Output Validation

What is our GOAL for this MODULE?

The goal of this module is to filter more exo-planets by applying statistics and machine learning.

What did we ACHIEVE in the class TODAY?

- Reviewed the output
- Debugged the code
- Fixed the code to get the right output

Which CONCEPTS/CODING BLOCKS did we cover today?

- Python
- Logical reasoning
- Debugging
- Exception Handling

How did we DO the activities?

1. In the last class, we came up with a dictionary containing all the planets and a list of their features (**gravity, type, goldilock and speed**), whether it supports it or not.
2. Find out the number of planets that have **gravity** listed as one of the features.

```
gravity_planet_count = 0
for key, value in final_dict.items():
 if "gravity" in value:
   gravity_planet_count += 1

print(gravity_planet_count)
```

3. The count comes out to be **3,951**. This is exactly the same as the number of planets we found out to have suitable gravity.
4. Find out the number of planets that have **planet_type** listed as one of the features.

```
type_planet_count = 0
for key, value in final_dict.items():
 if "planet_type" in value:
   type_planet_count += 1

print(type_planet_count)
```

5. Here, the count comes out to be **1,485** while we earlier got **1,452** when we found out the number of planets that are either **Terrestrial or Super Earth like**. That's because there are planets that do not support gravity but are of suitable types. Earlier, we only checked the type of planets that supported gravity.
6. Validate the same.

```
planet_not_gravity_support = []
for planet_data in planet_data_rows:
 if planet_data not in low_gravity_planets:
   planet_not_gravity_support.append(planet_data)

type_no_gravity_planet_count = 0
for planet_data in planet_not_gravity_support:
 if planet_data[6].lower() == "terrestrial" or
planet_data[6].lower() == "super earth":
```

```
    type_no_gravity_planet_count += 1

print(type_no_gravity_planet_count)
print(type_planet_count - type_no_gravity_planet_count)
```

7. We got the count for planets that do not support gravity but are of suitable type **33.** If we subtract 33 from **1,485,** we get **1,452** which matches with our findings earlier.
8. Find the number of planets that are in the **goldilock** zone and have suitable **speed** in the dictionary.

```
goldilock_planet_count = 0
for key, value in final_dict.items():
 if "goldilock" in value:
   goldilock_planet_count += 1

print(goldilock_planet_count)
```

```
speed_planet_count = 0
for key, value in final_dict.items():
 if "speed" in value:
   speed_planet_count += 1

print(speed_planet_count)
```

9. Planets in **Goldilock** zone came out to be **696** and those with suitable speed came out to be **8.**
10. Cross check if we are making any mistakes while creating the dictionary
11. If we revisit the code, we remember to make a few changes in columns. We are not performing any changes to any of the columns this time when we are creating a dictionary. We also did not handle **Unknown** values this time.
12. We are trying to add **goldilock** features to planets that are in the Goldilock zone, but it's in **X AU** format and we are not handling that. We have to split the string from a <space>, take the first element and convert it to float.

```
if float(planet_data[8].split(" ")[0]) > 0.38 and
float(planet_data[8].split(" ")[0]) < 2:
     features_list.append("goldilock")
```

13. We also have to fix the speed calculation. Again, we will fix the distance by converting it to float after we split it and we also have to convert **orbital_period** to days.

```
distance = 2 * 3.14 * (float(planet_data[8].split(" ")[0]) *
1.496e+9)
   time, unit = planet_data[9].split(" ")[0], planet_data[9].split("
")[1]
   if unit.lower() == "days":
     time = float(time)
   else:
     time = float(time) * 365
   time = time * 86400
   speed = distance / time
```

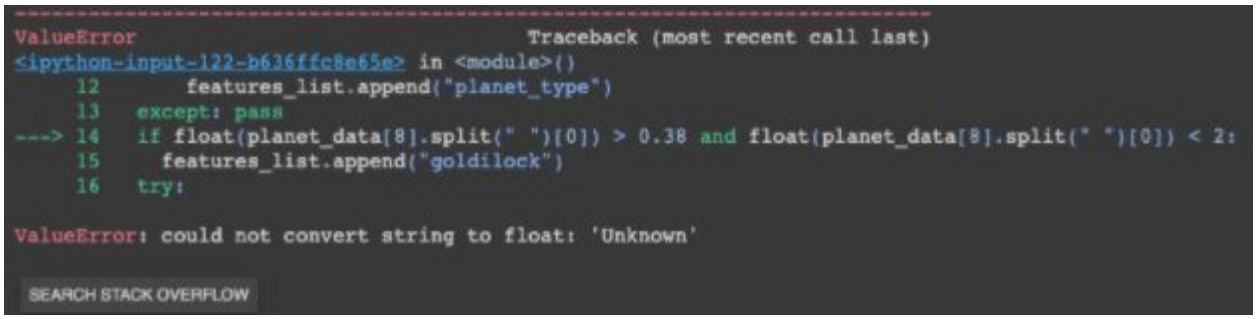
14. Now, our planets in the **Goldilock** zone were 25 and the planets that had suitable **speed** was 8. These planets also exhibited the properties of suitable gravity and planet type. Find out all such planets that exhibit all the properties.

```
goldilock_gravity_type_count = 0
for key, value in final_dict.items():
 if "goldilock" in value and "planet_type" in value and "gravity" in
value:
   goldilock_gravity_type_count += 1

print(goldilock_gravity_type_count)
```

```
speed_goldilock_gravity_type_count = 0
for key, value in final_dict.items():
 if "goldilock" in value and "planet_type" in value and "gravity" in
value and "speed" in value:
   speed_goldilock_gravity_type_count += 1

print(speed_goldilock_gravity_type_count)
```

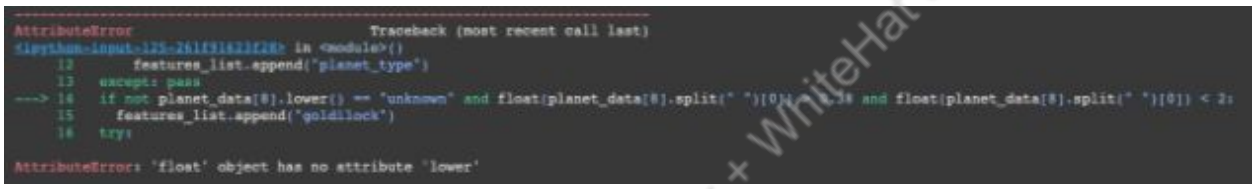
15. Here, the result is 0 for both of them. This does not match with the analysis result that we got earlier.
16. To debug this, remove the try-except statements from the code where we are adding

goldilock and speed properties and see what errors we are getting.



```
---------------------------------------------------------------------------
ValueError                                Traceback (most recent call last)
<ipython-input-122-b636ffc8e65e> in <module>()
     12         features_list.append("planet_type")
     13     except: pass
---> 14     if float(planet_data[8].split(" ")[0]) > 0.38 and float(planet_data[8].split(" ")[0]) < 2:
     15       features_list.append("goldilock")
     16     try:

ValueError: could not convert string to float: 'Unknown'

SEARCH STACK OVERFLOW
```

17. Handle the unknown values and re-run this.



```
---------------------------------------------------------------------------
AttributeError                            Traceback (most recent call last)
<ipython-input-125-261f91611f28> in <module>()
     12         features_list.append("planet_type")
     13     except: pass
---> 14     if not planet_data[8].lower() == "unknown" and float(planet_data[8].split(" ")[0]) > 0.38 and float(planet_data[8].split(" ")[0]) < 2:
     15       features_list.append("goldilock")
     16     try:

AttributeError: 'float' object has no attribute 'lower'
```

18. It now says that **float object has no attribute lower.** This means that there might be some values that are float. Fix this by handling these situations.

```
try:
  if float(planet_data[8].split(" ")[0]) > 0.38 and float(planet_data[8].split(" ")[0]) < 2:
    features_list.append("goldilock")
except:
  try:
    if planet_data[8] > 0.38 and planet_data[8] < 2:
      features_list.append("goldilock")
  except: pass
try:
  try:
    distance = 2 * 3.14 * (float(planet_data[8].split(" ")[0]) * 1.496e+9)
  except:
    try:
      distance = 2 * 3.14 * (float(planet_data[8]) * 1.496e+9)
    except: pass
  try:
    time, unit = planet_data[9].split(" ")[0], planet_data[9].split(" ")[1]
    if unit.lower() == "days":
      time = float(time)
    else:
      time = float(time) * 365
  except:
    time = planet_data[9]
  time = time * 86400
  speed = distance / time
  if speed < 200:
    features_list.append("speed")
```

19. Re-run the blocks that gave us the output 0 and 0 for planets supporting all the properties and this time, we shall get **25** for goldilock planets and **6** for planets with both goldilock and speed.
20. Awesome. The code has been successfully debugged.

What's NEXT?

In the next class, We will create a Flask API for our data so that we can integrate the API in a mobile application.

EXTEND YOUR KNOWLEDGE:

You can read the following blog on speed of our planet to understand more: https://www.safe.com/what-is/data-validation/#:~:text=Without%20validating%20your%20data%2C%20you,the%20data%20data%20model%20itself.