Jakub Bryl, numer indeksu: 293085
Link do repozytorium: github.com/jbryl7/PokemonDreamTeam

ALHE - Dobierz je wszystkie

Treść zadania:

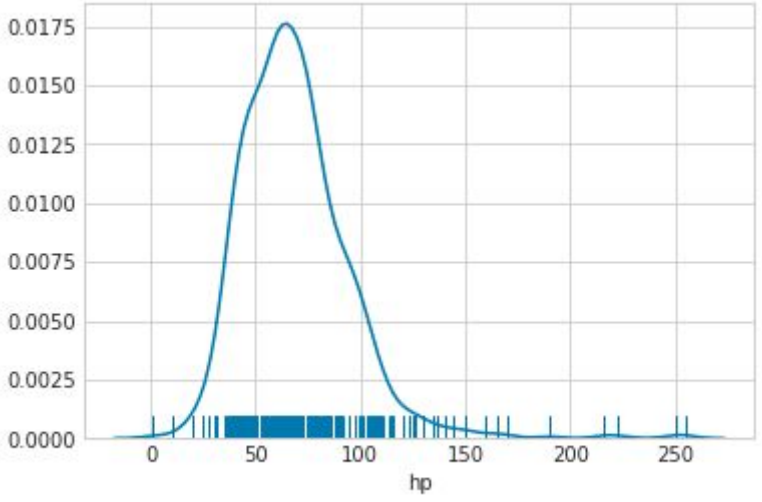
Znajdź idealny skład pokemonów które będą w stanie wygrać jak najwięcej pojedynków. Każdy pokemon charakteryzuje się innymi parametrami (atak, obrona) oraz ich skutecznością przeciwko konkretnym typom pokemonów. Z reguły np. wodne pokemony są skuteczne przeciwko ogniowym, natomiast mało skuteczne przeciwko trawiastym itp.. Bohater może posiadać jednocześnie maksymalnie 6 pokemonów, dlatego zadanie polegać będzie na znalezieniu takiej idealnej kombinacji, która będzie skuteczna przeciwko największej liczbie przeciwników, jednocześnie nie będąc zbyt wrażliwa na niektórych z nich. W dobieraniu kombinacji proszę wziąć pod uwagę częstotliwość występowania poszczególnych gatunków.
Dane:
https://www.kaggle.com/rounakbanik/pokemon

Kilka wniosków z analizy danych:

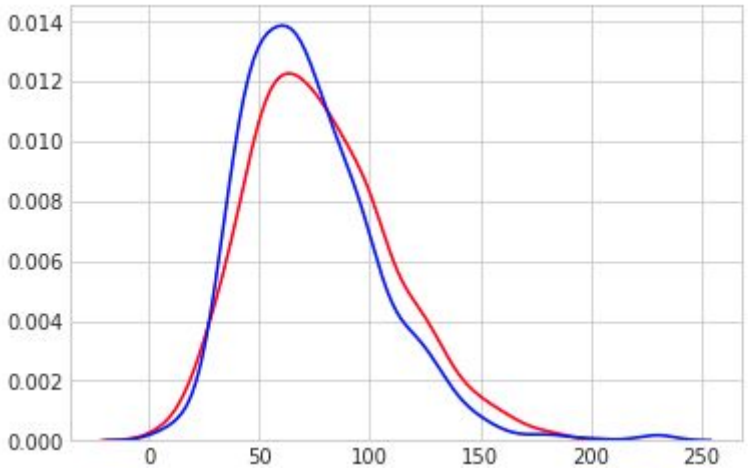
1. W zbiorze danych brakuje danych tylko w trzech nieistotnych dla naszego problemu kolumnach(zakładając że brak zdefiniowanej wartości w kolumnie type2 to zwyczajnie informacja że pokemon nie jest dwutypowy)
   <nazwa kolumny, liczba brakujących wartości>

```
height_m : 20
percentage_male : 98
type2 : 384
weight_kg : 20
```

2. Większość pokemonów ma zdrowie na poziomie od 25 do 125 punktów hp,



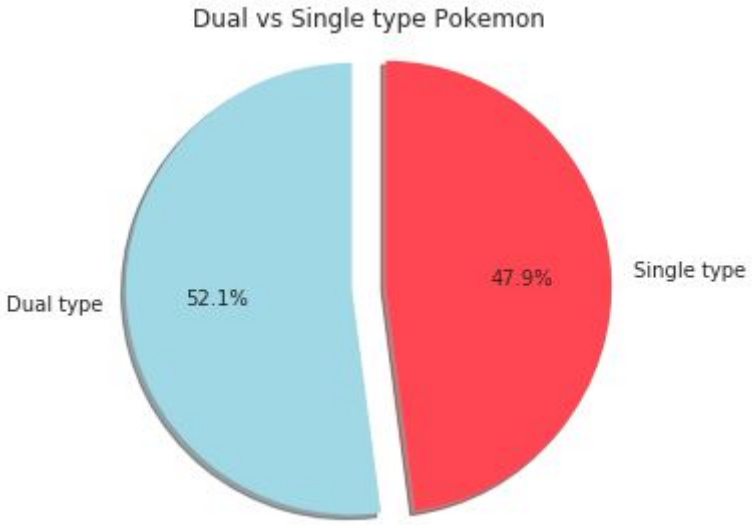
3. W podobnych granicach mieszczą się statystyki ataku(czerwony) i obrony(niebieski).



Sposób rozwiązania:

Przy wybieraniu pokemonów kluczowy jest sam sposób oceniania tego czy danego pokemona warto wybrać. Do tego celu wymagany jest sposób porównywania dwóch pokemonów ze sobą. Jest to realizowane poprzez symulację walki 1 vs 1.

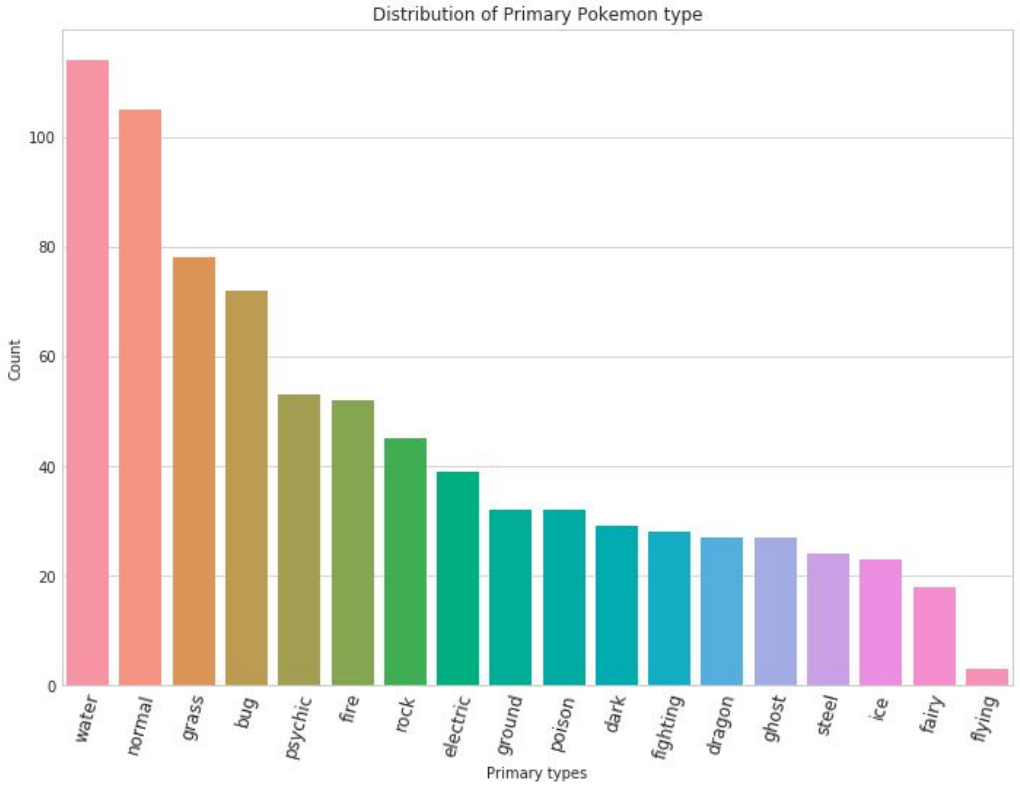
> **Wynik walki 1v1** jest obliczany na podstawie liczby ataków potrzebnych do pokonania jednego pokemona przez drugiego.
>
> To ile ataków należy wykonać aby pokonać danego przeciwnika obliczane jest ze wzoru:
> **moje_zdrowie / moc_uderzenia_od_przeciwnika - współczynnik_zadowolenia_przeciwnika * k_ataków - szybkość_przeciwnika * bonus_za_szybkość**
>
> **moc_uderzenia_od_przeciwnika = siła_ataku_przeciwnika * współczynnik_odporności_przeciwnika_na_mój_typ / moja_obrona**

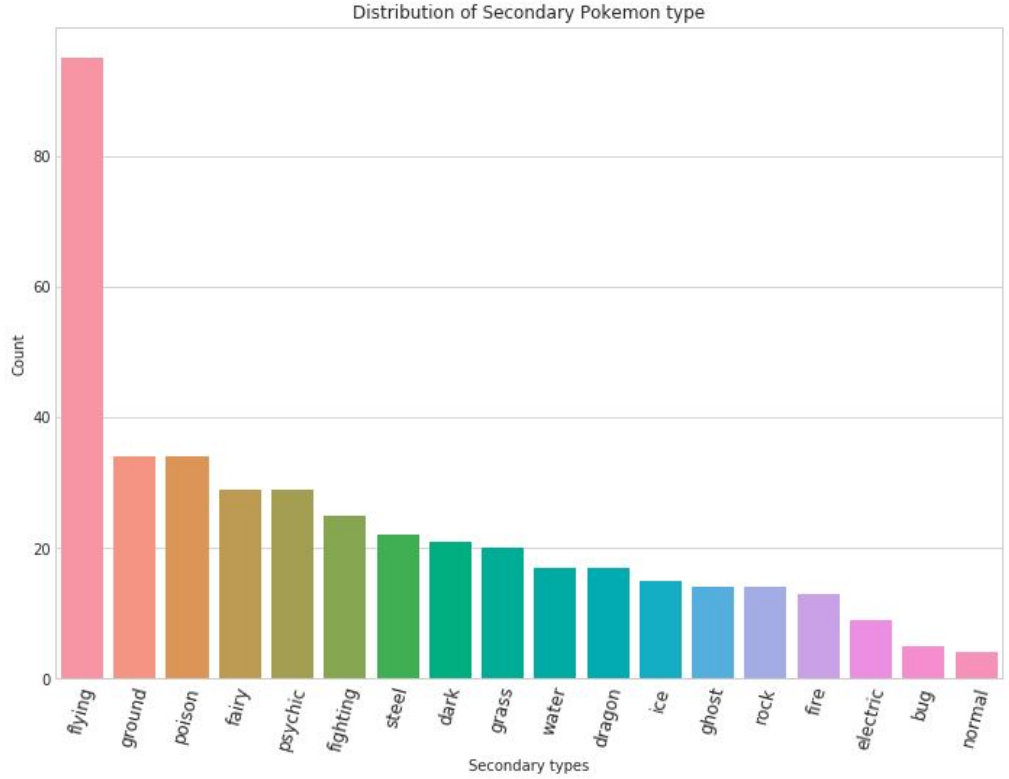


W związku z tym że pokemony mogą być dwutypowe oraz że ponad połowa pokemonów jest dwutypowa w rozstrzyganiu walki krytycznym czynnikiem będą typy walczących pokemonów. **Współczynnik_odporności_przeciwnika_na_mój_typ** jest wyliczany wykorzystując założenie że dwutypowy pokemon atakujący trochę częściej atakuje przy użyciu typu na który obrońca jest mniej odporny.
czyli dla przykładu:
Charizard będący typów ognistego i latającego przy atakowaniu Blastoise'a typu wodnego częściej będzie próbował używać ataków typu latającego ze względu na to że typ wodny z reguły ma dużą odporność na ogień.

współczynnik_odporności_przeciwnika_na_mój_typ =
p1 * odporność_przeciwnika_na_typ_dla_niego_groźniejszy + p2 * odporność_przeciwnika_na_drugi_typ

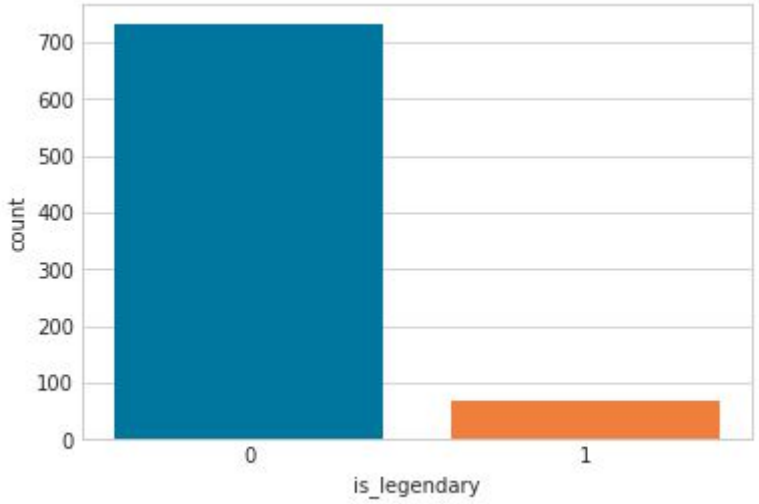
Zakładane wartości parametrów p1 i p2 to kolejno 3 i 2.
Początkowo wartości tych parametrów wynosiły 0.65 i 0.35, jednak okazały się one nieodpowiednie ponieważ drastycznie obniżały to jak bardzo typ pokemona jest ważny w rozstrzyganiu wyniku walki. Większość drużyn generowanych przez później opisane algorytmy składała się prawie zawsze tylko z pokemonów typu normalnego które często mają odrobinę wyższe statystyki podstawowe od pozostałych pokemonów.





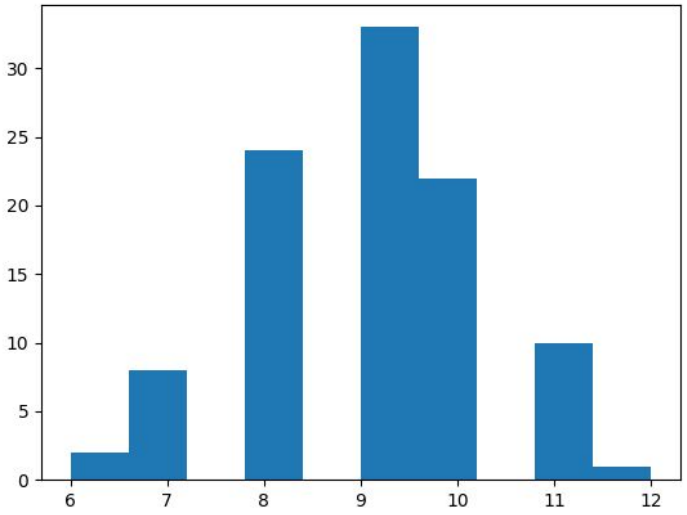
Wartym uwagi jest sama dystrybucja typów pokemonów.
Najwięcej jest pokemonów typu wodnego oraz normalnego(oczywiście, wiele z tych pokemonów ma też drugi typ). Na podstawie tych wykresów można wywnioskować, że silna drużyna powinna zawierać pokemona elektrycznego który jest silny przeciwko pokemonom

wodnym. Okazuje się jednak, pokemony elektryczne bardzo rzadko pojawiają się w drużynach bardzo dobrych. Zastępują je natomiast pokemony typu trawiastego lub smoczego które również są silne przeciwko typowi wodnemu.



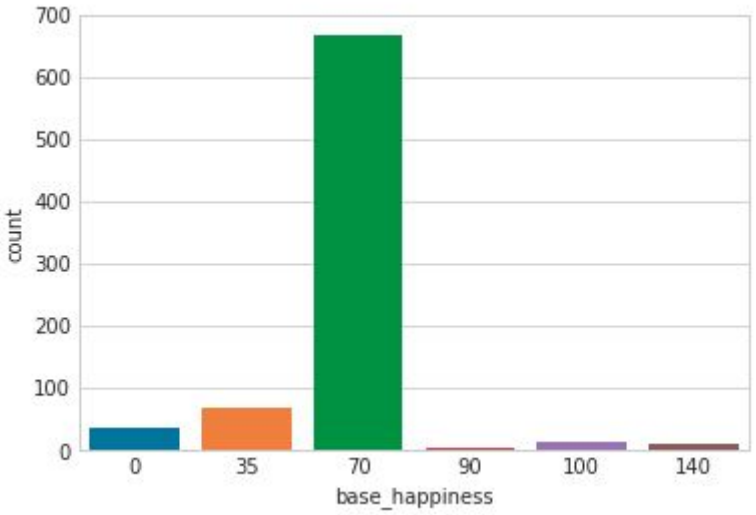
W badanym zbiorze danych około 10% pokemonów to pokemony legendarne. W związku z czym ważne jest to aby przy obliczaniu liczby ataków potrzebnych do pokonania danego pokemona uwzględniać to czy jest on legendarny. W przypadku takich pokemonów, początkowa liczba ataków potrzebnych do ich pokonania jest odpowiednio zmniejszana. Redukowane są też obrażenia zadawane przez pokemona legendarnego w przypadku kiedy to on jest “atakującym”.

Do wygenerowania poniższego wykresu wygenerowane zostało 100 losowych drużyn osiągających optimum. Można z niego wywnioskować, że warunkiem podstawowym tego, żeby drużyna osiągnęła optimum jest to żeby była ona różnorodna pod względem typów i że lista unikalnych typów występujących w drużynie była większa lub równa 6.



Oś y - liczba drużyn
Oś x - liczba różnych typów występujących w drużynie

Podczas walki uwzględniany jest również poziom zadowolenia pokemona gdyż ma on bezpośredni wpływ na to czy pokemonowi zależy na wygranej dla jego trenera. Pokemon szczęśliwy ma szansę pokonać przeciwnika wcześniej przy użyciu mniejszej liczby ataków. Pokemon niezadowolony częściej chybia atakami przez co zwiększa liczbę potrzebnych do pokonania przeciwnika ataków. U pokemona ze średnią wartością zadowolenia czynnik ten jest pomijany.



Warto jednak zauważyć, że zdecydowana większość pokemonów ma średni poziom zadowolenia co przekłada się na to, że jego wpływ można zauważyć w niewielu walkach.

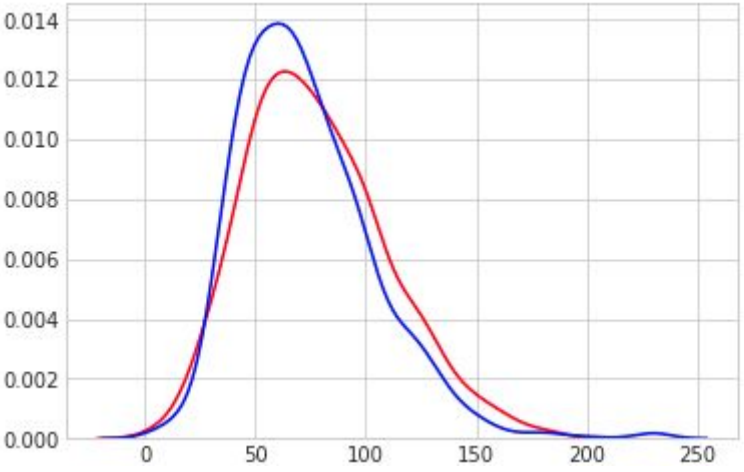
Wyniki walk między wszystkimi pokemonami są obliczane raz, na samym początku wykonywania programu i są oznaczane liczbami

Funkcja celu:

- **Liczba unikalnych pokemonów pokonanych przez drużynę** - Założyłem, że walka drużyny jest uznana za wygraną, gdy choć jeden z pokemonów z drużyny pokona atakowanego pokemona. Wydaje mi się, że głównie z tego powodu zazwyczaj bardzo szybko znajdowana jest bardzo dobra drużyna.Jest to spowodowane tym, że wystarczy znaleźć jednego silnego pokemona który pokrywa dużą część zbioru pokemonów do pokonania(np. Golisopod lub Metagross).
- **Wynik optymalny: 801**

Wykorzystane algorytmy poszukiwania:

Przy poszukiwaniu bardzo ważnym jest określenie co nazywamy sąsiadem. W tym przypadku sąsiadem nazywam drużynę która różni się od mojej drużyny jednym pokemonem.

- **Zachłanny** - W algorytmie tym wykorzystuję szukam samych najsilniejszych pokemonów. Kierowanie się tym kryterium przy szukaniu zespołu okazuje się bardzo łatwe do zrealizowania. Zrealizowałem to poprzez sumowanie wyników walk każdego z pokemonów i wybranie tych z największymi wynikami. Znaleziona drużyna (Golisopod, Salamence, Gyarados, Garchomp, Gyarados, Steelix) pokonuje 801 unikalnych pokemonów czyli jest w stanie wygrać z każdym pokemonem i jest to wynik optymalny. Steelix jest pokemonem który radzi sobie w walkach 1 na 1 najlepiej. Po przyjrzeniu się jego statystykom, można wywnioskować, że jest to spowodowane mocno odbiegającym od normy parametrem **obrona** który wynosi 200. Jest to wynik ponad 2 razy wyższy niż większość innych pokemonów.
  Mimo to, reszta wybranych pokemonów
  -
  

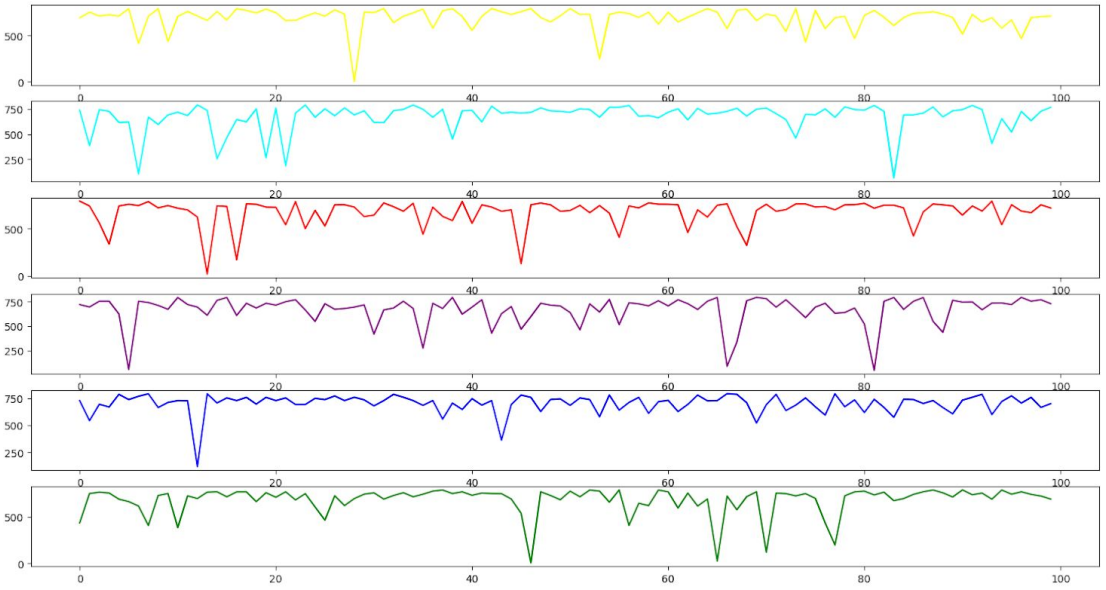
  Obrona na niebiesko
  oś x - wartość obrony/ataku pokemona

  Pozostałe pokemony wybrane do tej drużyny też mają ponadprzeciętne statystyki. Wartości obrony i ataku każdego z nich wynosi w prawie każdym przypadku około 125.

Typy pokemonów wybranych do drużyny:
steel, psychic, dragon, flying, bug, water, ground

- **Zachłanny2 -** W tym algorytmie wykorzystywana jest lista pokemonów “do pokonania”. Sześciokrotnie wybieramy pokemona który pokonuje najwięcej pokemonów z tej listy i usuwamy z tej listy tego pokemona oraz pokemony przez niego pokonane. Okazało się dla mnie zaskoczeniem, że wynik(**793**) tego algorytmu nie jest optymalny. Wygenerowana przez niego drużyna (Steelix, Jigglypuff, Pidgey, Fearow, Wartortle,Bulbasaur).

- **Algorytm wspinaczkowy -** w tym algorytmie losowo wybieram pokemony do drużyny, a następnie odwiedzam losowych sąsiadów i w razie gdy sąsiad osiąga większy wynik funkcji celu to zmieniam swoją drużynę na jego drużynę. Okazało się, że dobra drużyna jest często znajdowana bardzo szybko(często przy setnej iteracji osiągamy wynik ~795), a drużyna optymalna często w mniej niż 250 iteracjach. Okazało się też, że istnieje dużo drużyn optymalnych (np. Slowbro, Zweilous, Aggron, Rhyperior, Oricorio, Abomasnow. Czyli drużyna która nie zawiera żadnego z tych najsilniejszych pokemonów). Około 10% drużyn wybieranych przez ten algorytm zawiera w sobie pokemona legendarnego.



Oś Y - liczba pokonywanych przez danego pokemona pokemonów
Oś X - numer drużyny

Powyższy wykres przedstawia “moce” pokemonów ze stu drużyn wygenerowanych przez algorytm losowy których funkcja celu wynosi ponad 790. Można na nim zauważyć, że zazwyczaj prawie wszystkie pokemony dobrane do drużyny są bardzo silne. Zdarzają się jednak wyjątki w których występują pokemony słabe. Taki słaby pokemon jest do drużyny dobierany w momencie kiedy w drużynie występują dwa pokemony pokonujące prawie identyczny zbiór pokemonów, ale nie pokonujące tych kilku pokemonów które pokonuje ten słaby.

Przez to jak skuteczny jest najprostszy algorytm wspinaczkowy zacząłem podejrzewać błąd implementacyjny w metodzie rozstrzygania wyniku pojedynku między dwoma pokemonami. Nie udało mi się jednak wspomnianego błędu zlokalizować.

Implementacja

Do rozwiązania problemu zaimplementowałem 4 klasy:

- Pokemon - reprezentująca pokemona oraz definiująca sposób walki między pokemonami.
- PokemonList - Obiekt tej klasy tworzony jest na podstawie danych wejściowych. Zawiera listę wszystkich pokemonów oraz rezultaty przeprowadzonych między nimi walk.
- PokemonTeam - Przechowuje indeksy pokemonów które należą do drużyny.
- Solver - udostępnia metody pozwalające rozwiązać problem na różne sposoby oraz zawiera metody umożliwiające obliczenie funkcji celu danej drużyny.

Rozwiązanie zostało zaimplementowane w Pythonie przy wykorzystaniu bibliotek numpy oraz pandas.
Do repozytorium został załączony plik requirements.txt w którym znajdują się informacje o pakietach potrzebnych przy uruchamianiu środowiska pipenv.

Uruchamianie
> pipenv install -r requirements.txt
> pipenv shell
[pipenv] > python3 main.py [-g] [-g2] [-r -i <num_iterations>]
flagi:
g - greedy - uruchamia algorytm zachłanny
g2 - greedy2 - uruchamia algorytm zachłanny2
r - random - uruchamia algorytm wspinaczkowy. Może być uruchomiony z dodatkowym parametrem -i narzucającym liczbę iteracji jakie wykona algorytm(standardowo - 50).

Wniosek dodatkowy:

- Pokemony nie są bajką którą ogląda mi się tak dobrze jak w dzieciństwie. :(