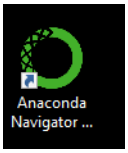
Lab1 – Data Acquisition and Exploration

In this lab, you will learn how to perform data acquisition to explore the dataset with Python Pandas library. By the end of this lab, you will successfully load the dataset into Jupyter Notebook, explore the dataset imported to gain some fundamental insights. The dataset named as auto.csv used in this lab.

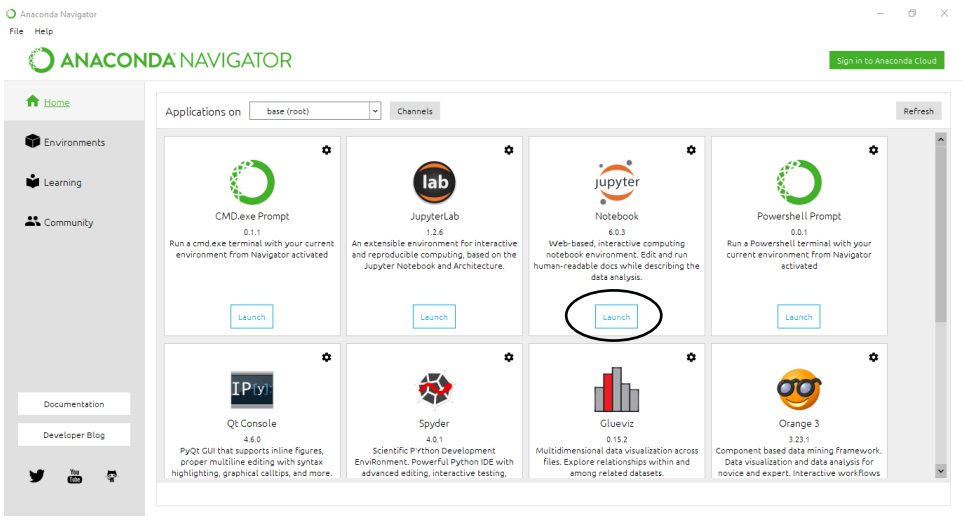
Software: Jupyter Notebook

Procedure

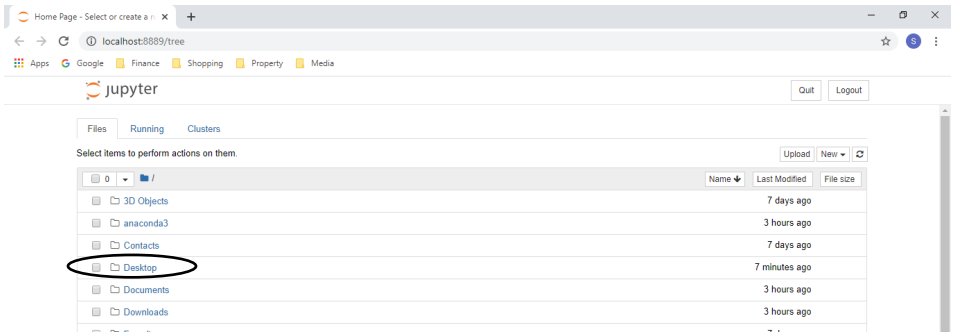
1. Launch the Anaconda in the desktop.



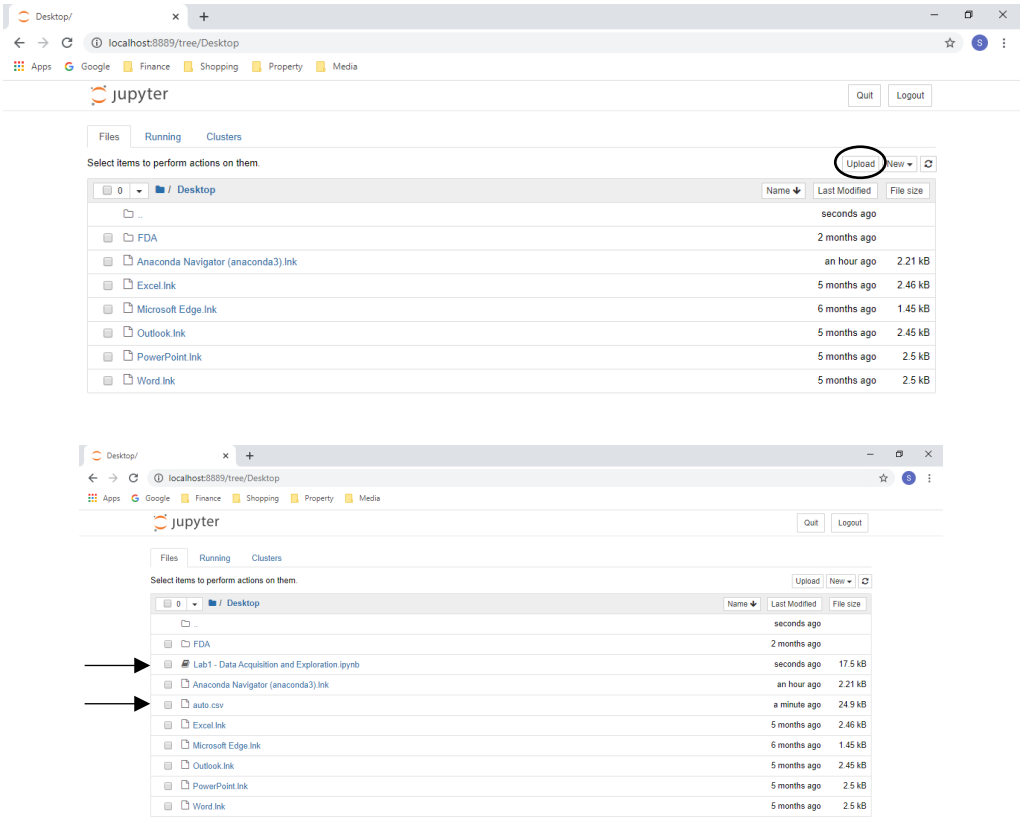
2. In the Anaconda Navigator launch the Jupyter Notebook.



3. Click on the Desktop directory to navigate there.



4. In the desktop directory, upload the auto.csv and Lab1 - Data Acquisition and Exploration – Participant Copy.ipynb) to the Jupyter Notebook.



5. Click the Lab1- Data Acquisition and Exploration – Participant Copy.ipynb file to start the lab.

Data Acquisition

Dataset can come in various formats (.csv, .json, .xlsx etc) and it can be located in your computer or sometimes online. You will learn how to load a dataset to Jupyter Notebook. The auto.csv is the dataset used in this lab, and it is in CSV (comma separated value) format.

The Pandas Library is a useful tool that enables us to read various datasets into a data frame. The Jupyter notebook platforms have a built-in Pandas Library to enable us to import Pandas without installing.

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

Read Data

We use `pandas.read_csv()` 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. Assign the dataset to the local file address.

In [ ]:
```
# import pandas library

import pandas as pd

# Read the file (auto.csv), and assign it to variable "df"

other_path = 'auto.csv'

df = pd.read_csv(other_path, header=None)
```

We use the `dataframe.head(n)` 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.

In [ ]:
```
# show the first 5 rows using dataframe.head() method

print("The first 5 rows of the dataframe")

df.head(5)
```

Out [ ]:

The first 5 rows of the dataframe

	0	1	2	3	4	5	6	7	8	9	...	16	17	18	19	20	21	22	23	24	25
0	3	?	alfa-romero	gas	std	two	convertible	rwd	front	88.6	...	130	mpfi	3.47	2.68	9.0	111	5000	21	27	13495
1	3	?	alfa-romero	gas	std	two	convertible	rwd	front	88.6	...	130	mpfi	3.47	2.68	9.0	111	5000	21	27	16500
2	1	?	alfa-romero	gas	std	two	hatchback	rwd	front	94.5	...	152	mpfi	2.68	3.47	9.0	154	5000	19	26	16500
3	2	164	audi	gas	std	four	sedan	fwd	front	99.8	...	109	mpfi	3.19	3.40	10.0	102	5500	24	30	13950
4	2	164	audi	gas	std	four	sedan	4wd	front	99.4	...	136	mpfi	3.19	3.40	8.0	115	5500	18	22	17450

5 rows × 26 columns

Exercise 1.1

Write the Python code to view the bottom 10 rows of the data fame.

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

Record the output below:

Out [ ]:	

Add Headers

From the dataset; we can see that pandas automatically set the header by an integer from 0. We can add a header manually by creating a list *headers* first. Then use *dataframe.columns = headers* to replace the *headers* by the list we created.

In [ ]:	`#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)`  `df.columns = headers`  `df.head(25)`

Record the output below:

Out [ ]:	

Exercise 1.2

Find the name of the columns of the dataframe:

Write the code to print the name of the columns in the dataframe

In [ ]:	`# Write your code below and press shift + enter to execute`
Out [ ]:	`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')`

Data Types

There are various types of data. The main types stored in Pandas dataframes are object, float, int, bool and datetime64. In order to better learn all these data types, it is better to print the data type of each column.

In [ ]:	`print(df.dtype)`

Out [ ]:

```
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
```

Another method you can use to check your dataset is `df.info()` which provides a concise summary of the dataframe.

In [ ]:

```
df.info()
```

Out[ ]:

```
<bound method DataFrame.info of      symboling normalized-losses         make fuel-type aspiration  \
0            3                 ?  alfa-romero       gas        std
1            3                 ?  alfa-romero       gas        std
2            1                 ?  alfa-romero       gas        std
3            2               164         audi       gas        std
4            2               164         audi       gas        std
..         ...               ...          ...       ...        ...
200         -1                95        volvo       gas        std
201         -1                95        volvo       gas      turbo
202         -1                95        volvo       gas        std
203         -1                95        volvo    diesel      turbo
204         -1                95        volvo       gas      turbo

    num-of-doors   body-style drive-wheels engine-location  wheel-base  ...  \
0            two  convertible          rwd           front        88.6  ...
1            two  convertible          rwd           front        88.6  ...
2            two    hatchback          rwd           front        94.5  ...
3           four        sedan          fwd           front        99.8  ...
4           four        sedan          4wd           front        99.4  ...
..           ...          ...          ...             ...         ...  ...
200         four        sedan          rwd           front       109.1  ...
201         four        sedan          rwd           front       109.1  ...
202         four        sedan          rwd           front       109.1  ...
203         four        sedan          rwd           front       109.1  ...
204         four        sedan          rwd           front       109.1  ...

     engine-size  fuel-system  bore  stroke compression-ratio horsepower  \
0            130         mpfi  3.47    2.68               9.0        111
1            130         mpfi  3.47    2.68               9.0        111
2            152         mpfi  2.68    3.47               9.0        154
3            109         mpfi  3.19    3.40              10.0        102
```

We are able to see the information of our data frame, with the top 5 rows and the bottom 5 rows. and, it also shows us the whole dataframe has 205 rows and 26 columns in total.

Describe()

In order to get a statistical summary of each column (count, mean, standard deviation, etc), we use the describe method `df.describe()` which will provide various summary statistics excluding `NaN` values.

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

# Use df.describe('include=all') to provide the statistical

# summary of all the column including object type

#df.describe(include='all')
```

Out [ ]:

	symboling	wheel-base	length	width	height	curb-weight	engine-size	compression-ratio	city-mpg	highway-mpg
count	205.000000	205.000000	205.000000	205.000000	205.000000	205.000000	205.000000	205.000000	205.000000	205.000000
mean	0.834146	98.756585	174.049268	65.907805	53.724878	2555.565854	126.907317	10.142537	25.219512	30.751220
std	1.245307	6.021776	12.337289	2.145204	2.443522	520.680204	41.642693	3.972040	6.542142	6.886443
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	166.300000	64.100000	52.000000	2145.000000	97.000000	8.600000	19.000000	25.000000
50%	1.000000	97.000000	173.200000	65.500000	54.100000	2414.000000	120.000000	9.000000	24.000000	30.000000
75%	2.000000	102.400000	183.100000	66.900000	55.500000	2935.000000	141.000000	9.400000	30.000000	34.000000
max	3.000000	120.900000	208.100000	72.300000	59.800000	4066.000000	326.000000	23.000000	49.000000	54.000000

Out [ ]:

	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	ho
count	205.000000	205	205	205	205	205	205	205	205	205.000000	...	205.000000	205	205	205	205.000000	
unique	NaN	52	22	2	2	3	5	3	2	NaN	...	NaN	8	39	37	NaN	
top	NaN	?	toyota	gas	std	four	sedan	fwd	front	NaN	...	NaN	mpfi	3.62	3.40	NaN	
freq	NaN	41	32	185	168	114	96	120	202	NaN	...	NaN	94	23	20	NaN	
mean	0.834146	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	98.756585	...	126.907317	NaN	NaN	NaN	10.142537	
std	1.245307	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	6.021776	...	41.642693	NaN	NaN	NaN	3.972040	
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	...	97.000000	NaN	NaN	NaN	8.600000	
50%	1.000000	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	97.000000	...	120.000000	NaN	NaN	NaN	9.000000	
75%	2.000000	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	102.400000	...	141.000000	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

Exercise 1.3

You can select the columns of a data frame by indicating the name of each column, for example, you can select the three columns and use the *describe()* to get the statistics of the columns of your interest: *df[[' column 1 ',column 2', 'column 3']].describe()*

Apply *.describe()* to the columns 'length', *'width'* and *'height'*.

Write the Python code to describe 'length', 'width' and 'height' of the datafame.

In [ ]:
```
# Write your code below and press shift + enter to execute
```

Record the output below:

Out [ ]:	

--- End of Lab1 ---