East West University
Department of Computer Science and Engineering

CSE 303: Statistics for Data Science
LAB 03 (Handout)
Course Instructor: Dr. Mohammad Rezwanul Huq

Pandas for Data Analysis

Lab Objective

Data analysis using Pandas in Python.

Lab Outcome

After completing this lab successfully, students will be able to:

1. **Understand** the Python Pandas operations for data manipulation, analysis and cleaning.
2. **Use** Pandas functions properly and **Write** appropriate Python programs for data analysis.

Psychomotor Learning Levels

This lab involves activities that encompass the following learning levels in psychomotor domain.

Level	Category	Meaning	Keywords
P1	Imitation	Copy action of another; observe and replicate.	Relate, Repeat, Choose, Copy, Follow, Show, Identify, Isolate.
P2	Manipulation	Reproduce activity from instruction or memory	Copy, response, trace, Show, Start, Perform, Execute, Recreate.

Required Applications/Tools

- Anaconda Navigator (Anaconda3)
  - Anaconda is a distribution of the Python and R programming languages for scientific computing (data science, machine learning applications, large-scale data processing, predictive analytics, etc.), that aims to simplify package management and deployment.
  - Popular Tools/IDEs: Spyder, Jupyter Notebook
- Google Colab: Colaboratory, or "Colab" for short, is a product from Google Research. Colab allows anybody to write and execute arbitrary python code through the browser, and is especially well suited to machine learning, data analysis and education.

Lab Activities

1. Pandas

Pandas is a fast, powerful, flexible and easy to use open source data analysis and manipulation tool, built on top of the Python programming language. Pandas is designed for working with tabular or heterogeneous data. Pandas must be imported before use.

```
import pandas as pd
```

2. Pandas Series

A **Series** is a one-dimensional array-like object containing a sequence of values.

```
obj = pd.Series([4, 7, -5, 3])
```

Filtering data: `obj[obj%2==0]`

Scalar Operations: `obj+5, obj * 2`

Creating a Series from a Dictionary

```
sdata = {'Ohio': 35000, 'Texas': 71000, 'Oregon': 16000,
         'Utah': 5000}
obj3 = pd.Series(sdata)
print(obj3)
```

3. Pandas DataFrame

A **DataFrame** represents a rectangular table of data and contains an ordered collection of columns, each of which can be a different value type. The DataFrame has both a row and column index; it can be thought of as a dictionary of Series all sharing the same index.

Constructing Pandas DataFrame:

1. From List

```
list1 = [['Alice',23,3.5,10],['Bob',24,3.4,6],['Charlie',22,3.9,8]]
df = pd.DataFrame(list1)
df.columns = ['name','age','cgpa','hoursStudied']
print(df.head())
```

2. From Dictionary

```
dict1 = {'id':[1,2,3],'name':['alice','bob','charlie'],
         'age':[20, 25, 32]}
df1 = pd.DataFrame(dict1)
print(df1)
```

3. From CSV File

```
df2 = pd.read_csv('sample_data_1.csv', header = None)
df2.columns=['id','state','population','murder_rate’]
print(df2)
df2.head() # displays first 5 rows
df2.tail() # displays last 5 rows
df2.count() # displays number of values for each column
```

4. Select, Add, Delete, Rename Indices, Rows or Columns of/from a DataFrame

Seleting the first cell

```
print(df1.iloc[0][0])
print(df1.loc[0]['name'])
```

Selecting a few columns

```
df3=df1[['name','cgpa']]
print(df3)
```

selecting a few rows

```
df4 = df1.loc[1:2]
print(df4)
df5 = df1.iloc[1:3]
print(df5)
```

selecting a few rows and columns

```
df4 = df1.loc[1:2,['name','age']]
print(df4)
df5 = df1.iloc[1:3,[0,1]]
print(df5)
```

appending two dataframes

```
list1 = [['Alice',23,3.5,10],['Bob',24,3.4,6],['Charlie',22,3.9,8]]
df = pd.DataFrame(list1)
df.columns = ['name','age','cgpa','hoursStudied']

list2 = [['Don',21,2.5,2],['Elton',25,2.75,4]]
df11 = pd.DataFrame(list2)
df11.columns = ['name','age','cgpa','hoursStudied']

df12 = df.append(df11, ignore_index=True)
print(df12)
```

deleting rows/columns from a dataframe

```
df12.drop([0,1], inplace=True)
df12.drop(['cgpa'], axis=1, inplace=True)
```

Renaming columns

```
new_cols = ['n','a','hs']
df12.columns=new_cols
print(df12)
```

5. Data Filtering, Sorting

```
cgpa_greater_than_three_point_five1 = df1[df1['cgpa'] > 3.5]
cgpa_greater_than_three_point_five2 = df1.loc[df1['cgpa'] > 3.5]
cgpa_greater_than_three_point_five3 = df1.query('cgpa > 3.5')


print(cgpa_greater_than_three_point_five1)
print(cgpa_greater_than_three_point_five2)
print(cgpa_greater_than_three_point_five3)

df1.sort_values(by='age',ascending=False)
```

6. Computing Descriptive Statistical Measures

`count`	Number of non-NA values
`describe`	Compute set of summary statistics for Series or each DataFrame column
`min, max`	Compute minimum and maximum values
`argmin, argmax`	Compute index locations (integers) at which minimum or maximum value obtained, respectively
`idxmin, idxmax`	Compute index labels at which minimum or maximum value obtained, respectively
`quantile`	Compute sample quantile ranging from 0 to 1
`sum`	Sum of values
`mean`	Mean of values
`median`	Arithmetic median (50% quantile) of values
`mad`	Mean absolute deviation from mean value
`prod`	Product of all values
`var`	Sample variance of values
`std`	Sample standard deviation of values
`skew`	Sample skewness (third moment) of values
`kurt`	Sample kurtosis (fourth moment) of values
`cumsum`	Cumulative sum of values
`cummin, cummax`	Cumulative minimum or maximum of values, respectively
`cumprod`	Cumulative product of values
`diff`	Compute first arithmetic difference (useful for time series)
`pct_change`	Compute percent changes

7. Visualization

1. Boxplot

```
df.boxplot(column=['col_name_1','col_name_2'])
```

2. Histogram

```
df.hist(column = ['col_name'], bins= 5)
```

3. Bar Chart

```
df['col_name'].value_counts().plot(kind = 'bar')
```

4. Pie Chart

```
df['col_name'].value_counts().plot(kind = 'pie')
```

5. Scallet Plot

```
df.plot.scatter(x='col_name_1',y='col_name_2')
```