itba-tp0-creaci195179n-dataset

August 15, 2024

Introducción al Código de Simulación de Captura de Pokémon

El objetivo de este código es generar un dataset que será utilizado para analizar la probabilidad de captura de diferentes Pokémon en el notebook "ITBA_TP0_análisis_de_datos.ipynb".

El código permite simular capturas variando parámetros como el tipo de Pokémon, su nivel, HP, estado, y el tipo de pokébola utilizada. Se generan múltiples combinaciones de estos factores para construir un dataset extenso. Este dataset, que incluye tanto la probabilidad como la efectividad de captura, se guardará para su análisis posterior.

[ ]:
```
!git clone https://github.com/santire/sia-tp0
```

```
Cloning into 'sia-tp0'…
remote: Enumerating objects: 48, done.
remote: Counting objects: 100% (15/15), done.
remote: Compressing objects: 100% (8/8), done.
remote: Total 48 (delta 9), reused 7 (delta 7), pack-reused 33 (from 1)
Receiving objects: 100% (48/48), 171.69 KiB | 4.40 MiB/s, done.
Resolving deltas: 100% (19/19), done.
```

[ ]:
```
%cd 'sia-tp0'
```

```
/content/sia-tp0
```

[ ]:
```
!pip install pipenv
```

```
Collecting pipenv
  Downloading pipenv-2024.0.1-py3-none-any.whl.metadata (19 kB)
Requirement already satisfied: certifi in /usr/local/lib/python3.10/dist-
packages (from pipenv) (2024.7.4)
Requirement already satisfied: setuptools>=67 in /usr/local/lib/python3.10/dist-
packages (from pipenv) (71.0.4)
Collecting virtualenv>=20.24.2 (from pipenv)
  Downloading virtualenv-20.26.3-py3-none-any.whl.metadata (4.5 kB)
Collecting distlib<1,>=0.3.7 (from virtualenv>=20.24.2->pipenv)
  Downloading distlib-0.3.8-py2.py3-none-any.whl.metadata (5.1 kB)
Requirement already satisfied: filelock<4,>=3.12.2 in
/usr/local/lib/python3.10/dist-packages (from virtualenv>=20.24.2->pipenv)
(3.15.4)
Requirement already satisfied: platformdirs<5,>=3.9.1 in
```

```
/usr/local/lib/python3.10/dist-packages (from virtualenv>=20.24.2->pipenv)
(4.2.2)
Downloading pipenv-2024.0.1-py3-none-any.whl (3.2 MB)
                           3.2/3.2 MB
29.2 MB/s eta 0:00:00
Downloading virtualenv-20.26.3-py3-none-any.whl (5.7 MB)
                           5.7/5.7 MB
57.7 MB/s eta 0:00:00
Downloading distlib-0.3.8-py2.py3-none-any.whl (468 kB)
                           468.9/468.9 kB
26.7 MB/s eta 0:00:00
Installing collected packages: distlib, virtualenv, pipenv
Successfully installed distlib-0.3.8 pipenv-2024.0.1 virtualenv-20.26.3
```

[ ]:
```
!pipenv install
```

```
Creating a virtualenv for this project…
Pipfile: /content/sia-tp0/Pipfile
Using /usr/local/bin/python (3.10.12) to
create virtualenv…
  Creating virtual environment…created virtual environment
CPython3.10.12.final.0-64 in 3070ms
  creator CPython3Posix(dest=/root/.local/share/virtualenvs/sia-tp0-lGmXv_eV,
clear=False, no_vcs_ignore=False, global=False)
  seeder FromAppData(download=False, pip=bundle, setuptools=bundle,
wheel=bundle, via=copy, app_data_dir=/root/.local/share/virtualenv)
    added seed packages: pip==24.1, setuptools==70.1.0, wheel==0.43.0
  activators BashActivator,CShellActivator,FishActivator,NushellActivator,PowerS
hellActivator,PythonActivator


  Successfully created virtual environment!
  Creating virtual environment…
Virtualenv location: /root/.local/share/virtualenvs/sia-
tp0-lGmXv_eV
To activate this project's virtualenv, run pipenv shell.
Alternatively, run a command inside the virtualenv with pipenv run.
To activate this project's virtualenv, run pipenv shell.
Alternatively, run a command inside the virtualenv with pipenv run.
Installing dependencies from Pipfile.lock
(159648)…
```

[ ]:
```
!pipenv run python main.py ./configs/caterpie.json
```

```
No noise:  (True, 0.332)
Noisy:  (True, 0.3707635505122667)
Noisy:  (False, 0.2934148534230771)
Noisy:  (True, 0.3630378769936585)
Noisy:  (False, 0.37672954888790117)
Noisy:  (False, 0.2923175605574392)
Noisy:  (False, 0.3538474519429606)
Noisy:  (False, 0.40552673760138225)
Noisy:  (False, 0.2944202516588919)
Noisy:  (True, 0.3903684509490291)
Noisy:  (False, 0.34071787805818865)
```

[ ]:
```
import json
import sys

from src.catching import attempt_catch
from src.pokemon import PokemonFactory, StatusEffect
import os
import pandas as pd
import numpy as np
```

[ ]:
```
factory = PokemonFactory("pokemon.json")
```

##Creación de Pokemon Con factorty.create se crea el pokemon, los argumentos son el tipo de pokemon como string, el HP (Vida) entre 0 y 1, el estados (enfermedad o no) y el nivel (0 y 100). Estos argumentos son las propiedades mutables del pokemon. Las propiedades inmutables están definidas por el tipo de pokemon.

##Atrapada de pokemon Para intentar atrapar el pokemon utilizo attempt_catch y es necesario utilizar como argumento el pokemon que deseo atrapar y el tipo de bola a utilizar. Esta función nos devuelve un bool, si atrapamos o no el pokemon y una probabilidad de atraparlo entre 0 y 1. A su vez es posible inducir un ruido, un valor entre 0 y 1.

[ ]:
```
pokemon = factory.create("onix", 100, StatusEffect.NONE, 1)
catch, prob= attempt_catch(pokemon, "ultraball") #Sin ruido
catch, prob
```

[ ]:
```
(False, 0.1172)
```

[ ]:
```
catch, prob= attempt_catch(pokemon, "ultraball", 0.1) #con ruido
catch, prob
```

[ ]:
```
(True, 0.13159522956361586)
```

##Propiedades inmutables

Son las características propias de cada pokemon, y no se modifican.

[ ]:
```
df_pokemon= pd.read_json("pokemon.json", orient="index")
df_pokemon= df_pokemon.reset_index()
df_type= pd.DataFrame(df_pokemon["type"].to_list(), columns=['type1', 'type2'])
df_stats=pd.DataFrame(df_pokemon["stats"].to_list(), columns=["base_hp",␣
 ↪"attack", "defense", "sp_attack", "Sp_defense", "speed"])
df_pokemon= pd.concat([df_pokemon["index"], df_type, df_stats,␣
 ↪df_pokemon["weight"],  df_pokemon["catch_rate"]], axis=1)
df_pokemon.rename(columns = {'index':'pokemon'}, inplace = True)
df_pokemon
```

[ ]:

	pokemon	type1	type2	base_hp	attack	defense	sp_attack	\
0	jolteon	ELECTRIC	NONE	65	65	60	110	
1	caterpie	BUG	NONE	45	30	35	20	
2	snorlax	NORMAL	NONE	160	110	65	65	
3	onix	ROCK	GROUND	35	45	160	30	
4	mewtwo	PSYCHIC	NONE	106	110	90	154	

	Sp_defense	speed	weight	catch_rate
0	95	130	54.0	45
1	20	45	6.4	255
2	110	30	1014.1	25
3	45	70	463.0	45
4	90	130	269.0	3

[ ]:
```
df_pokemon.to_csv("pokemon_inmutables.csv")
```

##Efectividad vs Probabilidad de captura. Si definimos la efectividad como:

$Efec. = \frac{Atrapadas}{Intentos}$

Podemos calcular la efectividad de diferentes cantidad de itentos y compararla con la probabilidad de captura.

Si probamos 100 veces

[ ]:
```
pokemon = factory.create("onix", 100, StatusEffect.NONE, 1)

EFECT_ACU= []
EFECT_ACU_SN= []
for i in range(100):
  catch, prob= attempt_catch(pokemon, "ultraball") #Sin ruido
  catch_sn, prob_sn= attempt_catch(pokemon, "ultraball", 0.05) #ruido 5%
  EFECT_ACU.append(catch)
  EFECT_ACU_SN.append(catch_sn)
efectividad= sum(EFECT_ACU)/len(EFECT_ACU)
efectividad_sn= sum(EFECT_ACU_SN)/len(EFECT_ACU_SN)
print(f'Sin ruido-> Efectividad: {efectividad} vs Probabilidad: {prob}')
print(f'Con ruido-> Efectividad: {efectividad_sn} vs Probabilidad: {prob_sn}')
```

```
Sin ruido-> Efectividad: 0.11 vs Probabilidad: 0.1172
Con ruido-> Efectividad: 0.1 vs Probabilidad: 0.11290997889123659
```

Mil veces

[ ]:
```python
pokemon = factory.create("onix", 1000, StatusEffect.NONE, 1)

EFECT_ACU= []
EFECT_ACU_SN= []
for i in range(1000):
  catch, prob= attempt_catch(pokemon, "ultraball") #Sin ruido
  catch_sn, prob_sn= attempt_catch(pokemon, "ultraball", 0.05) #ruido 5%
  EFECT_ACU.append(catch)
  EFECT_ACU_SN.append(catch_sn)
efectividad= sum(EFECT_ACU)/len(EFECT_ACU)
efectividad_sn= sum(EFECT_ACU_SN)/len(EFECT_ACU_SN)
print(f'Sin ruido-> Efectividad: {efectividad} vs Probabilidad: {prob}')
print(f'Con ruido-> Efectividad: {efectividad_sn} vs Probabilidad: {prob_sn}')
```

```
Sin ruido-> Efectividad: 0.131 vs Probabilidad: 0.1172
Con ruido-> Efectividad: 0.113 vs Probabilidad: 0.11186350355497765
```

10 mil veces

[ ]:
```python
pokemon = factory.create("onix", 1000, StatusEffect.NONE, 1)

EFECT_ACU= []
EFECT_ACU_SN= []
for i in range(10000):
  catch, prob= attempt_catch(pokemon, "ultraball") #Sin ruido
  catch_sn, prob_sn= attempt_catch(pokemon, "ultraball", 0.05) #ruido 5%
  EFECT_ACU.append(catch)
  EFECT_ACU_SN.append(catch_sn)
efectividad= sum(EFECT_ACU)/len(EFECT_ACU)
efectividad_sn= sum(EFECT_ACU_SN)/len(EFECT_ACU_SN)
print(f'Sin ruido-> Efectividad: {efectividad} vs Probabilidad: {prob}')
print(f'Con ruido-> Efectividad: {efectividad_sn} vs Probabilidad: {prob_sn}')
```

```
Sin ruido-> Efectividad: 0.1208 vs Probabilidad: 0.1172
Con ruido-> Efectividad: 0.1119 vs Probabilidad: 0.11403211746997473
```

Por lo tanto, podemos establecer que la efectividad y la probabilidad convergen en un mismo valor. Por ese motivo, de aquí en adelante se va a utilizar sólo la probabilidad.

0.1 Creación de dataset

Para crear el dataset se crearon pokemones utilizando los 5 pokemon preestablecidos, las 4 bolas disponibles, y variando todas sus propiedades mutables: - Variación de raza de Pokemon: jolteon, caterpie, snorlax, onix y mewtwo. - Variación de bolas: pokeball, ultraball, fastball y heavyball. - Variación de current HP (VIDA): de 0 a 1. - Variación de Level (nivel): de 0 a 100. - Variación de

estado: Burn, Freeze, None, Paralysis, Poison y Sleep.

##Definición de parámetros para iterar.

[ ]:
```
pokemons_names= ["jolteon", "caterpie", "snorlax", "onix", "mewtwo"] #Las 5␣
 ↪tipos de pokemon
balls= ["pokeball", "ultraball", "fastball" , "heavyball"] #Las 4 tipos de bolas
current_hp= np.arange( 0,1.1,0.1, dtype=np.float16) #Current HP de 0 a 1 con␣
 ↪salto de 0.05
levels= np.arange(0,101,5) #Niveles de 0 a 100 con saltos de 5

status_effect= [StatusEffect.BURN, #Todos los estados del pokemon
                StatusEffect.FREEZE,
                StatusEffect.NONE,
                StatusEffect.PARALYSIS,
                StatusEffect.POISON,
                StatusEffect.SLEEP ]
status_effect_names= ["BURN",
                "FREEZE",
                "NONE",
                "PARALYSIS",
                "POISON",
                "SLEEP"]
```

##Iteración

[ ]:
```
PROB= []
POKEMON_NAME= []
BALL=[]
LVL=[]
HP=[]
STATUS=[]


for ball in balls:
  for pokemon_name in pokemons_names:
    for level in levels:
      for i, status in enumerate(status_effect):
        for hp in current_hp:

          pokemon = factory.create(pokemon_name, level, status, hp)
          catch, prob= attempt_catch( pokemon, ball, 0.02)
          PROB.append(prob)
          POKEMON_NAME.append(pokemon_name)
          BALL.append(ball)
          LVL.append(level)
          HP.append(hp)
          STATUS.append(status_effect_names[i])
```

##Creación de dataframe con dataset

[ ]:
```
df_todos= pd.DataFrame({"Pokemon": POKEMON_NAME, "Ball": BALL, "Level": LVL,␣
↪"Hp": HP, "Status": STATUS, "Prob": PROB  })
```

[ ]:
```
len(df_todos)
```

[ ]: 27720

[ ]:
```
df_todos.head()
```

[ ]:

	Pokemon	Ball	Level	Hp	Status	Prob
0	jolteon	pokeball	0	0.000000	BURN	0.243883
1	jolteon	pokeball	0	0.099976	BURN	0.239376
2	jolteon	pokeball	0	0.199951	BURN	0.233513
3	jolteon	pokeball	0	0.299805	BURN	0.212150
4	jolteon	pokeball	0	0.399902	BURN	0.205306

[ ]:
```
df_todos.to_csv("pokemon.csv")
```

##Iteración con alto ruido (0.15)utilizando Efectividad. Se calcula el valor medio y el desvio standard. > Añadir blockquote

[ ]:
```
PROB= []
POKEMON_NAME= []
BALL=[]
LVL=[]
HP=[]
STATUS=[]
MEAN=[]
STD=[]
prob_= np.empty(100)

for ball in balls:
  for pokemon_name in pokemons_names:
    for level in levels:
      for i, status in enumerate(status_effect):
        for hp in current_hp:
          pokemon = factory.create(pokemon_name, level, status, hp)

          for g in range(100):
            catch, prob= attempt_catch( pokemon, ball, 0.1)
            prob_[g]= prob

          MEAN.append(prob_.mean())
          STD.append(prob_.std())
          PROB.append(prob)
          POKEMON_NAME.append(pokemon_name)
```

```
            BALL.append(ball)
            LVL.append(level)
            HP.append(hp)
            STATUS.append(status_effect_names[i])
```

[ ]:
```
df_todos_ruido= pd.DataFrame({"Pokemon": POKEMON_NAME, "Ball": BALL, "Level":␣
↪LVL, "Hp": HP, "Status": STATUS, "Prob": PROB, "Mean": MEAN, "Std": STD  })
```

[ ]:
```
df_todos_ruido.to_csv("pokemon_ruido.csv")
```