HW - Essential python for Data Analyst

- **HW1** : Request public API
- **HW2** : Simple ML model with sklearn

HW1 : Request public API

```
import requests
import time
import pandas as pd
```

```
# check url
url = "https://www.thecocktaildb.com/api/json/v1/1/search.php?f=a"
resp = requests.get(url)
result_api = resp.json()

print(result_api['drinks'][1])
print(len(result_api['drinks']))
```

```
{'idDrink': '13501', 'strDrink': 'ABC', 'strDrinkAlternate': None, 'strTags': None, 'strVideo': None, 'strCategory': 'Shot', 'strIBA':
25
```

```
#### Get List the first 10 cocktails of each first letter.
### choose cocktails name start with a, b and c

idDrink = []
strDrink = []
strCategory = []
strGlass = []
strInstructions = []

for i in ['a','b','c']:
  url = f"https://www.thecocktaildb.com/api/json/v1/1/search.php?f={i}"
  resp = requests.get(url)
  result_api = resp.json()
  for j in range(1,11):
    idDrink.append(result_api['drinks'][j]['idDrink'])
    strDrink.append(result_api['drinks'][j]['strDrink'])
    strCategory.append(result_api['drinks'][j]['strCategory'])
    strGlass.append(result_api['drinks'][j]['strGlass'])
    strInstructions.append(result_api['drinks'][j]['strInstructions'])
    time.sleep(1) # save server by taking 2 seconds before select the next one

df_cocktails = pd.DataFrame({
    'id' : idDrink,
    'name' : strDrink,
    'category' : strCategory,
    'glass' : strGlass,
    'instructions' : strInstructions
})

df_cocktails
```

	id	name	category	glass	instructions
0	13501	ABC	Shot	Shot glass	Layered in a shot glass.
1	17225	Ace	Cocktail	Martini Glass	Shake all the ingredients in a cocktail shaker...
2	14610	ACID	Shot	Shot glass	Poor in the 151 first followed by the 101 serv...
3	17837	Adam	Ordinary Drink	Cocktail glass	In a shaker half-filled with ice cubes, combin...
4	13938	AT&T	Ordinary Drink	Highball Glass	Pour Vodka and Gin over ice, add Tonic and Stir
5	17833	A. J.	Ordinary Drink	Cocktail glass	Shake ingredients with ice, strain into a cock...
6	17839	Affair	Ordinary Drink	Highball glass	Pour schnapps, orange juice, and cranberry jui...
7	15106	Apello	Other/Unknown	Collins Glass	Stirr. Grnish with maraschino cherry.
8	15266	Avalon	Ordinary Drink	Highball glass	Fill a tall glass with ice. Layer the Finlandi...
9	17835	Abilene	Ordinary Drink	Highball glass	Pour all of the ingredients into a highball gl...
10	13332	B-53	Shot	Collins Glass	Layer the Kahlua, Sambucca and Grand Marnier i...
11	17254	Bijou	Cocktail	Cocktail glass	Stir in mixing glass with ice and strain\r\n
12	11149	Boxcar	Ordinary Drink	Whiskey sour glass	In a shaker half-filled with ice cubes, combin...
13	13222	Big Red	Shot	Shot glass	Pour ingredients into 1 ounce shot glass
14	17195	Bellini	Ordinary Drink	Champagne Flute	Pour peach purée into chilled flute, add spark...
15	17210	Bramble	Ordinary Drink	Old-Fashioned glass	Fill glass with crushed ice. Build gin, lemon ...
16	11060	Balmoral	Ordinary Drink	Cocktail glass	In a mixing glass half-filled with ice cubes, ...
17	11120	Bluebird	Ordinary Drink	Cocktail glass	In a mixing glass half-filled with crushed ice...
18	178310	Brooklyn	Cocktail	Cocktail glass	Combine ingredients with ice and stir until we...
19	12572	Bora Bora	Cocktail	Highball glass	Prepare in a blender or shaker, serve in a hig...

HW1 analysis :

in df_cocktails dataframe,

- Q1 : How many cocktails in each category ?
- Q2 : Which one of glasses is the most ?
- Q3 : Are there 'shot' cocktail's categories ?

```
## Q1 : How many cocktails in each category ?
df_result1 = df_cocktails.groupby('category')['name'].agg('count').sort_values(ascending=False).reset_index()
df_result1.rename(columns = {'name':'count'}, inplace = True)

df_result1
```

	category	count
0	Ordinary Drink	17
1	Cocktail	6
2	Shot	4
3	Coffee / Tea	1
4	Other/Unknown	1
5	Soft Drink	1

```
## Q2 : Which one of glasses is the most ?
df_result2 = df_cocktails.groupby('glass')['name'].agg('count').sort_values(ascending=False).reset_index()
df_result2.rename(columns = {'name':'count'}, inplace = True)

df_result2.head(1)
```

	glass	count
0	Cocktail glass	9

```
## Q3 : Are there 'Shot' cocktail's categories ?
df_result3 = df_cocktails.loc[df_cocktails['category'] == 'Shot']

df_result3
```

	id	name	category	glass	instructions
0	13501	ABC	Shot	Shot glass	Layered in a shot glass.
2	14610	ACID	Shot	Shot glass	Poor in the 151 first followed by the 101 serv...
10	13332	B-53	Shot	Collins Glass	Layer the Kahlua, Sambucca and Grand Marnier i...
13	13222	Big Red	Shot	Shot glass	Pour ingredients into 1 ounce shot glass

HW2 : Simple ML Model with sklearn

from "mtcars" dataset

```
## read and preview data
mtcars = pd.read_csv("https://gist.githubusercontent.com/seankross/a412dfbd88b3db70b74b/raw/5f23f993cd87c283ce766e7ac6b329ee7cc2e1d1/mtcars.c
mtcars.head()
```

	model	mpg	cyl	disp	hp	drat	wt	qsec	vs	am	gear	carb
0	Mazda RX4	21.0	6	160.0	110	3.90	2.620	16.46	0	1	4	4
1	Mazda RX4 Wag	21.0	6	160.0	110	3.90	2.875	17.02	0	1	4	4
2	Datsun 710	22.8	4	108.0	93	3.85	2.320	18.61	1	1	4	1
3	Hornet 4 Drive	21.4	6	258.0	110	3.08	3.215	19.44	1	0	3	1
4	Hornet Sportabout	18.7	8	360.0	175	3.15	3.440	17.02	0	0	3	2

```
## find correlation
corr = mtcars.corr()
corr
```

	mpg	cyl	disp	hp	drat	wt	qsec	vs	am	g
mpg	1.000000	-0.852162	-0.847551	-0.776168	0.681172	-0.867659	0.418684	0.664039	0.599832	0.480
cyl	-0.852162	1.000000	0.902033	0.832447	-0.699938	0.782496	-0.591242	-0.810812	-0.522607	-0.492
disp	-0.847551	0.902033	1.000000	0.790949	-0.710214	0.887980	-0.433698	-0.710416	-0.591227	-0.555
hp	-0.776168	0.832447	0.790949	1.000000	-0.448759	0.658748	-0.708223	-0.723097	-0.243204	-0.125
drat	0.681172	-0.699938	-0.710214	-0.448759	1.000000	-0.712441	0.091205	0.440278	0.712711	0.699
wt	-0.867659	0.782496	0.887980	0.658748	-0.712441	1.000000	-0.174716	-0.554916	-0.692495	-0.583
qsec	0.418684	-0.591242	-0.433698	-0.708223	0.091205	-0.174716	1.000000	0.744535	-0.229861	-0.212
vs	0.664039	-0.810812	-0.710416	-0.723097	0.440278	-0.554916	0.744535	1.000000	0.168345	0.206
am	0.599832	-0.522607	-0.591227	-0.243204	0.712711	-0.692495	-0.229861	0.168345	1.000000	0.794
gear	0.480285	-0.492687	-0.555569	-0.125704	0.699610	-0.583287	-0.212682	0.206023	0.794059	1.000
carb	-0.550925	0.526988	0.394977	0.749812	-0.090790	0.427606	-0.656249	-0.569607	0.057534	0.274

1. Linear Regression

```
from sklearn.linear_model import LinearRegression
from sklearn.model_selection import train_test_split
import pandas as pd
import numpy as np
```

From correlation between each variables , to predict **hp** based on ...

- disp
- wt

```
## hp = f(disp,wt)

## prepare data
x = mtcars[["disp","wt"]]
y = mtcars["hp"]

# split data
x_train, x_test, y_train, y_test = train_test_split(
    x, y, test_size = 0.20, random_state = 33 #set.seed()
)
```

```
print(x_train.shape)
print(x_test.shape)
```

```
(25, 2)
(7, 2)
```

```
## predict new data (scoring)
# train model
model = LinearRegression()
model.fit(x_train, y_train)

# test model
p = model.predict(x_test)
print(p)
```

```
[196.82398288  80.7769203  102.57887161 143.65503636 222.74346743
 100.07148175 235.46602913]
```

```
## evaluate model
model.score(x_test,y_test)
```

```
0.6582630949816614
```

2. Logistic Regression

```
from sklearn.linear_model import LogisticRegression
```

From correlation between each variables , to predict **am** based on their other variables

```
## prepare data
x = mtcars[["mpg","drat","gear"]]
y = mtcars["am"]

# split data
x_train, x_test, y_train, y_test = train_test_split(
    x, y, test_size = 0.3, random_state = 55 #set.seed()
)
```

```
print(x_train.shape)
print(x_test.shape)
```

```
(22, 3)
(10, 3)
```

```
## predict new data (scoring)
# train model
glm_model = LogisticRegression()
glm_model.fit(x_train, y_train)

# test model
glm_p = glm_model.predict(x_test)
print(p)
```

```
[0 0 0 1 1 0 0]
```

```
## evaluate model
model.score(x_test,y_test)
```

```
1.0
```