# Module-03, Python for Data Analysis

## Data Preparation( Data Transformation )

Dostdar Ali
Instructor

Data science and Artificial Intelligence
3-Months Course
at
Karakaroum international Univrsity

January 11, 2024

## Table of Contents

## Data Transformation

- So far in this sub-module, we've been concerned with rearranging data. Filtering, cleaning, and other transformations are another class of important operations.
-

## Data Transformation

- So far in this sub-module, we’ve been concerned with rearranging data. Filtering, cleaning, and other transformations are another class of important operations.
-

## Removing Duplicates

- Duplicate rows may be found in a DataFrame for any number.

**Example**

Repeat of same information.
data = pd.DataFrame('k1': ['one', 'two'] * 3 + ['two'],
'k2': [1, 1, 2, 3, 3, 4, 4])

- The DataFrame method duplicated returns a boolean Series indicating whether each row is a duplicate (has been observed in a previous row) or not.
- Relatedly, drop-duplicates returns a DataFrame where the duplicated array is False:

## Removing Duplicates

- Duplicate rows may be found in a DataFrame for any number.

**Example**

Repeat of same information.
data = pd.DataFrame('k1': ['one', 'two'] * 3 + ['two'],
'k2': [1, 1, 2, 3, 3, 4, 4])

- The DataFrame method duplicated returns a boolean Series indicating whether each row is a duplicate (has been observed in a previous row) or not.
- Relatedly, drop-duplicates returns a DataFrame where the duplicated array is False:

## Removing Duplicates

- Duplicate rows may be found in a DataFrame for any number.

**Example**

Repeat of same information.
data = pd.DataFrame('k1': ['one', 'two'] * 3 + ['two'],
'k2': [1, 1, 2, 3, 3, 4, 4])

- The DataFrame method duplicated returns a boolean Series indicating whether each row is a duplicate (has been observed in a previous row) or not.
- Relatedly, drop-duplicates returns a DataFrame where the duplicated array is False:

## Transforming Data Using a Function or Mapping

- For many datasets, you may wish to perform some transformation based on the values in an array, Series, or column in a DataFrame. Consider the following hypothetical data collected about various kinds of meat:
- data = pd.DataFrame('food': ['bacon', 'pulled pork', 'bacon', 'Pastrami', 'corned beef', 'Bacon', 'pastrami', 'honey ham', 'nova lox'], 'ounces': [4, 3, 12, 6, 7.5, 8, 3, 5, 6])
- Suppose we wanted to add a column indicating the type of animal that each food came from. Let's write down a mapping of each distinct meat type to the kind of animal:
- The map method on a Series accepts a function or dict-like object containing a mapping, but here we have a small problem in that some of the meats are capitalized and others are not. Thus, we need to convert each value to lowercase using the str.lower Series method:

## Transforming Data Using a Function or Mapping

- For many datasets, you may wish to perform some transformation based on the values in an array, Series, or column in a DataFrame. Consider the following hypothetical data collected about various kinds of meat:
- data = pd.DataFrame('food': ['bacon', 'pulled pork', 'bacon', 'Pastrami', 'corned beef', 'Bacon', 'pastrami', 'honey ham', 'nova lox'], 'ounces': [4, 3, 12, 6, 7.5, 8, 3, 5, 6])
- Suppose we wanted to add a column indicating the type of animal that each food came from. Let's write down a mapping of each distinct meat type to the kind of animal:
- The map method on a Series accepts a function or dict-like object containing a mapping, but here we have a small problem in that some of the meats are capitalized and others are not. Thus, we need to convert each value to lowercase using the str.lower Series method:

## Transforming Data Using a Function or Mapping

- For many datasets, you may wish to perform some transformation based on the values in an array, Series, or column in a DataFrame. Consider the following hypothetical data collected about various kinds of meat:
- data = pd.DataFrame('food': ['bacon', 'pulled pork', 'bacon', 'Pastrami', 'corned beef', 'Bacon', 'pastrami', 'honey ham', 'nova lox'], 'ounces': [4, 3, 12, 6, 7.5, 8, 3, 5, 6])
- Suppose we wanted to add a column indicating the type of animal that each food came from. Let's write down a mapping of each distinct meat type to the kind of animal:
- The map method on a Series accepts a function or dict-like object containing a mapping, but here we have a small problem in that some of the meats are capitalized and others are not. Thus, we need to convert each value to lowercase using the str.lower Series method:

## Transforming Data Using a Function or Mapping

- For many datasets, you may wish to perform some transformation based on the values in an array, Series, or column in a DataFrame. Consider the following hypothetical data collected about various kinds of meat:
- data = pd.DataFrame('food': ['bacon', 'pulled pork', 'bacon', 'Pastrami', 'corned beef', 'Bacon', 'pastrami', 'honey ham', 'nova lox'], 'ounces': [4, 3, 12, 6, 7.5, 8, 3, 5, 6])
- Suppose we wanted to add a column indicating the type of animal that each food came from. Let's write down a mapping of each distinct meat type to the kind of animal:
- The map method on a Series accepts a function or dict-like object containing a mapping, but here we have a small problem in that some of the meats are capitalized and others are not. Thus, we need to convert each value to lowercase using the str.lower Series method:

## Renaming Axis Indexes

- Like values in a Series, axis labels can be similarly transformed by a function or mapping of some form to produce new, differently labeled objects. You can also modify the axes in-place without creating a new data structure. Here's a simple example.
- data = pd.DataFrame(np.arange(12).reshape((3, 4)), index=['Ohio', 'Colorado', 'New York'], columns=['one', 'two', 'three', 'four'])
- Lab work will explore more.

## Renaming Axis Indexes

- Like values in a Series, axis labels can be similarly transformed by a function or mapping of some form to produce new, differently labeled objects. You can also modify the axes in-place without creating a new data structure. Here's a simple example.
- data = pd.DataFrame(np.arange(12).reshape((3, 4)), index=['Ohio', 'Colorado', 'New York'], columns=['one', 'two', 'three', 'four'])
- Lab work will explore more.

## Renaming Axis Indexes

- Like values in a Series, axis labels can be similarly transformed by a function or mapping of some form to produce new, differently labeled objects. You can also modify the axes in-place without creating a new data structure. Here's a simple example.
- data = pd.DataFrame(np.arange(12).reshape((3, 4)), index=['Ohio', 'Colorado', 'New York'], columns=['one', 'two', 'three', 'four'])
- Lab work will explore more.

## Computing Indicator/Dummy Variables

- Another type of transformation for statistical modeling or machine learning applications is converting a categorical variable into a “dummy” or “indicator” matrix. If a column in a DataFrame has k distinct values, you would derive a matrix or DataFrame with k columns containing all 1s and 0s. pandas has a get-dummies function for doing this, though devising one yourself is not difficult. Let’s return to an earlier example DataFrame,
- df = pd.DataFrame('key': ['b', 'b', 'a', 'c', 'a', 'b'], 'data1': range(6))

## Computing Indicator/Dummy Variables

- Another type of transformation for statistical modeling or machine learning applications is converting a categorical variable into a "dummy" or "indicator" matrix. If a column in a DataFrame has k distinct values, you would derive a matrix or DataFrame with k columns containing all 1s and 0s. pandas has a get-dummies function for doing this, though devising one yourself is not difficult. Let's return to an earlier example DataFrame,
- df = pd.DataFrame('key': ['b', 'b', 'a', 'c', 'a', 'b'], 'data1': range(6))

Great Job
Thank you