Python For Data Analysis

By: Ani Anisyah, M.T.
Universitas Pendidikan Indonesia

Content

1. Overview of Python Libraries for Data Scientists
2. Reading Data
3. Plotting Data
4. Descriptive Statistics
5. Assignment

Python Libraries for Data Science

Many popular Python Libraries such as:

1. Numpy
2. SciPy
3. Pandas
4. Scikit-Learn
5. Matplotlib
6. Seaborn
7. And Many More

Numpy

- Numpy is a module which provides the basic data structures, implementing multi-dimensional arrays and matrices. Besides that the module supplies the necessary functionalities to create and manipulate these data structures as well as functions that allow to easily perform advanced mathematical and statistical operations on those objects.

- Provides vectorization of mathematical operations on arrays and matrices which significantly improves the performance.

- Link : https://numpy.org/

SciPy

- SciPy is based on top of Numpy, i.e. it uses the data structures provided by NumPy. It extends the capabilities of NumPy with further useful functions for minimization, regression, Fourier-transformation and many others.
- SciPy have Collection of algorithms for linear algebra, differential equations, numerical integration, optimization, statistics and more.
- Link : https://scipy.org/

Pandas

- Pandas provides tools for data manipulation: reshaping, merging, sorting, slicing, aggregation etc. it provides add data structures and tools designed to work with table-like data (similar to Series and Data Frames in R). Besides that, it allows handling missing data.
- The special focus of Pandas consists in offering data structures and operations for manipulating numerical tables and time series.
- Link : http://pandas.pydata.org/

Scikit-Learn

- Scikit-learn provides simple and efficient tools for data mining and data analysis and is accessible to everyone and reusable in various contexts.
- Scikit-learn is known for its user-friendly interface, comprehensive documentation, and wide range of algorithms. It provides machine learning algorithms such as classification, regression, clustering, model validation, etc.
- Scikit-learn is a popular machine learning library for Python, built on NumPy, SciPy, and matplotlib
- Link : http://scikit-learn.org/

Matplotlib

- Matplotlib is a comprehensive library for creating static, animated, and interactive visualizations in Python. It is widely used for generating plots, histograms, power spectra, bar charts, error charts, scatterplots, etc.,
- Python 2D plotting library which produces publication quality figures in a variety of hardcopy formats
- Having relatively low-level; some effort needed to create advanced visualization
- Link : http://matplotlib.org/

Seaborn

- Seaborn is a Python visualization library based on Matplotlib that provides a high-level interface for drawing attractive and informative statistical graphics. It simplifies the process of creating complex visualizations and enhances the visual appeal of Matplotlib plots.
- Similar (in style) to the popular ggplot2 library in R
- Link : https://seaborn.pydata.org/

2

Reading Data with Python Libraries

Selecting and Filtering the Data; Data manipulation, sorting, grouping, rearranging

Loading Python Libraries

```python
#import python libraries
import numpy as np
import scipy as sp
import pandas as pd
import matplotlib as mpl
import seaborn as sns
```

Reading data Using Pandas Libraries

```
from google.colab import files
uploaded = files.upload()
```

Import files libraries from external sources

Choose Files Salaries.csv

- **Salaries.csv**(text/csv) - 2126 bytes, last modified: 6/25/2024 - 100% done

```
Saving Salaries.csv to Salaries.csv
```

```
[5] #read csv file
df = pd.read_csv("Salaries.csv")
```

Read csv data from imported file

```
[8] #reading list of the data
df.head()
```

Read the first 5 records with function head()

	rank	discipline	phd	service	sex	salary
0	Prof	B	56	49	Male	186960
1	Prof	A	12	6	Male	93000
2	Prof	A	23	20	Male	110515
3	Prof	A	40	31	Male	131205
4	Prof	B	20	18	Male	104800

Data Frame

Pandas Type	Native Python Type	Description
object	string	The most general dtype. Will be assigned to your column if column has mixed types (numbers and strings).
int64	int	Numeric characters. 64 refers to the memory allocated to hold this character.
float64	float	Numeric characters with decimals. If a column contains numbers and NaNs(see below), pandas will default to float64, in case your missing value has a decimal.
datetime64, timedelta[ns]	N/A (but see the datetime module in Python's standard library)	Values meant to hold time data. Look into these for time series experiments.

Data Frame Data Types

```
[9]  #check a particular column type
     df['salary'].dtype
```

```
dtype('int64')
```

```
[10] #check types for all columns
     df.dtypes
```

```
rank          object
discipline    object
phd            int64
service        int64
sex           object
salary         int64
```

Data Frame Attributes

df.attribute	description
dtypes	list the types of the columns
columns	list the column names
axes	list the row labels and column names
ndim	number of dimensions
size	number of elements
shape	return a tuple representing the dimensionality
values	numpy representation of the data

Data Frame Method

df.method()	description
head([n]), tail([n])	first/last n rows
describe()	generate descriptive statistics (for numeric columns only)
max(), min()	return max/min values for all numeric columns
mean(), median()	return mean/median values for all numeric columns
std()	standard deviation
sample([n])	returns a random sample of the data frame
dropna()	drop all the records with missing values

Data Frame Method

Give the max of salary of the first 5 records in the dataset

```
#get max values of the first 5 records
df['salary'].head(5).max()
```

```
186960
```

Data Frame with Groupby Method

Using "group by" method we can:

- Split the data into groups based on some criteria
- Calculate statistics (or apply a function) to each group

```
data = pd.read_csv("Salaries (2).csv")
#make dataframe from data
df_data = pd.DataFrame(data)

#group data using sex
df_sex = df_data.groupby('sex')
```

```
#calculate mean value for each numeric column per each group
df_sex['salary'].mean()
```

```
sex
Female     101002.410256
Male       115045.153846
Name: salary, dtype: float64
```

```
[ ] Start coding or generate with AI.
```

Data Frame: Filtering

- To subset the data we can apply Boolean indexing. This indexing is commonly known as a filter.
- For example if we want to subset the rows in which the salary value is greater than $150K:

```
#find salary more than $150K
df_sub = df[ df['salary'] > 150000 ]
df_sub
```

	rank	discipline	phd	service	sex	salary
0	Prof	B	56	49	Male	186960
13	Prof	B	35	33	Male	162200
14	Prof	B	25	19	Male	153750
15	Prof	B	17	3	Male	150480
19	Prof	A	29	27	Male	150500
27	Prof	A	45	43	Male	155865
31	Prof	B	22	21	Male	155750
44	Prof	B	23	19	Female	151768
72	Prof	B	24	15	Female	161101

Any logical operator can be used:

Symbol	Description
>	Greater
<	Less
==	Equal
>=	Greater or equal
<=	Less or equal
!=	Not equal

Data Frame: Filtering

Any logical operator can be used:

Symbol	Description
>	Greater
<	Less
==	Equal
>=	Greater or equal
<=	Less or equal
!=	Not equal

```
#Select only those rows that contain female professors:
df_f = df[ df['sex'] == 'Female' ]
df_f
```

	rank	discipline	phd	service	sex	salary
39	Prof	B	18	18	Female	129000
40	Prof	A	39	36	Female	137000
41	AssocProf	A	13	8	Female	74830
42	AsstProf	B	4	2	Female	80225
43	AsstProf	B	5	0	Female	77000
44	Prof	B	23	19	Female	151768
45	Prof	B	25	25	Female	140096
46	AsstProf	B	11	3	Female	74692

Data Frame: Slicing

There are number of ways to subset the data frame:

1. One or more columns
2. One or more rows
3. A subset of rows and columns

Data Frame: Slicing

Example: selecting one column

```
#select column salary
df['salary']
```

```
0     186960
1      93000
2     110515
3     131205
4     104800
       ...
73    105450
74    104542
75    124312
76    109954
77    109646
Name: salary, Length: 78, dtype: int64
```

it is possible to use single set of brackets, but the resulting object will be a Series (not a DataFrame)

Example: selecting more than one column

```
#Select column salary and rank:
df[['rank','salary']]
```

	rank	salary
0	Prof	186960
1	Prof	93000
2	Prof	110515
3	Prof	131205
4	Prof	104800
...	...	...
73	Prof	105450
74	AssocProf	104542
75	Prof	124312
76	Prof	109954
77	Prof	109646

78 rows × 2 columns

Data Frame: Selecting Rows

Example: selecting rows by their position

```python
#select rows by their position
df[10:20]

#note: the first row has position 0
```

	rank	discipline	phd	service	sex	salary
10	Prof	B	39	33	Male	128250
11	Prof	B	23	23	Male	134778
12	AsstProf	B	1	0	Male	88000
13	Prof	B	35	33	Male	162200
14	Prof	B	25	19	Male	153750
15	Prof	B	17	3	Male	150480
16	AsstProf	B	8	3	Male	75044
17	AsstProf	B	4	0	Male	92000

Data Frame: method iloc

If we need to select a range of rows and/or columns, using their positions we can use method iloc

```
#Select rows by their labels:
df.iloc[10:20,[0, 3, 4, 5]]
```

	rank	service	sex	salary
10	Prof	33	Male	128250
11	Prof	23	Male	134778
12	AsstProf	0	Male	88000
13	Prof	33	Male	162200
14	Prof	19	Male	153750
15	Prof	3	Male	150480
16	AsstProf	3	Male	75044
17	AsstProf	0	Male	92000
18	Prof	7	Male	107300
19	Prof	27	Male	150500

Data Frame: method iloc

```
df.iloc[0]   # First row of a data frame
df.iloc[i]   #(i+1)th row
df.iloc[-1] # Last row
```

```
df.iloc[:, 0]   # First column
df.iloc[:, -1] # Last column
```

```
df.iloc[0:7]        #First 7 rows
df.iloc[:, 0:2]     #First 2 columns
df.iloc[1:3, 0:2]   #Second through third rows and first 2 columns
df.iloc[[0,5], [1,3]]   #1st and 6th rows and 2nd and 4th columns
```

Data Frame: sorting

We can sort the data by a value in the column. By default the sorting will occur in ascending order and a new data frame is return.

```
# Create a new data frame from the original sorted by the column Salary
df_sorted = df.sort_values( by ='service')
df_sorted.head()
```

	rank	discipline	phd	service	sex	salary
55	AsstProf	A	2	0	Female	72500
23	AsstProf	A	2	0	Male	85000
43	AsstProf	B	5	0	Female	77000
17	AsstProf	B	4	0	Male	92000
12	AsstProf	B	1	0	Male	88000

Data Frame: sorting

Example : sorting using 2 or more columns

```
[79] df_sorted = df.sort_values( by =['service', 'salary'], ascending = [True, False])
     df_sorted.head(10)
```

	rank	discipline	phd	service	sex	salary
52	Prof	A	12	0	Female	105000
17	AsstProf	B	4	0	Male	92000
12	AsstProf	B	1	0	Male	88000
23	AsstProf	A	2	0	Male	85000
43	AsstProf	B	5	0	Female	77000
55	AsstProf	A	2	0	Female	72500
57	AsstProf	A	3	1	Female	72500
28	AsstProf	B	7	2	Male	91300
42	AsstProf	B	4	2	Female	80225
68	AsstProf	A	4	2	Female	77500

3

Plotting Data

Missing Value

- Missing values are marked as NaN
- There are a number of methods to deal with missing values in the data frame:

df.method()	description
dropna()	Drop missing observations
dropna(how='all')	Drop observations where all cells is NA
dropna(axis=1, how='all')	Drop column if all the values are missing
dropna(thresh = 5)	Drop rows that contain less than 5 non-missing values
fillna(0)	Replace missing values with zeros
isnull()	returns True if the value is missing
notnull()	Returns True for non-missing values

Missing Value

- When summing the data, missing values will be treated as zero
- If all values are missing, the sum will be equal to NaN
- cumsum() and cumprod() methods ignore missing values but preserve them in the resulting arrays
- Missing values in GroupBy method are excluded (just like in R)
- Many descriptive statistics methods have skipna option to control if missing data should be excluded . This value is set to True by default (unlike R)

4 Descriptive Analysis

Aggregation Function in Pandas

In pandas, aggregation functions are used to perform operations on data, typically after grouping it. Common aggregation functions include sum, mean, median, min, max, count, and others. Here's how you can use them with pandas DataFrames:

1. **Grouping data using groupby**:
   - The groupby method is used to split the data into groups based on some criteria.
2. **Applying aggregation functions**:
   - Once the data is grouped, you can apply aggregation functions using methods like agg, sum, mean, etc.

Basic Descriptive Statistics

df.method()	description
describe	Basic statistics (count, mean, std, min, quantiles, max)
min, max	Minimum and maximum values
mean, median, mode	Arithmetic average, median and mode
var, std	Variance and standard deviation
sem	Standard error of mean
skew	Sample skewness
kurt	kurtosis

Aggregation Function in Pandas

agg() method are useful when multiple statistics are computed per column:

```
df['salary'].agg(['min', 'max', 'mean'])
```

```
min      57800.000000
max     186960.000000
mean    108023.782051
Name: salary, dtype: float64
```

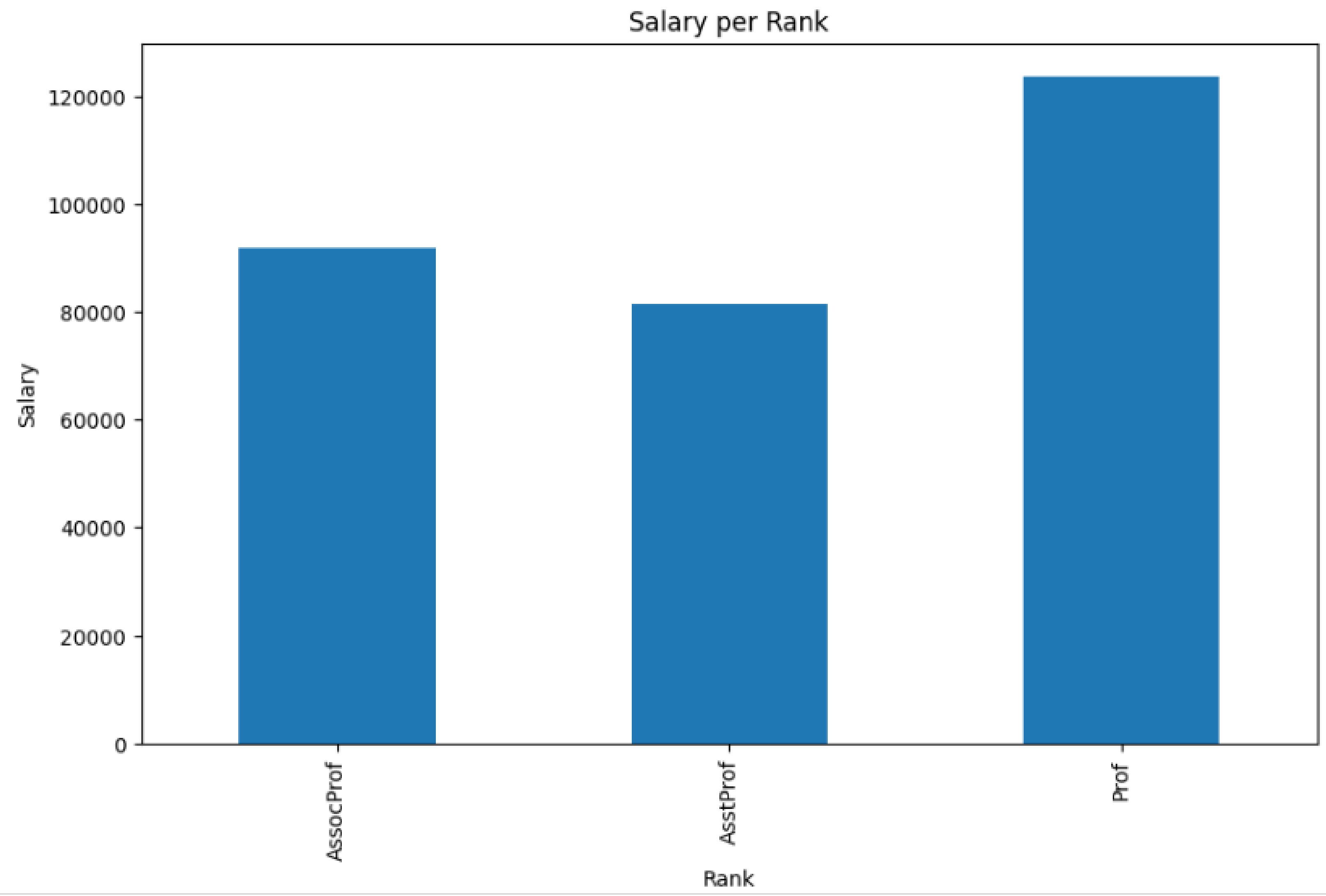
Graphic to explore the data

Visualize the data with matplotlib library

```
import matplotlib.pyplot as plt
class_averages = data.groupby('rank')['salary'].mean()

# Membuat grafik bar untuk data rata-rata
plt.figure(figsize=(10,6))
class_averages.plot(kind='bar')
plt.xlabel('Rank')
plt.ylabel('Salary')
plt.title('Salary per Rank')

# Menunjukan grafik bar
plt.show()
```



5

Assignment

About Data

This dataset is having the 50000 sales orders data that consist of columns as following :

1. Order ID
2. Quantity
3. Product Id
4. Seller Id
5. Freight Value
6. Customer Id
7. Order Status
8. Purchase Status
9. Payment Type
10. Product Category Name
11. Product Weight in gram

Objectives

- Conduct exploratory Data Analysis (EDA): Perform EDA to understand the distribution and relationships between variables from the data
- Analyzing Sales Dataset is to identify salles patterns from order data that resonate with consumers and propel them to purchase.
- Translate insights into actionable recommendations that optimize product development, inform marketing strategies, and boost competitive edge.

Description Task

Exploring the data Sales involves a step-by-step process:

1. Check and prepare data to clean and handling missing values and ensuring consistency.
2. Summaries the data with statistical analysis: Use descriptive statistics with aggregation function (i.e sum, count, average, min, max) for searching meaningful information such as: top product sales, total amount, average amount, etc
3. Use Statistical methods to identify significant correlation/comparative/ distribution/trending between variables from the data
4. Visualize the data with charts and graphs to see patterns and relationships (min. 3 graph)
5. Use related python library to handle all of tasks
6. Upload your source code with python extension file such as .py or .ipynb and file .rawgraphs (if you used visualize data using rawgraphs)
7. Tomorrow some of you will present the result of your assignment

Python Libraries

- We will use the following libraries:
  1. Pandas: Data manipulation and analysis
  2. Numpy: Numerical operations and calculations
  3. Matplotlib: Data visualization and plotting
  4. Seaborn: Enhanced data visualization and statistical graphics
  5. Scipy: Scientific computing and advanced mathematical operations
  6. RawGraph: A free and open source tool for data visualization (https://www.rawgraphs.io/)

"Summer Course 2024"

Thank You

Any Question?