Portfolio 1 – APIs and Data Cleaning

Qiaoyu Wang

0 Introduction

This week we looked at APIs, which could be a way we make requests for some specific pieces of data. When I worked as an AI product manager before, it always happened that we used the APIs from our collaborators' product for joint development and adjustment, which made me feel very interested in exploring APIs.

In class, I followed the notebook to use of API of The Metropolitan Museum of Art (https://metmuseum.github.io/) to explore some art. It worked well when I use the original code to find out some pictures involved 'child'. But when I tried to adjust the code to find some 'cats' pictures, it didn't work well. And also I tried 'dogs', 'columns', 'eyes', unfortunately I always got the same results. **After discussing with my groupmates, we thought that maybe it was this API's limitation, which couldn't offer us the results we wanted.**

1 Exploring new APIs

Then I started considering alternative APIs to explore some other datasets. Recently I have been obsessed with Plants vs. Zombies, which made me wonder if I could get the API to analyze plants and zombies. Luckily, I found it on the official website (https://pvz-2-api.vercel.app/docs) and it was free and open to use. I made a request

for it and got a plant list, as well as the information of the specific plant:

Details: {'name': 'Chili Bean', 'cost': 50, 'recharge': 20, 'Usage': 'When eaten', 'special': 'Eating zombie is destroyed and releases stunning gas', 'family': 'Ail-mint', 'description': 'Chili Beans deliver a crippling bout of gastrointestinal distress.', 'image': '/assets/plants/Chili Bean.png'}.

From this, I thought that it would be interesting if I could get the information of every plant and then make some data analysis. So I wrote a for-loop, using the plant list to make several API requests for each plant, and then covert them into a dataframe.

2 Cleaning the dataset

However, it was not going well as I thought it would be. Lots of problems appeared.

Firstly, the data was very messy. There were lots of same columns but in different case letters, such as **'Toughness' and 'toughness'**, which really needed to be combined.

```
df.columns
```

```
Index(['name', 'Sun cost', 'recharge', 'damage', 'range', 'family',
       'description', 'image', 'Recharge', 'Duration', 'Weakness', 'Family',
       'Toughness', 'Damage', 'Area', 'Range', 'Usage', 'Special', 'cost',
       'toughness', 'powerup', 'usage', 'special',
       '120 damage per shot (blue bulb)', 'Damage details', 'area', 'weakness',
       'Range details', 'sun-production'],
      dtype='object')
```

Secondly, I also found lots of **NAN** values emerged in different columns, which made these columns insufficient to be analyzed. So **I dropped lots of columns and left valid data.**

```python
nan_counts = new_df.isna().sum()
nan_counts
```

```
[30]
name              0
Sun cost          0
recharge          1
damage           61
description      93
image             0
Duration        125
Weakness        118
Family            9
Toughness        97
Area            105
Range             81
Usage             96
Special           77
powerup          133
Damage details   135
Range details    135
sun-production   136
dtype: int64
```

```python
nan_columns_to_drop = ['Duration', 'Weakness', 'Area', 'powerup', 'Damage details', 'Range details', 'sun-production', 'descriptio
plant_df = new_df.drop(columns=nan_columns_to_drop)
plant_df
```

✓ 0.0s

	name	Sun cost	recharge	damage	image	Family
0	A.K.E.E.	175	5.0	0/50/40/30 damage per shot (the first, second,...	/assets/plants/A.K.E.E..png	Arma-mint
1	Ail-mint	0	85 seconds	NaN	/assets/plants/Ail-mint.png	Ail-mint
2	Aloe	75	20 seconds	NaN	/assets/plants/Aloe.png	Reinforce-mint
3	Apple Mortar	250	8 seconds	30 damage per shot	/assets/plants/Apple Mortar.png	Arma-mint
4	Arma-mint	0	85 seconds	1200 dps	/assets/plants/Arma-mint.png	Arma-mint
...	...	...	...	...	...	...
132	Wasabi Whip	150	5 seconds	40 damage per shot	/assets/plants/Wasabi Whip.png	Enforce-mint
133	Winter Melon	500	5.0	80	/assets/plants/Winter Melon.png	Winter-mint
134	Winter-mint	0	85 seconds	NaN	/assets/plants/Winter-mint.png	Winter-mint
135	Witch Hazel	200	30 seconds	NaN	/assets/plants/Witch Hazel.png	Enchant-mint
136	Zoybean Pod	200	20 seconds	NaN	/assets/plants/Zoybean Pod.png	Enchant-mint

137 rows × 6 columns

Thirdly, the data types under the "Sun cost" and "Recharge" column were also different. We could see str, int, and float. To make them the same data type, **I used regex to clean them and just left integer for later analysis.**

```python
Recharge = df['recharge'].values
Recharge
```

```
array([5.0, '85 seconds', '20 seconds', '8 seconds', '85 seconds',
       '5 seconds', '10 seconds', 5.0, '5 seconds', 5.0, '85 seconds',
       5.0, '5', '5 seconds', 5.0, '45 seconds', '15 seconds', 15.0, 35.0,
       20.0, 5.0, 5.0, '5 seconds', '85 seconds', '85', '8.5 seconds',
       '5 seconds', '5 seconds', '10 seconds', 25.0, '15 seconds',
       '85 seconds', 15.0, '85 seconds', '85 seconds', '20 seconds',
       '10 seconds', '10 seconds', '85 seconds', '5 seconds',
       '20 seconds', '15 seconds', '5 seconds', '75 seconds', 30.0,
       '35 seconds', 10.0, '20 seconds', '12', '7 seconds', '20',
       '20 seconds', '15 seconds', '15 seconds', 10.0, 20.0, '5 seconds',
       15.0, '20 seconds', '20 seconds', '35 seconds', 5.0, '15 seconds',
       5.0, '15 seconds', 5.0, '5 seconds', '15 seconds', 5.0, nan, 5.0,
       '5 seconds', '10 seconds (Mediocre in China)', '10 seconds',
       '5 seconds', '7 seconds', '5 seconds', 5.0, '5', '15 seconds', 5.0,
       '85 seconds', 20.0, '30 seconds', 20.0, '60 seconds', '5 seconds',
       '5 seconds', '25 seconds', '5 seconds', 'Fast', '5 seconds',
       '20 seconds', '12 seconds', '10 seconds', 5.0, '85 seconds', 5.0,
       5.0, '5 seconds', '5 seconds,Fast (Chinese version)',
       '10 seconds (Mediocre in China)', '5', '15 seconds', 5.0, '20',
       '25 seconds', 5.0, 5.0, 5.0, '5 seconds', 20.0, '20 seconds', 20.0,
       '5 seconds', '10 seconds', 20.0, '20 seconds', '5 seconds', 5.0,
       '20 seconds', 20.0, '20 seconds', '20 seconds', 5.0, '60 seconds',
       '7 seconds', 10.0, '5 seconds', '5.5 seconds', 10.0, 20.0,
       '5 seconds', 5.0, '85 seconds', '30 seconds', '20 seconds'],
```

```python
import re

clean_suncost = []

for sun in Suncost:
    if isinstance(sun, str):
        clean_sun = re.sub(r',.*', '', re.sub(r'\(.*?\)', '', sun)).strip()
        if clean_sun.isdigit():
            clean_suncost.append(int(clean_sun))
        else:
            clean_suncost.append(clean_sun)
    elif isinstance(sun, int):
        clean_suncost.append(sun)

clean_suncost
```

✓ 0.0s

```
[175,
 0,
 75,
 250,
 0,
 500,
 175,
 175,
 150,
```

```python
plant_df['Sun cost'] = clean_suncost
plant_df['recharge'] = clean_recharge
plant_df
```

✓ 0.0s

	name	Sun cost	recharge	image	Family
0	A.K.E.E.	175	5.0	/assets/plants/A.K.E.E..png	Arma-mint
1	Ail-mint	0	85	/assets/plants/Ail-mint.png	Ail-mint
2	Aloe	75	20	/assets/plants/Aloe.png	Reinforce-mint
3	Apple Mortar	250	8	/assets/plants/Apple Mortar.png	Arma-mint
4	Arma-mint	0	85	/assets/plants/Arma-mint.png	Arma-mint
...	...	...	...	...	...
132	Wasabi Whip	150	5	/assets/plants/Wasabi Whip.png	Enforce-mint
133	Winter Melon	500	5.0	/assets/plants/Winter Melon.png	Winter-mint
134	Winter-mint	0	85	/assets/plants/Winter-mint.png	Winter-mint
135	Witch Hazel	200	30	/assets/plants/Witch Hazel.png	Enchant-mint
136	Zoybean Pod	200	20	/assets/plants/Zoybean Pod.png	Enchant-mint

137 rows × 5 columns

3 Analyzing the dataset

Finally, I finished the data cleaning and calculated the ‘mean’, ‘median’ and ‘mode’ numbers out. The analysis revealed that the mean sun cost required to purchase a new plant is approximately 119, while the median stands at 112.5. This slight deviation between the mean and median suggests a relatively balanced distribution of sun costs among the plants.

Regarding the recharge time, which denotes the duration for a seed packet to be replenished, the mean value is 20, with a median of 10. This indicates that while the mean suggests a moderate recharge time, the median highlights a significant portion of plants with shorter recharge periods.

```
mean = plant_df["Sun cost"].mean()
print(mean)
median = plant_df["Sun cost"].median()
print(median)
mode = plant_df["Sun cost"].mode()
print(mode)
```

```
118.56617647058823
112.5
0    150
Name: Sun cost, dtype: object
```

```
mean = plant_df["recharge"].mean()
print(mean)
median = plant_df["recharge"].median()
print(median)
mode = plant_df["recharge"].mode()
print(mode)
```

✓ 0.0s

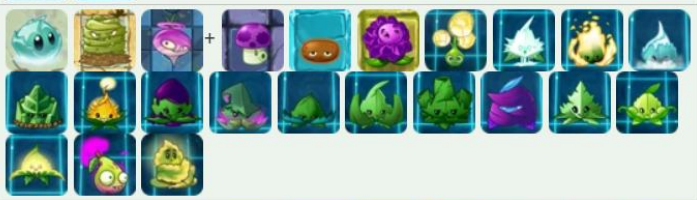
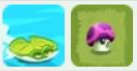
```
20.762962962962963
10.0
0    5.0
Name: recharge, dtype: object
```

What intrigued me was the discovery of **a mode value of 0 in the sun cost data**, which struck me as unusual and contradicted my memory! To investigate further, I decided to print out the details of all the plants with a zero-sun cost. This discrepancy prompted me to question the accuracy of the data I obtained and made me try to seek additional evidence to prove it.

	name	Sun cost	recharge	image	Family
1	Ail-mint	0	85	/assets/plants/Ail-mint.png	Ail-mint
4	Arma-mint	0	85	/assets/plants/Arma-mint.png	Arma-mint
10	Bombard-mint	0	85	/assets/plants/Bombard-mint.png	Bombard-mint
23	Conceal-mint	0	85	/assets/plants/Conceal-mint.png	Conceal-mint
24	Contain-mint	0	85	/assets/plants/Contain-mint.png	Contain-mint
31	Enchant-mint	0	85	/assets/plants/Enchant-mint.png	Enchant-mint
33	Enforce-mint	0	85	/assets/plants/Enforce-mint.png	Enforce-mint
34	Enlighten-mint	0	85	/assets/plants/Enlighten-mint.png	Enlighten-mint
38	Fila-mint	0	85	/assets/plants/Fila-mint.png	Fila-mint
43	Gold Bloom	0	75	/assets/plants/Gold Bloom.png	Enlighten-mint
46	Grave Buster	0	10.0	/assets/plants/Grave Buster.png	Contain-mint
54	Hot Potato	0	10.0	/assets/plants/Hot Potato.png	Pepper-mint
55	Iceberg Lettuce	0	20.0	/assets/plants/Iceberg Lettuce.png	Winter-mint
56	Imp Pear	0	5	/assets/plants/Imp Pear.png	Ail-mint
81	Pepper-mint	0	85	/assets/plants/Pepper-mint.png	Pepper-mint
91	Puff-shroom	0	5	/assets/plants/Puff-shroom.png	Ail-mint
96	Reinforce-mint	0	85	/assets/plants/Reinforce-mint.png	Reinforce-mint
105	Solar Sage	0	20	/assets/plants/Solar Sage.png	Enlighten-mint
113	Stallia	0	20.0	/assets/plants/Stallia.png	Contain-mint
127	Tile Turnip	0	10.0	/assets/plants/Tile Turnip.png	NaN
134	Winter-mint	0	85	/assets/plants/Winter-mint.png	Winter-mint

Upon searching for "plants that cost no sun," I discovered that many new plants were introduced in PvZs 2! Since I haven't fully explored this version yet, it became apparent why I was initially confused about encountering zero sun cost plants in the data.

These plants have a sun cost of 0 and therefore require no sun for planting.

Quick-search table	
Plants vs. Zombies	
Plants vs. Zombies 2	
Plants vs. Zombies 2 (Chinese version)	
Plants vs. Zombies Online	
Plants vs. Zombies Heroes	
Plants vs. Zombies 3	

(Image from:https://plantsvszombies.fandom.com/wiki/Category:Plants_that_cost_no_sun)

Let's also display the zero-sun cost plants' images for comparation. I downloaded the entire plant images folder from GitHub (https://github.com/code-cracked/plants-vs-zombies-api/tree/main) and extracted specific images whose filenames are listed in the "image" column of our Dataframe.

Seems like every piece of evidence can align together! Now I believed the real existence of each zero-cost plant and also the correctness of the data I got.

This whole process leaded me lots of questions to reflect:

When I get some data from API which looked very messy, is that necessary for me to doubt the data confidence? And if I doubt it, how can I deal with it?

I think it's essential to make sure that the data is reliable. We need to evaluate the reputation and credibility of the API provider, finding out some methods to implement data validation and verification, such as cross-referencing the data with other reliable sources.

If we do find a problem with the data, providing feedback to the API provider is a good choice. They may take steps to address data quality issues or provide

clarifications.

4 Conclusion

An API is like a bridge that connects different types of systems and facilitates seamless communication and data exchange between them. It acts as an intermediary layer that enables different software applications, services, and platforms to interact with each other, regardless of the programming languages or technologies they are built with. Exploring APIs indeed open the opportunities for me to have a rich conversation with corresponding developers, exchanging lots of knowledge.