

Value

Definition: The usefulness of the data.
Example: Customer behavior data can be highly valuable for targeted marketing.

Types of Data Analysis

Different approaches to analyzing data:

Descriptive Analysis

Definition: Summarizes historical data to describe what has happened.
Example: A company reviews sales figures from the last quarter to understand overall performance.

Diagnostic Analysis

Definition: Explains why something happened by identifying patterns or anomalies.
Example: A website analyzes user data to determine why bounce rates spiked.

Types of Data Analysis

Predictive Analysis

Definition: Forecasts future outcomes using historical data.
Example: An e-commerce platform predicts future sales based on previous holiday shopping patterns.

Prescriptive Analysis

Definition: Suggests actions or decisions based on data predictions.
Example: A delivery service uses data to optimize delivery routes, saving time and costs.

Stages of Data Analysis

- **Data Gathering:** Collecting data from various sources. For example, a marketing team gathers customer survey responses.
- **Data Cleaning:** Removing or correcting errors or inconsistencies in the data. For example, a data scientist fills in missing values in a dataset to ensure accuracy.
- **Data Analysis:** Applying statistical or computational methods to examine the data. For example, a healthcare analyst uses statistical models to study patient outcomes.
- **Data Interpretation:** Making sense of the results and drawing conclusions. For example, a business analyst interprets customer feedback data to make recommendations for product improvements.
- **Data Visualization:** Presenting data findings in charts, graphs, or other visual formats. For example, Sales trends are displayed in a line chart for the management team to review.

Current Section

1. Data Analysis
   - Why Data Analysis?
   - 4 (or 5) V's of Data
   - Types of Data Analysis
   - Stages of Data Analysis
2. Working with Data File
   - Terminology
   - Pandas
   - Jupyter Notebook
3. Coding
   - Installing Pandas
   - Example: CSV Import and Dataframe Exploration
   - Example: Exploring a Dataframe
   - Dataset Verification
4. Useful DataFrame Methods

Terminology

Data point

Definition: A single fact; a discrete unit of information.
Example: The price of a product on a specific day is a data point.

Dataset

Definition: A collection of data.
Example: A file containing multiple rows of customer transactions is a dataset.

Series

Definition: A one-dimensional array.
Example: A list of product prices over time can be represented as a Series in Pandas.

Terminology

Data Frame

Definition: A two-dimensional labeled data structure (columns).
Example: A table of product prices, quantities, and dates can be represented as a DataFrame in Pandas.

Pandas

Pandas is a Python data analysis library. Key Features of Pandas are:

- **Data manipulation:** Suited for manipulating numeric tables and time series.
- **Versatility:** Works well with other libraries for data visualization and analysis.
- **Data Structures:** Intuitive data structures for flexible and powerful data manipulation. For example, DataFrame and Series.
- **Comprehensive Functions:** Pandas provides functions for data cleaning, transformation, and aggregation. For example, use `fillna()` to handle missing data, `groupby()` for aggregation.
- **Integration with Libraries:** Seamless integration with other Python libraries such as Matplotlib, Seaborn, and Scikit-learn. For example, easily create visualizations or apply machine learning models using these libraries.

Jupyter Notebook

The Jupyter Notebook is a web-based interactive computing platform.

Definition

Definition: A document-based environment that can contain both code and documentation along with the output of text and visualizations.

Usage

Platforms: You can use Jupyter Notebooks in tools like Visual Studio Code or web-based environments such as Google Colab.

File Extension

Definition: The typical file extension is `.ipynb`.
Example: A saved Jupyter Notebook file will have the extension `filename.ipynb`.

Current Section

1. Data Analysis
   - Why Data Analysis?
   - 4 (or 5) V's of Data
   - Types of Data Analysis
   - Stages of Data Analysis
2. Working with Data File
   - Terminology
   - Pandas
   - Jupyter Notebook
3. Coding
   - Installing Pandas
   - Example: CSV Import and Dataframe Exploration
   - Example: Exploring a Dataframe
   - Dataset Verification
4. Useful DataFrame Methods

Installing Pandas

Before using Pandas, you may need to install it. For example, if you see a red underline in VS Code:

Command

```
python -m pip install pandas
```

Make sure Python is in your system's PATH.

Typical Naming in a Notebook

In most cases, Pandas is imported as `pd`, and dataframes are commonly named `df`.

Example: Importing CSV Data

Pandas allows importing data from CSV files.

Importing data from a CSV file.

```
import pandas as pd
df = pd.read_csv('/users/dbedf/nba.csv', header=0)
df = pd.read_csv('http://somefakesite.com/somefile.csv',
                 header=0)

df.head(5)
df.describe()
```

Example: Exploring a Dataframe

We will generate simple data and demonstrate how to explore it using Pandas:

Data:

Name	Age	Salary
Alice	30	50000
Bob	25	60000
Charlie	35	70000
David	40	80000
Eve	28	55000

Convert a data frame using Pandas:

```
import pandas as pd

# Generate simple dataset
data = {'Name': ['Alice', 'Bob', 'Charlie', 'David', 'Eve'],
        'Age': [30, 25, 35, 40, 28],
        'Salary': [50000, 60000, 70000, 80000, 55000]}

df = pd.DataFrame(data)
```

Displaying Dataset Information

We will now display basic information about the dataset:

```
# Display basic information about the dataset
print("Dataset Information:")
print(df.info())

# Output:
# <class 'pandas.core.frame.DataFrame'>
# RangeIndex: 5 entries, 0 to 4
# Data columns (total 3 columns):
#  #   Column  Non-Null Count  Dtype
# ---  ------  --------------  -----
#  0   Name    5 non-null      object
#  1   Age     5 non-null      int64
#  2   Salary  5 non-null      int64
# dtypes: int64(2), object(1)
# memory usage: 248.0+ bytes
```

Displaying First Few Rows of the Dataset

We will display the first few rows of the dataset:

```
# Display the first few rows of the dataset
print("\nFirst few rows of the dataset:")
print(df.head())

# Output:
#       Name  Age  Salary
# 0    Alice   30   50000
# 1      Bob   25   60000
# 2  Charlie   35   70000
# 3    David   40   80000
# 4      Eve   28   55000
```

Displaying Summary Statistics

We will display summary statistics for the numerical columns:

```
# Summary statistics of numerical columns
print("\nSummary Statistics:")
print(df.describe())

# Output:
#                Age         Salary
# count    5.000000       5.000000
# mean    31.600000   63000.000000
# std      5.979131   11401.754251
# min     25.000000   50000.000000
# 25%     28.000000   55000.000000
# 50%     30.000000   60000.000000
# 75%     35.000000   70000.000000
# max     40.000000   80000.000000
```

Checking for Missing Values

We will check for missing values in the dataset:

```
# Check for missing values
print("\nMissing Values:")
print(df.isnull().sum())

# Output:
# Name       0
# Age        0
# Salary     0
# dtype: int64
```

Dataset Verification

Steps for verifying a dataset include:

Verification Steps

- Review and understand all columns.
- Ensure no missing values (NaN).
- Ensure correct data types.
- Detect and remove or adjust outliers.
- Remove any useless columns.
- Split data into training/test datasets.

Current Section

1. Data Analysis
   - Why Data Analysis?
   - 4 (or 5) V's of Data
   - Types of Data Analysis
   - Stages of Data Analysis
2. Working with Data File
   - Terminology
   - Pandas
   - Jupyter Notebook
3. Coding
   - Installing Pandas
   - Example: CSV Import and Dataframe Exploration
   - Example: Exploring a Dataframe
   - Dataset Verification
4. Useful DataFrame Methods

Dataset for Example

We will use the following dataset to demonstrate the Pandas functions:

```
import pandas as pd

# Generate a simple dataset
data = {'Name': ['Alice', 'Bob', 'Charlie', 'David', 'Eve'],
        'Age': [30, 25, 35, 40, 28],
        'Salary': [50000, 60000, 70000, 80000, 55000],
        'Department': ['HR', 'Finance', 'IT', 'IT', 'HR'],
        'Years_At_Company': [5, 3, 8, 10, 6]}

df = pd.DataFrame(data)
```

read_csv()

The `read_csv()` function is used to read data from a CSV file:

```
# Reading CSV file (assuming dataset saved as 'data.csv')
df = pd.read_csv('data.csv')
```

head() or tail()

The `head()` and `tail()` functions are used to display the first or last few rows of a DataFrame:

```
# Displaying the first few rows
print(df.head())
# Output:
#       Name  Age  Salary Department  Years_At_Company
# 0    Alice   30   50000         HR                 5
# 1      Bob   25   60000    Finance                 3
# 2  Charlie   35   70000         IT                 8
# 3    David   40   80000         IT                10
# 4      Eve   28   55000         HR                 6
```

describe()

The `describe()` function provides summary statistics for numerical columns:

```
# Summary statistics
print(df.describe())
# Output:
#                Age         Salary  Years_At_Company
# count    5.000000       5.000000          5.000000
# mean    31.600000   63000.000000          6.400000
# std      5.979131   11401.754251          2.701851
# min     25.000000   50000.000000          3.000000
# 25%     28.000000   55000.000000          5.000000
# 50%     30.000000   60000.000000          6.000000
# 75%     35.000000   70000.000000          8.000000
# max     40.000000   80000.000000         10.000000
```

memory_usage()

The `memory_usage()` function shows the memory usage of each column:

```
# Memory usage
print(df.memory_usage())
# Output:
# Index              128
# Name                40
# Age                 40
# Salary              40
# Department          40
# Years_At_Company    40
# dtype: int64
```

info()

The `info()` function provides a concise summary of the DataFrame:

```
# Dataset information
print(df.info())
# Output:
# <class 'pandas.core.frame.DataFrame'>
# RangeIndex: 5 entries, 0 to 4
# Data columns (total 5 columns):
#  #   Column            Non-Null Count  Dtype
# ---  ------            --------------  -----
#  0   Name              5 non-null      object
#  1   Age               5 non-null      int64
#  2   Salary            5 non-null      int64
#  3   Department        5 non-null      object
#  4   Years_At_Company  5 non-null      int64
# dtypes: int64(3), object(2)
# memory usage: 328.0 bytes
```

groupby()

The `groupby()` function groups data by one or more columns:

```
# Group by Department and calculate average salary
grouped = df.groupby('Department')['Salary'].mean()
print(grouped)
# Output:
# Department
# Finance    60000.0
# HR         52500.0
# IT         75000.0
# Name: Salary, dtype: float64
```

merge()

The `merge()` function combines two DataFrames:

```
# Merge two DataFrames
df2 = pd.DataFrame({'Department': ['HR', 'Finance', 'IT'],
'Budget': [100000, 200000, 300000]})
merged = pd.merge(df, df2, on='Department')
print(merged)
# Output:
#       Name  Age  Salary Department  Years_At_Company  Budget
# 0    Alice   30   50000         HR                 5  100000
# 1      Eve   28   55000         HR                 6  100000
# 2      Bob   25   60000    Finance                 3  200000
# 3  Charlie   35   70000         IT                 8  300000
# 4    David   40   80000         IT                10  300000
```

sort_values()

The `sort_values()` function sorts the DataFrame by a specified column:

```
# Sort by Salary
sorted_df = df.sort_values('Salary', ascending=False)
print(sorted_df)
# Output:
#       Name  Age  Salary Department  Years_At_Company
# 3    David   40   80000         IT                10
# 2  Charlie   35   70000         IT                 8
# 1      Bob   25   60000    Finance                 3
# 4      Eve   28   55000         HR                 6
# 0    Alice   30   50000         HR                 5
```

fillna()

The `fillna()` function replaces missing values in a DataFrame:

```
# Replace missing values in Salary with the average
df_with_missing = df.copy()
df_with_missing['Salary'][2] = None  # Introduce a missing value
df_filled = df_with_missing.fillna(
df_with_missing['Salary'].mean())
print(df_filled)
# Output:
#       Name  Age    Salary Department  Years_At_Company
# 0    Alice   30  50000.0         HR                 5
# 1      Bob   25  60000.0    Finance                 3
# 2  Charlie   35  63000.0         IT                 8
# 3    David   40  80000.0         IT                10
# 4      Eve   28  55000.0         HR                 6
```

count()

The `count()` function counts non-null values in each column:

```
# Count non-null values
print(df.count())
# Output:
# Name                5
# Age                 5
# Salary              5
# Department          5
# Years_At_Company    5
# dtype: int64
```

cut()

The `cut()` function bins continuous data into intervals:

```
# Bin Age into categories
age_bins = pd.cut(df['Age'], bins=[20, 30, 40, 50],
labels=['Young', 'Middle-aged', 'Senior'])
print(age_bins)
# Output:
# 0        Middle-aged
# 1              Young
# 2        Middle-aged
# 3             Senior
# 4        Middle-aged
# Name: Age, dtype: category
# Categories (3, object): ['Young' < 'Middle-aged' < 'Senior']
```