Importing Dataset

Estimated time needed: **15** minutes

Objectives

After completing this lab you will be able to:

- Acquire data in various ways
- Obtain insights from data with Pandas library

Table of Contents

1. Data Acquisition
2. Basic Insights from the Data set

Data Acquisition

A data set is typically a file containing data stored in one of several formats. Common file formats containing data sets include: .csv, .json, .xlsx etc. The data set can be stored in different places, on your local machine, on a server or a websiite, cloud storage and so on.

To analyse data in a Python notebook, we need to bring the data set into the notebook. In this section, you will learn how to load a data set into our Jupyter Notebook.

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

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

The Pandas Library is a very popular and very 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.

In [ ]:
```
# uncomment the lines below if you need to install specific version of libraries if using this notebook
# in an environment where these libraries are not installed
#! mamba install pandas==1.3.3  -y
#! mamba install numpy=1.21.2 -y
```

In [ ]:
```
# import pandas library
import pandas as pd
import numpy as np
```

Read Data

We utilize the `pandas.read_csv()` function for reading CSV files. However, in this version of the lab, which operates on JupyterLite, the dataset needs to be downloaded to the interface using the provided code below.

The functions below will download the dataset into your browser:

In [2]:
```
from pyodide.http import pyfetch

async def download(url, filename):
    response = await pyfetch(url)
    if response.status == 200:
        with open(filename, "wb") as f:
            f.write(await response.bytes())
```

In [3]:
```
file_path='https://cf-courses-data.s3.us.cloud-object-storage.appdomain.cloud/IBMDeveloperSkillsNetwork-DA0101EN-SkillsNetwork/labs/Data%20files/aut
```

To obtain the dataset, utilize the download() function as defined above:

In [4]:
```
await download(file_path, "auto.csv")
file_name="auto.csv"
```

Utilize the Pandas method `read_csv()` to load the data into a dataframe.

In [5]:
```
df = pd.read_csv(file_name)
```

> Note: This version of the lab is working on JupyterLite, which requires the dataset to be downloaded to the interface.While working on the downloaded version of this notebook on their local machines(Jupyter Anaconda), the learners can simply **skip the steps above,** and simply use the URL directly in the `pandas.read_csv()` function. You can uncomment and run the statements in the cell below.

In [ ]:
```
#filepath = "https://cf-courses-data.s3.us.cloud-object-storage.appdomain.cloud/IBMDeveloperSkillsNetwork-DA0101EN-SkillsNetwork/labs/Data%20files/a
#df = pd.read_csv(filepath, header=None)
```

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

In [6]:
```
# show the first 5 rows using dataframe.head() method
print("The first 5 rows of the dataframe")
df.head(5)
```

```
The first 5 rows of the dataframe
```

Out[6]:

	3	?	alfa-romero	gas	std	two	convertible	rwd	front	88.6	...	130	mpfi	3.47	2.68	9.0	111	5000	21	27	13495
0	3	?	alfa-romero	gas	std	two	convertible	rwd	front	88.6	...	130	mpfi	3.47	2.68	9.0	111	5000	21	27	16500
1	1	?	alfa-romero	gas	std	two	hatchback	rwd	front	94.5	...	152	mpfi	2.68	3.47	9.0	154	5000	19	26	16500
2	2	164	audi	gas	std	four	sedan	fwd	front	99.8	...	109	mpfi	3.19	3.40	10.0	102	5500	24	30	13950
3	2	164	audi	gas	std	four	sedan	4wd	front	99.4	...	136	mpfi	3.19	3.40	8.0	115	5500	18	22	17450
4	2	?	audi	gas	std	two	sedan	fwd	front	99.8	...	136	mpfi	3.19	3.40	8.5	110	5500	19	25	15250

5 rows × 26 columns

Question #1:

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

In [7]:
```
# Write your code below and press Shift+Enter to execute
df.tail(10)
```

Out[7]:

	3	?	alfa-romero	gas	std	two	convertible	rwd	front	88.6	...	130	mpfi	3.47	2.68	9.0	111	5000	21	27	13495
194	-1	74	volvo	gas	std	four	wagon	rwd	front	104.3	...	141	mpfi	3.78	3.15	9.5	114	5400	23	28	13415
195	-2	103	volvo	gas	std	four	sedan	rwd	front	104.3	...	141	mpfi	3.78	3.15	9.5	114	5400	24	28	15985
196	-1	74	volvo	gas	std	four	wagon	rwd	front	104.3	...	141	mpfi	3.78	3.15	9.5	114	5400	24	28	16515
197	-2	103	volvo	gas	turbo	four	sedan	rwd	front	104.3	...	130	mpfi	3.62	3.15	7.5	162	5100	17	22	18420
198	-1	74	volvo	gas	turbo	four	wagon	rwd	front	104.3	...	130	mpfi	3.62	3.15	7.5	162	5100	17	22	18950
199	-1	95	volvo	gas	std	four	sedan	rwd	front	109.1	...	141	mpfi	3.78	3.15	9.5	114	5400	23	28	16845
200	-1	95	volvo	gas	turbo	four	sedan	rwd	front	109.1	...	141	mpfi	3.78	3.15	8.7	160	5300	19	25	19045
201	-1	95	volvo	gas	std	four	sedan	rwd	front	109.1	...	173	mpfi	3.58	2.87	8.8	134	5500	18	23	21485
202	-1	95	volvo	diesel	turbo	four	sedan	rwd	front	109.1	...	145	idi	3.01	3.40	23.0	106	4800	26	27	22470
203	-1	95	volvo	gas	turbo	four	sedan	rwd	front	109.1	...	141	mpfi	3.78	3.15	9.5	114	5400	19	25	22625

10 rows × 26 columns

▸ Click here for the solution

Add Headers

Take a look at the data set. Pandas automatically set the header with an integer starting from 0.

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

Thus, you have to add headers manually.

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

In [8]:
```
# 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)
```

```
headers
 ['symboling', 'normalized-losses', 'make', 'fuel-type', 'aspiration', 'num-of-doors', 'body-style', 'drive-wheels', 'engine-location', 'wheel-bas
e', 'length', 'width', 'height', 'curb-weight', 'engine-type', 'num-of-cylinders', 'engine-size', 'fuel-system', 'bore', 'stroke', 'compression-rati
o', 'horsepower', 'peak-rpm', 'city-mpg', 'highway-mpg', 'price']
```

Replace headers and recheck our data frame:

In [9]:
```
df.columns = headers
df.columns
```

Out[9]:
```
Index(['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'],
      dtype='object')
```

You can also see the first 10 entries of the updated data frame and note that the headers are updated.

In [10]:
```
df.head(10)
```

Out[10]:

	symboling	normalized-losses	make	fuel-type	aspiration	num-of-doors	body-style	drive-wheels	engine-location	wheel-base	...	engine-size	fuel-system	bore	stroke	compression-ratio	horsepower
0	3	?	alfa-romero	gas	std	two	convertible	rwd	front	88.6	...	130	mpfi	3.47	2.68	9.0	111
1	1	?	alfa-romero	gas	std	two	hatchback	rwd	front	94.5	...	152	mpfi	2.68	3.47	9.0	154
2	2	164	audi	gas	std	four	sedan	fwd	front	99.8	...	109	mpfi	3.19	3.40	10.0	102
3	2	164	audi	gas	std	four	sedan	4wd	front	99.4	...	136	mpfi	3.19	3.40	8.0	115
4	2	?	audi	gas	std	two	sedan	fwd	front	99.8	...	136	mpfi	3.19	3.40	8.5	110
5	1	158	audi	gas	std	four	sedan	fwd	front	105.8	...	136	mpfi	3.19	3.40	8.5	110
6	1	?	audi	gas	std	four	wagon	fwd	front	105.8	...	136	mpfi	3.19	3.40	8.5	110
7	1	158	audi	gas	turbo	four	sedan	fwd	front	105.8	...	131	mpfi	3.13	3.40	8.3	140
8	0	?	audi	gas	turbo	two	hatchback	4wd	front	99.5	...	131	mpfi	3.13	3.40	7.0	160
9	2	192	bmw	gas	std	two	sedan	rwd	front	101.2	...	108	mpfi	3.50	2.80	8.8	101

10 rows × 26 columns

Now, we need to replace the "?" symbol with NaN so the dropna() can remove the missing values:

In [11]:
```
df1=df.replace('?',np.NaN)
```

You can drop missing values along the column "price" as follows:

In [12]:
```
df=df1.dropna(subset=["price"], axis=0)
df.head(20)
```

Out[12]:

	symboling	normalized-losses	make	fuel-type	aspiration	num-of-doors	body-style	drive-wheels	engine-location	wheel-base	...	engine-size	fuel-system	bore	stroke	compression-ratio	horsepow
0	3	NaN	alfa-romero	gas	std	two	convertible	rwd	front	88.6	...	130	mpfi	3.47	2.68	9.00	1
1	1	NaN	alfa-romero	gas	std	two	hatchback	rwd	front	94.5	...	152	mpfi	2.68	3.47	9.00	1
2	2	164	audi	gas	std	four	sedan	fwd	front	99.8	...	109	mpfi	3.19	3.40	10.00	1
3	2	164	audi	gas	std	four	sedan	4wd	front	99.4	...	136	mpfi	3.19	3.40	8.00	1
4	2	NaN	audi	gas	std	two	sedan	fwd	front	99.8	...	136	mpfi	3.19	3.40	8.50	1
5	1	158	audi	gas	std	four	sedan	fwd	front	105.8	...	136	mpfi	3.19	3.40	8.50	1
6	1	NaN	audi	gas	std	four	wagon	fwd	front	105.8	...	136	mpfi	3.19	3.40	8.50	1
7	1	158	audi	gas	turbo	four	sedan	fwd	front	105.8	...	131	mpfi	3.13	3.40	8.30	1
9	2	192	bmw	gas	std	two	sedan	rwd	front	101.2	...	108	mpfi	3.50	2.80	8.80	1
10	0	192	bmw	gas	std	four	sedan	rwd	front	101.2	...	108	mpfi	3.50	2.80	8.80	1
11	0	188	bmw	gas	std	two	sedan	rwd	front	101.2	...	164	mpfi	3.31	3.19	9.00	1
12	0	188	bmw	gas	std	four	sedan	rwd	front	101.2	...	164	mpfi	3.31	3.19	9.00	1
13	1	NaN	bmw	gas	std	four	sedan	rwd	front	103.5	...	164	mpfi	3.31	3.19	9.00	1
14	0	NaN	bmw	gas	std	four	sedan	rwd	front	103.5	...	209	mpfi	3.62	3.39	8.00	1
15	0	NaN	bmw	gas	std	two	sedan	rwd	front	103.5	...	209	mpfi	3.62	3.39	8.00	1
16	0	NaN	bmw	gas	std	four	sedan	rwd	front	110.0	...	209	mpfi	3.62	3.39	8.00	1
17	2	121	chevrolet	gas	std	two	hatchback	fwd	front	88.4	...	61	2bbl	2.91	3.03	9.50	
18	1	98	chevrolet	gas	std	two	hatchback	fwd	front	94.5	...	90	2bbl	3.03	3.11	9.60	
19	0	81	chevrolet	gas	std	four	sedan	fwd	front	94.5	...	90	2bbl	3.03	3.11	9.60	
20	1	118	dodge	gas	std	two	hatchback	fwd	front	93.7	...	90	2bbl	2.97	3.23	9.41	

20 rows × 26 columns

Here, `axis=0` means that the contents along the entire row will be dropped wherever the entity 'price' is found to be NaN

Now, you have successfully read the raw data set and added the correct headers into the data frame.

Question #2:

Find the name of the columns of the dataframe.

In [14]:
```
# Write your code below and press Shift+Enter to execute
print(df.columns)
```

```
Index(['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'],
      dtype='object')
```

▸ Click here for the solution

Save Dataset

Correspondingly, Pandas enables you to save the data set 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 save the data frame **df** as **automobile.csv** to your local machine, you may use the syntax below, where `index = False` means the row names will not be written.

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

You can also read and save other file formats. You can use similar functions like **`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

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()`
...	...	...

Basic Insights from the Data set

After reading data into Pandas dataframe, it is time for you to explore the data set.

There are several ways to obtain essential insights of the data to help you better understand it.

Data Types

Data has a variety of types.

The main types stored in Pandas data frames are **object**, **float**, **int**, **bool** and **datetime64**. In order to better learn about each attribute, you should always know the data type of each column. In Pandas:

```
In [15]: df.dtypes
```

```
Out[15]: symboling              int64
         normalized-losses     object
         make                  object
         fuel-type             object
         aspiration            object
         num-of-doors          object
         body-style            object
         drive-wheels          object
         engine-location       object
         wheel-base           float64
         length               float64
         width                float64
         height               float64
         curb-weight            int64
         engine-type           object
         num-of-cylinders      object
         engine-size            int64
         fuel-system           object
         bore                  object
         stroke                object
         compression-ratio    float64
         horsepower            object
         peak-rpm              object
         city-mpg               int64
         highway-mpg            int64
         price                 object
         dtype: object
```

Returns a series with the data type of each column.

As shown above, you can clearly 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; you 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., use the describe method:

dataframe.describe()

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

```
In [17]: df.describe()
```

Out[17]:

	symboling	wheel-base	length	width	height	curb-weight	engine-size	compression-ratio	city-mpg	highway-mpg
count	200.000000	200.000000	200.000000	200.000000	200.000000	200.000000	200.000000	200.000000	200.000000	200.000000
mean	0.830000	98.848000	174.228000	65.898000	53.791500	2555.705000	126.860000	10.170100	25.200000	30.705000
std	1.248557	6.038261	12.347132	2.102904	2.428449	518.594552	41.650501	4.014163	6.432487	6.827227
min	-2.000000	86.600000	141.100000	60.300000	47.800000	1488.000000	61.000000	7.000000	13.000000	16.000000
25%	0.000000	94.500000	166.675000	64.175000	52.000000	2163.000000	97.750000	8.575000	19.000000	25.000000
50%	1.000000	97.000000	173.200000	65.500000	54.100000	2414.000000	119.500000	9.000000	24.000000	30.000000
75%	2.000000	102.400000	183.500000	66.675000	55.525000	2928.250000	142.000000	9.400000	30.000000	34.000000
max	3.000000	120.900000	208.100000	72.000000	59.800000	4066.000000	326.000000	23.000000	49.000000	54.000000

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 you 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. Try it again.

In [18]:
```
# describe all the columns in "df"
df.describe(include = "all")
```

Out[18]:

	symboling	normalized-losses	make	fuel-type	aspiration	num-of-doors	body-style	drive-wheels	engine-location	wheel-base	...	engine-size	fuel-system	bore	stroke	compression-ratio	hors
count	200.000000	164	200	200	200	198	200	200	200	200.000000	...	200.000000	200	196	196	200.000000	
unique	NaN	51	22	2	2	2	5	3	2	NaN	...	NaN	8	38	36	NaN	
top	NaN	161	toyota	gas	std	four	sedan	fwd	front	NaN	...	NaN	mpfi	3.62	3.40	NaN	
freq	NaN	11	32	180	164	113	94	118	197	NaN	...	NaN	91	23	19	NaN	
mean	0.830000	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	98.848000	...	126.860000	NaN	NaN	NaN	10.170100	
std	1.248557	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	6.038261	...	41.650501	NaN	NaN	NaN	4.014163	
min	-2.000000	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	86.600000	...	61.000000	NaN	NaN	NaN	7.000000	
25%	0.000000	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	94.500000	...	97.750000	NaN	NaN	NaN	8.575000	
50%	1.000000	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	97.000000	...	119.500000	NaN	NaN	NaN	9.000000	
75%	2.000000	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	102.400000	...	142.000000	NaN	NaN	NaN	9.400000	
max	3.000000	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	120.900000	...	326.000000	NaN	NaN	NaN	23.000000	

11 rows × 26 columns

Now it provides the statistical summary of all the columns, including object-typed attributes.

YOu can now see how many unique values there, which one is the top value, and the frequency of the top value in the object-typed columns.

Some values in the table above show "NaN". Those numbers are not available regarding a particular column type.

Question #3:

You can select the columns of a dataframe 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'.

In [19]:
```
# Write your code below and press Shift+Enter to execute
df[['length','compression-ratio']].describe()
```

Out[19]:

	length	compression-ratio
count	200.000000	200.000000
mean	174.228000	10.170100
std	12.347132	4.014163
min	141.100000	7.000000
25%	166.675000	8.575000
50%	173.200000	9.000000
75%	183.500000	9.400000
max	208.100000	23.000000

▸ Click here for the solution

Info

You can also use another method to check your data set:

dataframe.info()

It provides a concise summary of your data frame.

This method prints information about a data frame including the index dtype and columns, non-null values and memory usage.

In [20]:
```
# look at the info of "df"
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Index: 200 entries, 0 to 203
Data columns (total 26 columns):
 #   Column             Non-Null Count  Dtype  
---  ------             --------------  -----  
 0   symboling          200 non-null    int64  
 1   normalized-losses  164 non-null    object 
 2   make               200 non-null    object 
 3   fuel-type          200 non-null    object 
 4   aspiration         200 non-null    object 
 5   num-of-doors       198 non-null    object 
 6   body-style         200 non-null    object 
 7   drive-wheels       200 non-null    object 
 8   engine-location    200 non-null    object 
 9   wheel-base         200 non-null    float64
 10  length             200 non-null    float64
 11  width              200 non-null    float64
 12  height             200 non-null    float64
 13  curb-weight        200 non-null    int64  
 14  engine-type        200 non-null    object 
 15  num-of-cylinders   200 non-null    object 
 16  engine-size        200 non-null    int64  
 17  fuel-system        200 non-null    object 
 18  bore               196 non-null    object 
 19  stroke             196 non-null    object 
 20  compression-ratio  200 non-null    float64
 21  horsepower         198 non-null    object 
 22  peak-rpm           198 non-null    object 
 23  city-mpg           200 non-null    int64  
 24  highway-mpg        200 non-null    int64  
 25  price              200 non-null    object 
dtypes: float64(5), int64(5), object(16)
memory usage: 29.7+ KB
```

Excellent! You have just completed the Introduction Notebook.

Thank you for completing this lab.

Author

Joseph Santarcangelo

Other Contributors

Mahdi Noorian PhD

Bahare Talayian

Eric Xiao

Steven Dong

Parizad