/ labs / DA0101EN /

Name ▲	Last Modified
review-introduction.ipynb	seconds ago

Launcher ×   review-introduction.ipynb ×

Markdown ▾   Python ○

COGNITIVE CLASS

Data Analysis with Python

Introduction

Welcome!

In this section, you will learn how to approach data acquisition in various ways, and obtain necessary insights from a dataset. By the end of this lab, you will successfully load the data into Jupyter Notebook, and gain some fundamental insights via Pandas Library.

Table of Contents

1. Data Acquisition
2. Basic Insight of Dataset

Estimated Time Needed: **10 min**

Data Acquisition

There are various formats for a dataset, .csv, .json, .xlsx etc. The dataset can be stored in different places, on your local machine or sometimes online.

In this section, you will learn how to load a dataset into our Jupyter Notebook.

In our case, the Automobile Dataset is an online source, and it is in CSV (comma separated value) format. Let's use this dataset as an example to practice data reading.

- data source: https://archive.ics.uci.edu/ml/machine-learning-databases/autos/imports-85.data
- data type: csv

The Pandas Library is a useful tool that enables us to read various datasets into a data frame; our Jupyter notebook platforms have a built-in **Pandas Library** so that all we need to do is import Pandas without installing.

```python
# import pandas library
import pandas as pd
```
•••

Read Data

We use `pandas.read_csv()` function to read the csv file. In the bracket, we put the file path along with a quotation mark, so that pandas will read the file into a data frame from that address. The file path can be either an URL or your local file address.

Because the data does not include headers, we can add an argument `headers = None` inside the `read_csv()` method, so that pandas will not automatically set the first row as a header.

You can also assign the dataset to any variable you create.

This dataset was hosted on IBM Cloud object click HERE for free storage.

```python
# Import pandas library
import pandas as pd

# Read the online file by the URL provides above, and assign it to variable "df"
other_path = "https://s3-api.us-geo.objectstorage.softlayer.net/cf-courses-data/CognitiveClass/DA0101EN/auto.csv"
df = pd.read_csv(other_path, header=None)
```

After reading the dataset, we can use the `dataframe.head(n)` method to check the top n rows of the dataframe; where n is an integer. Contrary to `dataframe.head(n)`, `dataframe.tail(n)` will show you the bottom n rows of the dataframe.

```python
# show the first 5 rows using dataframe.head() method
print("The first 5 rows of the dataframe")
df.head(5)
```

Question #1:

check the bottom 10 rows of data frame "df".

```python
# Write your code below and press Shift+Enter to execute
```

Question #1 Answer:

Run the code below for the solution!

Double-click **here** for the solution.

Add Headers

Take a look at our dataset; pandas automatically set the header by an integer from 0.

To better describe our data we can introduce a header, this information is available at: https://archive.ics.uci.edu/ml/datasets/Automobile

Thus, we have to add headers manually.

Firstly, we create a list "headers" that include all column names in order. Then, we use `dataframe.columns = headers` to replace the headers by the list we created.

```
# create headers list
headers = ["symboling","normalized-losses","make","fuel-type","aspiration", "num-of-doors","body-style",
         "drive-wheels","engine-location","wheel-base", "length","width","height","curb-weight","engine-type",
         "num-of-cylinders", "engine-size","fuel-system","bore","stroke","compression-ratio","horsepower",
         "peak-rpm","city-mpg","highway-mpg","price"]
print("headers\n", headers)
```

We replace headers and recheck our data frame

```
df.columns = headers
df.head(10)
```

we can drop missing values along the column "price" as follows

```
df.dropna(subset=["price"], axis=0)
```

Now, we have successfully read the raw dataset and add the correct headers into the data frame.

Question #2:

Find the name of the columns of the dataframe

```
# Write your code below and press Shift+Enter to execute
```

Double-click **here** for the solution.

Save Dataset

Correspondingly, Pandas enables us to save the dataset to csv by using the `dataframe.to_csv()` method, you can add the file path and name along with quotation marks in the brackets.

For example, if you would save the dataframe **df** as **automobile.csv** to your local machine, you may use the syntax below:

```
df.to_csv("automobile.csv", index=False)
```

We can also read and save other file formats, we can use similar functions to `pd.read_csv()` and `df.to_csv()` for other data formats, the functions are listed in the following table:

Read/Save Other Data Formats

Data Formate	Read	Save
csv	pd.read_csv()	df.to_csv()
json	pd.read_json()	df.to_json()
excel	pd.read_excel()	df.to_excel()
hdf	pd.read_hdf()	df.to_hdf()
sql	pd.read_sql()	df.to_sql()
...	...	...

Did you know? IBM Watson Studio lets you build and deploy an AI solution, using the best of open source and IBM software and giving your team a single environment to work in. Learn more here.

Basic Insight of Dataset

After reading data into Pandas dataframe, it is time for us to explore the dataset.
There are several ways to obtain essential insights of the data to help us better understand our dataset.

Data Types

Data has a variety of types.
The main types stored in Pandas dataframes are **object**, **float**, **int**, **bool** and **datetime64**. In order to better learn about each attribute, it is always good for us to know the data type of each column. In Pandas:

```
df.dtypes
```

returns a Series with the data type of each column.

```
# check the data type of data frame "df" by .dtypes
print(df.dtypes)
```

As a result, as shown above, it is clear to see that the data type of "symboling" and "curb-weight" are `int64`, "normalized-losses" is `object`, and "wheel-base" is `float64`, etc.

These data types can be changed; we will learn how to accomplish this in a later module.

Describe

If we would like to get a statistical summary of each column, such as count, column mean value, column standard deviation, etc. We use the describe method:

```
dataframe.describe()
```

This method will provide various summary statistics, excluding `NaN` (Not a Number) values.

```
df.describe()
```

This shows the statistical summary of all numeric-typed (int, float) columns.
For example, the attribute "symboling" has 205 counts, the mean value of this column is 0.83, the standard deviation is 1.25, the minimum value is -2, 25th percentile is 0, 50th percentile is 1, 75th percentile is 2, and the maximum value is 3.
However, what if we would also like to check all the columns including those that are of type object.

You can add an argument `include = "all"` inside the bracket. Let's try it again.

```
# describe all the columns in "df"
df.describe(include = "all")
```

Now, it provides the statistical summary of all the columns, including object-typed attributes.
We can now see how many unique values, which is the top value and the frequency of top value in the object-typed columns.
Some values in the table above show as "NaN", this is because those numbers are not available regarding a particular column type.
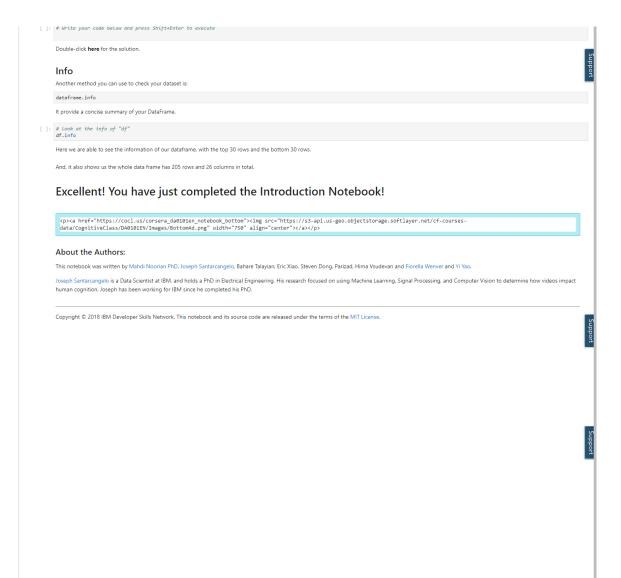

Question #3:

You can select the columns of a data frame by indicating the name of each column, for example, you can select the three columns as follows:

```
dataframe[[' column 1 ',column 2', 'column 3']]
```

Where "column" is the name of the column, you can apply the method ".describe()" to get the statistics of those columns as follows:

```
dataframe[[' column 1 ',column 2', 'column 3' ]].describe()
```

Apply the method to ".describe()" to the columns 'length' and 'compression-ratio'.

```
[ ]: # Write your code below and press Shift+Enter to execute
```

Double-click **here** for the solution.

Info

Another method you can use to check your dataset is:

```
dataframe.info
```

It provide a concise summary of your DataFrame.

```
[ ]: # look at the info of "df"
     df.info
```

Here we are able to see the information of our dataframe, with the top 30 rows and the bottom 30 rows.

And, it also shows us the whole data frame has 205 rows and 26 columns in total.

Excellent! You have just completed the Introduction Notebook!

```
<p><a href="https://cocl.us/corsera_da0101en_notebook_bottom"><img src="https://s3-api.us-geo.objectstorage.softlayer.net/cf-courses-data/CognitiveClass/DA0101EN/Images/BottomAd.png" width="750" align="center"></a></p>
```

About the Authors:

This notebook was written by Mahdi Noorian PhD, Joseph Santarcangelo, Bahare Talayian, Eric Xiao, Steven Dong, Parizad, Hima Vsudevan and Fiorella Wenver and Yi Yao.

Joseph Santarcangelo is a Data Scientist at IBM, and holds a PhD in Electrical Engineering. His research focused on using Machine Learning, Signal Processing, and Computer Vision to determine how videos impact human cognition. Joseph has been working for IBM since he completed his PhD.