tp0

August 13, 2023

[81]:
```
import pandas as pd
import numpy as np
import plotly.express as px
import plotly.io as pio
pio.renderers.default += "+pdf"  # Renderer for PDF exports
from src.catching import attempt_catch
from src.pokemon import PokemonFactory, StatusEffect
```

[82]:
```
factory = PokemonFactory("pokemon.json")
df = pd.read_json("pokemon.json")
pokemons = list(df.columns)
pokeballs = ["pokeball", "ultraball", "fastball", "heavyball"]
```

[83]:
```
def estimate_catchrate(pokemon_instance, pokeball, noise, n):
    return np.average([attempt_catch(pokemon_instance, pokeball, noise)[0] for␣
 ↪_ in range(n)])
```

0.0.1 a) Ejecutando la función 100 veces, para cada Pokemon en condiciones ideales (HP:100 %, LVL 100) ¿Cuál es la probabilidad de captura promedio para cada pokebola?

[84]:
```
data = {}
for pokemon in pokemons:
    bicho = factory.create(pokemon, 100, StatusEffect.NONE, 1)
    data[pokemon] = {}
    for ball in pokeballs:
        data[pokemon][ball] = estimate_catchrate(bicho, ball, 0, 100)
print(data)
```

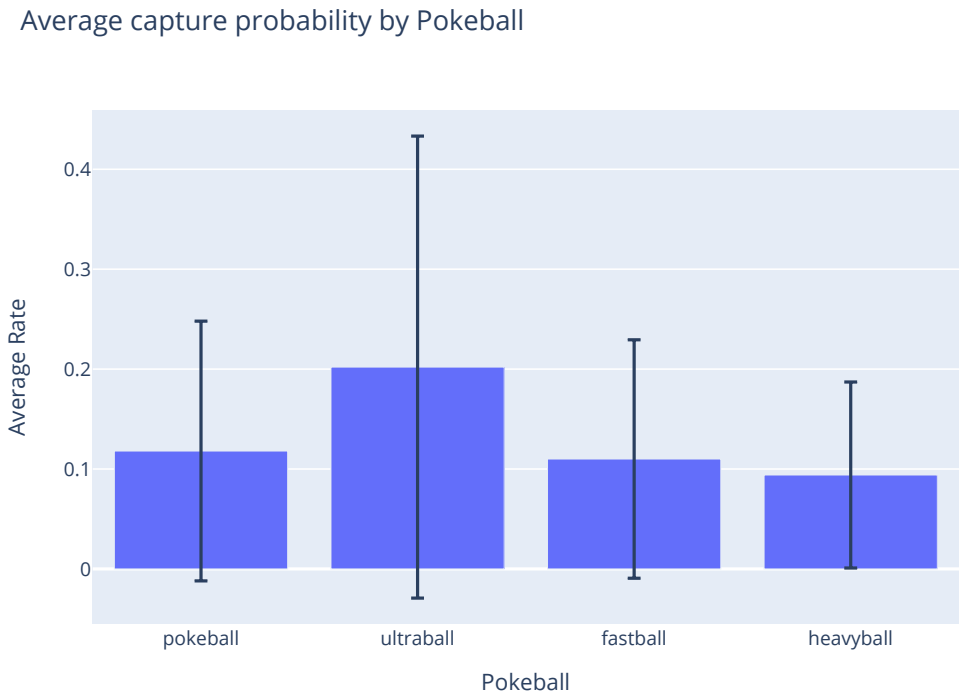
```
{'jolteon': {'pokeball': 0.1, 'ultraball': 0.14, 'fastball': 0.2, 'heavyball':
0.04}, 'caterpie': {'pokeball': 0.37, 'ultraball': 0.65, 'fastball': 0.3,
'heavyball': 0.27}, 'snorlax': {'pokeball': 0.06, 'ultraball': 0.06, 'fastball':
0.04, 'heavyball': 0.09}, 'onix': {'pokeball': 0.06, 'ultraball': 0.16,
'fastball': 0.01, 'heavyball': 0.07}, 'mewtwo': {'pokeball': 0.0, 'ultraball':
0.0, 'fastball': 0.0, 'heavyball': 0.0}}
```

[85]:
```
average_rates = {
```

```
    ball: np.average([values[ball] for values in data.values()]) for ball in␣
  ↪pokeballs
}
errors = {
    ball: np.std([values[ball] for values in data.values()]) for ball in␣
  ↪pokeballs
}

df = pd.DataFrame({'Pokeball': pokeballs,
                   'Average Rate': list(average_rates.values()),
                   'Error': list(errors.values())})

print(df)
```

```
    Pokeball  Average Rate     Error
0   pokeball         0.118  0.129985
1  ultraball         0.202  0.231206
2   fastball         0.110  0.119331
3  heavyball         0.094  0.093081
```

[86]:
```
fig = px.bar(
    data_frame=df,
    x="Pokeball",
    y="Average Rate",
    title="Average capture probability by Pokeball",
    labels={
        "x": "Pokeball type",
        "y": "Average capture rate"
    },
    error_y="Error"
)
fig.show()
```



Está clara la diferencia de efectividad entre pokebolas. Por ejemplo, la Ultraball tiene cerca del doble de efectividad que la Pokebola común. Sin embargo se pueden apreciar diferencias significativas en el error. Esto se debe a que no se segregan los datos por Pokemon, por lo cual la diferencia entre ellos se ve marcada en el error.

0.0.2 1b) ¿Es cierto que algunas pokebolas son más o menos efectivas dependiendo de propiedades intrinsecas de cada Pokemon? Justificar.

[87]:
```
data = {}
for pokemon in pokemons:
    bicho = factory.create(pokemon, 100, StatusEffect.NONE, 1)
    data[pokemon] = {}
    for ball in pokeballs:
        data[pokemon][ball] = estimate_catchrate(bicho, ball, 0, 3000)
print(data)
```

```
{'jolteon': {'pokeball': 0.06633333333333333, 'ultraball': 0.123, 'fastball':
0.22633333333333333, 'heavyball': 0.03266666666666666}, 'caterpie': {'pokeball':
0.322, 'ultraball': 0.6793333333333333, 'fastball': 0.32366666666666666,
'heavyball': 0.2906666666666667}, 'snorlax': {'pokeball': 0.033, 'ultraball':
0.06266666666666666, 'fastball': 0.03666666666666667, 'heavyball': 0.085},
'onix': {'pokeball': 0.06133333333333333, 'ultraball': 0.11966666666666667,
'fastball': 0.058666666666666666, 'heavyball': 0.08733333333333333}, 'mewtwo':
```

```
{'pokeball': 0.004333333333333333, 'ultraball': 0.007, 'fastball':
0.015333333333333332, 'heavyball': 0.0023333333333333335}}
```

Volvemos a intentar las capturas, pero con un número de intentos una orden de magnitud mayor. Esto lo hacemos ya que con solo 100 intentos, había pokemons que no eran atrapados, lo cual dificultaba la comparación.

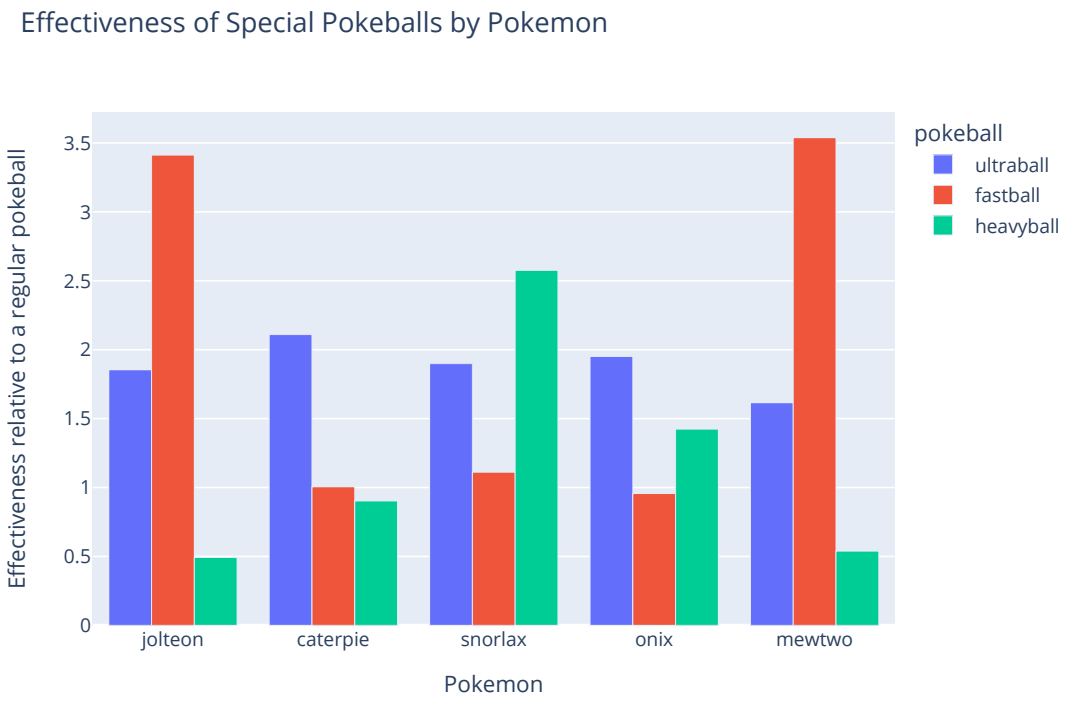
[88]:
```
rates_by_pokemon = {
    pokemon: {
        pokeball: data[pokemon][pokeball] / data[pokemon]["pokeball"] for␣
 ↪pokeball in data[pokemon].keys() if pokeball != "pokeball"
    } for pokemon in data.keys()
}
print(rates_by_pokemon)
```

```
{'jolteon': {'ultraball': 1.8542713567839197, 'fastball': 3.412060301507538,
'heavyball': 0.4924623115577889}, 'caterpie': {'ultraball': 2.109730848861284,
'fastball': 1.005175983436853, 'heavyball': 0.9026915113871636}, 'snorlax':
{'ultraball': 1.8989898989898988, 'fastball': 1.1111111111111112, 'heavyball':
2.5757575757575757}, 'onix': {'ultraball': 1.9510869565217392, 'fastball':
0.9565217391304348, 'heavyball': 1.423913043478261}, 'mewtwo': {'ultraball':
1.6153846153846154, 'fastball': 3.5384615384615383, 'heavyball':
0.5384615384615385}}
```

[89]:
```
table = []
for pokemon in rates_by_pokemon.keys():
    for pokeball in rates_by_pokemon[pokemon].keys():
        table.append([pokemon, pokeball, rates_by_pokemon[pokemon][pokeball]])
df = pd.DataFrame(table, columns=["pokemon", "pokeball", "rate"])
print(df)
```

```
     pokemon   pokeball      rate
0    jolteon  ultraball  1.854271
1    jolteon   fastball  3.412060
2    jolteon  heavyball  0.492462
3   caterpie  ultraball  2.109731
4   caterpie   fastball  1.005176
5   caterpie  heavyball  0.902692
6    snorlax  ultraball  1.898990
7    snorlax   fastball  1.111111
8    snorlax  heavyball  2.575758
9       onix  ultraball  1.951087
10      onix   fastball  0.956522
11      onix  heavyball  1.423913
12    mewtwo  ultraball  1.615385
13    mewtwo   fastball  3.538462
14    mewtwo  heavyball  0.538462
```

[90]:
```
fig = px.bar(
    data_frame=df,
    x="pokemon",
    y="rate",
    title="Effectiveness of Special Pokeballs by Pokemon",
    labels={
        "pokemon": "Pokemon",
        "rate": "Effectiveness relative to a regular pokeball"
    },
    barmode="group",
    color="pokeball"
)
fig.show()
```



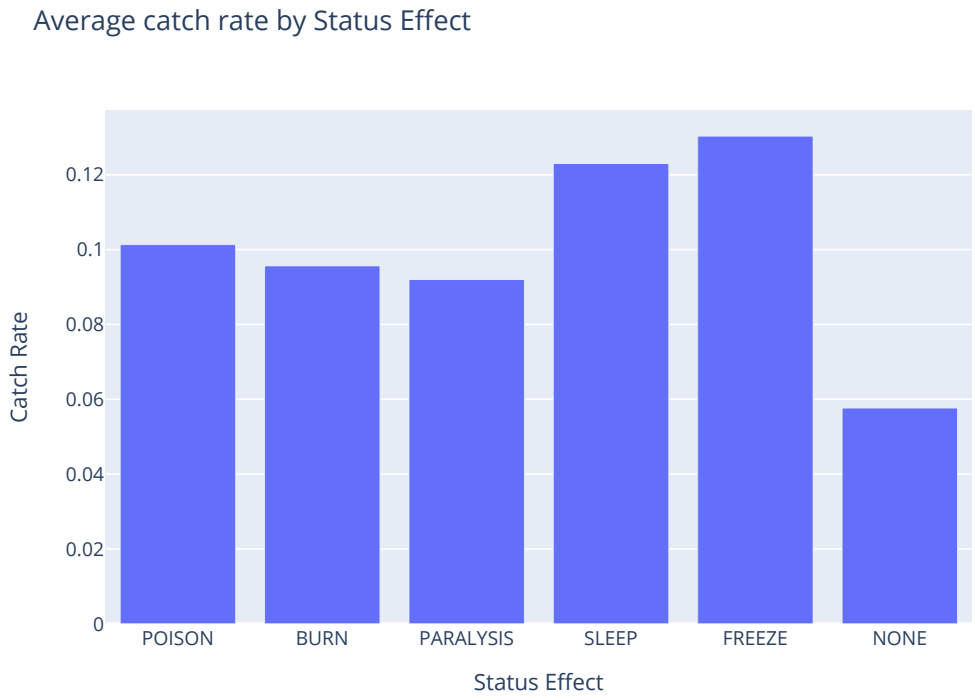
Podemos ver desde el grafico que es cierto que algunos tipos de pokebolas son (en comparacion a la pokebola base) más efectivas en ciertos Pokemones. Por ejemplo, las fastball son mucho mejores contra pokemons como Jolteon y Mewtwo, y las heavyballs son mejores contra Snorlax, pero peores contra Jolteon o Mewtwo. En cambio, la ultraball no parece depender del Pokemon a atrapar.

0.0.3 2a)¿Las condiciones de salud tienen algún efecto sobre la efectividad de la captura? Si es así, ¿Cuál es más o menos efectiva?

[91]:
```
pokemon_test = "jolteon"
variants = {
    status: factory.create(pokemon_test, 100, status, 1) for status in␣
 ↪StatusEffect
}
catchrate_by_variant = {
    status: estimate_catchrate(pokemon, pokeballs[0], 1, 3000) for (status,␣
 ↪pokemon) in variants.items()
}
print(catchrate_by_variant)
```

```
{<StatusEffect.POISON: ('poison', 1.5)>: 0.10133333333333333,
<StatusEffect.BURN: ('burn', 1.5)>: 0.09566666666666666,
<StatusEffect.PARALYSIS: ('paralysis', 1.5)>: 0.092, <StatusEffect.SLEEP:
('sleep', 2)>: 0.123, <StatusEffect.FREEZE: ('freeze', 2)>: 0.13033333333333333,
<StatusEffect.NONE: ('none', 1)>: 0.057666666666666665}
```

[92]:
```
fig = px.bar(
    x = list(map(lambda status : status.name, catchrate_by_variant.keys())),
    y = list(catchrate_by_variant.values()),
    title = "Average catch rate by Status Effect",
    labels= {
        "x": "Status Effect",
        "y": "Catch Rate"
    }
)
fig.show()
```



Podemos ver que los más efectivos son SLEEP y FREEZE. Mientras que POISON, BURN y PARALYSIS no llegan a ser tan efectivos. Sin embargo, todos los status es mejor que no tener ningún estado afectandolo.

0.0.4 2b) ¿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 %

[93]:
```
def get_catchrate_with_error(pokemon_instance, pokeball, noise, n):
    values = [attempt_catch(pokemon_instance, pokeball, noise)[1] for _ in␣
 ↪range(n)]
    return {
        "rate": np.average(values),
        "error": np.std(values)
    }
```

[94]:
```
import decimal

def drange(x, y, jump):
  while x < y + jump:
    yield float(x)
    x += decimal.Decimal(jump)
```

```
health_vars = {
    percentage: factory.create(pokemon_test, 100, StatusEffect.NONE,␣
  ↪percentage) for percentage in drange(0, 1, 0.05)
}

catchrate_by_health = {
    percentage: get_catchrate_with_error(pokemon, pokeballs[0], 0.15, 3000) for␣
  ↪(percentage, pokemon) in health_vars.items()
}
print(catchrate_by_health)
```

```
{0.0: {'rate': 0.17528505975333686, 'error': 0.025925570123004945}, 0.05:
{'rate': 0.1711039561625922, 'error': 0.025209271258806124}, 0.1: {'rate':
0.16484045529875768, 'error': 0.02484476373161205}, 0.15000000000000002:
{'rate': 0.15896439619578734, 'error': 0.023270342015779735}, 0.2: {'rate':
0.1532530227149025, 'error': 0.02334160814257883}, 0.25: {'rate':
0.14672555479640506, 'error': 0.021877892504154617}, 0.30000000000000004:
{'rate': 0.1404786843264369, 'error': 0.02158793107002247}, 0.35000000000000003:
{'rate': 0.13534592307193957, 'error': 0.020471166396412013}, 0.4: {'rate':
0.12919648185229854, 'error': 0.01972929595920458}, 0.45: {'rate':
0.12332871683762528, 'error': 0.018692132657800675}, 0.5: {'rate':
0.11777830478333423, 'error': 0.01770413786469458}, 0.55: {'rate':
0.11133076845371431, 'error': 0.01665911103321095}, 0.6000000000000001: {'rate':
0.10583951429894262, 'error': 0.01614166580973011}, 0.65: {'rate':
0.10009583920049919, 'error': 0.014734739248776333}, 0.7000000000000001:
{'rate': 0.09447393371704021, 'error': 0.01397444736255454}, 0.75: {'rate':
0.0882212650764073, 'error': 0.013388664044984504}, 0.8: {'rate':
0.08280357875188948, 'error': 0.012456832797024114}, 0.8500000000000001:
{'rate': 0.07708814737534425, 'error': 0.011540776948010138}, 0.9: {'rate':
0.0711806876396498, 'error': 0.010753703905440516}, 0.9500000000000001: {'rate':
0.06500297863407116, 'error': 0.009623977384582594}, 1.0: {'rate':
0.0587465588835596, 'error': 0.008942972224241815}}
```

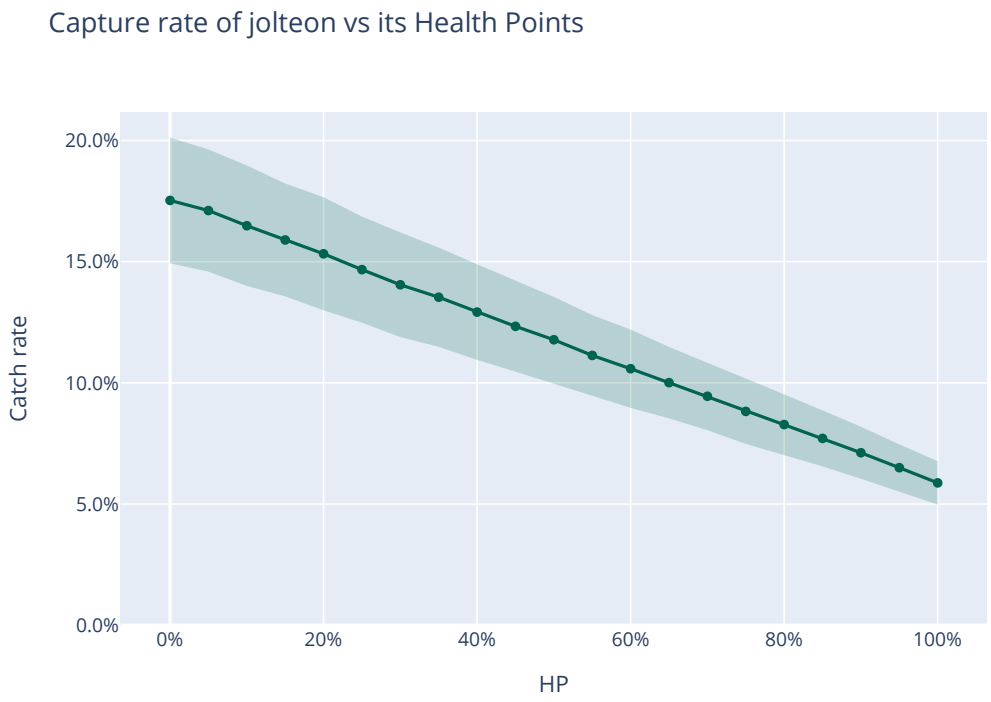
[95]:
```
values = {
    "x": [],
    "y": [],
    "upper_y": [],
    "lower_y": []
}
for item in catchrate_by_health.items():
    values["x"].append(item[0])
    values["y"].append(item[1]["rate"])
    values["upper_y"].append(item[1]["rate"] + item[1]["error"])
    values["lower_y"].append(item[1]["rate"] - item[1]["error"])
print(values)
```

```
{'x': [0.0, 0.05, 0.1, 0.15000000000000002, 0.2, 0.25, 0.30000000000000004,
```

```
0.35000000000000003, 0.4, 0.45, 0.5, 0.55, 0.6000000000000001, 0.65,
0.7000000000000001, 0.75, 0.8, 0.8500000000000001, 0.9, 0.9500000000000001,
1.0], 'y': [0.17528505975333686, 0.1711039561625922, 0.16484045529875768,
0.15896439619578734, 0.1532530227149025, 0.14672555479640506,
0.1404786843264369, 0.13534592307193957, 0.12919648185229854,
0.12332871683762528, 0.11777830478333423, 0.11133076845371431,
0.10583951429894262, 0.10009583920049919, 0.09447393371704021,
0.0882212650764073, 0.08280357875188948, 0.07708814737534425,
0.0711806876396498, 0.06500297863407116, 0.0587465588835596], 'upper_y':
[0.2012106298763418, 0.1963132274213983, 0.18968521903036972,
0.18223473821156708, 0.1765946308574813, 0.16860344730055968,
0.16206661539645936, 0.1558170894683516, 0.14892577781150312,
0.14202084949542595, 0.1354824426480288, 0.12798987948692525,
0.12198118010867273, 0.11483057844927552, 0.10844838107959476,
0.1016099291213918, 0.0952604115489136, 0.0886289243233544, 0.08193439154509032,
0.07462695601865375, 0.06768953110780142], 'lower_y': [0.1493594896303319,
0.14589468490378607, 0.13999569156714564, 0.1356940541800076,
0.12991141457232366, 0.12484766229225044, 0.11889075325641443,
0.11487475667552756, 0.10946718589309395, 0.1046365841798246,
0.10007416691863966, 0.09467165742050336, 0.0896978484892125,
0.08536109995172286, 0.08049948635448567, 0.07483260103142281,
0.07034674595486537, 0.0655473704273341, 0.06042698373420928,
0.055379001249488566, 0.04980358669317785]}
```

[96]:
```
import plotly.graph_objs as go

fig = go.Figure([
    go.Scatter(
        x=values["x"],
        y=values["y"],
        line=dict(color='rgb(0,100,80)'),
        mode='lines+markers',
        showlegend=False,
        name="average rate"
    ),
    go.Scatter(
        x=values["x"]+values["x"][::-1], # x, then x reversed
        y=values["upper_y"]+values["lower_y"][::-1], # upper, then lower␣
 ↪reversed
        fill='toself',
        fillcolor='rgba(0,100,80,0.2)',
        line=dict(color='rgba(255,255,255,0)'),
        hoverinfo="skip",
        showlegend=False,
    )
])
fig.update_layout(
```

```
    title = f"Capture rate of {pokemon_test} vs its Health Points",
    xaxis_title="HP",
    yaxis_title="Catch rate",
)
fig.update_yaxes(
    rangemode="tozero",
    tickformat=".1%",
)
fig.update_xaxes(
    tickformat=".0%"
)
fig.show()
```



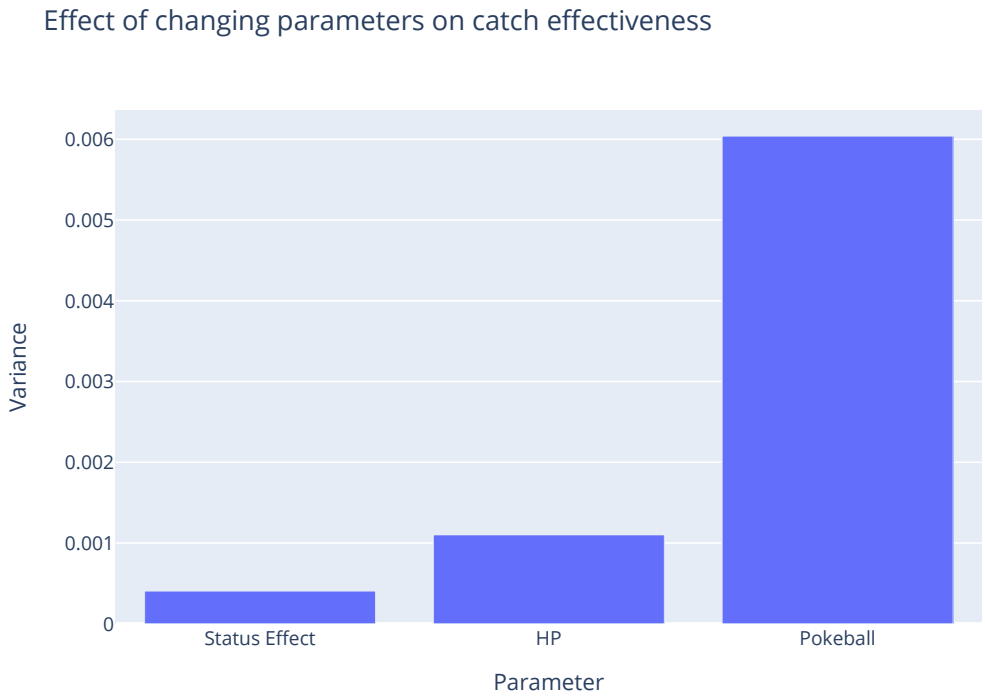
Podemos ver que a medida que aumentamos la vida del pokemon, menor será la posibilidad de captura de manera prácticamente lineal.

0.0.5 2c) ¿Qué parámetros son los que más afectan la probabilidad de captura?

Para ver que parámetros afectan más a la probabilidad de captura, calculamos la varianza de los valores calculados variando el parámetro y manteniendo los otros constantes. Los parametros son: - La vida - El Efecto de estado - La pokebola

Tomamos un Jolteon como ejemplo.

[97]:
```
data_status = []
data_hp = []
data_pb = []
for hp in range(10):
    data_hp.append(attempt_catch(
        factory.create(pokemon_test, 100, StatusEffect.NONE, hp*0.1),
        pokeballs[0]
    )[1])
for status in StatusEffect:
    data_status.append(attempt_catch(
        factory.create(pokemon_test, 100, status, 1),
        pokeballs[0]
    )[1])
for pb in pokeballs:
    data_pb.append(attempt_catch(
        factory.create(pokemon_test, 100, StatusEffect.NONE, 1),
        pb
    )[1])

data_var = [np.var(data_status), np.var(data_hp), np.var(data_pb)]
df = pd.DataFrame(data_var,index=['Status␣
 ↪Effect','HP','Pokeball'],columns=['Variance'])
px.bar(df,x=df.index,y="Variance",title="Effect of changing parameters on catch␣
 ↪effectiveness",labels={
    "index": "Parameter"
}).show()
```

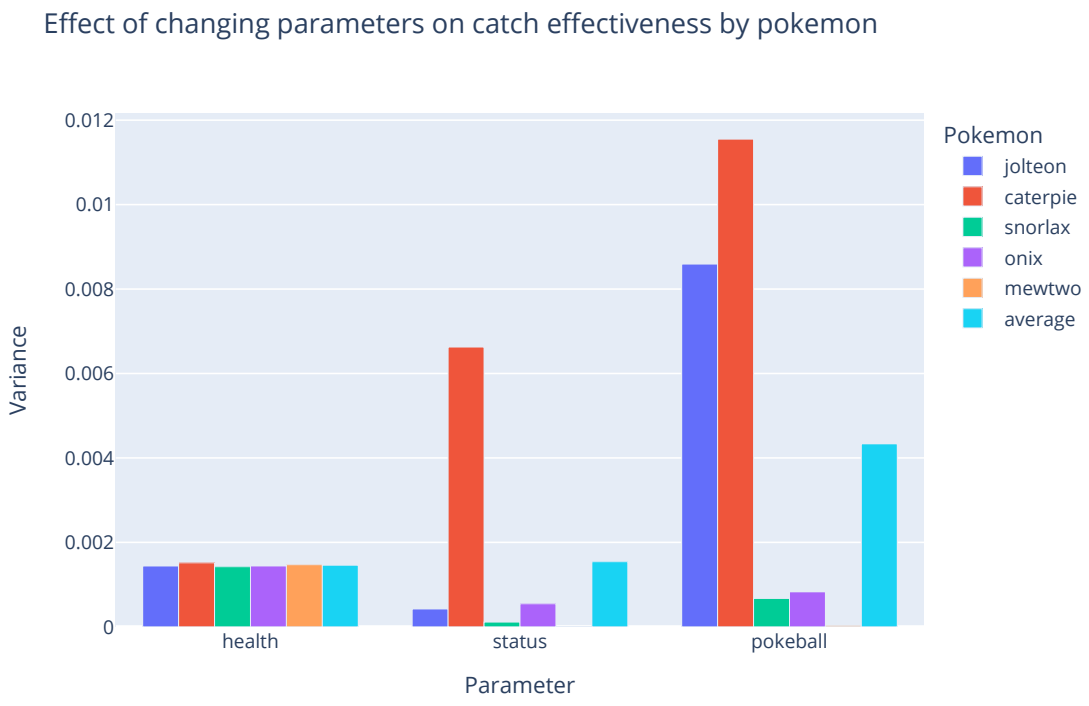


Para Jolteon en particular, la mayor varianza se obtiene cambiando el tipo de pokebola que se utiliza. Debido a que la efectividad de una pokebola depende de los atributos del pokemon, esto puede cambiar.

[98]:
```
# Funcion devuelve una tupla (varianza_por_tipo, max y minimo por tipo)
def calculate_all_variances(pokemon, level, iterations):
    health_vars = { percentage: factory.create(pokemon_test, level,␣
 ↪StatusEffect.NONE, percentage) for percentage in drange(0, 1, 0.05) }
    catchrate_by_health = { percentage: estimate_catchrate(pokemon,␣
 ↪pokeballs[0], 1, iterations) for (percentage, pokemon) in health_vars.
 ↪items() }

    variants = { status: factory.create(pokemon, level, status, 1) for status␣
 ↪in StatusEffect }
    catchrate_by_variant = { status.name: estimate_catchrate(pokemon,␣
 ↪pokeballs[0], 1, iterations) for (status, pokemon) in variants.items() }

    healthy_pokemon = factory.create(pokemon, level, StatusEffect.NONE, 1)
    catchrate_by_pokeball = { pokeball: estimate_catchrate(healthy_pokemon,␣
 ↪pokeball, 1, 3000) for pokeball in pokeballs }
    return {
        "health": np.var(list(catchrate_by_health.values())),
```

```
        "status": np.var(list(catchrate_by_variant.values())),
        "pokeball": np.var(list(catchrate_by_pokeball.values()))
    }

all_pokemons_variances = { pokemon: calculate_all_variances(pokemon, 100, 3000)␣
 ↪for pokemon in pokemons }

df = pd.DataFrame.from_dict(all_pokemons_variances)
df = df.assign(average=df.mean(axis=1))
print(df)
fig5 = px.bar(df, barmode="group",title="Effect of changing parameters on catch␣
 ↪effectiveness by pokemon",labels={
    "index": "Parameter",
    "value": "Variance",
    "variable": "Pokemon"
})
fig5.show()
```

	jolteon	caterpie	snorlax	onix	mewtwo	average
health	0.001441	0.001516	0.001424	0.001441	0.001471	0.001459
status	0.000423	0.006627	0.000112	0.000547	0.000004	0.001542
pokeball	0.008592	0.011551	0.000672	0.000829	0.000028	0.004334



El gráfico nos muestra cuáles son las varianzas por cada Pokemon. Podemos ver que caterpie es afectado mucho más por el estado o la pokebola, mientras que mewtwo no es prácticamente afectado por estos parámetros. Por otro lado, la incidencia de la vida del pokemon en la efectividad no parece depender del pokemon en cuestión. Agarrando el promedio de todas las varianzas, parece que la vida y la pokebola afectan bastante en comparacion al estado y nivel.

0.0.6 2d) Teniendo en cuenta uno o dos pokemones distintos: ¿Qué combinación de condiciones (propiedades mutables) y pokebola conviene utilizar para capturarlos?

[99]:
```
max_catchrate = {
    pokemon: { "rate" : 0 } for pokemon in pokemons
}
for status in StatusEffect:
    for pokeball in pokeballs:
        for pokemon in max_catchrate.keys():
            catchrate = attempt_catch(
                factory.create(pokemon, 100, status, 1),
                pokeball
            )[1]
            if max_catchrate[pokemon]["rate"] < catchrate:
                max_catchrate[pokemon] = {
                    "rate": catchrate,
                    "status": status.name,
                    "pokeball": pokeball
                }
print(pd.DataFrame.from_dict(max_catchrate))
```

```
           jolteon   caterpie    snorlax       onix    mewtwo
rate        0.4688          1     0.1693     0.2344    0.0313
status       SLEEP      SLEEP      SLEEP      SLEEP     SLEEP
pokeball  fastball  ultraball  heavyball  ultraball  fastball
```

Considerando que entre más bajo sean los HP de los Pokemon, mayor será la chance de captura y las condiciones no cambian su efectividad en base al nivel de salud, directamente utilizamos HP = 100%. Podemos apreciar que SLEEP es el estado que más efectivo en general, junto con FREEZE, sin importar qué Pokemon estamos analizando. Sin embargo, la Pokebola a utilizar sí tiene grandes incidencias dependiendo del Pokemon que estemos analizando. Por ejemplo, la Heavyball es significativamente mejor en Snorlax que en Jolteon, para el cual la mejor pokebola es la Fastball. Por lo tanto podemos afirmar que la Pokebola depende del Pokemon, mientras que siempre conviene utilizar SLEEP o FREEZE. Independientemente de esto, siempre se debe buscar que el Pokemon tenga el menor porcentaje de salud posible.

0.0.7 2e) A partir del punto anterior, ¿sería efectiva otra combinación de parámetros teniendo en cuenta un nivel del pokemon más bajo (o más alto)?

Repetimos el anterior pero con un nivel de 40 en vez de 100

[100]:
```
max_catchrate = {
    pokemon: { "rate" : 0 } for pokemon in pokemons
}
for status in StatusEffect:
    for pokeball in pokeballs:
        for pokemon in max_catchrate.keys():
            catchrate = attempt_catch(
                factory.create(pokemon, 40, status, 1),
                pokeball
            )[1]
            if max_catchrate[pokemon]["rate"] < catchrate:
                max_catchrate[pokemon] = {
                    "rate": catchrate,
                    "status": status.name,
                    "pokeball": pokeball
                }
print(pd.DataFrame.from_dict(max_catchrate))
```

```
           jolteon   caterpie    snorlax       onix    mewtwo
rate        0.4688          1     0.1693     0.2344    0.0313
status       SLEEP      SLEEP      SLEEP      SLEEP     SLEEP
pokeball  fastball  ultraball  heavyball  ultraball  fastball
```

El nivel no parecería afectar qué parametros usar.