Week 1: Pandas Tutorial for Data Wrangling

Team DSS: Kevin Miao, Youngli Hong, Sandeep Sainath, Amanda Ma

Agenda

1. Introduction
2. Data structures
3. Data selection
4. Applying Functions
5. Reshaping Data
6. Merging Data
7. Data Wrangling in Practice

Introduction

Why Pandas?

- The Python ecosystem is universally popular in practical machine learning applications
- Pandas is the most popular choice for data wrangling

To begin:

- conventional Pandas import statement in Python is "import pandas as pd".
- This will be assumed in conjunction with the conventional NumPy import statement "import numpy as np"

Data Structures

DataFrames:

analog of tabular datasets in Pandas

2-dimensional data structure with rows and columns

To create your own DataFrame, use the pd.DataFrame() method to pass in dict-like data

Series:

analog of columns of data in Pandas

take the form of arrays and lists, with a common data type

array-like operations are element-wise

To create your own Series, use the pd.Series() method

Data Types:

integers, floats, strings, booleans, and datetime objects

all classified as **dtypes** in Pandas

.dtypes attribute returns all available data types in a particular DataFrame or Series

Data Selection - 1

Indexing: .loc() vs. iloc()

.loc() function is used to access a group of rows and columns by **label**

.iloc function is used to access the same by **position**

also possible to use slicing in place of a label or position

Transforming the Index:

set_index(): set a column as the index

reset_index(): remove the index, make it column

Subsetting Data:

all rows in a DataFrame with a numerical value of a particular column greater than 10 AND string value of a particular column equal to "Hello":

```
df[(df['column_1'] > 10) & (df['column_2'] == "Hello")]
```

Data Selection - 2

Creating/Dropping/Renaming Columns:

df['new_column_name'] = Series_or_list_of_data

Dropping: df.drop('column_name', axis=1)

The axis=1 parameter indicates that we want to drop a column, not a row (to drop rows, use axis=0).

Renaming: df.rename(columns={'old1':'new1', 'old2':'new2'})

"inplace" default parameter: indicates whether this method will mutate an existing DataFrame or return a new one with the specified column(s) dropped. This parameter is False by default.

Sorting:

Values in a DataFrame and Series can be sorted in any order using the sort_values() method.

Sort the "order_number" column in descending order:

df["number"].sort_values(axis=0, ascending=False)

Dropping Duplicates:

df.drop_duplicates() method returns a DataFrame with duplicate rows removed

Applying Functions

The .apply() function allows you to pass in any function, built-in or custom, to apply along any axis of a DataFrame

the "axis" parameter in df.apply() with value 0 corresponds to applying a function over its rows, while value 1 corresponds to columns.

- using the NumPy function "np.sum" aggregation function to calculate the arithmetic sum in .apply() will return one value, aka the sum of the specified Series.

- using a custom function such as "lambda x: x + 2", which adds 2 to every row, will return a DataFrame/Series with the column modified appropriately.

Reshaping Data - Groupbys

groupby operation involves some combination of splitting the object, applying a function, and combining the results

df.groupby() function enables this functionality. The function returns a DataFrameGroupBy object, a special class in Pandas.

Aggregation functions can take the form of both built-in functions and custom functions. Common built-in functions include .mean(), .max(), .min(), .sum(), and .count().

For custom functions, the .agg() function takes in any lambda or pre-defined function and applies to a DataFrameGroupBy object.

Reshaping Data - Pivots, Melts

Pivots:

a rotational transformation of a DataFrame

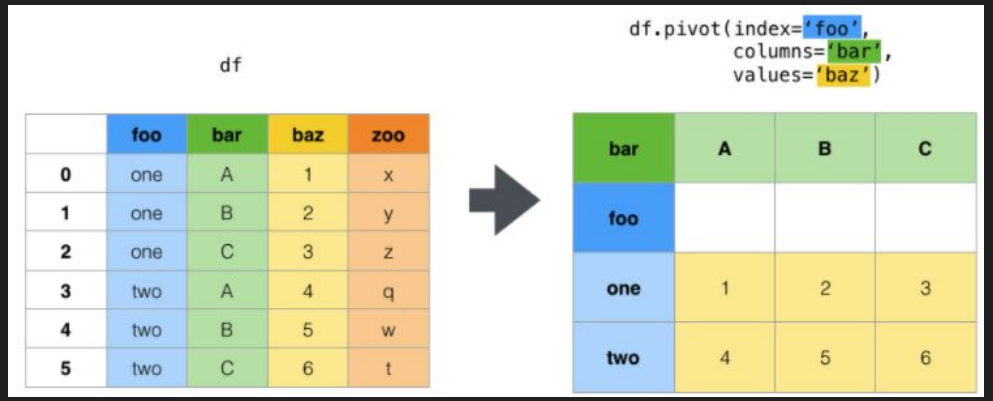
reorganizes a DataFrame such that the table is indexed by one or more of its columns

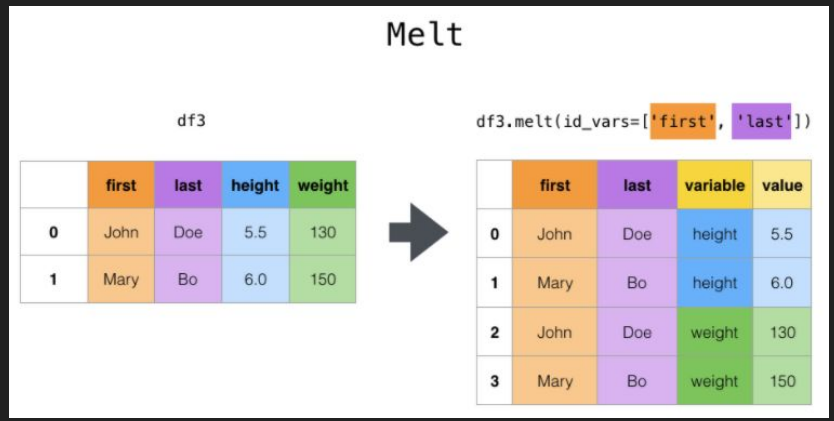
df.pivot() method takes in three main parameters: "index", "columns", and "values"

Melts:

involves "unpivoting" columns over columns that are identifier variables.

df.melt() function specifies this with the "id_vars" parameter through a list

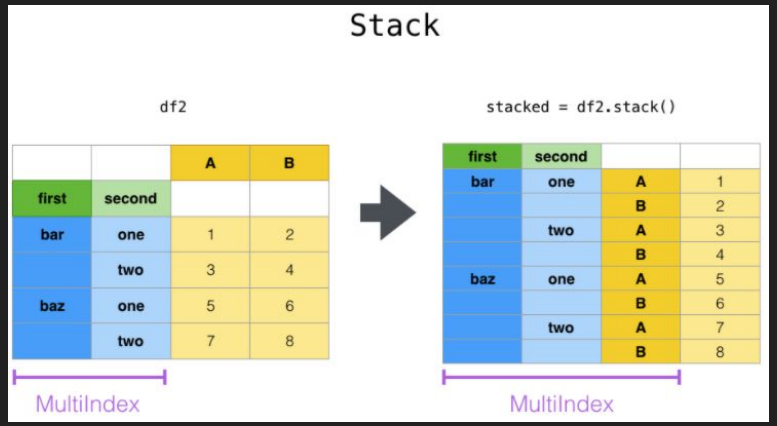




Reshaping Data - Stacking, Unstacking

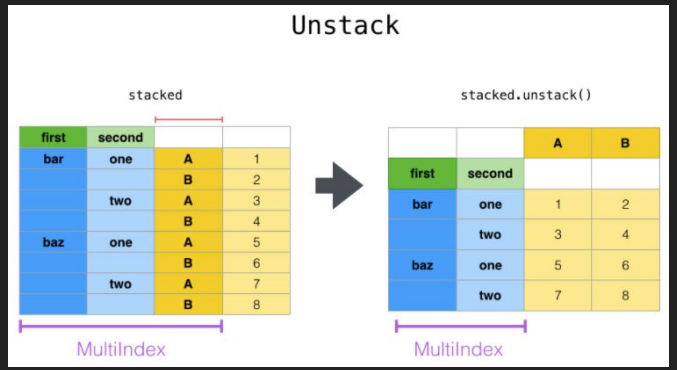
Stacking:

involves adding a column as the next additional level to a DataFrame



Unstacking:

removing a column from the index of a DataFrame and reverting it to a column



Merging Data

Pandas has extensive functionality to perform traditional database-style joins. These include inner joins, left joins, right joins, and outer joins.

pd.merge() function takes several key parameters:

left: the DataFrame on the left.

right: the DataFrame on the right.

how: the type of join ("inner" by default, "left", "right", "outer")

on: column name to join on; use this if this column name is present in both DataFrames

left_on: joiner column from left; unnecessary if **on** is correctly specified.

right_on: joiner column from right: unnecessary if **on** is correctly specified.

suffixes: takes in a tuple of two suffixes corresponding to the left and right DataFrames; these suffixes are added to column names if they exist in both DataFrames (which results in duplicate names for varying data).