2. Vizualizace dat

Zadání:

V jednom ze cvičení jste probírali práci s moduly pro vizualizaci dat. Mezi nejznámější moduly patří matplotlib (a jeho nadstavby jako seaborn), pillow, opencv, aj. Vyberte si nějakou zajímavou datovou sadu na webovém portále Kaggle a proveďte datovou analýzu datové sady. Využijte k tomu různé typy grafů a interpretujte je (minimálně alespoň 5 zajímavých grafů). Příklad interpretace: z datové sady pro počasí vyplynulo z liniového grafu, že v létě je vyšší rozptyl mezi minimální a maximální hodnotou teploty. Z jiného grafu vyplývá, že v létě je vyšší průměrná vlhkost vzduchu. Důvodem vyššího rozptylu může být absorpce záření vzduchem, který má v létě vyšší tepelnou kapacitu.

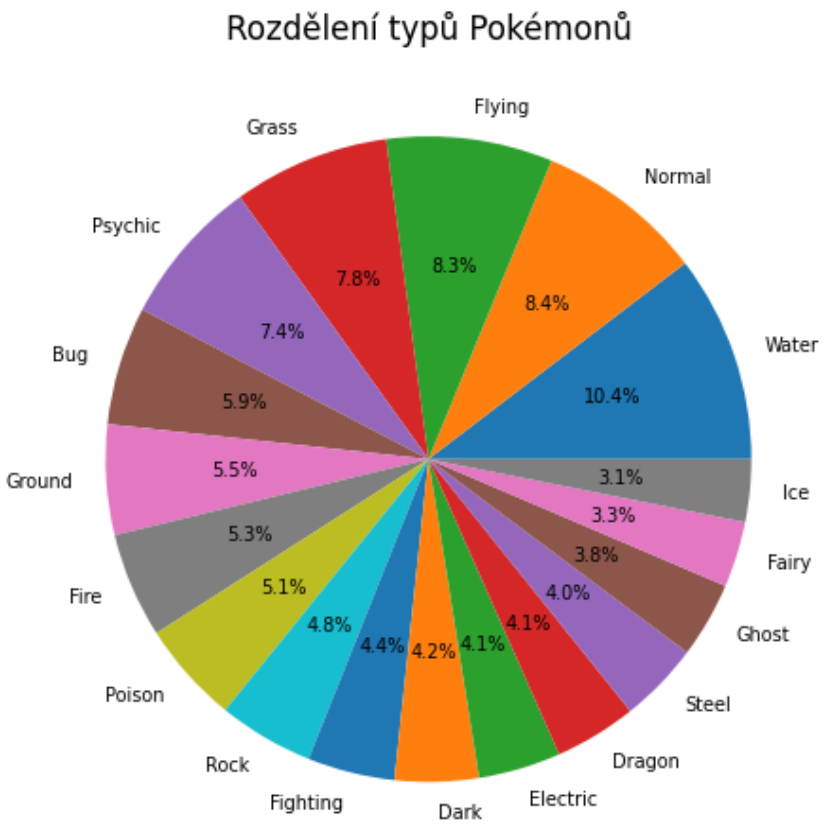
Řešení:

Rozhodla jsem se použít knihovnu matplotlib k vizualizaci dat z datové sady Pokémonů stažené z webové stránky Kaggle (odkaz: https://www.kaggle.com/datasets/abcsds/pokemon). V datové sadě se nacházely kolonky ID, Name, Type 1, Type 2, Total (sečtené všechny hlavní staty), HP, Attack, Defense, SP. Atk, SP. Def, Speed, Generation, Legendary. S těmito informacemi jsem vymyslela různé grafy k interpretaci. Jako první jsem si naimportovala potřebné knihovny a uložila si datovou sadu do proměnné.

První kód a graf:

První graf nám ukazuje „Rozdělení typů Pokémonů“, spočítá tedy, kolikrát se daný typ v datové sadě objeví. Pokémoni můžou mít více než jeden typ a v datové sadě jsou rozděleny do sloupců „Type 1“ a Type 2“, proto jsem si pomocí knihovny pandas tyto sloupce spojila. Concat funkce mi vrátí všechny typy jednou a pomocí value_counts spočítá, kolikrát se v datové sadě objevily. Jednotlivé typy si poté uložím do listu a z těchto dvou proměnných vytvořím koláčový graf.

```
type_counts = pd.concat([pokemon_dataset["Type 1"], pokemon_dataset["Type 2"]]).value_counts()
types = type_counts.index.tolist()

#Vytvoří koláčový graf
plt.pie(type_counts, labels = types, autopct = "%1.1f%%", textprops = {"fontsize": 7})

plt.title("Rozdělení typů Pokémonů")

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

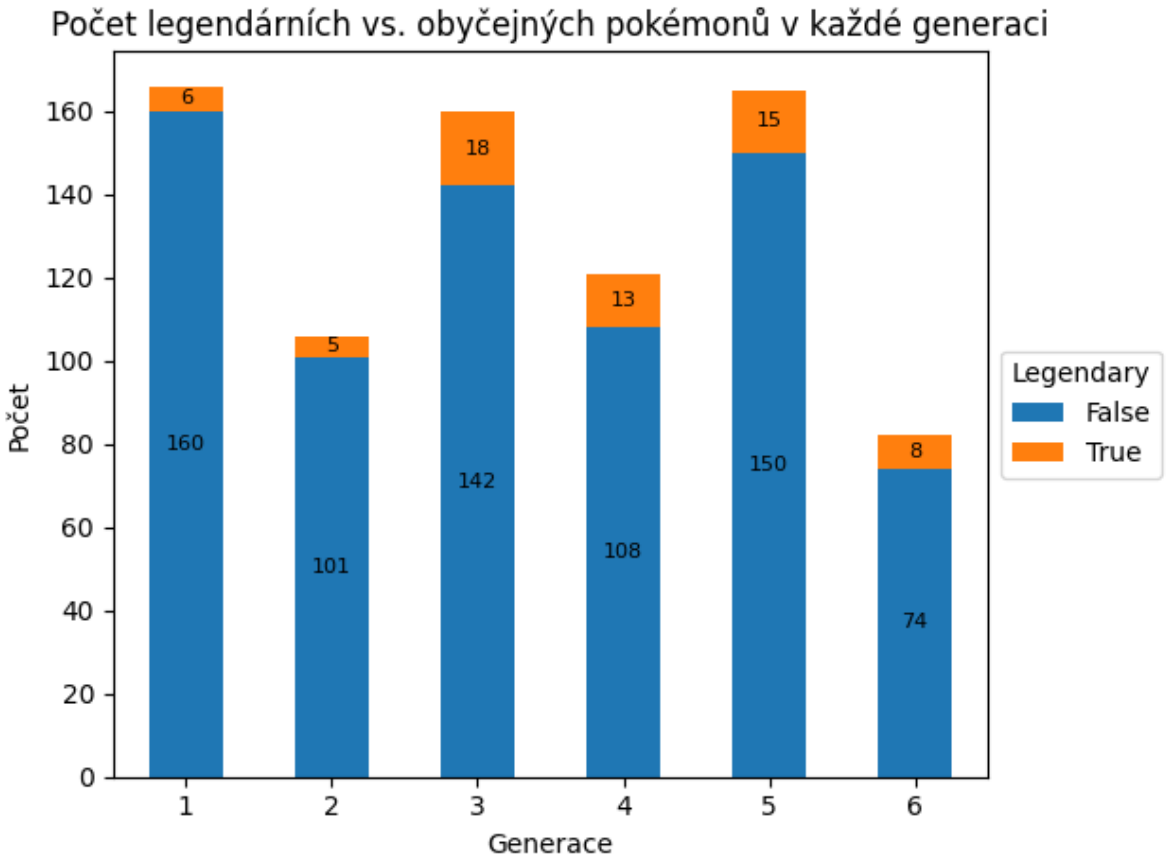


Jako první si můžeme všimnout, že nejčastější typ, který se objevuje je water, který má jediný přes 10 %. Dále jsou časté typy normal, flying, grass a psychic. Naopak jako nejméně časté pod 4 % se objevuje ice, fairy a ghost. V rozmezí od 6 % do 4 % se pak objevuje zbytek 10 typů, které mezi sebou už moc velké procentuální rozdíly nemají. Celkově můžeme říct, že rozdělení typů je ve směs vyrovnané. První a poslední nejčastější typ má mezi sebou rozdíl pouhých 7,3 %, což s ohledem na počet typů, kterých je 18, je celkem málo. Původně bychom mohli čekat, že nejčastějšími typy budou water, grass a fire s ohledem na hry, kde si hráč může vždy na začátku hry vybrat z těchto tří typů pokémonů. U jediného typu water je toto pravda, ovšem grass je až na 4. místě a fire dokonce až na 8. místě.

Druhý kód a graf:

Druhý graf nám ukazuje „Počet legendárních vs. Obyčejných Pokémonů v každé generaci“, spočítá tedy, kolik je legendárních pokémonů v jednotlivých generací. První vytvořím skupinu se sloupci „Generation“ a „Legendary“ pomocí groupby, size mi poté spočítá, kolikrát se všechny unikátní kombinace těchto dvou sloupců objevily. Unstack mi přetvoří data tak, aby se z nich dal vytvořit skládaný sloupcový graf. Poté už jenom upravuji vizuální stránku grafu. Pomocí funkce for projdu všechny jednotlivé sloupce a přidám do nich počet pokémonů. Na konec upravím legendu, aby byla vedle grafu.

```python
generation_legendary_counts = pokemon_dataset.groupby(["Generation", "Legendary"]).size().unstack()

#Vytvoří skládaný sloupcový graf
ax = generation_legendary_counts.plot(kind = "bar", stacked = True)

plt.title("Počet legendárních vs. obyčejných pokémonů v každé generaci")
plt.xlabel("Generace")
plt.ylabel("Počet")

#Přídá přesný počet do sloupců a upraví čísla generací
for container in ax.containers:
    ax.bar_label(container, label_type = "center", fontsize = 8)
plt.xticks(rotation = 0)

#Posune legendu
ax.legend(title = "Legendary", loc = "center left", bbox_to_anchor = (1, 0.5))

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



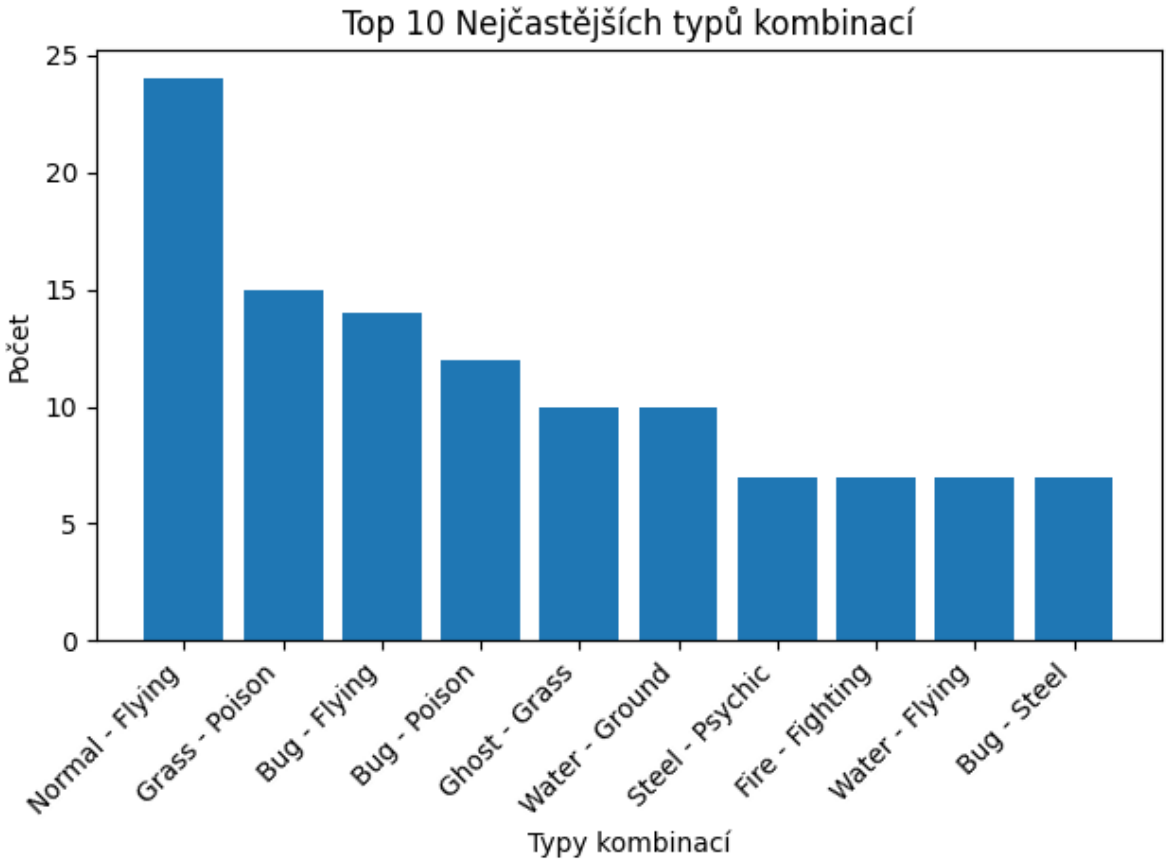
Jako první vidíme, že legendárních pokémonů je v každé generaci méně než obyčejných, což není nic překvapujícího. Ovšem můžeme si na příklad všimnout, že 3. generace má největší počet legendárních pokémonů, i když je až na 3. místě s nejvyšším počtem celkových pokémonů, což nám dává cca 11 % legendárních pokémonů z celkového počtu. V další generaci se sice celkový počet pokémonů snížil a tím pádem se snížil i počet legendárních pokémonů, ale poměr legendárních pokémonů ku celkovému počtu se stále drží v podobných procentuálních hodnotách (9-11 %), stejně tomu tak je i u 5. a 6. generace. Jedinými výjimkami

je 1. a 2. generace, kde je počet legendárních pokémonů z celkového počtu v obou případech pod 5 % u 1. generace dokonce pod 4 %. Celkově tedy můžeme říct, že od 3. generace se objevuje více legendárních pokémonů. Z mého pohledu to měli nejlépe právě první dvě generace. Legendární pokémoni by měli něco výjimečného, co člověk jen tak nemá šanci potkat.

Třetí kód a graf:

Druhý graf nám ukazuje „Top 10 nejčastějších typů kombinací“, spočítá tedy, kolikrát se spolu v kombinaci nachází všechny typy. kolik je legendárních pokémonů v jednotlivých generací. První vytvořím skupinu se sloupci „Type 1“ a „Type 2“ pomocí groupby, size mi poté spočítá, kolikrát se všechny unikátní kombinace těchto dvou sloupců objevily. Reset_index mi přetvoří data tak, aby se z nich dal vytvořit sloupcový graf a vytvoří nový sloupec „Sum“, kam uloží výsledky. Poté už jenom setřídím podle tohoto nového sloupce „Sum“ a vezmu pouze prvních 10 kombinací pomocí head(10). Na konec pouze vytvořím sloupcový graf a upravím popisky na ose x.

```
type_combinations_counts = pokemon_dataset.groupby(["Type 1", "Type 2"]).size().reset_index(name = "Sum")

type_combinations_counts = type_combinations_counts.sort_values("Sum", ascending = False)
top_10_combinations = type_combinations_counts.head(10)

#Vytvoří sloupcový graf
plt.bar(top_10_combinations["Type 1"] + " - " + top_10_combinations["Type 2"], top_10_combinations["Sum"])

plt.title("Top 10 Nejčastějších typů kombinací")
plt.xlabel("Typy kombinací")
plt.ylabel("Počet")

#Otočí a posune jména kombinací
plt.xticks(rotation = 45, ha = "right")

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



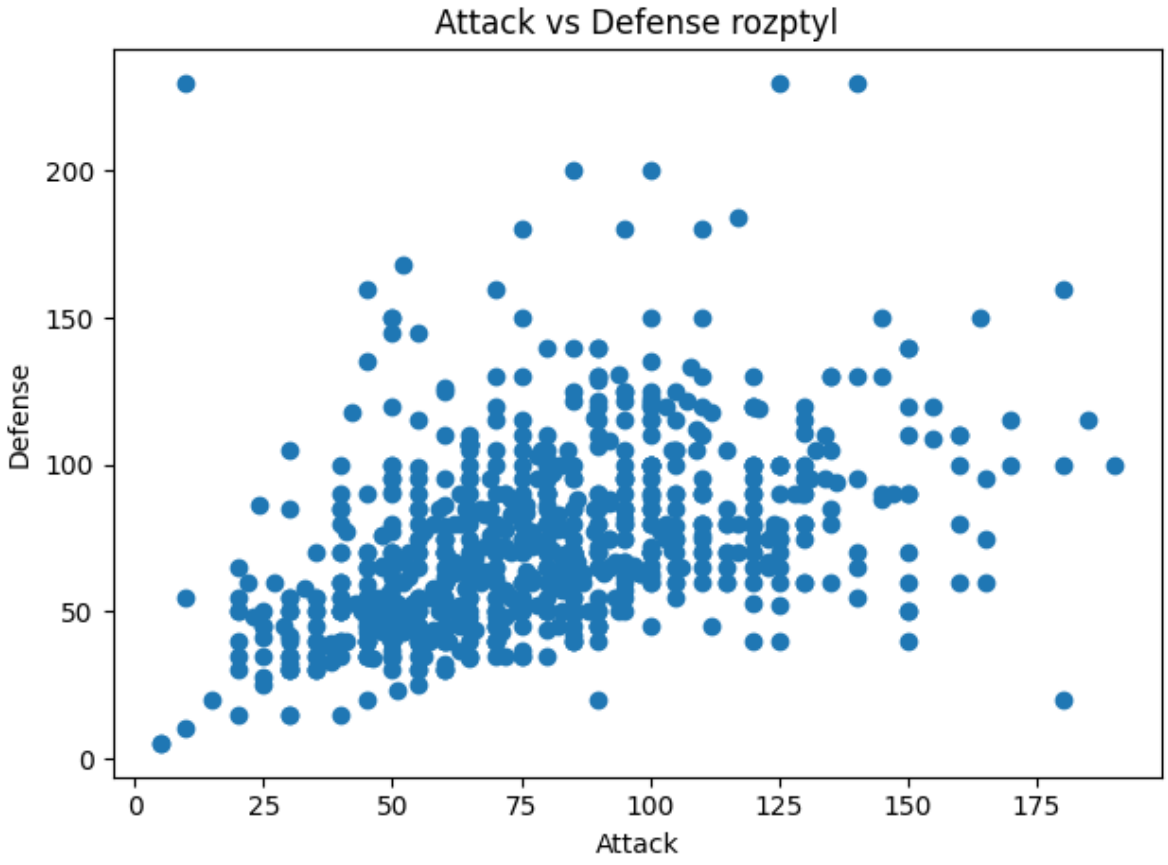
Tento graf je ve velice přímočarý. Vidíme, že nejčastější typ kombinace je normal – flying, který ostatní o dost převyšuje. Poté už jsou kombinace vesměs v podobných počtech, poslední 4 mají dokonce úplně stejný počet pokémonů. Na příklad v kombinaci s prvním grafem „Rozdělení typů Pokémonů“, tak jsme mohli nejčastěji očekávat pokémony s typem water, normal, flying, grass and psychic, což je správné očekávání, protože jsou tady všechny tyto typy zastoupeny.

Tento kód se dá jednoduše přepsat, když budeme chtít na příklad nejméně časté kombinace, tak pouze přepíšeme ascending na True, a nebo v head(10) můžeme 10 nahradit jakýmkoliv jiným počtem, který bychom chtěli ve finálním grafu zobrazit, graf ovšem pak může být velice dlouhý.

Čtvrtý kód a graf:

Třetí graf nám ukazuje „Attack vs Defense rozptyl“, vytvoří tedy na grafu bod pro každého pokémona na základě jeho útoku a obrany. První jsem si do proměnných uložila dva staty, které jsem chtěla porovnávat, v tomto případě „Attack“ a „Defense“. A poté už jsem pouze vytvořila bodový graf.

```python
x_stat = "Attack"
y_stat = "Defense"

#Vytvoří graf rozptylu
plt.scatter(pokemon_dataset[x_stat], pokemon_dataset[y_stat])

plt.title(f"{x_stat} vs {y_stat} rozptyl")
plt.xlabel(x_stat)
plt.ylabel(y_stat)

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



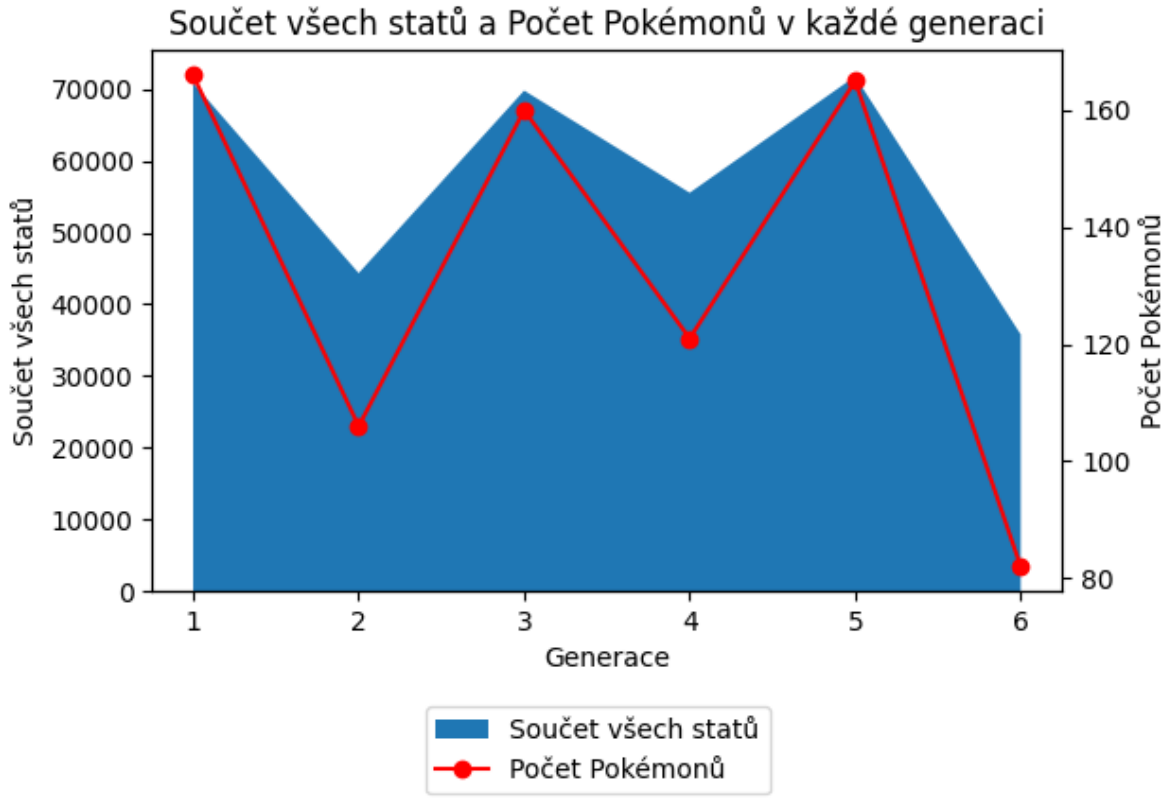
Bodový graf nám vlastně ukazuje závislost mezi dvěma proměnnýma. V tomto případě nám říká, že když se zvyšuje útok pokémona, jestli se stejně zvyšuje i jeho obrana. Jak můžeme vidět z grafu, tak se dá říct, že moc velká závislost mezi těmito dvěma proměnnýma nebude. To bude pravděpodobně tím, že většina pokémonů vyniká právě buď v útoku, anebo obraně, zároveň však stále mají decentní svojí druhou proměnnou. Samozřejmě existují i pokémoni, kteří jsou vybalancování v obojím a ti nám právě vytváří pozitivní korelaci. Zároveň však můžeme vidět i velice extrémní případy, kdy mají někteří pokémoni velice vysokou obranu, ale žádný útok a naopak, ti nám zase vytváří negativní korelaci. Celkově tedy můžeme říct, že jakási závislost zde existuje, ale nebude moc veliká.

Tento kód se dá jednoduše přepsat, když budeme chtít na příklad porovnávat jiné sloupce, tak pouze přepíšeme proměnné x_stat a y_stat.

Pátý kód a graf:

Poslední graf nám ukazuje „Součet všech statů a Počet Pokémonů v každé generaci“. První si tedy vytvořím proměnou, která seskupí data na základě sloupce „Generation“ a sečte sloupce „Total“ pro každou jednotlivou generaci. Dostanu tedy novou řadu s generací jako indexem a staty jako values. Dále spočítám počet pokémonů v každé generaci a seřadí je podle indexu (generace). Dostanu tedy řadu s generací jako indexem a počtem pokémonů jako values. Teď už pouze stačí tyto proměnné dosadit do grafu. Abych mohla mít data na levé i pravé straně musím si vytvořit osu ax1, která mi bude sloužit pro „Součet všech statů“, tu pak vytvořím znovu na druhé straně pomocí ax1.twinx(), tím pádem generace na ose x zůstanou a přibudeme mi pouze „Počet pokémonů“ na ose y. Na konec zkombinuji obě legendy, aby graf byl přehledný.

```
generation_sum_stats = pokemon_dataset.groupby("Generation")["Total"].sum()

#Vypočítá celkový počet pokemonů v každé generaci
generation_pokemon_count = pokemon_dataset["Generation"].value_counts().sort_index()

#Vytvoří "pozadí" a osu
fig1, ax1 = plt.subplots()

ax1.stackplot(generation_sum_stats.index, generation_sum_stats.values, labels = ["Součet všech statů"])
ax1.set_xlabel("Generace")
ax1.set_ylabel("Součet všech statů")

#Vytvoří osu na druhé straně
ax2 = ax1.twinx()
ax2.plot(generation_pokemon_count.index, generation_pokemon_count.values, color = "red", marker = "o", label = "Počet Pokémonů")
ax2.set_ylabel("Počet Pokémonů")

#Zkombinuje legendu pro obě osy
handles1, labels1 = ax1.get_legend_handles_labels()
handles2, labels2 = ax2.get_legend_handles_labels()
ax1.legend(handles1 + handles2, labels1 + labels2, loc = "lower center", bbox_to_anchor = (0.49, -0.4))

plt.title("Součet všech statů a Počet Pokémonů v každé generaci")

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



Hned z prvního pohledu můžeme vidět, že součet všech statů(síly) je závislý na počtu pokémonů v každé generaci. První, třetí a pátá generace, které mají největší počet pokémonů, mají také největší sílu. Z těchto tří generací pouze u třetí vidíme, že i když má menší počet pokémonů než ostatní dvě, tak sílu má o trošku větší. Z toho můžeme odvodit, že v této generaci bylo např. více legendárních pokémonů, kteří bývají silnější. Ostatní tři generace mají mezi sebou už větší odchylky, co se týče počtu pokémonů a tím pádem i co se týče celkové síly. Jediným vybočujícím úkazem můžeme označit šestou generaci, kde je vysoký propad na obouch stranách osy y, také to je první generace, která nedosahuje ani ke stovce pokémonů. Dalším úkaz, kterého si můžeme všimnout, je klesající a poté rostoucí změna v počtu pokémonů s každou generací. Můžeme tedy předpokládat, že sedmá generace bude mít zase větší počet pokémonů, než měla generace šestá. Celkově to však vypadá, že celková síla odpovídá počtu pokémonů v generaci. Generace s největším počtem pokémonů má největší sílu, druhá generace s největším počtem pokémonů má druhá největší sílu atd. až k poslední.