

```
In [ ]: from src.catching import attempt_catch
from src.pokemon import PokemonFactory, StatusEffect
import json
import pandas as pd
import matplotlib
import matplotlib.pyplot as plt
import numpy as np
```

```
In [ ]: pokeballs = ['pokeball', 'ultraball', 'fastball', 'heavyball']
with open('pokemon_clean.json') as f:
    pokes = json.load(f)
factory = PokemonFactory('pokemon_clean.json')
```

Ejercicio 2.a

Analizar el efecto del estado de salud sobre la chance de captura

```
In [ ]: aux = []
for pok, detail in pokes.items():
    for status in StatusEffect:
        beast = factory.create(pok, 100, status, 1)
        for _ in range(2500):
            success, catch_rate = attempt_catch(beast, 'pokeball', 0)
            aux.append({'pokemon': pok, 'statusEffect': status.name, 'pokeball': 'pokeball'})
df = pd.DataFrame(aux)
```

Out[]:

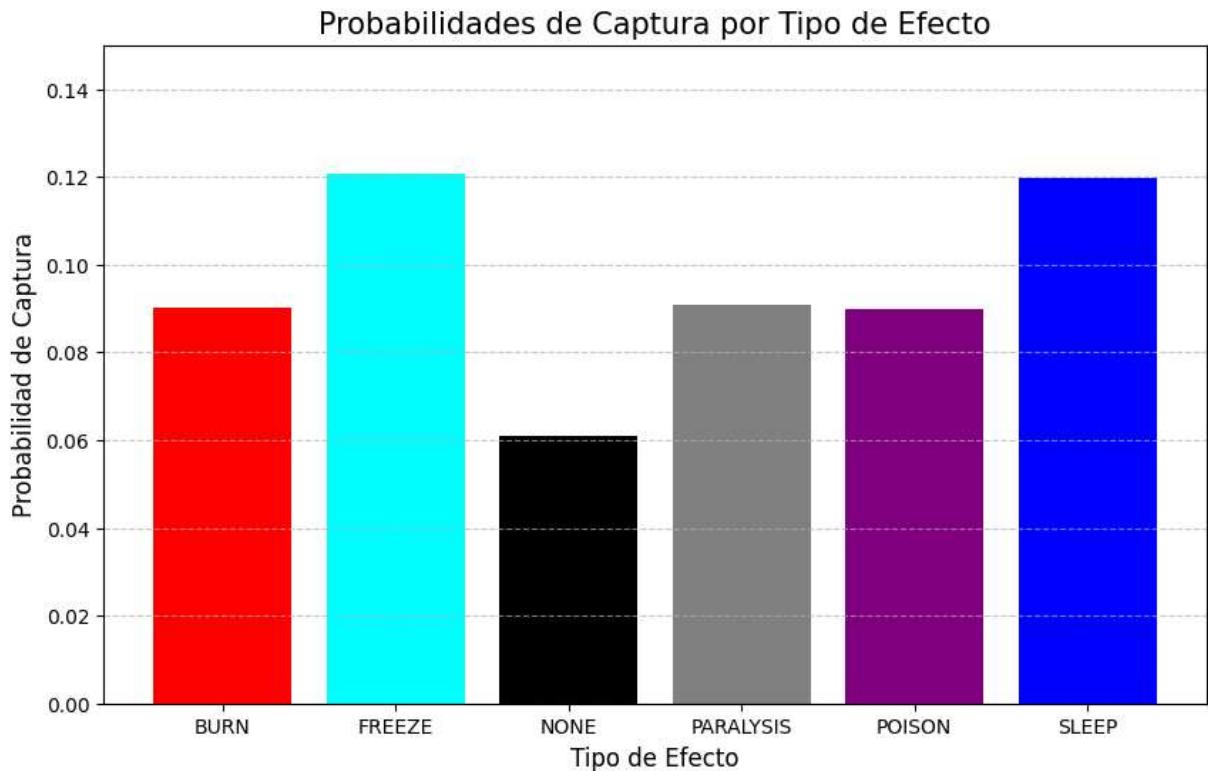
| | pokemon | statusEffect | pokeball | success | noise | weight | speed | catch_rate |
|--------|-----------|--------------|----------|---------|-------|--------|-------|------------|
| 0 | jolteon | POISON | pokeball | False | 0 | 54.0 | 130 | 0.0879 |
| 1 | jolteon | POISON | pokeball | False | 0 | 54.0 | 130 | 0.0879 |
| 2 | jolteon | POISON | pokeball | False | 0 | 54.0 | 130 | 0.0879 |
| 3 | jolteon | POISON | pokeball | True | 0 | 54.0 | 130 | 0.0879 |
| 4 | jolteon | POISON | pokeball | True | 0 | 54.0 | 130 | 0.0879 |
| ... | ... | ... | ... | ... | ... | ... | ... | ... |
| 179995 | dragonite | NONE | pokeball | False | 0 | 210.0 | 80 | 0.0586 |
| 179996 | dragonite | NONE | pokeball | False | 0 | 210.0 | 80 | 0.0586 |
| 179997 | dragonite | NONE | pokeball | False | 0 | 210.0 | 80 | 0.0586 |
| 179998 | dragonite | NONE | pokeball | False | 0 | 210.0 | 80 | 0.0586 |
| 179999 | dragonite | NONE | pokeball | False | 0 | 210.0 | 80 | 0.0586 |

180000 rows × 8 columns

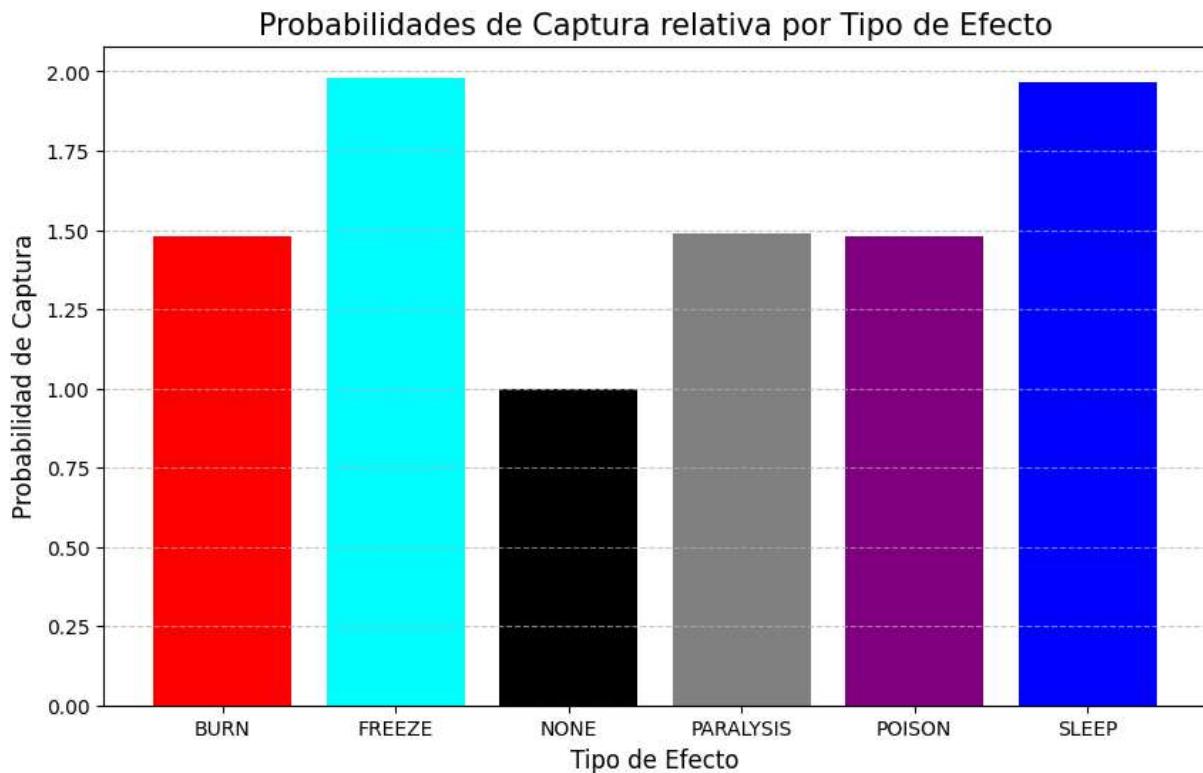
```
In [ ]: probabilidades = df.groupby(['statusEffect'])['success'].mean()
probabilidades
```

```
Out[ ]: statusEffect
BURN      0.090100
FREEZE    0.120567
NONE      0.060867
PARALYSIS 0.090667
POISON    0.089967
SLEEP     0.119567
Name: success, dtype: float64
```

```
In [ ]: plt.figure(figsize=(10,6))
plt.bar(probabilidades.index.values, probabilidades, color=['red','cyan','black','gray'])
plt.title('Probabilidades de Captura por Tipo de Efecto', fontsize=15)
plt.xlabel('Tipo de Efecto', fontsize=12)
plt.ylabel('Probabilidad de Captura', fontsize=12)
plt.ylim(0, 0.15)
plt.grid(axis='y', linestyle='--', alpha=0.7)
plt.show()
```



```
In [ ]: min_prob = min(probabilidades)
probs_rel = [prob / min_prob for prob in probabilidades]
plt.figure(figsize=(10,6))
plt.bar(probabilidades.index.values, probs_rel, color=['red','cyan','black','gray'],
plt.title('Probabilidades de Captura relativa por Tipo de Efecto', fontsize=15)
plt.xlabel('Tipo de Efecto', fontsize=12)
plt.ylabel('Probabilidad de Captura', fontsize=12)
# plt.ylim(0, 0.15)
plt.grid(axis='y', linestyle='--', alpha=0.7)
plt.show()
```



Claramente todos los statusEffects tienen un efecto positivo en la chance de capturar un pokemon. Los mas utiles son FREEZE y SLEEP, ya que duplican la tasa de captura.

Ejercicio 2.b

Analizar el efecto de los puntos de vida sobre la tasa de captura.

```
In [ ]: aux = []
for pok, detail in pokes.items():
    for life in np.arange(0.01, 1.01, 0.01):
        beast = factory.create(pok, 100, StatusEffect.NONE, life) #pokemon con nivel 1
        for _ in range(1000):
            success, catch_rate = attempt_catch(beast, 'pokeball', 0.05) #attempt_succ
            aux.append({'pokemon': pok, 'statusEffect': 'NONE', 'pokeball': 'pokeball',
df2 = pd.DataFrame(aux)
df2
```

Out[]:

| | pokemon | statusEffect | pokeball | success | noise | weight | speed | healthPercentag |
|----------------|----------------|---------------------|-----------------|----------------|--------------|---------------|--------------|------------------------|
| 0 | jolteon | NONE | pokeball | False | 0.05 | 54.0 | 130 | 0.0 |
| 1 | jolteon | NONE | pokeball | False | 0.05 | 54.0 | 130 | 0.0 |
| 2 | jolteon | NONE | pokeball | True | 0.05 | 54.0 | 130 | 0.0 |
| 3 | jolteon | NONE | pokeball | False | 0.05 | 54.0 | 130 | 0.0 |
| 4 | jolteon | NONE | pokeball | False | 0.05 | 54.0 | 130 | 0.0 |
| ... | ... | ... | ... | ... | ... | ... | ... | ... |
| 1199995 | dragonite | NONE | pokeball | False | 0.05 | 210.0 | 80 | 1.0 |
| 1199996 | dragonite | NONE | pokeball | False | 0.05 | 210.0 | 80 | 1.0 |
| 1199997 | dragonite | NONE | pokeball | False | 0.05 | 210.0 | 80 | 1.0 |
| 1199998 | dragonite | NONE | pokeball | False | 0.05 | 210.0 | 80 | 1.0 |
| 1199999 | dragonite | NONE | pokeball | False | 0.05 | 210.0 | 80 | 1.0 |

1200000 rows × 10 columns



In []: `probs = df2.groupby(['health'])['success'].mean()
probs`

Out[]: `health`

| | |
|-----|----------|
| 1 | 0.181250 |
| 2 | 0.179583 |
| 3 | 0.173583 |
| 4 | 0.182083 |
| 5 | 0.179167 |
| | ... |
| 109 | 0.060778 |
| 110 | 0.057600 |
| 111 | 0.061500 |
| 112 | 0.053000 |
| 113 | 0.034000 |

Name: success, Length: 113, dtype: float64

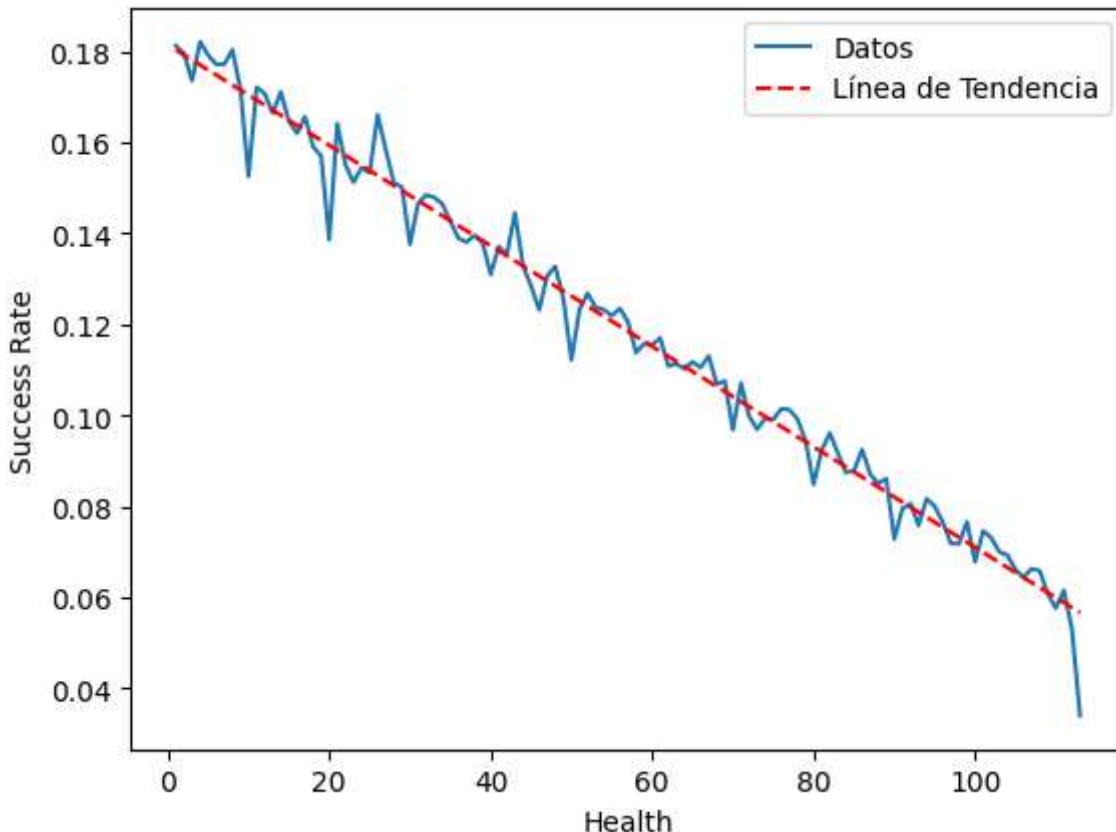
In []: `probs = probs.reset_index()`

```
# regresión Lineal
z = np.polyfit(probs['health'], probs['success'], 1)
p = np.poly1d(z)

plt.plot(probs['health'],probs['success'], label='Datos')
plt.plot(probs['health'], p(probs['health']), 'r--', label='Línea de Tendencia')

plt.xlabel('Health')
plt.ylabel('Success Rate')
```

```
plt.legend()  
plt.show()
```

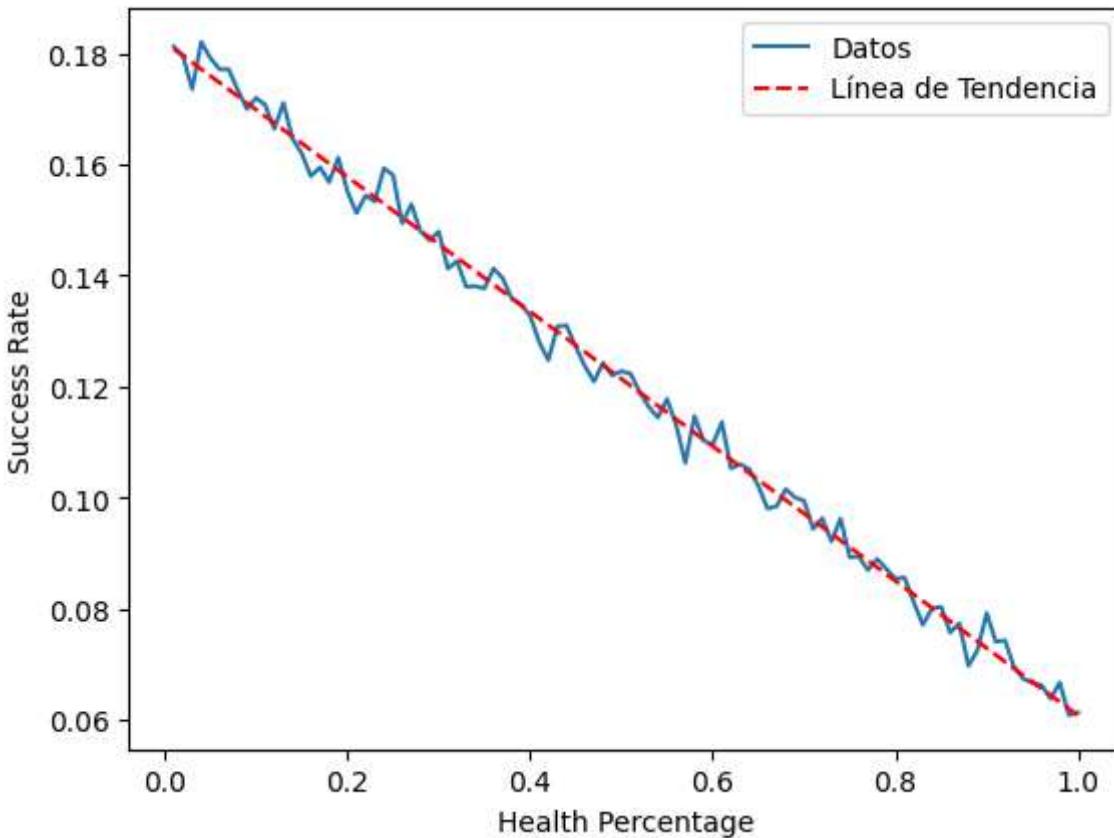


```
In [ ]: probs_p = df2.groupby(['healthPercentage'])['success'].mean()  
probs_p
```

```
Out[ ]: healthPercentage  
0.01    0.181250  
0.02    0.179583  
0.03    0.173583  
0.04    0.182083  
0.05    0.179167  
...  
0.96    0.066167  
0.97    0.063750  
0.98    0.066667  
0.99    0.060750  
1.00    0.061250  
Name: success, Length: 100, dtype: float64
```

```
In [ ]: probs_p = probs_p.reset_index()  
# regresión Lineal  
z = np.polyfit(probs_p['healthPercentage'], probs_p['success'], 1)  
p = np.poly1d(z)  
  
plt.plot(probs_p['healthPercentage'],probs_p['success'], label='Datos')  
plt.plot(probs_p['healthPercentage'], p(probs_p['healthPercentage']), 'r--', label=  
  
plt.xlabel('Health Percentage')  
plt.ylabel('Success Rate')
```

```
plt.legend()  
plt.show()
```



Se puede observar que es muy claro el efecto de la cantidad de puntos de vida sobre la tasa de captura. Es mas claro para el analisis observar las tasas dadas porcentajes de vida, ya que parece ser la razon principal del efecto.

Ejercicio 2.c

Que parametros son los que mas afectan la tasa de captura?

Como parametros entendemos aquellos que el jugador puede regular en un combate pokemon. Por lo tanto solo vamos a considerar statusEffect y vida del pokemon.

```
In [ ]: matriz_correlacion_vida = np.corrcoef(probs_p[ 'healthPercentage' ], probs_p[ 'success' ])  
correlacion = matriz_correlacion_vida[0, 1]  
  
print("Correlación vida-captura:", correlacion)  
print("Minima tasa captura:", min(probs_p[ 'success' ]), "\t Maxima tasa captura:", m
```

Correlación vida-captura: -0.9969599250451918
Minima tasa captura: 0.06075 Maxima tasa captura: 0.18208333333333335
Maximo multiplier: 2.9972565157750344

Del analisis anterior sabemos que los statusEffects pueden a lo sumo tener un efecto x2 en la tasa de captura. Mientras que el porcentaje de vida puede llegar a lo sumo a x3, un numero mucho mayor. Aun asi, en las condiciones de un combate pokemon es muy

complicado llegar al punto de vida minimo deseado (golpes criticos, statusEffects, etc) sin eliminar al contrincante.

Es facil ver que la mejor estrategia es un statusEffect de alta eficacia como FREEZE o SLEEP, un bajo punto de vida, y la pokebola indicada (Ultraball para la gran mayoria de pokemones, Heavy o Fast para pokemones particularmente pesados o rapidos)

Ejercicio 2.d

Crear la mejor estrategia para un pokemon dado.

Vamos a elegir a Snorlax por su gran peso asi podemos considerar la HeavyBall. Por otro lado vamos a analizar como es la mejor forma de atraparlo.

```
In [ ]: aux = []
for life in np.arange(0.1, 1.01, 0.1):
    for status in StatusEffect:
        sleepy_boy = factory.create('snorlax', 100, status, life)
        for ball in pokeballs:
            for _ in range(750):
                success, catch_rate = attempt_catch(sleepy_boy, ball, 0)
                aux.append({'pokemon': sleepy_boy.name, 'statusEffect': status.name,
df3 = pd.DataFrame(aux)
df3
```

Out[]:

| | pokemon | statusEffect | pokeball | success | noise | weight | speed | healthPercentage |
|--------|---------|--------------|-----------|---------|-------|--------|-------|------------------|
| 0 | snorlax | POISON | pokeball | False | 0 | 1014.1 | 30 | 0.1 |
| 1 | snorlax | POISON | pokeball | False | 0 | 1014.1 | 30 | 0.1 |
| 2 | snorlax | POISON | pokeball | True | 0 | 1014.1 | 30 | 0.1 |
| 3 | snorlax | POISON | pokeball | False | 0 | 1014.1 | 30 | 0.1 |
| 4 | snorlax | POISON | pokeball | False | 0 | 1014.1 | 30 | 0.1 |
| ... | ... | ... | ... | ... | ... | ... | ... | .. |
| 179995 | snorlax | NONE | heavyball | False | 0 | 1014.1 | 30 | 1.0 |
| 179996 | snorlax | NONE | heavyball | False | 0 | 1014.1 | 30 | 1.0 |
| 179997 | snorlax | NONE | heavyball | False | 0 | 1014.1 | 30 | 1.0 |
| 179998 | snorlax | NONE | heavyball | False | 0 | 1014.1 | 30 | 1.0 |
| 179999 | snorlax | NONE | heavyball | False | 0 | 1014.1 | 30 | 1.0 |

180000 rows × 10 columns



```
In [ ]: prob_snor = df3.groupby(['pokeball', 'healthPercentage'])['success'].mean()
```

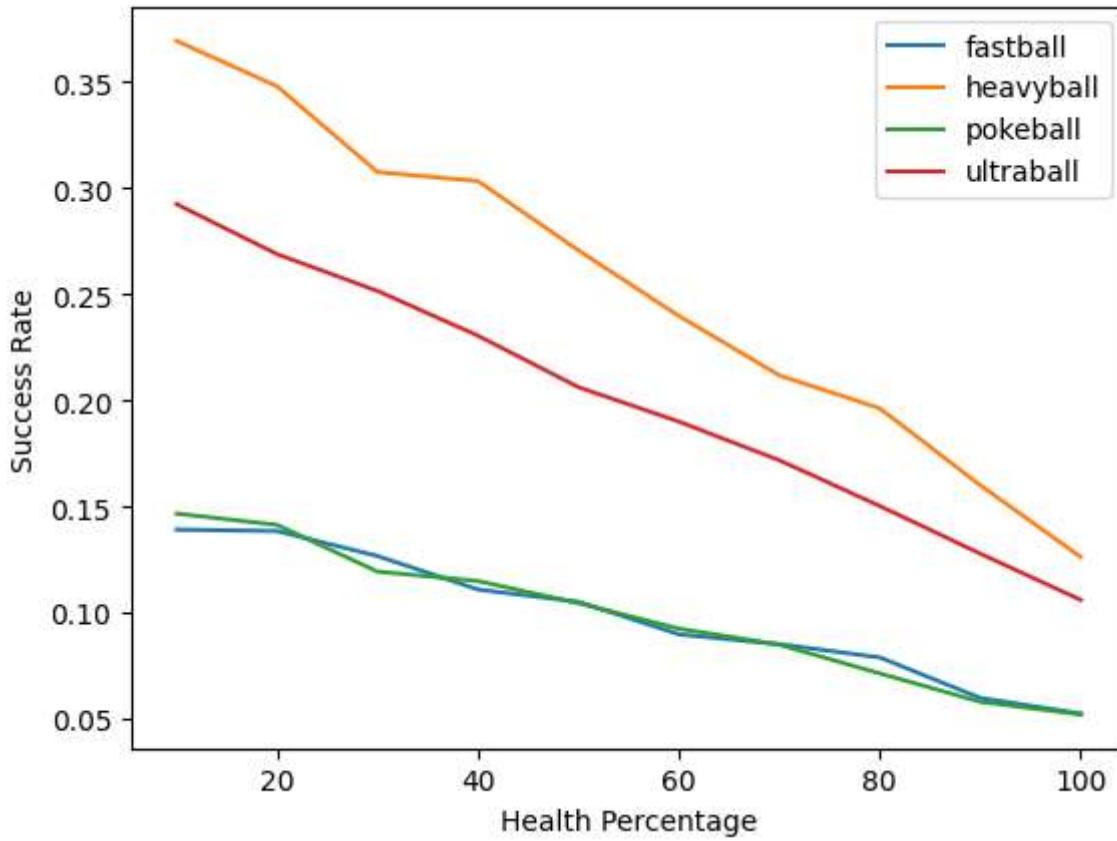
```
prob_snor
```

```
Out[ ]: pokeball    healthPercentage
fastball     0.1           0.139111
              0.2           0.138444
              0.3           0.126667
              0.4           0.110889
              0.5           0.105111
              0.6           0.089778
              0.7           0.085111
              0.8           0.078889
              0.9           0.059778
              1.0           0.052444
heavyball    0.1           0.369333
              0.2           0.348000
              0.3           0.307556
              0.4           0.303333
              0.5           0.270667
              0.6           0.239778
              0.7           0.211778
              0.8           0.196222
              0.9           0.160000
              1.0           0.126222
pokeball     0.1           0.146667
              0.2           0.141333
              0.3           0.119333
              0.4           0.114889
              0.5           0.104444
              0.6           0.092444
              0.7           0.084889
              0.8           0.071333
              0.9           0.058000
              1.0           0.052000
ultraball    0.1           0.292444
              0.2           0.268889
              0.3           0.251556
              0.4           0.230444
              0.5           0.206222
              0.6           0.190000
              0.7           0.171778
              0.8           0.150222
              0.9           0.127778
              1.0           0.106000
Name: success, dtype: float64
```

```
In [ ]: df_reset = prob_snor.reset_index()
pokeballs = df_reset['pokeball'].unique()

for pokeball in pokeballs:
    subset = df_reset[df_reset['pokeball'] == pokeball]
    plt.plot(subset['healthPercentage']*100, subset['success'], label=pokeball)

plt.xlabel('Health Percentage')
plt.ylabel('Success Rate')
plt.legend()
plt.show()
```

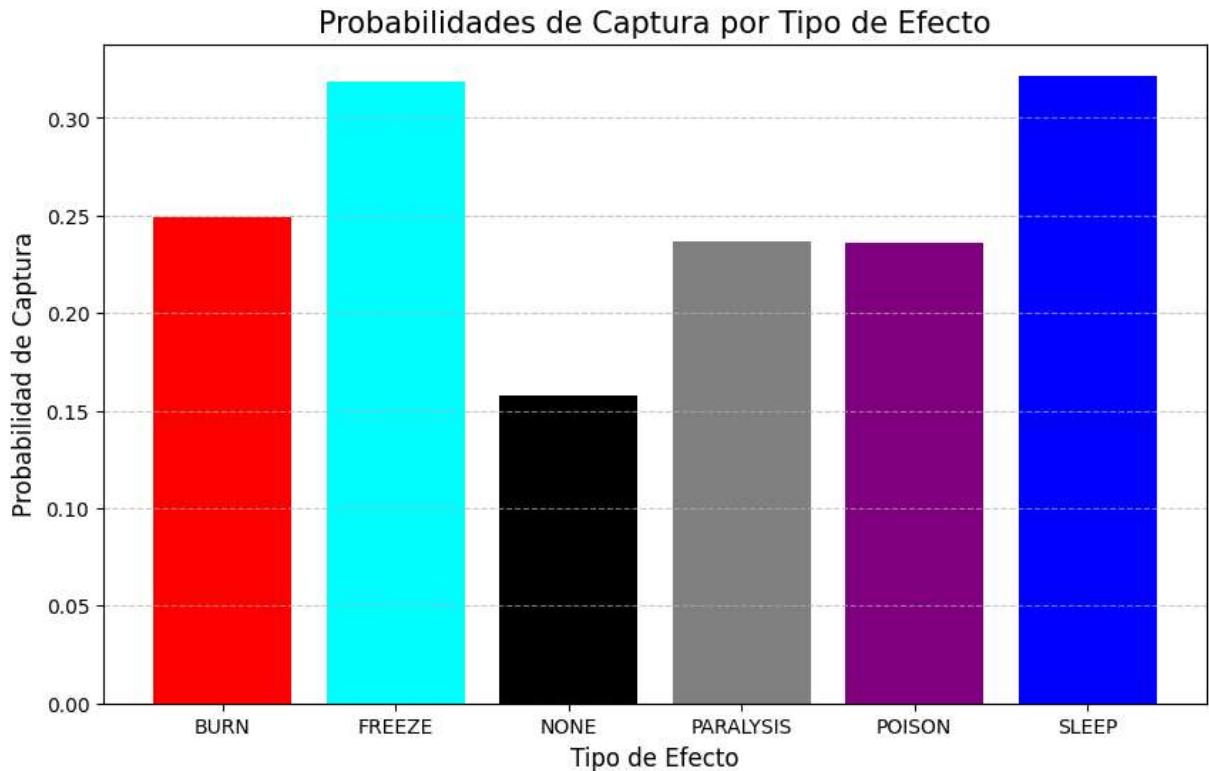


Como era de esperarse, la HeavyBall es ideal para capturar a Snorlax. Una UltraBall no es una mala alternativa, pero es clara la ventaja de la HeavyBall.

Ahora analizaremos que statusEffect sera ideal darle a Snorlax para capturarlo, dado los datos anteriores, esperamos que sea FREEZE o SLEEP.

```
In [ ]: ultra_prob = df3[df3['pokeball'] == 'heavyball']
ultra_effects = ultra_prob.groupby(['statusEffect'])['success'].mean()

plt.figure(figsize=(10,6))
plt.bar(ultra_effects.index.values, ultra_effects, color=['red','cyan','black','gray'])
plt.title('Probabilidades de Captura por Tipo de Efecto', fontsize=15)
plt.xlabel('Tipo de Efecto', fontsize=12)
plt.ylabel('Probabilidad de Captura', fontsize=12)
# plt.ylim(0, 0.15)
plt.grid(axis='y', linestyle='--', alpha=0.7)
plt.show()
```



Una vez mas, gracias a nuestro analisis previo, podemos con certeza afirmar cual es la tactica ideal sin la necesidad de realizar pruebas especificas sobre los pokemones. En este caso, recomendamos SLEEP sobre FREEZE debido a que Snorlax disfruta mucho una siesta.

Por lo tanto, queda finalizada la tactica. Atacar hasta llegar al punto mas bajo posible de HP, inducir SLEEP/FREEZE en Snorlax y utilizar una HeavyBall.

Ejercicio 2.e

Tiene el nivel del pokemon un efecto en la decision de la tactica empleada?

```
In [ ]: aux = []
for pok, detail in pokes.items():
    for life in np.arange(0.1, 1.01, 0.1):
        for lvl in np.arange(5,100,5):
            for status in StatusEffect:
                beast = factory.create(pok, lvl, status, life)
                for ball in pokeballs:
                    for _ in range(100):
                        success, catch_rate = attempt_catch(beast, ball, 0.05)
                        aux.append({'pokemon': pok, 'statusEffect':status.name, 'pok
df4 = pd.DataFrame(aux)
df4
```

Out[]:

| | pokemon | statusEffect | pokeball | success | noise | weight | speed | healthPercentag |
|---------|-----------|--------------|-----------|---------|-------|--------|-------|-----------------|
| 0 | jolteon | POISON | fastball | True | 0.05 | 54.0 | 130 | 0. |
| 1 | jolteon | POISON | fastball | True | 0.05 | 54.0 | 130 | 0. |
| 2 | jolteon | POISON | fastball | True | 0.05 | 54.0 | 130 | 0. |
| 3 | jolteon | POISON | fastball | True | 0.05 | 54.0 | 130 | 0. |
| 4 | jolteon | POISON | fastball | True | 0.05 | 54.0 | 130 | 0. |
| ... | ... | ... | ... | ... | ... | ... | ... | ... |
| 5471995 | dragonite | NONE | ultraball | True | 0.05 | 210.0 | 80 | 1. |
| 5471996 | dragonite | NONE | ultraball | False | 0.05 | 210.0 | 80 | 1. |
| 5471997 | dragonite | NONE | ultraball | False | 0.05 | 210.0 | 80 | 1. |
| 5471998 | dragonite | NONE | ultraball | False | 0.05 | 210.0 | 80 | 1. |
| 5471999 | dragonite | NONE | ultraball | False | 0.05 | 210.0 | 80 | 1. |

5472000 rows × 11 columns

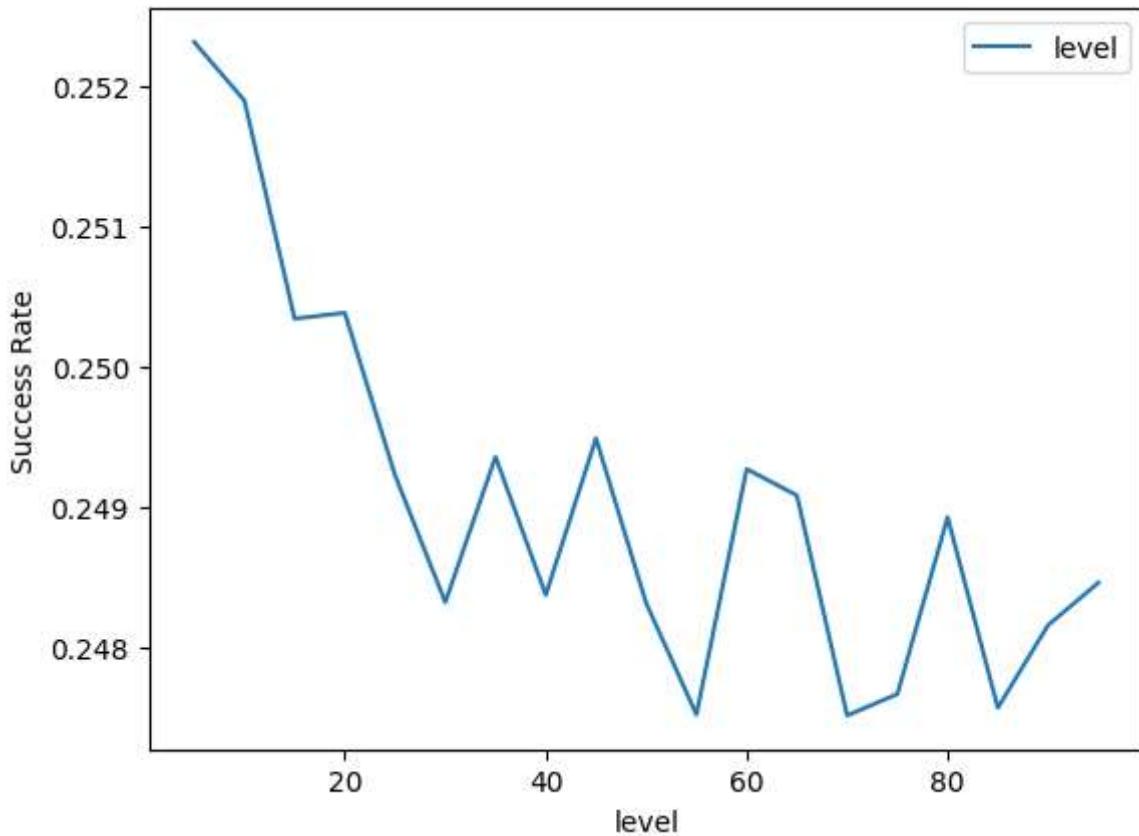


In []: prob_2 = df4.groupby(['level'])['success'].mean()
prob_2

Out[]: level
5 0.252309
10 0.251892
15 0.250340
20 0.250382
25 0.249226
30 0.248323
35 0.249358
40 0.248375
45 0.249490
50 0.248316
55 0.247524
60 0.249271
65 0.249083
70 0.247517
75 0.247670
80 0.248927
85 0.247573
90 0.248160
95 0.248462
Name: success, dtype: float64

In []: df_reset = prob_2.reset_index()
plt.plot(df_reset['level'],df_reset['success'], label='level')

```
plt.xlabel('level')
plt.ylabel('Success Rate')
plt.legend()
plt.show()
```



La primera observación es que el nivel tiene un efecto extremadamente pequeño en la tasa de éxito, tan pequeño que hasta podría considerarse dentro del error.

Para estar seguros vamos a observar si algún punto de vida se ve desproporcionalmente afectado por el nivel.

```
In [ ]: prob = df4.groupby(['level','healthPercentage'])['success'].mean()
prob
```

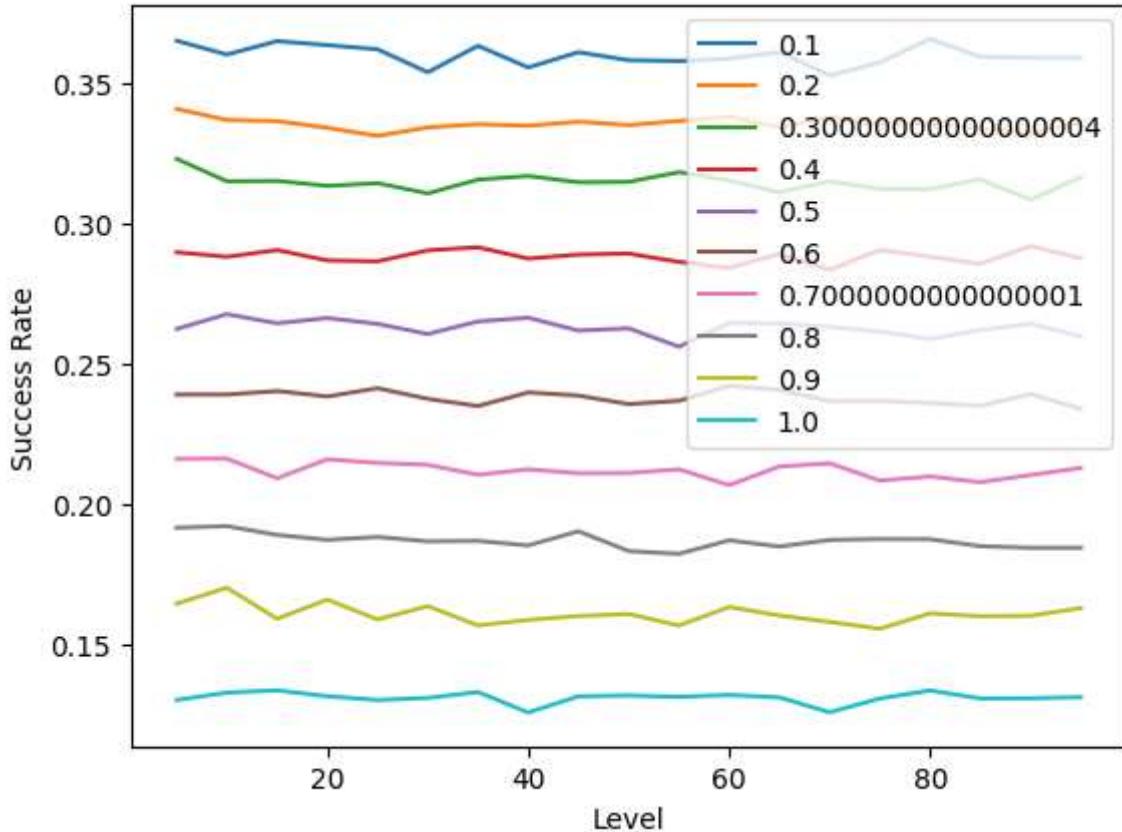
```
Out[ ]: level  healthPercentage
      5       0.1           0.365139
                 0.2           0.340868
                 0.3           0.323056
                 0.4           0.289792
                 0.5           0.262431
                  ...
      95      0.6           0.233993
                 0.7           0.212951
                 0.8           0.184514
                 0.9           0.163021
                 1.0           0.131285
Name: success, Length: 190, dtype: float64
```

A simple vista el efecto del nivel es muy pequeño

```
In [ ]: df_reset = prob.reset_index()
effects = df_reset['healthPercentage'].unique()

for effect in effects:
    subset = df_reset[df_reset['healthPercentage'] == effect]
    plt.plot(subset['level'], subset['success'], label=effect)

plt.xlabel('Level')
plt.ylabel('Success Rate')
plt.legend()
plt.show()
```



Ningun punto de vida es desproporcionalmente afectado por el nivel para su tasa de exito.
Todos los puntos de vida parecen tener un efecto nulo dado el nivel.

Si observamos las pokebolas y los statusEffects tambien esperamos el mismo resultado.

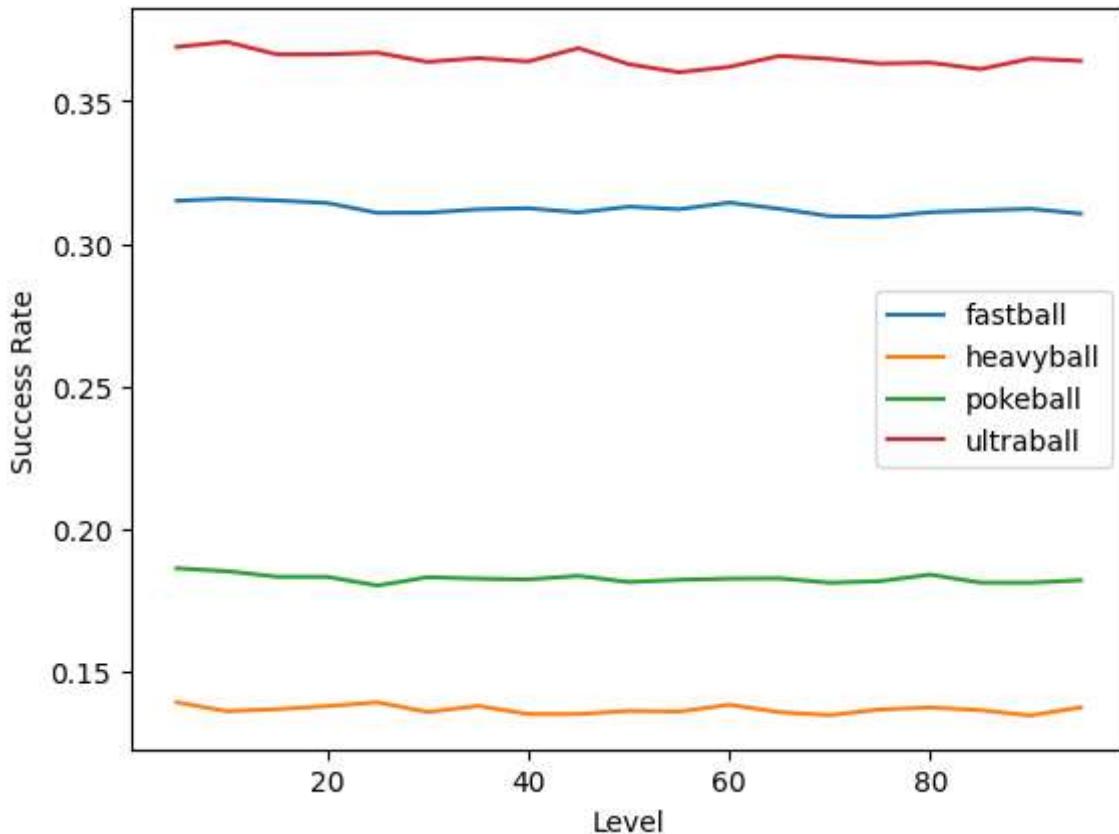
```
In [ ]: prob_3 = df4.groupby(['pokeball','level'])['success'].mean()
```

```
In [ ]: df_reset = prob_3.reset_index()
pokeballs = df_reset['pokeball'].unique()

for pokeball in pokeballs:
    subset = df_reset[df_reset['pokeball'] == pokeball]
    plt.plot(subset['level'], subset['success'], label=pokeball)

plt.xlabel('Level')
```

```
plt.ylabel('Success Rate')
plt.legend()
plt.show()
```

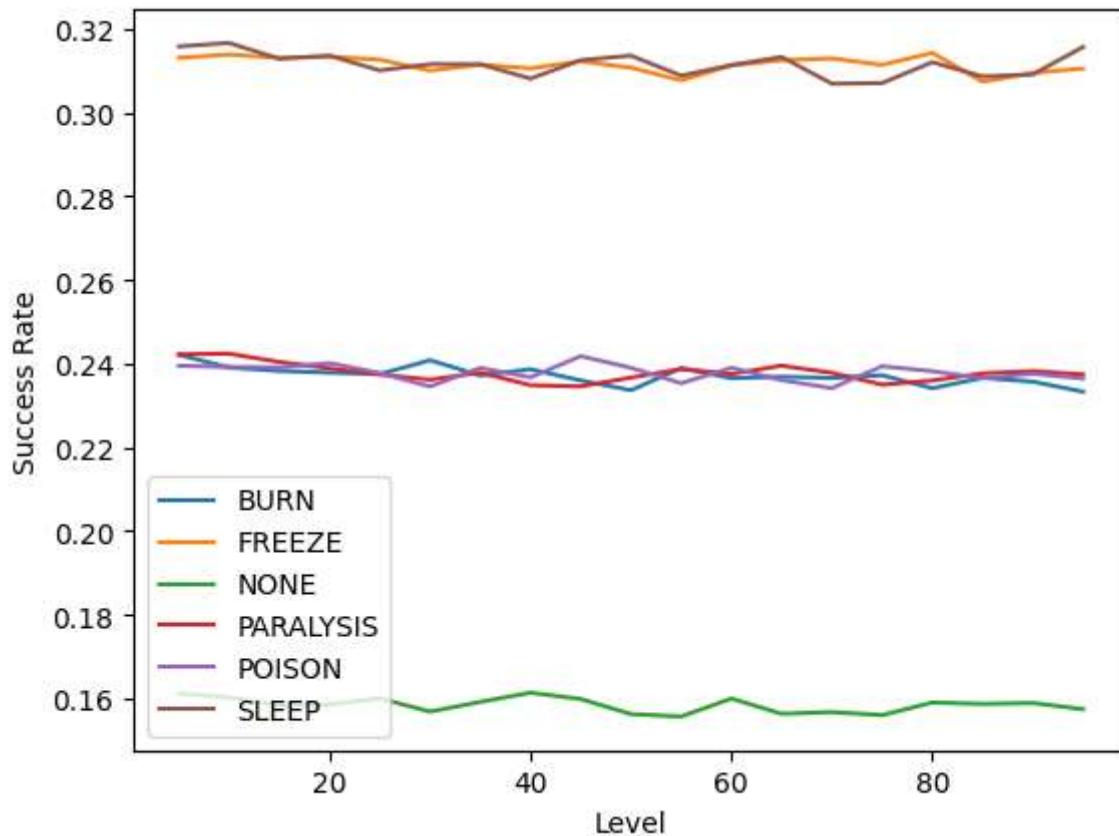


```
In [ ]: prob_4 = df4.groupby(['statusEffect','level'])['success'].mean()

df_reset = prob_4.reset_index()
pokeballs = df_reset['statusEffect'].unique()

for pokeball in pokeballs:
    subset = df_reset[df_reset['statusEffect'] == pokeball]
    plt.plot(subset['level'], subset['success'], label=pokeball)

plt.xlabel('Level')
plt.ylabel('Success Rate')
plt.legend()
plt.show()
```



Usando este estudio dado todos los pokemones, y la experiencia del inciso anterior donde el estudio de los pokemones en conjunto permite realizar conclusiones sobre tacticas particulares, podemos definir que el nivel no tiene ningun efecto significante sobre la capacidad de capturar un pokemon o la tactica empleada. Por supuesto en una situacion de juego real, un pokemon de alto nivel sera mas complejo de llevar a un punto de vida mas bajo.