Instituto Tecnológico de Buenos Aires

TP 0

Introducción al Análisis de Datos

Boullosa Gutierrez, Juan Cruz - 63414
Deyheralde, Ben - 63559
Freire, Tomás - 62027
Stanfield, Theo - 63403

Sistema de Inteligencia Artificial - 72.27

Primer cuatrimestre 2025 - Grupo 6

Tabla de contenidos

1 Preguntas Guía

Se desea evaluar qué factores influyen en la captura de un Pokemon teniendo en cuenta los parámetros del mismo junto con la pokebola utilizada. Para ello se deberán generar distintos gráficos que representen las respuestas a las siguientes preguntas guía:

1. Acerca de las pokebolas:
    1. Ejecutando la función 100 veces, para cada Pokemon en condiciones ideales (HP: 100%, LVL 100) ¿Cuál es la probabilidad de capturar promedio para cada pokebola?
    2. ¿Es cierto que algunas pokebolas son más o menos efectivas dependiendo de propiedades intrínsecas de cada Pokemon? Justificar.
        > **Sugerencia**: Comparar efectividad (*success/total_attempts*) como proporción de la efectividad de la Pokebola básica para cada Pokemon
2. Acerca del estado del Pokemon:
    1. ¿Las condiciones de salud tienen algún efecto sobre la efectividad de la captura? Si es así, ¿Cuál es más o menos efectiva?
    2. ¿Cómo afectan los puntos de vida a la efectividad de la captura?
        > **Sugerencia**: Elegir uno o dos Pokemones y manteniendo el resto de los parámetros constantes, calcular la probabilidad de captura para distintos HP %
    3. ¿Qué parámetros son los que más afectan la probabilidad de captura?
    4. Teniendo en cuenta uno o dos pokemones distintos: ¿Qué combinación de condiciones (propiedades mutables) y pokebola conviene utilizar para capturarlos?
    5. A partir del punto anterior, ¿sería efectiva otra combinación de parámetros teniendo en cuenta un nivel del pokemon más bajo (o más alto)?

2 Preparando el entorno

Para poder realizar las consignas solicitadas primero debemos preparare el entorno de trabajo, importando los módulos necesarios:

- `matplotlib`: módulo encargado de la realización de gráficos.
- `numpy`: módulo que simplifica los cálculos con vectores y matrices.

A su vez tenemos también módulos locales:

- `attempt_catch`: es la función encargada de calcular la probabilidad de atrapar al **Pokemon**.
- `PokemonFactory`: clase encargada de generar **Pokemones** en base a un archivo de entrada de tipo *json*.
- `StatusEffect`: estructura que simula toda la información respecto a los posibles estados de salud del **Pokemon**.

[1]:
```
import matplotlib.pyplot as plt
import numpy as np

from src.catching import attempt_catch
from src.pokemon import PokemonFactory, StatusEffect
```

Ahora, creamos un *factory* encargado de generar todos los **Pokemones** en base a un archivo *json* que los define.

[2]:
```
factory = PokemonFactory("pokemon.json")
```

Definamos también los posibles **Pokemones**, junto con las posibles **Pokeballs** a utilizar en la resolución de las consignas.

[3]:
```
pokemons = ["caterpie", "onix", "jolteon", "snorlax", "mewtwo"]
pokeballs = ["pokeball", "ultraball", "fastball", "heavyball"]
```

3 Resolución

3.1 Punto 1a

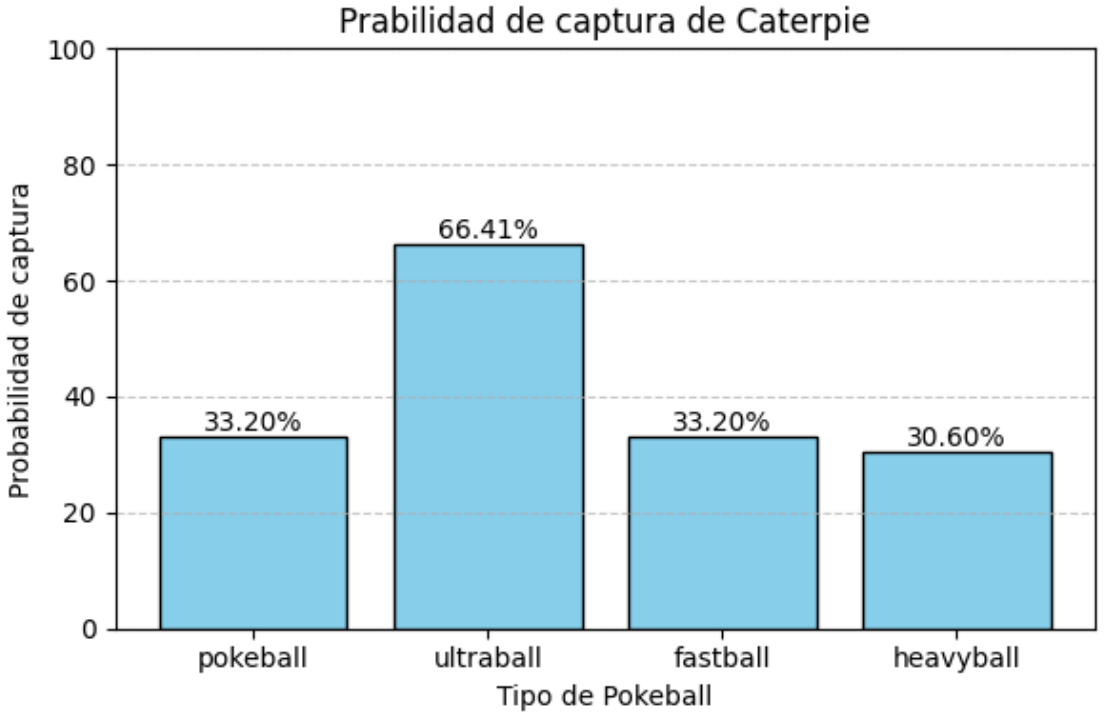
Para resolver este punto, realizaremos 100 intentos de captura con cada **Pokeball** por cada **Pokemon**. Calcularemos luego el promedio de todos los *capture_rate* obtenidos.

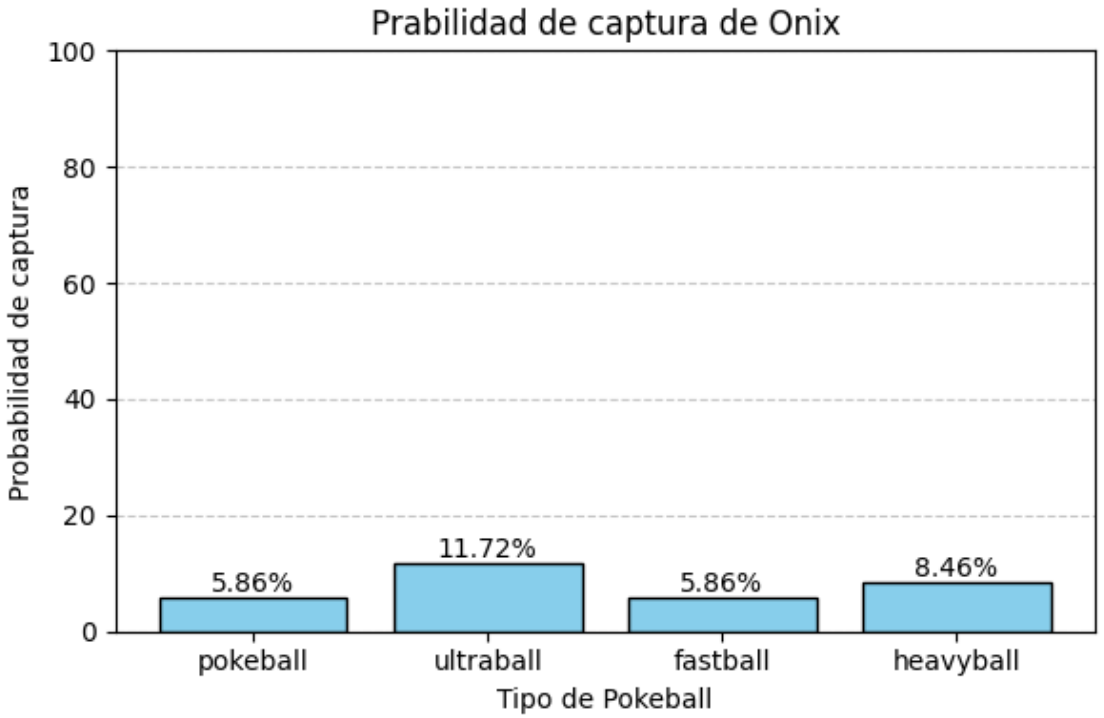
[4]:
```
results = {}

for pokemon_name in pokemons:
    results[pokemon_name] = {}
    pokemon = factory.create(pokemon_name, 100, StatusEffect.NONE, 1)

    for ball in pokeballs:
        results[pokemon_name][ball] = []

        for _ in range(100):
            _, capture_rate = attempt_catch(pokemon, ball)
            results[pokemon_name][ball].append(capture_rate)
```

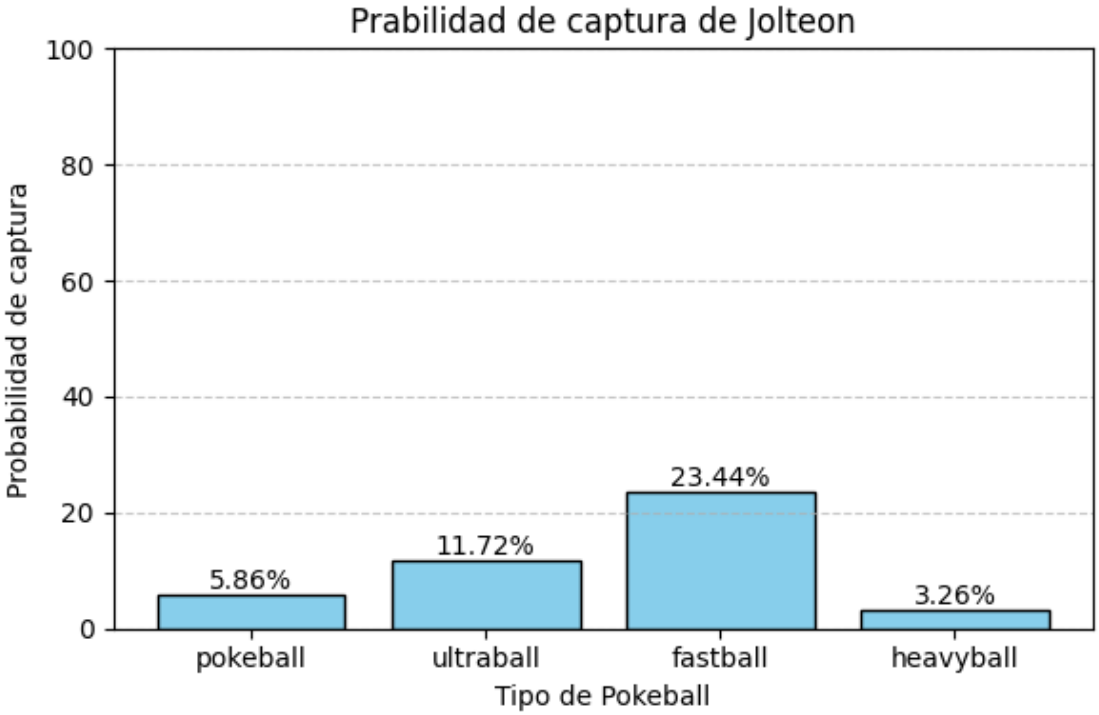
Ahora que ya tenemos los resultados, calculemos el promedio de la probabilidad de capturar a cada **Pokemon**. Para eso realizaremos un gráfico por cada **Pokemon** que se intentó capturar, mostrando el promedio de la probabilidad de captura para cada **Pokeball**.

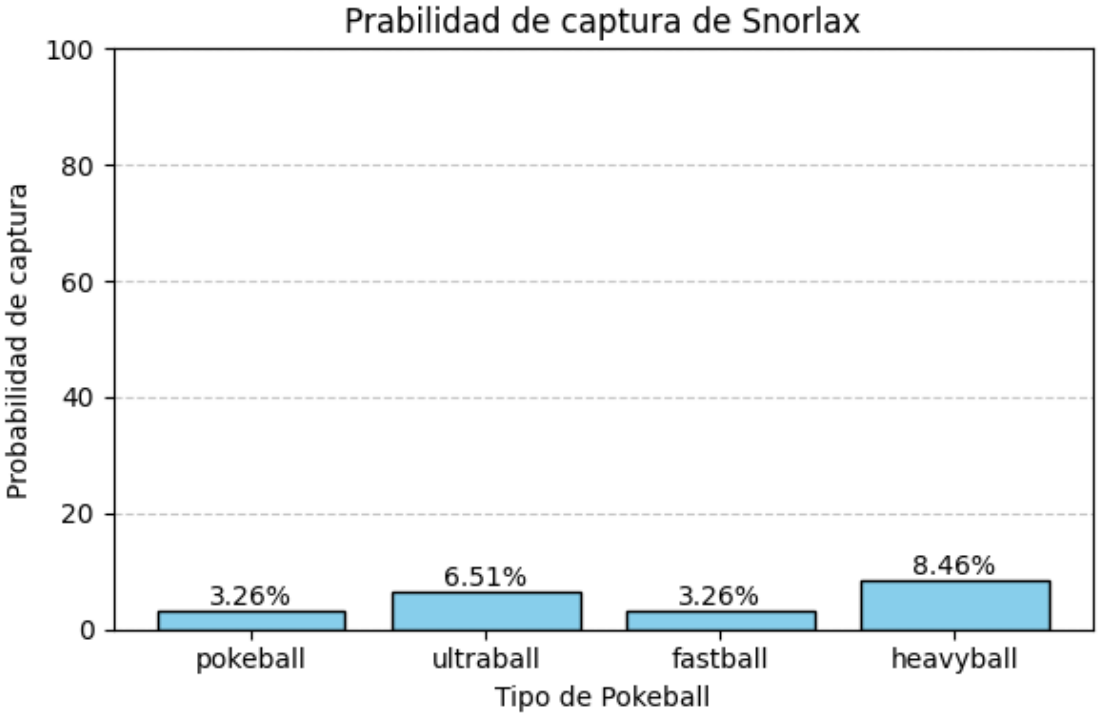
[5]:
```
for pokemon, result_data in results.items():
    data = {ball: np.mean(r) * 100 for ball, r in result_data.items()}

    plt.figure(figsize=(6, 4))
    plt.bar(data.keys(), data.values(), edgecolor="black", color="skyblue")

    plt.xlabel("Tipo de Pokeball")
    plt.ylabel("Probabilidad de captura")
    plt.title(f"Probabilidad de captura de {pokemon.capitalize()}")

    plt.grid(axis="y", linestyle="--", alpha=0.7)
    plt.ylim(0, 100)

    for i, rate in enumerate(data.values()):
        plt.text(i, rate, f"{rate:.2f}%", ha="center", va="bottom")
```

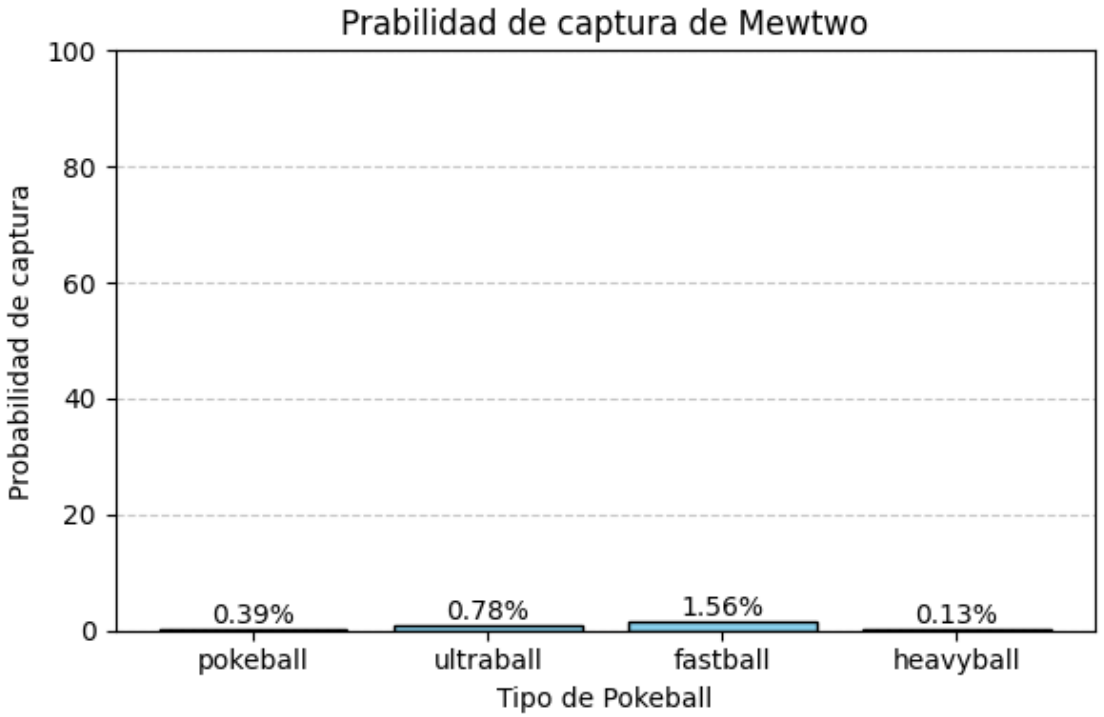
```
    plt.tight_layout()
    plt.show(block=False)
```






Prabilidad de captura de Jolteon
Probabilidad de captura
100
80
60
40
20
0
5.86%
11.72%
23.44%
3.26%
pokeball
ultraball
fastball
heavyball
Tipo de Pokeball


Prabilidad de captura de Snorlax
Probabilidad de captura
100
80
60
40
20
0
3.26%
6.51%
3.26%
8.46%
pokeball
ultraball
fastball
heavyball
Tipo de Pokeball



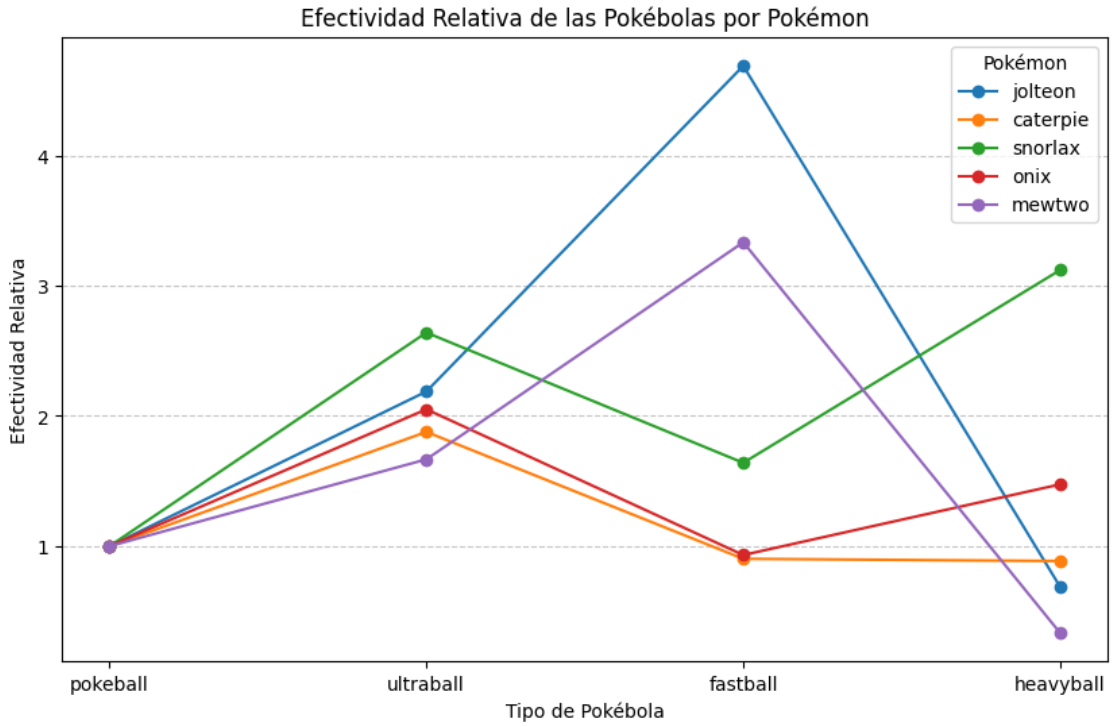
Podemos observar que la **Ultraball** duplica la efectividad de la **Pokeball** básica mientras que la **Heavyball** y la **Fastball** varían dependiendo

3.2 Punto 1b

Para resolver este punto realizamos 1000 intentos de captura con cada **Pokeball** por cada **Pokemon**. Luego se calcula el *success rate* calculando el promedio de éxitos de cada **Pokeball** por cada **Pokemon**. Por último por cada **Pokeball** se calcula la efectividad relativa dividiendo su *success rate* por la de la **Pokeball** básica. Lo hacemos con 1000 intentos porque al hacerlo con 100 nos pasaba que la probabilidad de captura para la **Pokeball** básica quedaba muy cercano a 0 y al calcular el *success rate* relativo nos daban números no representativos.

[5]:
```
results = {pokemon: {ball: 0 for ball in pokeballs} for pokemon in␣
 ↪pokemon_names}

for pokemon_name in pokemon_names:
    for ball in pokeballs:
        pokemon = factory.create(pokemon_name, 50, StatusEffect.NONE, 1.0)
        rates = [attempt_catch(pokemon, ball, 0.15)[0] for _ in range(1000)]
        results[pokemon_name][ball] = np.mean(rates)

relative_effectiveness = {}
```

```
for pokemon in pokemon_names:
    pokeball_success = results[pokemon]["pokeball"]
    relative_effectiveness[pokemon] = {ball: results[pokemon][ball] /␣
 ↪pokeball_success for ball in pokeballs}

plt.figure(figsize=(10, 6))
for pokemon, effectiveness in relative_effectiveness.items():
    plt.plot(pokeballs, list(effectiveness.values()), marker='o', label=pokemon)

plt.xlabel("Tipo de Pokébola")
plt.ylabel("Efectividad Relativa")
plt.title("Efectividad Relativa de las Pokébolas por Pokémon")
plt.legend(title="Pokémon")
plt.grid(axis="y", linestyle="--", alpha=0.7)
plt.show()
```



En el gráfico podemos ver claramente la efectividad de cada **Pokebola** en relación a la **Pokebola** básica y llegamos a la conclusión de que efectivamente es cierto que algunas **Pokebolas** son más o menos efectivas dependiendo de las propiedades intrínsecas de cada **Pokemon**. Por ejemplo la **Heavyball** es más efectiva que la **Pokebola** básica para atrapar al **Snorlax** pero menos efectiva para atrapar al **Mewtwo**. Otro ejemplo que también se puede ver en el gráfico es que la mejor **Pokebola** para atrapar al **Jolteon** es la **Fastball**. Estos resultados reflejan que la elección de la **Pokebola** adecuada puede influir significativamente en la probabilidad de captura de cada

Pokemon.

3.3 Punto 2a

Veamos ahora como se ve afectada la efectividad para la captura de los **Pokemones** dependiendo de los estados de salud de los mismos. Para eso vamos a definir los estados de salud que vamos a estar analizando.

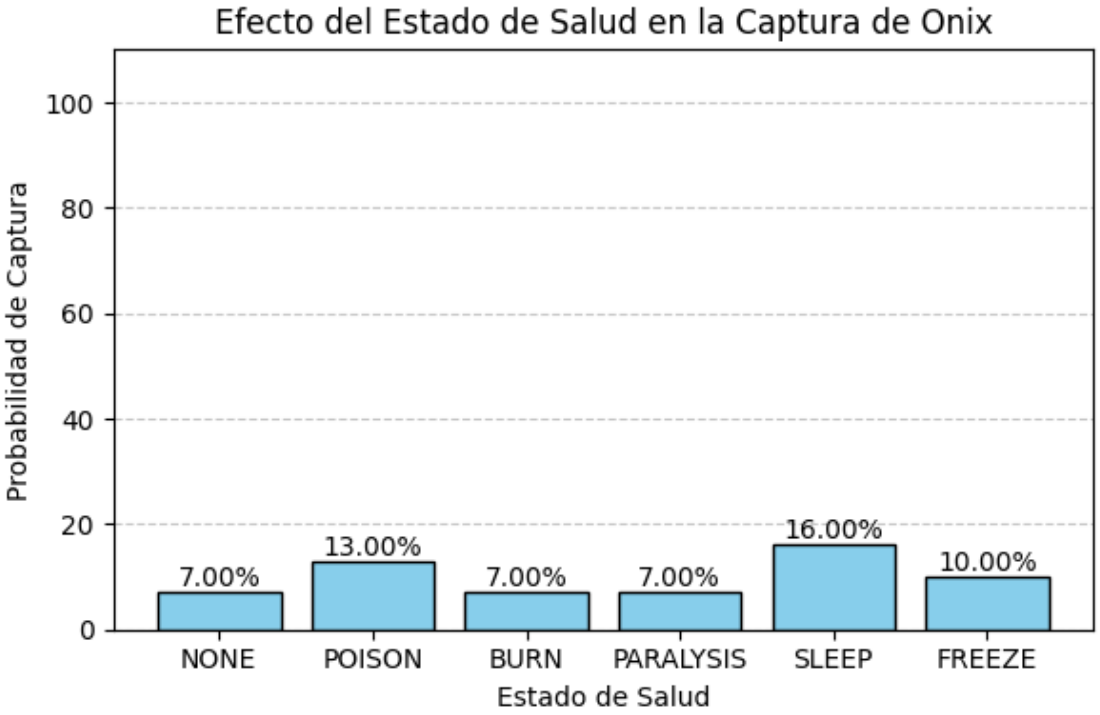
[7]:
```
effects = [StatusEffect.NONE, StatusEffect.POISON, StatusEffect.BURN,␣
↪StatusEffect.PARALYSIS, StatusEffect.SLEEP, StatusEffect.FREEZE]
```

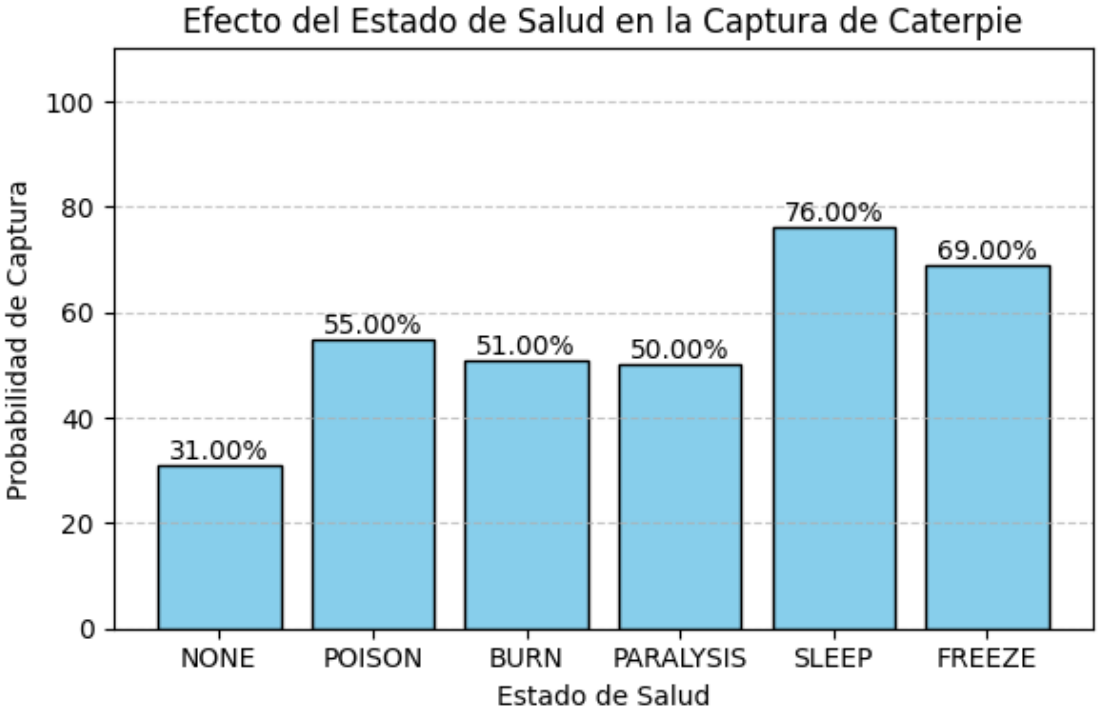
Intentemos ahora calcular a **Onix** y a **Caterpie** 100 veces cada uno en cada uno de los posibles estados de salud que definimos.

[8]:
```
results = {}

for pokemon_name in ["onix", "caterpie"]:
    results[pokemon_name] = {}

    for effect in effects:
        results[pokemon_name][effect.name] = []
        pokemon = factory.create(pokemon_name, 100, effect, 1)

        for _ in range(100):
            success, _ = attempt_catch(pokemon, "pokeball", 0.15)
            results[pokemon_name][effect.name].append(success)
```

Ahora grafiquemos para ver como varía la efectividad de los diferentes estados de salud.

[9]:
```
for pokemon, result_data in results.items():
    data = {effect: np.mean(r) * 100 for effect, r in result_data.items()}

    plt.figure(figsize=(6, 4))
    plt.bar(data.keys(), data.values(), edgecolor="black", color="skyblue")

    plt.xlabel("Estado de Salud")
    plt.ylabel("Probabilidad de Captura")
    plt.title(f"Efecto del Estado de Salud en la Captura de {pokemon.
↪capitalize()}")

    plt.grid(axis="y", linestyle="--", alpha=0.7)
    plt.ylim(0, 110)

    for i, rate in enumerate(data.values()):
        plt.text(i, rate, f"{rate:.2f}%", ha="center", va="bottom")

    plt.tight_layout()
    plt.show()
```


Efecto del Estado de Salud en la Captura de Onix
Probabilidad de Captura
0
20
40
60
80
100
7.00%
13.00%
7.00%
7.00%
16.00%
10.00%
NONE
POISON
BURN
PARALYSIS
SLEEP
FREEZE
Estado de Salud


Efecto del Estado de Salud en la Captura de Caterpie
Probabilidad de Captura
0
20
40
60
80
100
31.00%
55.00%
51.00%
50.00%
76.00%
69.00%
NONE
POISON
BURN
PARALYSIS
SLEEP
FREEZE
Estado de Salud

Por más que el porcentaje de efectividad en cada caso es diferente, podemos ver claramente como el estado de "*sueño*" y "*congelamiento*" afectan en mucho mayor medida a la efectividad en la captura de los **Pokemones**. Incluso en menores grados, podemos ver que todos los efectos de salud, aumentan a la efectividad de la captura, ya que en comparación al caso de que no se haya aplicado ningún efecto, estos son mucho mayores.

3.4 Punto 2b

Veamos ahora, de forma similar, como afectan los puntos de vida restantes al intentar capturar a los **Pokemones**. Para este caso, usaremos los mismos **Pokemones** del caso anterior (**Onix** y **Caterpie**), manteniendo el nivel en 100, sin efectos de salud y utilizando la **Pokeball** básica.

Definamos entonces para eso los valores de vida que estaremos utilizando.

[10]:
```
healths = ["1.00", "0.75", "0.50", "0.25", "0.01"]
```

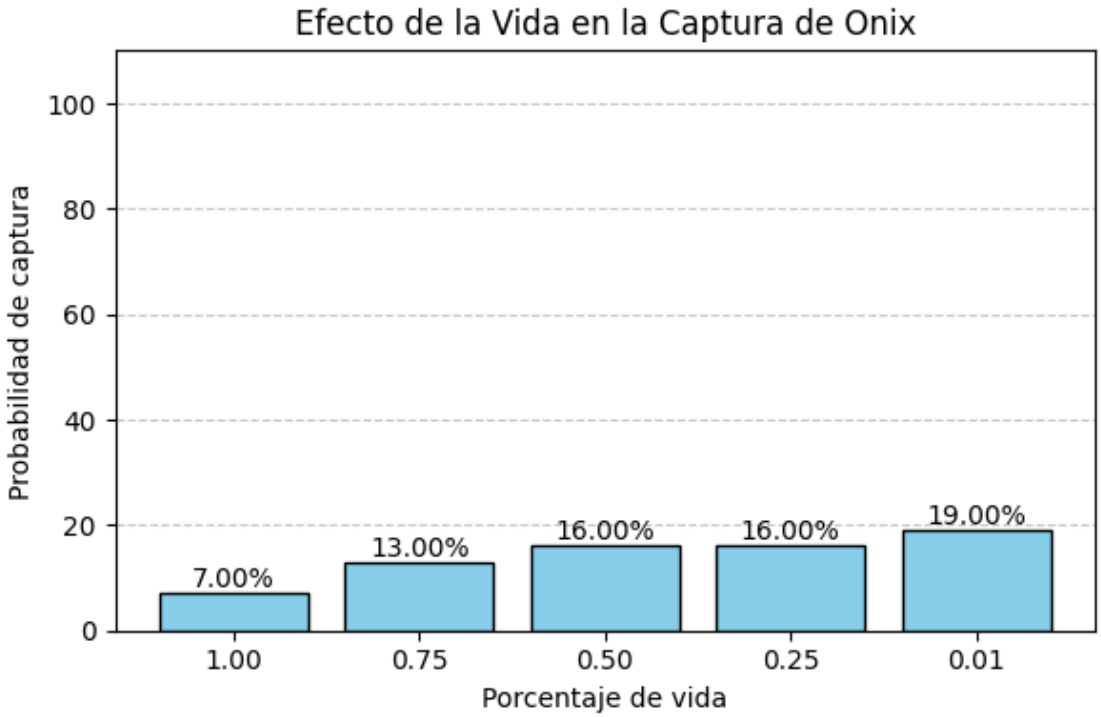
Ahora, intentemos atrapar a los **Pokemones** mencionados, 100 veces cada uno por cada puntaje de vida disponible.

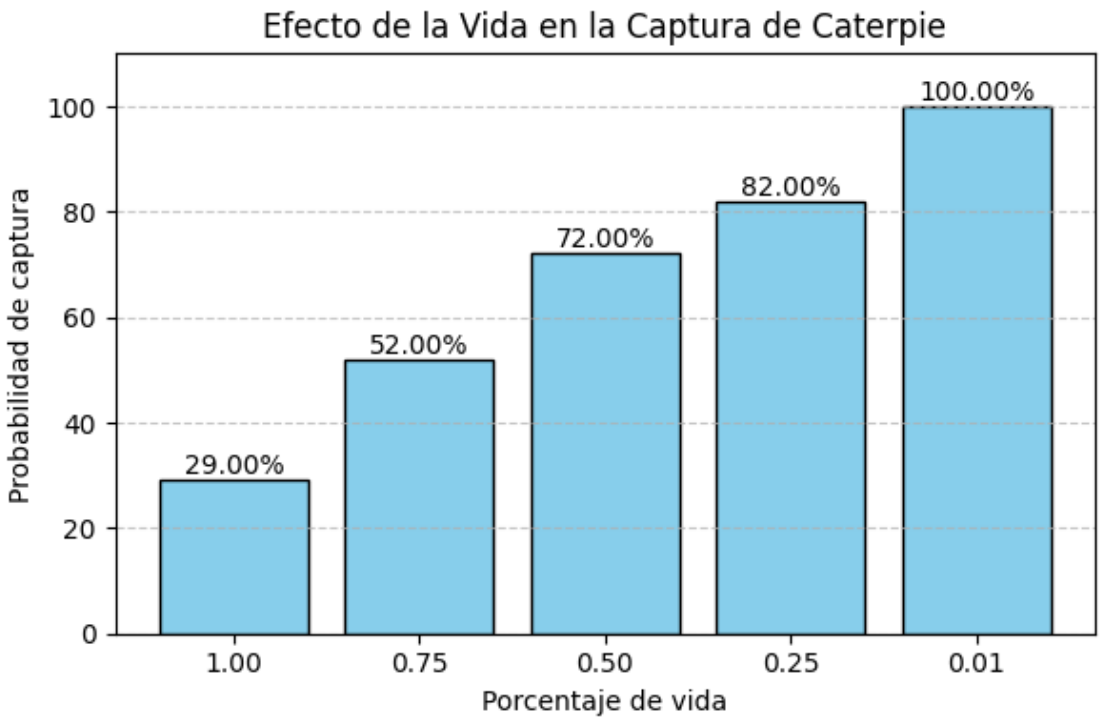
[11]:
```
results = {}

for pokemon_name in ["onix", "caterpie"]:
    results[pokemon_name] = {}

    for health in healths:
        results[pokemon_name][health] = []
        pokemon = factory.create(pokemon_name, 100, StatusEffect.NONE,␣
 ↪float(health))

        for _ in range(100):
            success, _ = attempt_catch(pokemon, "pokeball")
            results[pokemon_name][health].append(success)
```

Con los resultados disponibles, calculemos la efectividad en cada caso y grafiquemos los resultados.

[12]:
```
for pokemon, result_data in results.items():
    data = {health: np.mean(r) * 100 for health, r in result_data.items()}

    plt.figure(figsize=(6, 4))
    plt.bar(data.keys(), data.values(), edgecolor="black", color="skyblue")

    plt.xlabel("Porcentaje de vida")
    plt.ylabel("Probabilidad de captura")
    plt.title(f"Efecto de la Vida en la Captura de {pokemon.capitalize()}")

    plt.grid(axis="y", linestyle="--", alpha=0.7)
    plt.ylim(0, 110)

    for i, rate in enumerate(data.values()):
```

```
        plt.text(i, rate, f"{rate:.2f}%", ha="center", va="bottom")

    plt.tight_layout()
    plt.show()
```





Se puede observar que cuando el nivel de vida es muy bajo, aumenta considerablemente la probabilidad de captura.

3.5 Punto 2c

Ya analizamos como varía la efectividad al agregar efectos de salud y al modificar los puntos de vida del **Pokemon**. Veamos por último como varía la efectividad al modificar el nivel de **Pokemon**. Nuevamente, probaremos con **Onix** y **Caterpie**, que son los que estuvimos utilizando hasta el momento. Definamos entonces los posibles niveles a probar.

[14]:
```
levels = ["1", "25", "50", "75", "100"]
```

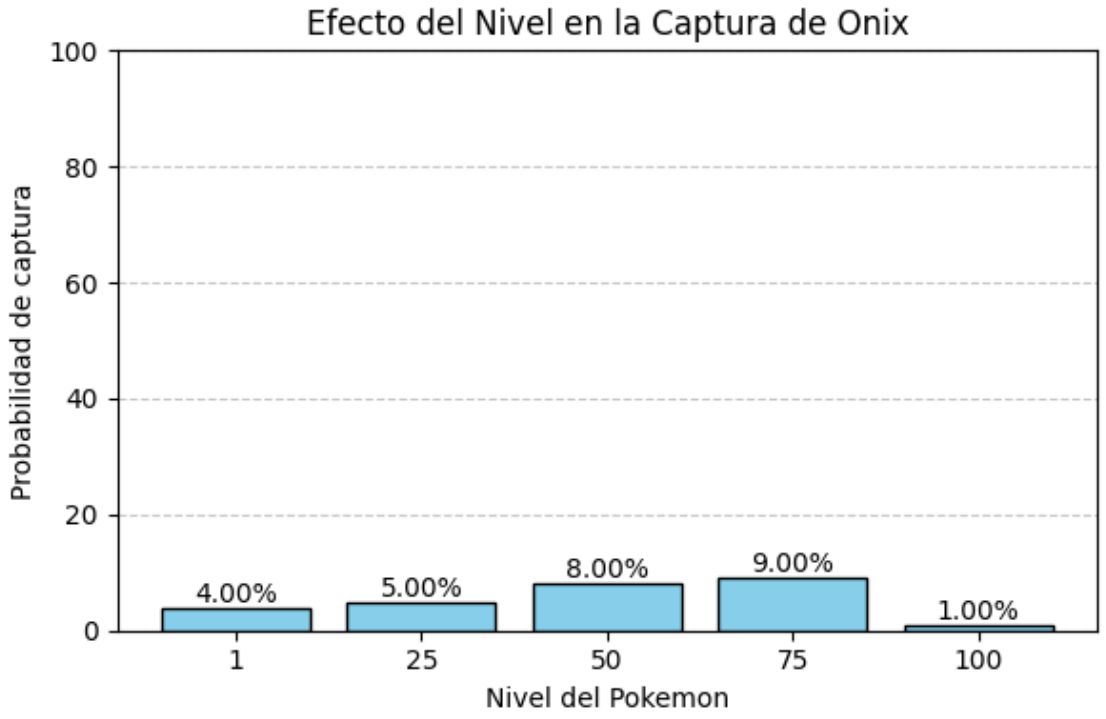
Manteniendo la vida al máximo y sin aplicar efectos de salud, veamos como varía la efectividad al intentar atrapar a los **Pokemones** mencionados 100 veces.

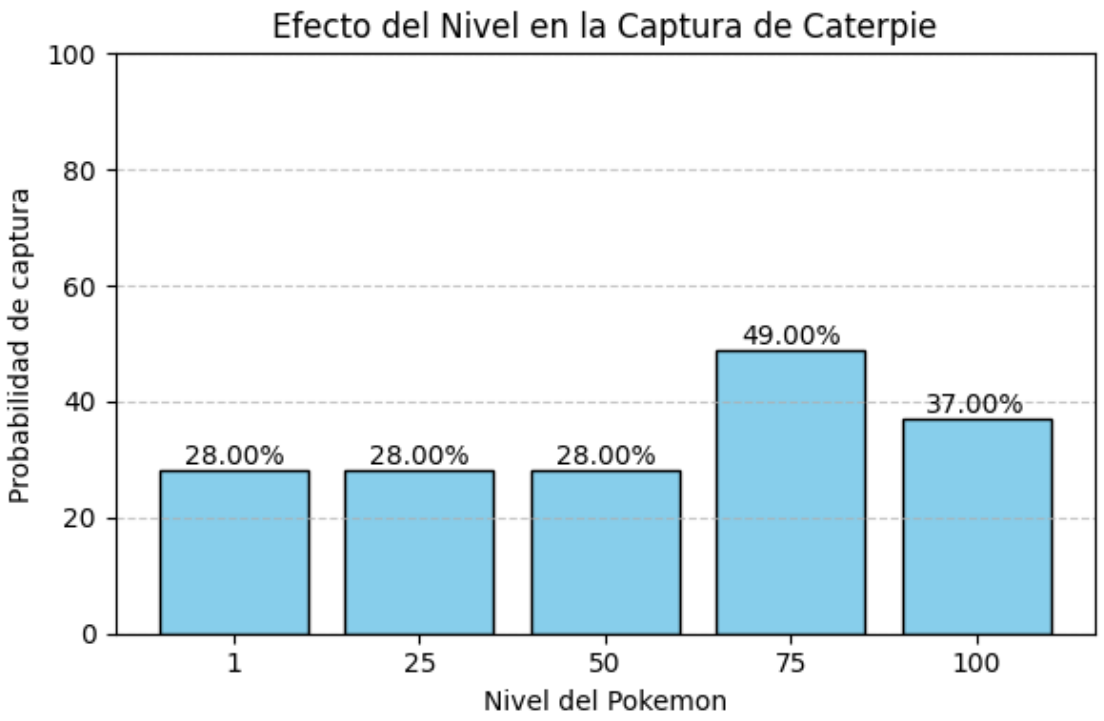
[15]:
```
results = {}

for pokemon_name in ["onix", "caterpie"]:
    results[pokemon_name] = {}

    for level in levels:
        results[pokemon_name][level] = []
        pokemon = factory.create(pokemon_name, int(level), StatusEffect.NONE, 1)

        for _ in range(100):
```

```
            success, _ = attempt_catch(pokemon, "pokeball")
            results[pokemon_name][level].append(success)
```

Con los resultados disponibles, calculemos la efectividad por cada nivel y grafiquemos los valores obtenidos.

[16]:
```
for pokemon, result_data in results.items():
    data = {level: np.mean(r) * 100 for level, r in result_data.items()}

    plt.figure(figsize=(6, 4))
    plt.bar(data.keys(), data.values(), edgecolor="black", color="skyblue")

    plt.xlabel("Nivel del Pokemon")
    plt.ylabel("Probabilidad de captura")
    plt.title(f"Efecto del Nivel en la Captura de {pokemon.capitalize()}")

    plt.grid(axis="y", linestyle="--", alpha=0.7)
    plt.ylim(0, 100)

    for i, rate in enumerate(data.values()):
        plt.text(i, rate, f"{rate:.2f}%", ha="center", va="bottom")

    plt.tight_layout()
    plt.show()
```





[17]:
```
El parametro que mas afecta la probabilidad de captura es el nivel de vida de␣
↪un pokemon seguido por su estado de salud y por ultimo su nivel.
```

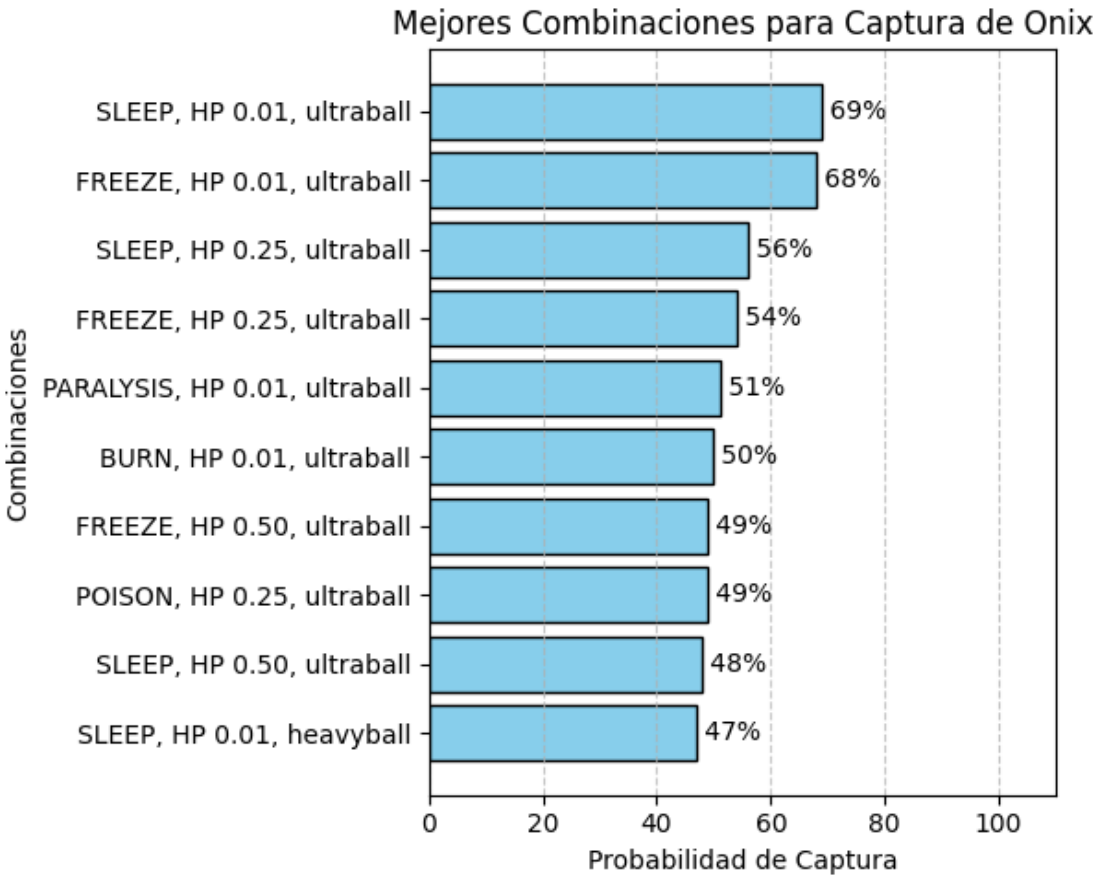
3.6 Punto 2d

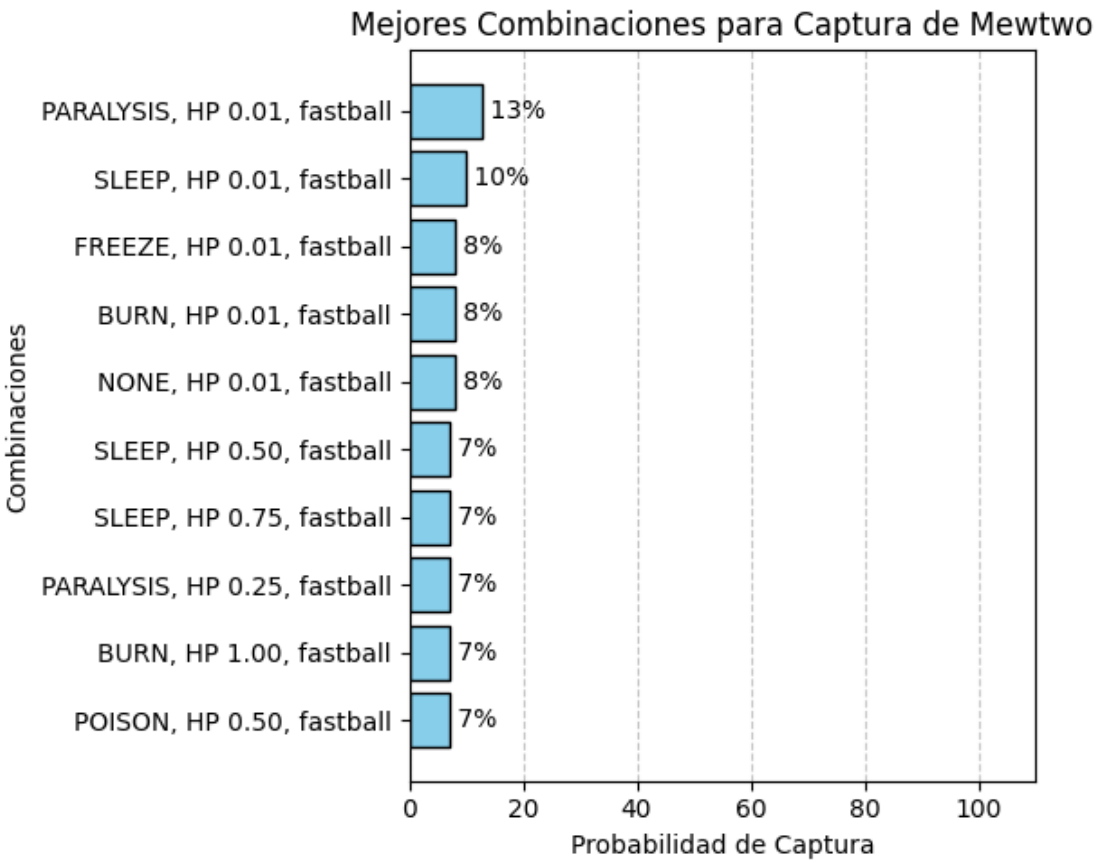
Ya analizamos como afectan a la efectividad la variación de efectos de salud, puntos de vida restantes y el nivel del **Pokemon**. Veamos ahora como la combinación de diferentes valores de puntos de vida, estados de salud y **Pokeballs** diferentes.

[18]:
```
results = {}

for pokemon_name in ["onix", "mewtwo"]:
    results[pokemon_name] = []

    for effect in effects:
        for health in healths:
            for ball in pokeballs:
                pokemon = factory.create(pokemon_name, 100, effect,␣
↪float(health))
                probs = []

                for _ in range(100):
                    success, _ = attempt_catch(pokemon, ball)
```

```
                    probs.append(success)

                results[pokemon_name].append((effect.name, health, ball, np.
 ↪mean(probs) * 100))
```

[19]:
```
for pokemon, result_data in results.items():
    data = sorted(result_data, key=lambda x: x[3])[-10:]
    labels = [f"{x[0]}, HP {x[1]}, {x[2]}" for x in data]
    probs = [x[3] for x in data]

    plt.figure(figsize=(6, 5))
    plt.barh(labels, probs, edgecolor="black", color="skyblue")

    plt.xlabel("Probabilidad de Captura")
    plt.ylabel("Combinaciones")
    plt.title(f"Mejores Combinaciones para Captura de {pokemon.capitalize()}")

    plt.grid(axis="x", linestyle="--", alpha=0.7)
    plt.xlim(0, 110)

    for i, rate in enumerate(data):
        plt.text(rate[3], i, f" {rate[3]:.0f}%", ha="left", va="center")

    plt.tight_layout()
    plt.show()
```


Mejores Combinaciones para Captura de Onix
SLEEP, HP 0.01, ultraball
FREEZE, HP 0.01, ultraball
SLEEP, HP 0.25, ultraball
FREEZE, HP 0.25, ultraball
PARALYSIS, HP 0.01, ultraball
BURN, HP 0.01, ultraball
FREEZE, HP 0.50, ultraball
POISON, HP 0.25, ultraball
SLEEP, HP 0.50, ultraball
SLEEP, HP 0.01, heavyball
69%
68%
56%
54%
51%
50%
49%
49%
48%
47%
Combinaciones
0
20
40
60
80
100
Probabilidad de Captura



Como podemos ver para el caso del **Onix**, la mejor combinación es un bajo nivel de vida (1%) y utilizar la **Ultraball**. Luego, como se puede ver, que el **Pokemon** se encuentre en estado de *sueño* o *congelamiento* afectan positivamente a la efectividad de la captura siendo las combinaciones más efectivas las siguientes:

- Sueño - 1% HP - Ultraball
- Congelamiento - 1% HP - Ultraball

Por otro lado, para el caso de **Mewtwo**, el condicionante más relevante es la utilización de la **Fastball**. Luego, lógicamente además de la **Pokeball** utilizada, el hecho de que tenga poca vida, aumenta la efectividad de la captura, sin embargo, los efectos de los puntos de vida restantes no son tan relevantes. El efecto de *parálisis* es el que más aumenta la efectividad, luego viene *congelamiento*. Por lo tanto las combinaciones más efectivas son:

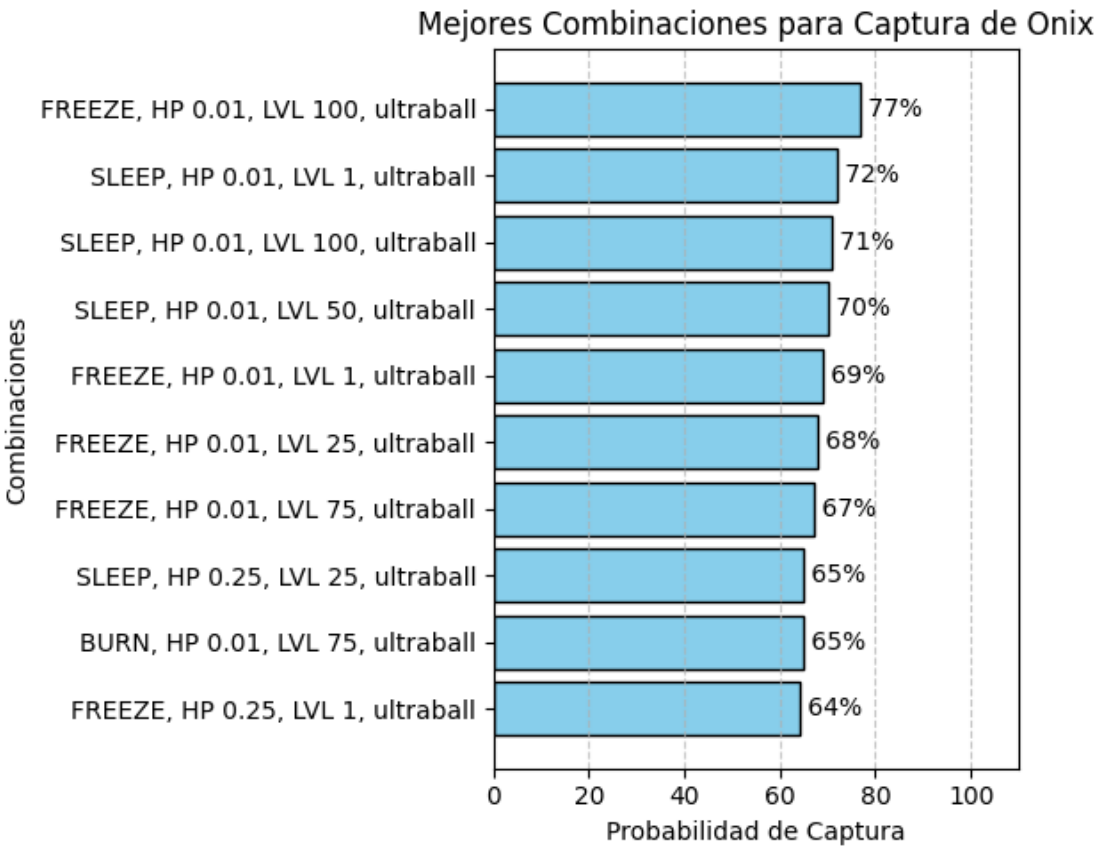
- Parálisis - 1% HP - Fastball
- Congelamiento - 25% HP - Fastball
- Congelamiento - 1% HP - Fastball

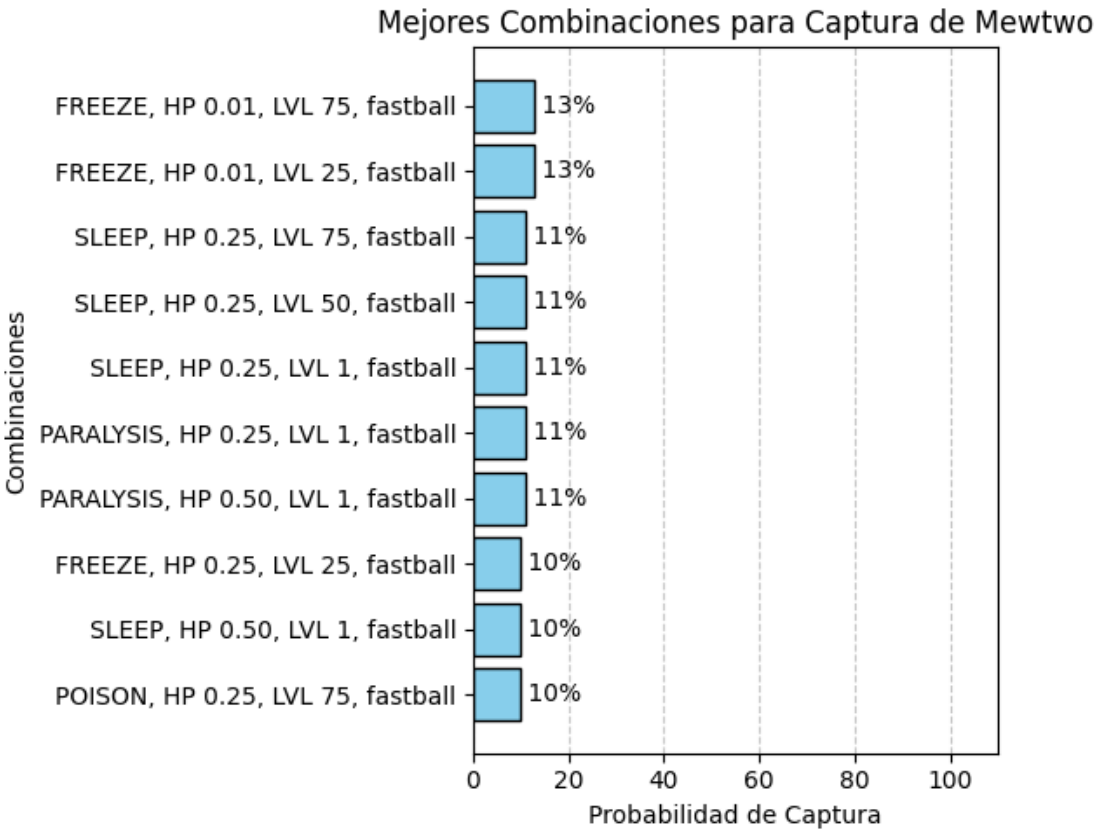
3.7 Punto 2e

Ya comparamos la combinación de efectos de salud, niveles de vida y tipos de **Pokeballs**. Veamos ahora que pasa si agregamos a la combinación, el nivel del **Pokemon**.

[20]:
```
results = {}

for pokemon_name in ["onix", "mewtwo"]:
    results[pokemon_name] = []

    for effect in effects:
        for health in healths:
            for level in levels:
                for ball in pokeballs:
                    pokemon = factory.create(pokemon_name, int(level), effect,␣
 ↪float(health))
                    probs = []

                    for _ in range(100):
                        success, _ = attempt_catch(pokemon, ball)
                        probs.append(success)

                    results[pokemon_name].append((effect.name, health, level,␣
 ↪ball, np.mean(probs) * 100))
```

[21]:
```
for pokemon, result_data in results.items():
    data = sorted(result_data, key=lambda x: x[4])[-10:]
    labels = [f"{x[0]}, HP {x[1]}, LVL {x[2]}, {x[3]}" for x in data]
    probs = [x[4] for x in data]

    plt.figure(figsize=(6, 5))
    plt.barh(labels, probs, edgecolor="black", color="skyblue")

    plt.xlabel("Probabilidad de Captura")
    plt.ylabel("Combinaciones")
    plt.title(f"Mejores Combinaciones para Captura de {pokemon.capitalize()}")

    plt.grid(axis="x", linestyle="--", alpha=0.7)
    plt.xlim(0, 110)

    for i, rate in enumerate(data):
        plt.text(rate[4], i, f" {rate[4]:.0f}%", ha="left", va="center")

    plt.tight_layout()
    plt.show()
```


Mejores Combinaciones para Captura de Onix
FREEZE, HP 0.01, LVL 100, ultraball
SLEEP, HP 0.01, LVL 1, ultraball
SLEEP, HP 0.01, LVL 100, ultraball
SLEEP, HP 0.01, LVL 50, ultraball
FREEZE, HP 0.01, LVL 1, ultraball
FREEZE, HP 0.01, LVL 25, ultraball
FREEZE, HP 0.01, LVL 75, ultraball
SLEEP, HP 0.25, LVL 25, ultraball
BURN, HP 0.01, LVL 75, ultraball
FREEZE, HP 0.25, LVL 1, ultraball
77%
72%
71%
70%
69%
68%
67%
65%
65%
64%
Combinaciones
0
20
40
60
80
100
Probabilidad de Captura



Como se puede observar, el hecho de variar el nivel del **Pokemon** afecta a la efectividad de captura. Evidentemente a niveles más bajos, la efectividad es más alta, esto se puede ver evidentemente tanto para **Onix** como para **Mewtwo**. Por lo tanto si se puede concluir que es efectiva la variación del nivel a un valor menor que el máximo.