Working with Data Files 2

Course:

INFO-6145 Data Science and Machine Learning

Revised by:
Mohammad Noorchenarboo

September 19, 2024

Contents

Current Section

1. Data Files Continued
   - NumPy and Pandas
   - Working with DataFrames: Indexing
   - Combining and Appending Data
   - Exploring Data
   - Shape and Manipulating Data
   - Data Cleaning
   - Data Analysis: Correlations
   - Data Visualization

NumPy and Pandas Data Types

Pandas Classes:

- **Series:** A one-dimensional labeled array for holding any data type (e.g., integers, strings).
- **DataFrame:** A two-dimensional data structure, similar to a table, with labeled rows and columns.

NumPy Data Types:

- `float`, `int`, `bool`, `datetime64[ns]`: Common data types used in NumPy arrays.

Pandas adds these dtypes:

- `object`: Any Python object.
- `stringDtype`: For handling strings explicitly in Pandas.

Can a DataFrame Index be Text?

Yes! The index of a DataFrame can be text labels.
Example:

Text Indexing

```
df = pd.DataFrame({'A':[2,3,5], 'Z':[4,5,6]},
                  index=['Uw', 'A', 'Ur'])
print(df)
```

Combining Series into a DataFrame

Overview:

- A Series is a one-dimensional array.
- Multiple Series can be combined to form a DataFrame.

Example:

Combining Series

```
s1 = pd.Series(['100', '200', 'python',
                '300.12', '400'])
s2 = pd.Series(['10', '20', 'php', '30.12', '40'])
df = pd.concat([s1, s2], axis=1)
print(df)
```

Appending Data to an Empty DataFrame

Appending Rows:

- Data can be appended dynamically to a DataFrame.

Example:

Appending Rows

```
df = pd.DataFrame()
data = pd.DataFrame({"col1": range(3),
                     "col2": range(3)})
df = df.append(data)
print(df)
```

Exploring Data in Pandas

Statistical Summary:

- `df.describe()` gives a quick summary of data.

Example:

Using `describe()`

```
df.describe()
```

Slicing Rows:

- Use slicing to select multiple rows.

Slicing Rows

```
df[2:4]
```

Exploring Shape and Manipulating Data

Shape of Data:

- `df.shape` returns the number of rows and columns.

Transposing Data:

- `df.T` transposes the DataFrame (swapping rows and columns).

Formulas:

- You can apply mathematical operations to columns.

Creating New Columns

```
df['NewSalary'] = df['Salary'] * 1.05
```

Data Cleaning Techniques

Drop Columns or Rows:

- `df.drop()` removes unneeded columns or rows.

Dropping Columns

```
df.drop(['RowNumber', 'CustomerId'],
        axis=1, inplace=True)
```

Handling Missing Data:

- Use `dropna()` to remove rows with missing data.

Dropping Missing Data

```
df.dropna(inplace=True)
```

Removing Duplicates:

- Use `drop_duplicates()` to remove duplicate rows.

Data Analysis: Correlations

Overview:

- The `corr()` method calculates correlations between columns.

Perfect Correlation:

Example

```
df.corr()
```

Data Visualization with Matplotlib

Overview:

- Matplotlib can be used to visualize data in Pandas DataFrames.

Example: Plotting Data

Using `matplotlib`

```
import matplotlib.pyplot as plt
df.plot()
plt.show()
```

Sources

- w3resource
- w3schools
- pydata userguide
- pydata api towardsdatascience