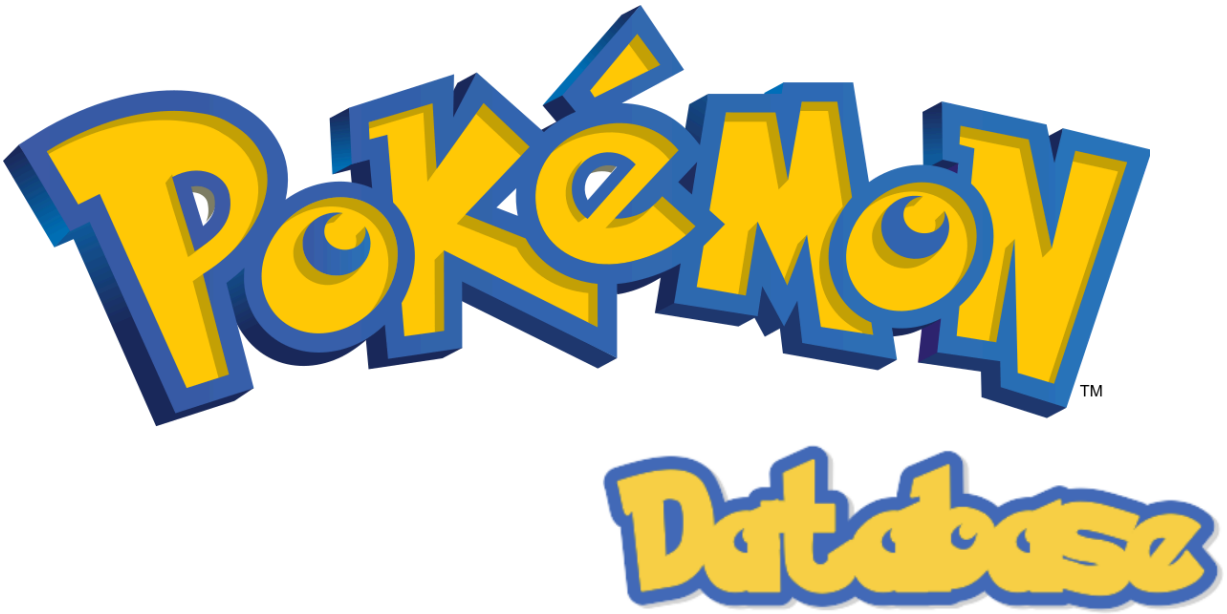

Aleix Rodriguez
Bader-Eddine Maoulay
ERP
Informe Del Projecte

Aleix Rodriguez, Bader Maoulay

INDEX

Aleix Rodriguez, Bader Maoulay

Informe del Projecte

Introducció

Hem escollit el tema de pokémon per el nostre projecte, en concret analitzar i comparar el nivell de poder de cada generació, comparan el increment o decrement de la mitja de estadístiques, aquestas son, punts de vida, atac físic, defensa fisica, atac especial, defensa física i velocitat, també el número/percentatge de cada tipus, amb aquestes dades si detectem alguna tendencia, podría, fins i tot predir com sera la seguent en quant a pokemons es refereix.

Datasets seleccionats

Utilitzarem aquests DataSets:
https://www.kaggle.com/datasets/mrdew25/pokemon-database
El primer DataSet es diu Pokemon DataBase i conté absoultament tota la informació de cada pokemon, habilitats, evolucions, formes alternes i el tipus dels pokemons (inclos si son dos tipus).
https://www.kaggle.com/datasets/hamzacyberpatcher/data-of-1010-pokemons
Aquest DataSet te la informació bàsica de la anterior(els tipus i les formes alternes a més dels noms) però també té inclós, la altura i el pes dels pokemons, a més del rank(si son legendaris o bebes mitics, etc.)
Pokemons (kaggle.com)
Aquesta ultim DataSet funciona com una pokedex nacional, el que vol dir que els pokemons van en orden de la regió i generació (primer va Kanto, després Jotto,etc.)

Modificacions als datasets

Com la nostra intenció en principi es comparar estadístiques base, els seus totals, i els la densitat dels tipus entre i de cada generació, tenim un parell de bases de dades que ens proporcionen totes aquestes dades amb cada generació corresponent si que una de elles dona extres com el pes, altura, habilitats etc. De moment no tenim intenció d'utilitzar cap d'aquestes així que no serán necesaries.

Aleix Rodriguez, Bader Maoulay

Problemes Trobats

Hem trobat que no son necesaries les columnes de pes, altura, i numero de la pokedex, de forma que les hem de eliminar, concretament eliminarem,el numero de la pokedex de i de regió de la base de dades (Pokemons (kaggle.com)), i la altura, pes i rank de la base de dades (https://www.kaggle.com/datasets/hamzacyberpatcher/data-of-1010-pokemons).

Sprints

Hem eliminat les columnes de alçada i pes al DataSet ja que no el necessitem:

Pokemon Name	Classification	Alternate Form Name	Original Pokemon ID	Legendary Type	Pokemon Height
"Bulbasaur"	"Seed Pokémon"	NULL	NULL	NULL	0.7
"Ivysaur"	"Seed Pokémon"	NULL	NULL	NULL	1.0
"Venusaur"	"Seed Pokémon"	NULL	NULL	NULL	2.0
"Venusaur"	"Seed Pokémon"	"Mega"	3	NULL	2.4
"Venusaur"	"Seed Pokémon"	"Gigantamax"	3	NULL	24.0
"Charmander"	"Lizard Pokémon"	NULL	NULL	NULL	0.6
"Charmeleon"	"Flame Pokémon"	NULL	NULL	NULL	1.1
"Charizard"	"Flame Pokémon"	NULL	NULL	NULL	1.7
"Charizard"	"Flame Pokémon"	"Mega X"	7	NULL	1.7
"Charizard"	"Flame Pokémon"	"Mega Y"	7	NULL	1.7
"Charizard"	"Flame Pokémon"	"Gigantamax"	7	NULL	28.0
"Squirtle"	"Tiny Turtle Pokémon"	NULL	NULL	NULL	0.5
"Wartortle"	"Turtle Pokémon"	NULL	NULL	NULL	1.0
"Blastoise"	"Shellfish Pokémon"	NULL	NULL	NULL	1.6
"Blastoise"	"Shellfish Pokémon"	"Mega"	12	NULL	1.6
"Blastoise"	"Shellfish Pokémon"	"Gigantamax"	12	NULL	25.0
"Caterpie"	"Worm Pokémon"	NULL	NULL	NULL	0.3
"Metapod"	"Cocoon Pokémon"	NULL	NULL	NULL	0.7
"Butterfree"	"Butterfly Pokémon"	NULL	NULL	NULL	1.1
"Butterfree"	"Butterfly Pokémon"	"Gigantamax"	24	NULL	17.0
"Weedle"	"Hairy Bug Pokémon"	NULL	NULL	NULL	0.3
"Kakuna"	"Cocoon Pokémon"	NULL	NULL	NULL	0.6
"Beedrill"	"Poison Bee Pokémon"	NULL	NULL	NULL	1.0
"Beedrill"	"Poison Bee Pokémon"	"Mega"	27	NULL	1.4
"Pidgey"	"Tiny Bird Pokémon"	NULL	NULL	NULL	0.3

Aleix Rodriguez, Bader Maoulay

	Pokemon Id	Pokedex Number	Pokemon Name	Classification	Alternate Form Name	Original Pokemon ID	Legendary Type	Primary Type	Secondary Type	Primary Ability	...	Speed EV	EV Yield Total
0	1	1	"Bulbasaur"	"Seed Pokémon"	NaN	NaN	NaN	"Grass"	"Poison"	"Overgrow"	...	0	1
1	2	2	"Ivysaur"	"Seed Pokémon"	NaN	NaN	NaN	"Grass"	"Poison"	"Overgrow"	...	0	2
2	3	3	"Venusaur"	"Seed Pokémon"	NaN	NaN	NaN	"Grass"	"Poison"	"Overgrow"	...	0	3
3	4	3	"Venusaur"	"Seed Pokémon"	"Mega"	3.0	NaN	"Grass"	"Poison"	"Thick Fat"	...	0	3
4	1526	3	"Venusaur"	"Seed Pokémon"	"Gigantamax"	3.0	NaN	"Grass"	"Poison"	"Overgrow"	...	0	3
5	5	4	"Charmander"	"Lizard Pokémon"	NaN	NaN	NaN	"Fire"	NaN	"Blaze"	...	1	1
6	6	5	"Charmeleon"	"Flame Pokémon"	NaN	NaN	NaN	"Fire"	NaN	"Blaze"	...	1	2
7	7	6	"Charizard"	"Flame Pokémon"	NaN	NaN	NaN	"Fire"	"Flying"	"Blaze"	...	0	3
8	8	6	"Charizard"	"Flame Pokémon"	"Mega X"	7.0	NaN	"Fire"	"Dragon"	"Tough Claws"	...	0	3
9	9	6	"Charizard"	"Flame Pokémon"	"Mega Y"	7.0	NaN	"Fire"	"Flying"	"Drought"	...	0	3

En les imatges es mostrar el canvi.
També eliminar l'id del pokemon i el numero de la pokedex:

	Pokemon Name	Classification	Alternate Form Name	Original Pokemon ID	Legendary Type	Primary Type	Secondary Type	Primary Ability	Primary Ability Description	Secondary Ability	...	Speed EV	I Yie Tot
0	"Bulbasaur"	"Seed Pokémon"	NaN	NaN	NaN	"Grass"	"Poison"	"Overgrow"	"Powers up Grass-type moves when the Pokémon's...	NaN	...	0	
1	"Ivysaur"	"Seed Pokémon"	NaN	NaN	NaN	"Grass"	"Poison"	"Overgrow"	"Powers up Grass-type moves when the Pokémon's...	NaN	...	0	
2	"Venusaur"	"Seed Pokémon"	NaN	NaN	NaN	"Grass"	"Poison"	"Overgrow"	"Powers up Grass-type moves when the Pokémon's...	NaN	...	0	
3	"Venusaur"	"Seed Pokémon"	"Mega"	3.0	NaN	"Grass"	"Poison"	"Thick Fat"	"The Pokémon is protected by a layer of thick ...	NaN	...	0	
		"Seed							"Powers up Grass-type				

El nom el deixem ja que el necessitem per saber quin pokemon es.

Aleix Rodriguez, Bader Maoulay

Les commandes quedarien així:

```
[2]: import pandas as pd

[13]: df = pd.read_csv(r'C:\Users\cacac\Downloads\Pokemon_Database.csv')

[14]: df = df.drop('Pokemon Height', axis=1)

[19]: df.head(10)

[19]:
```

Original ... EV

```
0]: df = df.drop('Pokemon Weight', axis=1)

1]: df.head(10)

22]: df = df.drop('Pokemon Id', axis=1)

23]: df = df.drop('Pokedex Number', axis=1)

24]: df.head(10)

24]:
```

Així quedaría una vegada eliminat tots els camps que no usarem, si necessitem un altre el afegim conforme el necessitem.

	Pokemon Name	Primary Type	Secondary Type	Game(s) of Origin	Health Stat	Attack Stat	Defense Stat	Special Attack Stat	Special Defense Stat	Speed Stat	Base Stat Total
0	"Bulbasaur"	"Grass"	"Poison"	"Red"	45	49	49	65	65	45	318
1	"Ivysaur"	"Grass"	"Poison"	"Red"	60	62	63	80	80	60	405
2	"Venusaur"	"Grass"	"Poison"	"Red"	80	82	83	100	100	80	525
3	"Venusaur"	"Grass"	"Poison"	"X"	80	100	123	122	120	80	625
4	"Venusaur"	"Grass"	"Poison"	"Sword"	80	82	83	100	100	80	525
...	...	...	...	...	...	...	...	...	...	...	...
1377	"Iron Crown"	"Steel"	"Psychic"	"Scarlet"	90	72	100	122	108	98	590
1378	"Terapagos"	"Normal"	NaN	"Violet"	160	105	110	130	110	85	700
1379	"Terapagos"	"Normal"	NaN	"Violet"	90	65	85	65	85	60	450
1380	"Terapagos"	"Normal"	NaN	"Scarlet"	95	95	110	105	110	85	600
1381	"Pecharunt"	"Poison"	"Ghost"	"Scarlet"	88	88	160	88	88	88	600

Aleix Rodriguez, Bader Maoulay

Optimització

Ja que el nostre objectiu és que la database es pugui utilitzar per comparar pokemon entre generacions, haurém d'afegir aquestes

Primer carraguem la base de dades.

```
import pandas as pd
```

```
df = pd.read_csv('modified_pokemon.csv')
```

```
df
```

	Pokemon Name	Primary Type	Secondary Type	Secondary Ability	Base Happiness	Game(s) of Origin	Health Stat	Attack Stat	Defense Stat	Special Attack Stat	Special Defense Stat	Speed Stat	Base Stat Total
0	"Bulbasaur"	"Grass"	"Poison"	NaN	50	"Red"	45	49	49	65	65	45	318
1	"Ivysaur"	"Grass"	"Poison"	NaN	50	"Red"	60	62	63	80	80	60	405
2	"Venusaur"	"Grass"	"Poison"	NaN	50	"Red"	80	82	83	100	100	80	525
3	"Venusaur"	"Grass"	"Poison"	NaN	70	"X"	80	100	123	122	120	80	625
4	"Venusaur"	"Grass"	"Poison"	NaN	50	"Sword"	80	82	83	100	100	80	525
...	...	...	...	...	...	...	...	...	...	...	...	...	...
1377	"Iron Crown"	"Steel"	"Psychic"	NaN	0	"Scarlet"	90	72	100	122	108	98	590
1378	"Terapagos"	"Normal"	NaN	NaN	0	"Violet"	160	105	110	130	110	85	700
1379	"Terapagos"	"Normal"	NaN	NaN	0	"Violet"	90	65	85	65	85	60	450
1380	"Terapagos"	"Normal"	NaN	NaN	0	"Scarlet"	95	95	110	105	110	85	600
1381	"Pecharunt"	"Poison"	"Ghost"	NaN	0	"Scarlet"	88	88	160	88	88	88	600

1382 rows × 13 columns

Eliminem les columnes que sobren, com Secondary Ability i Base happiness.

```
[25]: df=df.drop('Base Happiness', axis=1)
```

```
[26]: df
```

[26]:

	Pokemon Name	Primary Type	Secondary Type	Gam
0	"Bulbasaur"	"Grass"	"Poison"	
1	"Ivysaur"	"Grass"	"Poison"	
2	"Venusaur"	"Grass"	"Poison"	
3	"Venusaur"	"Grass"	"Poison"	
4	"Venusaur"	"Grass"	"Poison"	
...	...	...	...	
1377	"Iron Crown"	"Steel"	"Psychic"	
1378	"Terapagos"	"Normal"	NaN	
1379	"Terapagos"	"Normal"	NaN	
1380	"Terapagos"	"Normal"	NaN	
1381	"Pecharunt"	"Poison"	"Ghost"	

1382 rows × 11 columns

```
[23]: df=df.drop('Secondary Ability', axis=1)
```

```
[24]: df
```

[24]:

	Pokemon Name	Primary Type	Secondary Type	Base Happiness	Game(s) of Origin	Health Stat	Attack Stat	Defense Stat	At
0	"Bulbasaur"	"Grass"	"Poison"	50	"Red"	45	49	49	
1	"Ivysaur"	"Grass"	"Poison"	50	"Red"	60	62	63	
2	"Venusaur"	"Grass"	"Poison"	50	"Red"	80	82	83	
3	"Venusaur"	"Grass"	"Poison"	70	"X"	80	100	123	
4	"Venusaur"	"Grass"	"Poison"	50	"Sword"	80	82	83	
...	...	...	...	...	...	...	...	...	
1377	"Iron Crown"	"Steel"	"Psychic"	0	"Scarlet"	90	72	100	
1378	"Terapagos"	"Normal"	NaN	0	"Violet"	160	105	110	
1379	"Terapagos"	"Normal"	NaN	0	"Violet"	90	65	85	
1380	"Terapagos"	"Normal"	NaN	0	"Scarlet"	95	95	110	
1381	"Pecharunt"	"Poison"	"Ghost"	0	"Scarlet"	88	88	160	

1382 rows × 12 columns

Aleix Rodriguez, Bader Maoulay

Guardem aquesta versió de la base de dades abans de fer cap insert:

```
[28]: df.to_csv('modified_pokemon_noGens.csv', index=False)

[29]: pokedb = pd.read_csv('modified_pokemon_noGens.csv')
```

Mostrem quants jocs per separat hi han:

```
[84]: games = df['Game(s) of Origin'].drop_duplicates()
      print(games)

      0                  "Red"
      3                    "X"
      4                "Sword"
      23           "Omega Ruby"
      29                "Sun"
      38      "Let's Go Pikachu"
      86       "Legends Arceus"
      192             "Scarlet"
      229               "Gold"
      309               "Ruby"
      532           "Fire Red"
      533          "Leaf Green"
      534             "Emerald"
      535             "Diamond"
      645            "Platinum"
      668               "Black"
      840             "Black 2"
      1018          "Ultra Sun"
      1097         "Pokémon GO"
      1374             "Violet"
      Name: Game(s) of Origin, dtype: object
```

Definim una funció que afegeix una columna “Gen” i a partir del joc del pokemon d’origen del pokémon asigni a aquella fila una generació:

```
def asignar_generacion(df):
    Gen = {
        'Red' : 'Gen1',
        'X' : 'Gen6',
        'Sword' : 'Gen8',
        'Omega Ruby' : 'Gen6',
        'Sun' : 'Gen7',
        'Let\'s Go Pikachu' : 'Gen7',
        'Legends Arceus' : 'Gen8',
        'Scarlet' : 'Gen9',
        'Gold' : 'Gen2',
        'Ruby' : 'Gen3',
        'Fire Red' : 'Gen3',
        'Leaf Green' : 'Gen3',
        'Emerald' : 'Gen3',
        'Diamond' : 'Gen4',
        'Platinum' : 'Gen4',
        'Black' :  'Gen5',
        'Black 2' : 'Gen5',
        'Ultra Sun' : 'Gen7',
        'Pokémon GO' : 'Gen7',
        'Violet' : 'Gen9',
    }
    df['Gen'] = df['Game(s) of Origin'].apply(lambda x: Gen.get(x.strip('"').strip(), 'Generación desconocida'))
    return df

pokedb = asignar_generacion(pokedb)
```

Mostrem la taula per comprobar si ha funcionat, ha inserit les generacions correctament:

pokedb

	Pokemon Name	Primary Type	Secondary Type	Game(s) of Origin	Health Stat	Attack Stat	Defense Stat	Special Attack Stat	Special Defense Stat	Speed Stat	Base Stat Total	Gen
0	"Bulbasaur"	"Grass"	"Poison"	"Red"	45	49	49	65	65	45	318	Gen1
1	"Ivysaur"	"Grass"	"Poison"	"Red"	60	62	63	80	80	60	405	Gen1
2	"Venusaur"	"Grass"	"Poison"	"Red"	80	82	83	100	100	80	525	Gen1
3	"Venusaur"	"Grass"	"Poison"	"X"	80	100	123	122	120	80	625	Gen6
4	"Venusaur"	"Grass"	"Poison"	"Sword"	80	82	83	100	100	80	525	Gen8

I finalment guardem la base de dades:

```
[8]: pokedb.to_csv('modified_pokemon_withGens.csv', index=False)
```

D'aquesta manera la base de dades està més optimitzada ja que normalment de manera competitiva a pokemon es parla per generacions.

Comprovació de la nostra idea:

Primer hem de definir la funció que hem estat pensat.

```
# Definir la funció per agafar els 6 pokemons mes poderosos per generacio
def millor_pokemon_per_generacio(df, gen):

    generacio = df[df['Gen'] == gen]
    totalEstadistiques = generacio.sort_values(by='Base Stat Total', ascending=False)
    top6 = totalEstadistiques.head(6)
    print(top6)

    #return top6
```

Una vegada la tenim tenim definida, escollint per les estadístiques podem comprovar-ho amb diferent generacions. Com veiem, Podem veure el nom del pokemon, els seus tipus i les seves estadístiques.

```
#Comprovacio amb la gen1
millor_pokemon = millor_pokemon_per_generacio(pokedb, 'Gen1')
```

```
    Pokemon Name Primary Type Secondary Type Game(s) of Origin  Health Stat  \
225     "Mewtwo"    "Psychic"            NaN             "Red"          106
228        "Mew"    "Psychic"            NaN             "Red"          100
224  "Dragonite"     "Dragon"       "Flying"             "Red"           91
220    "Moltres"       "Fire"       "Flying"             "Red"           90
216   "Articuno"        "Ice"       "Flying"             "Red"           90
218     "Zapdos"   "Electric"       "Flying"             "Red"           90

     Attack Stat  Defense Stat  Special Attack Stat  Special Defense Stat  \
225          110            90                  154                    90
228          100           100                  100                   100
224          134            95                  100                   100
220          100            90                  125                    85
216           85           100                   95                   125
218           90            85                  125                    90

     Speed Stat  Base Stat Total   Gen
225         130              680  Gen1
228         100              600  Gen1
224          80              600  Gen1
220          90              580  Gen1
216          85              580  Gen1
218         100              580  Gen1
```

Aquí tenim la funció però amb la generació 9, per veure els millors pokemons d'aquesta generació.

```
#Comprovacio amb la gen 9
millor_pokemon = millor_pokemon_per_generacio(pokedb, 'Gen9')
```

	Pokemon Name	Primary Type	Secondary Type	Game(s) of Origin	Health Stat	\
1378	"Terapagos"	"Normal"	NaN	"Violet"	160	
1356	"Koraidon"	"Fighting"	"Dragon"	"Scarlet"	100	
1357	"Miraidon"	"Electric"	"Dragon"	"Scarlet"	100	
1308	"Palafin"	"Water"	NaN	"Scarlet"	100	
1381	"Pecharunt"	"Poison"	"Ghost"	"Scarlet"	88	
1372	"Archaludon"	"Steel"	"Dragon"	"Scarlet"	90	

	Attack Stat	Defense Stat	Special Attack Stat	Special Defense Stat	\
1378	105	110	130	110	
1356	135	115	85	100	
1357	85	100	135	115	
1308	160	97	106	87	
1381	88	160	88	88	
1372	105	130	125	65	

	Speed Stat	Base Stat Total	Gen
1378	85	700	Gen9
1356	135	670	Gen9
1357	135	670	Gen9
1308	100	650	Gen9
1381	88	600	Gen9
1372	85	600	Gen9

Repositori

https://github.com/BadarNH/Projecte-ERP

Aleix Rodriguez, Bader Maoulay

Bibliografía

La pagina utilitzada es de la que hem extret els datasets:
https://www.kaggle.com