# Rahman_Final

May 7, 2019

Aysha Rahman
PHY 231

## 1 Final Project

### 1.1 Introduction

As a person who grew up playing Pokemon video games, the Pokemon franchise is near and dear to my heart. There are 7 generations of Pokemon, which each have a set of video games they were released for. I wanted to look at the Complete Pokemon Dataset (https://www.kaggle.com/rounakbanik/pokemon/version/1) from Kaggle, which contains all current seven generations of Pokemon with their types, stats, and other information, all scraped from the site serebii.net. There are a total of 801 Pokemon, with 41 columns of information in the dataset.

I want to explore the data and look at correlations of different statistics, as well as create different teams of six Pokemon each for various purposes, and ultimately create an ideal Pokemon team to battle a particular character from one of the games.

In [82]:
```
#importing whatever I can think of, just in case I need it
import pandas as pd
import ast
import numpy as np
from sklearn.linear_model import LinearRegression, LogisticRegression
from sklearn.metrics import mean_squared_error
import matplotlib
import matplotlib.pyplot as plt
import seaborn as sbn
from altair import Chart, X, Y, Color, Scale
import altair as alt
from vega_datasets import data #error
import requests
from bs4 import BeautifulSoup
matplotlib.style.use('ggplot')
from prettytable import PrettyTable as pt
from tabulate import tabulate as tb
```

In [83]:
```
#reading the csv file
poke = pd.read_csv("pokemon.csv")
poke.head()
```

Out[83]:

| | abilities | against_bug | against_dark | against_dragon | \ |
|---|---|---|---|---|---|
| 0 | ['Overgrow', 'Chlorophyll'] | 1.0 | 1.0 | 1.0 | |
| 1 | ['Overgrow', 'Chlorophyll'] | 1.0 | 1.0 | 1.0 | |
| 2 | ['Overgrow', 'Chlorophyll'] | 1.0 | 1.0 | 1.0 | |
| 3 | ['Blaze', 'Solar Power'] | 0.5 | 1.0 | 1.0 | |
| 4 | ['Blaze', 'Solar Power'] | 0.5 | 1.0 | 1.0 | |

| | against_electric | against_fairy | against_fight | against_fire | \ |
|---|---|---|---|---|---|
| 0 | 0.5 | 0.5 | 0.5 | 2.0 | |
| 1 | 0.5 | 0.5 | 0.5 | 2.0 | |
| 2 | 0.5 | 0.5 | 0.5 | 2.0 | |
| 3 | 1.0 | 0.5 | 1.0 | 0.5 | |
| 4 | 1.0 | 0.5 | 1.0 | 0.5 | |

| | against_flying | against_ghost | ... | percentage_male | \ |
|---|---|---|---|---|---|
| 0 | 2.0 | 1.0 | ... | 88.1 | |
| 1 | 2.0 | 1.0 | ... | 88.1 | |
| 2 | 2.0 | 1.0 | ... | 88.1 | |
| 3 | 1.0 | 1.0 | ... | 88.1 | |
| 4 | 1.0 | 1.0 | ... | 88.1 | |

| | pokedex_number | sp_attack | sp_defense | speed | type1 | type2 | weight_kg | \ |
|---|---|---|---|---|---|---|---|---|
| 0 | 1 | 65 | 65 | 45 | grass | poison | 6.9 | |
| 1 | 2 | 80 | 80 | 60 | grass | poison | 13.0 | |
| 2 | 3 | 122 | 120 | 80 | grass | poison | 100.0 | |
| 3 | 4 | 60 | 50 | 65 | fire | NaN | 8.5 | |
| 4 | 5 | 80 | 65 | 80 | fire | NaN | 19.0 | |

| | generation | is_legendary |
|---|---|---|
| 0 | 1 | 0 |
| 1 | 1 | 0 |
| 2 | 1 | 0 |
| 3 | 1 | 0 |
| 4 | 1 | 0 |

[5 rows x 41 columns]

### 1.2 Pokemon Stats

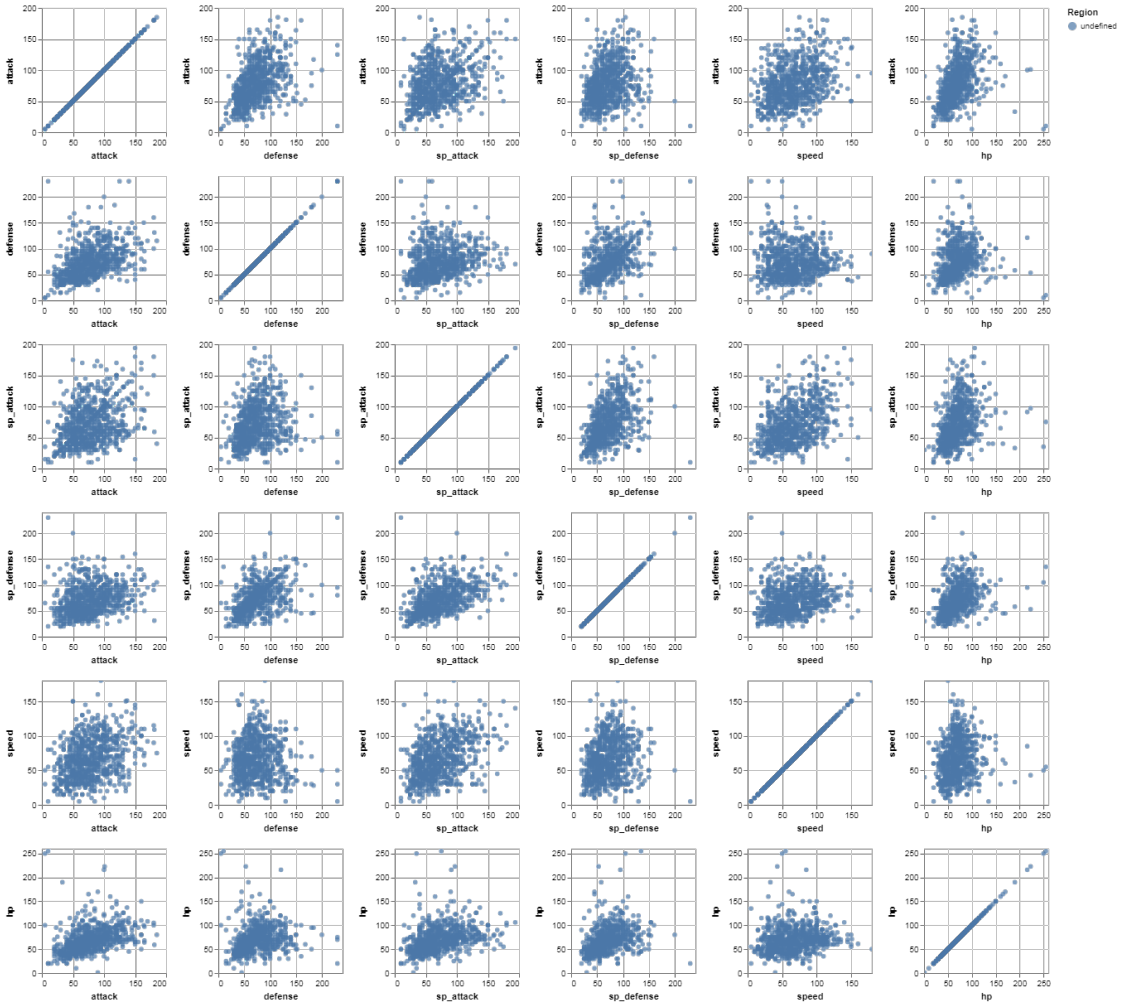
I want to look at how each Pokemon statistic (attack, defense, special attack, special defense, speed, hit points, and total stats) compares to the others. Is there any correlation, and could we predict one stat based off of others? Let's look at it graphically.

In [84]:
```
#Graphs comparing stats

alt.Chart(poke).mark_circle().encode(
    alt.X(alt.repeat("column"), type='quantitative'),
    alt.Y(alt.repeat("row"), type='quantitative'),
```

```
        color='Region:N'
    ).properties(
        width=150,
        height=150
    ).repeat(
        row=['attack', 'defense', 'sp_attack', 'sp_defense', 'speed', 'hp'],
        column=['attack', 'defense', 'sp_attack', 'sp_defense', 'speed', 'hp']
    ).interactive()
```

Out[84]:




Now let us look at the actual correlation between each stat.

```
In [85]: #Looking at the correlation between each stat
         stat='attack', 'defense', 'sp_attack', 'sp_defense', 'speed', 'hp'#, 'base_total'
         for x in stat:
             a = poke[x]
```

```
            for y in stat:
                b=a.corr(poke[y])
                print(x,"vs", y, b)
```

```
attack vs attack 1.0
attack vs defense 0.4689149139189302
attack vs sp_attack 0.3681539995495995
attack vs sp_defense 0.26583622612926466
attack vs speed 0.35270264003657553
attack vs hp 0.41061578930650766
defense vs attack 0.4689149139189302
defense vs defense 0.9999999999999999
defense vs sp_attack 0.24188203283262197
defense vs sp_defense 0.5263482757215796
defense vs speed 0.007934069032646924
defense vs hp 0.24237815560221193
sp_attack vs attack 0.36815399954959954
sp_attack vs defense 0.24188203283262197
sp_attack vs sp_attack 0.9999999999999998
sp_attack vs sp_defense 0.5114955077107626
sp_attack vs speed 0.43898126771331236
sp_attack vs hp 0.36597264465583984
sp_defense vs attack 0.26583622612926466
sp_defense vs defense 0.5263482757215796
sp_defense vs sp_attack 0.5114955077107625
sp_defense vs sp_defense 1.0
sp_defense vs speed 0.22597698037922195
sp_defense vs hp 0.3669707444115091
speed vs attack 0.3527026400365756
speed vs defense 0.007934069032646924
speed vs sp_attack 0.43898126771331236
speed vs sp_defense 0.22597698037922195
speed vs speed 1.0
speed vs hp 0.16075981807940234
hp vs attack 0.4106157893065077
hp vs defense 0.24237815560221193
hp vs sp_attack 0.3659726446558399
hp vs sp_defense 0.36697074441150906
hp vs speed 0.16075981807940237
hp vs hp 1.0
```

There doesn't seem to be any correlation. Still, I want to see how well we can predict one Pokemon statistic using others. The two most related stats seem to be defense and special defense, and then special defense and special attack. So, let's pick special attack, special defense, and defense as our starting example to see if one can be predicted using the others.

```
In [86]: model = LinearRegression()
         model.fit(poke[['sp_attack','sp_defense']], poke.defense)
```

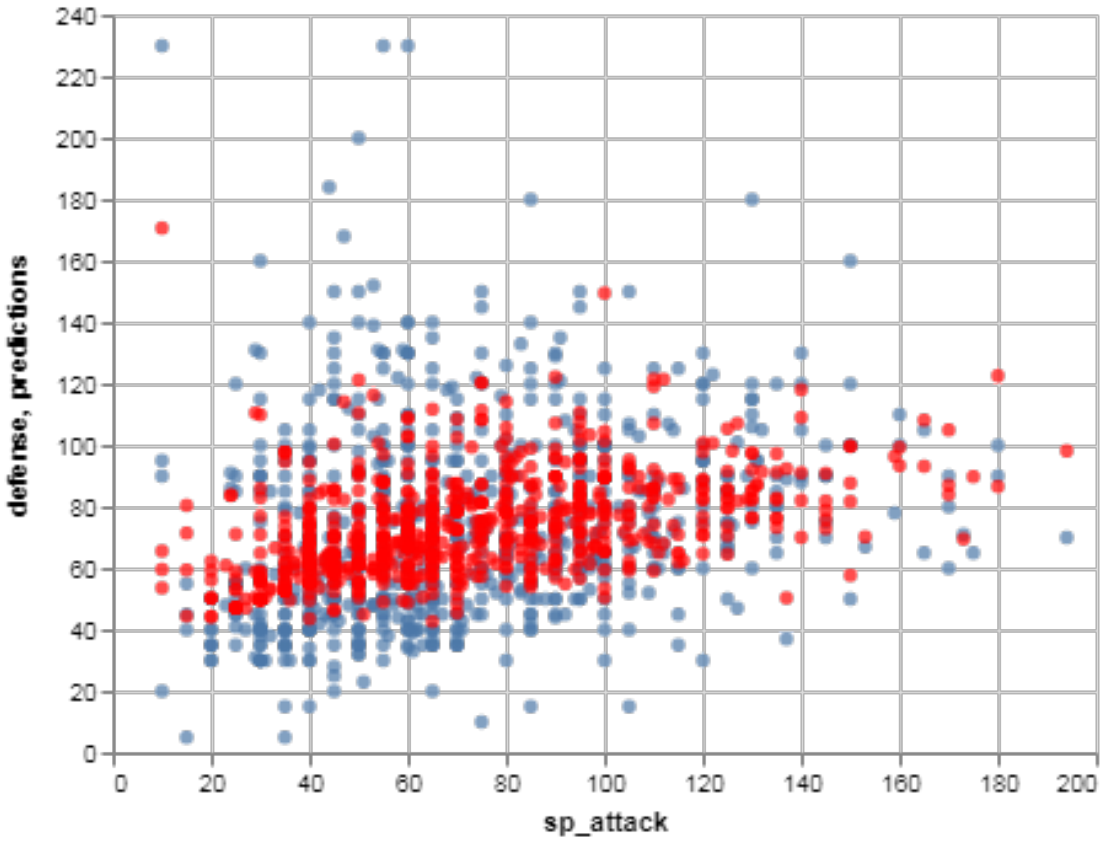
```
Out[86]: LinearRegression(copy_X=True, fit_intercept=True, n_jobs=None,
                          normalize=False)

In [87]: mean_squared_error(poke.defense, model.predict(poke[['sp_attack','sp_defense']]))

Out[87]: 682.6417169748061

In [88]: poke['predictions'] = model.predict(poke[['sp_attack','sp_defense']])
         Chart(poke).mark_circle().encode(x='sp_attack', y='defense') + \
         Chart(poke).mark_circle(color='red').encode(x='sp_attack', y='predictions')

Out[88]:
```

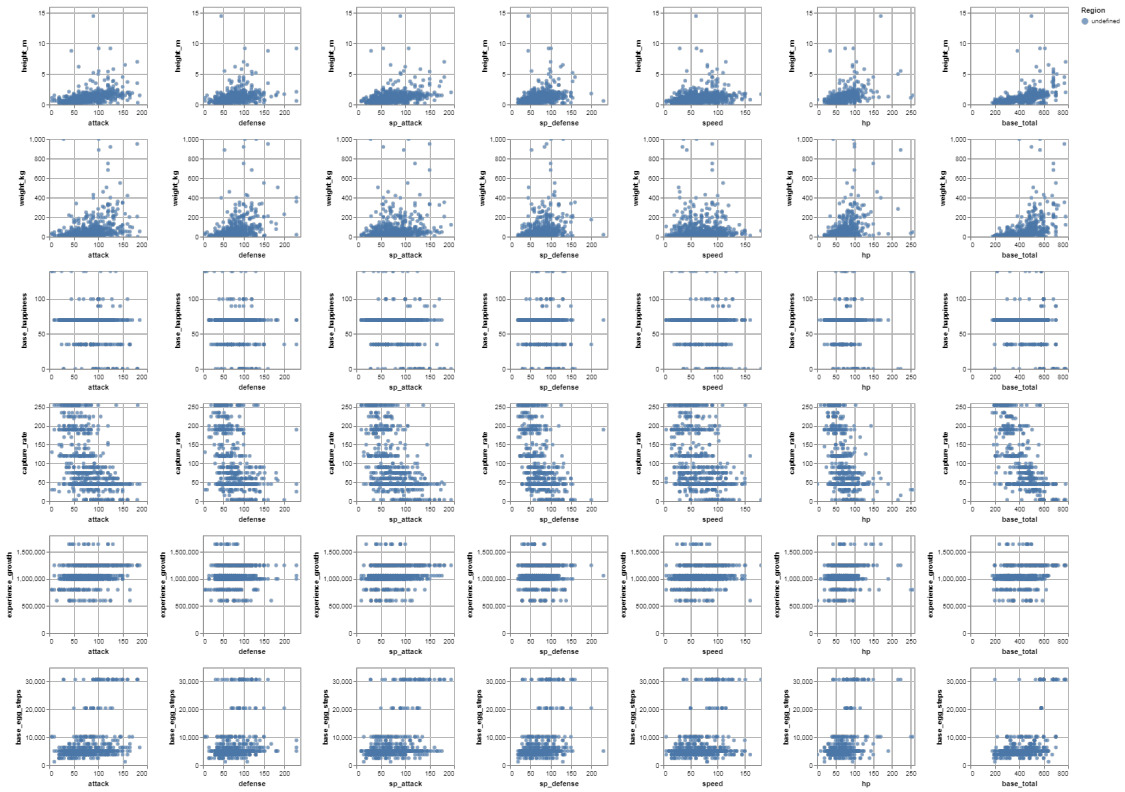



There doesn't seem to be much of a correlation between any of the stats, so it makes sense that the predictions are not super close to our actual data. They are visually a lot closer than I would expect, given the huge mean squared error. That's an interesting observation, but still far off. And since these are the three most correlated stats, trying to make other predictions using battle stats will not be a fruitful endeavor.

However, there are other items we could look at besides battle stats. Could properties such as height, weight, base happiness, capture rate, experience growth, and base egg steps correlate with any of the battle stats? Let's look at the same thing we did with the battle stats, but this time look at battle stats vs other properties of each species.

```
In [89]: alt.Chart(poke).mark_circle().encode(
             alt.X(alt.repeat("column"), type='quantitative'),
             alt.Y(alt.repeat("row"), type='quantitative'),
             color='Region:N'
         ).properties(
             width=150,
             height=150
         ).repeat(
             row=['height_m', 'weight_kg', 'base_happiness', 'capture_rate', 'experience_growth
             column=['attack', 'defense', 'sp_attack', 'sp_defense', 'speed', 'hp','base_total
         ).interactive()
```

Out[89]:




Even moreso than the battle stats against each other, it's pretty clear that these other properties of Pokemon species have no relationship with battle stats. This doesn't tell us anything new about what makes a Pokemon strong, but it has satisfied my curiosity, and it reinforces what we found previously about Pokemon characteristics being pretty arbitrary. It makes sense that these properties would be unpredictable such as to increase diversity amongst Pokemon species.

Before we get on to creating teams, let's look at different Pokemon types. First, we'll see what all the different types are, and then we can create dataframes for each type that we can use later. In creating the dataframes, we will consider the fact that some Pokemon have two types, so we want those Pokemon to show up in the dataframes for both of their types.

In [90]:
```
#To make dataframes for each type, I'm starting by creating a series that includes ea

types = pd.Series(poke.type1.unique())
#types = poke.type1.unique()
types
```

Out[90]:
```
0        grass
1         fire
2        water
3          bug
4       normal
5       poison
6     electric
7       ground
8        fairy
9     fighting
10     psychic
11        rock
12       ghost
13         ice
14      dragon
15        dark
16       steel
17      flying
dtype: object
```

In [92]:
```
#Using a for loop to create a dataframe for each type using both primary 'type1' and
for i in types:
    print (i)
    vars()[i] = poke[(poke.type1==i) | (poke.type2==i)]
    print("Dataframe has been created.")
```

```
grass
Dataframe has been created.
fire
Dataframe has been created.
water
Dataframe has been created.
bug
Dataframe has been created.
normal
Dataframe has been created.
poison
Dataframe has been created.
electric
Dataframe has been created.
ground
Dataframe has been created.
```

```
fairy
Dataframe has been created.
fighting
Dataframe has been created.
psychic
Dataframe has been created.
rock
Dataframe has been created.
ghost
Dataframe has been created.
ice
Dataframe has been created.
dragon
Dataframe has been created.
dark
Dataframe has been created.
steel
Dataframe has been created.
flying
Dataframe has been created.
```

We've created a dataframe for each type using a for loop. A Pokemon can have two types and we stated that we wanted the same Pokemon to show up in the dataframes for both of its types, so our loop looks at both its primary and secondary type to see if it belongs in each dataframe. Now, let us use these dataframes to create yet another dataframe, this time one that will tell us the strongest Pokemon of each type based off of the sum of their total stats, which in the dataset is listed under the column "base_total".

```
In [93]: #A dataframe with the strongest Pokemon of each type, based off of the sum of their s
         bytype = pd.DataFrame()
         for i in types:
             i = poke[(poke.type1==i) | (poke.type2==i)]
             bytype = bytype.append([i[i.is_legendary==False].sort_values(['base_total'],ascen
         bytype[['name','type1','type2']]
```

Out[93]:

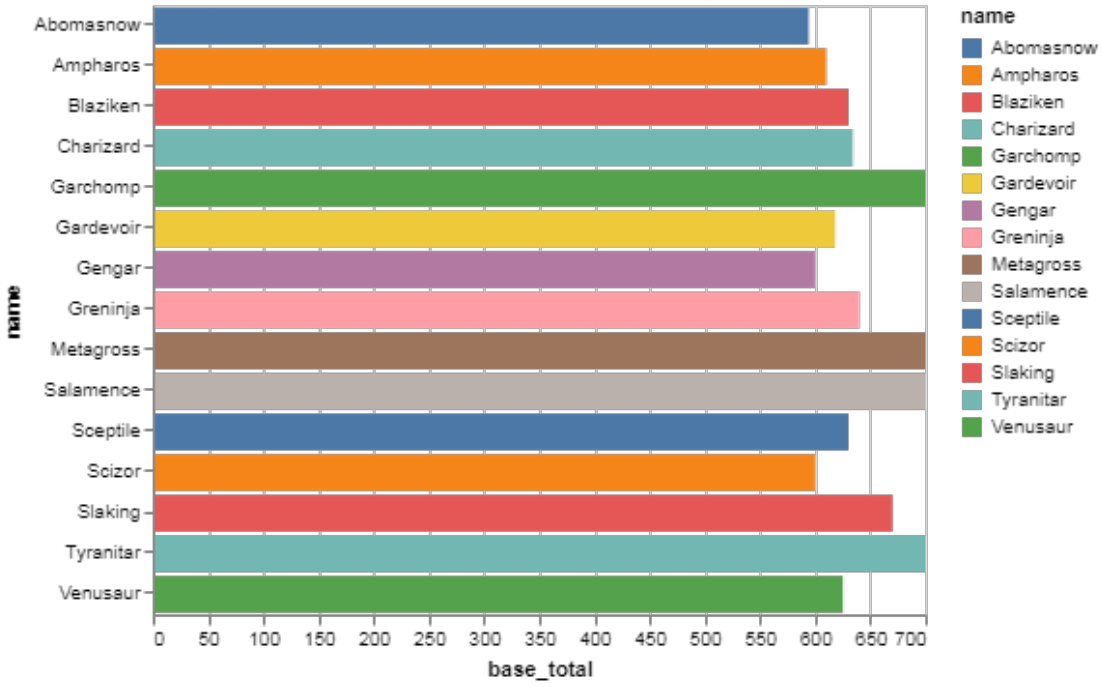
| | name | type1 | type2 |
|---|---|---|---|
| 253 | Sceptile | grass | NaN |
| 5 | Charizard | fire | flying |
| 657 | Greninja | water | dark |
| 211 | Scizor | bug | steel |
| 288 | Slaking | normal | NaN |
| 2 | Venusaur | grass | poison |
| 180 | Ampharos | electric | NaN |
| 444 | Garchomp | dragon | ground |
| 281 | Gardevoir | psychic | fairy |
| 256 | Blaziken | fire | fighting |
| 375 | Metagross | steel | psychic |

```
247  Tyranitar      rock      dark
93      Gengar     ghost    poison
459  Abomasnow     grass       ice
372  Salamence    dragon    flying
247  Tyranitar      rock      dark
375  Metagross     steel   psychic
372  Salamence    dragon    flying
```

There we have it! These are the strongest Pokemon of each type. Notice Salamence appears twice; since it is both dragon and flying type, it seems that it is the strongest Pokemon of both those types. Now, I'm interested in knowing: what are the strongest stats for this group of strongest Pokemon; is there a particular stat that most of them are highest in?

In [94]:
```
#base total chart for strongest Pokemon of each type
alt.Chart(bytype).mark_bar().encode(
    x='base_total',
    y='name',
    color='name',
)
```

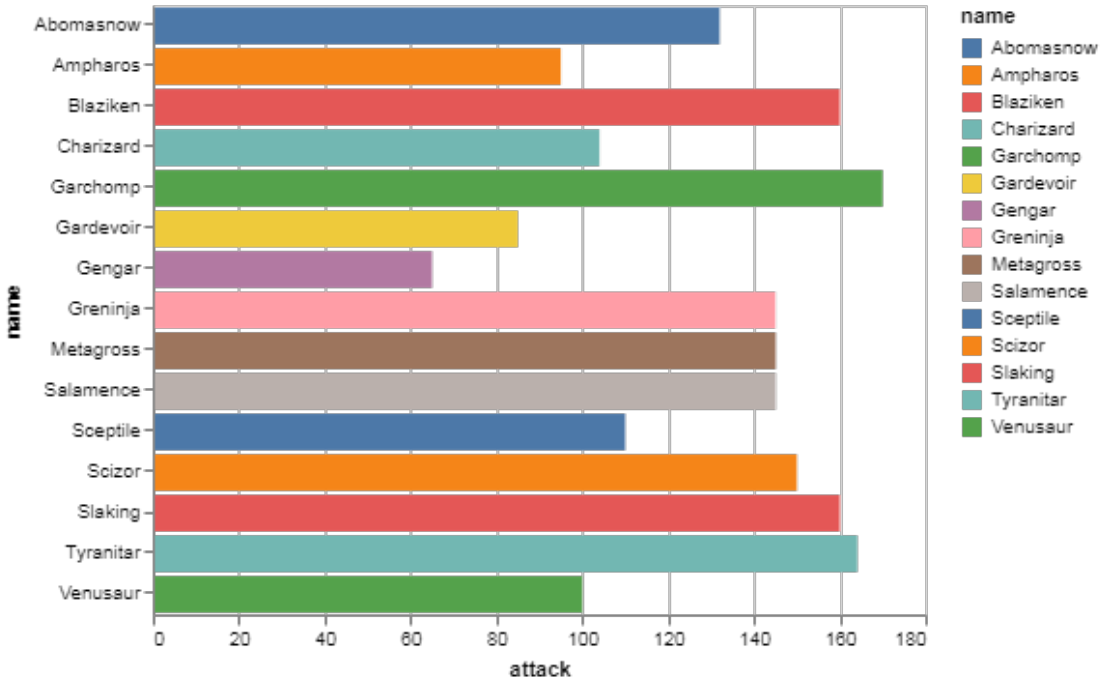
Out[94]:




In [95]:
```
#attack chart for strongest Pokemon of each type
alt.Chart(bytype).mark_bar().encode(
    x='attack',
```

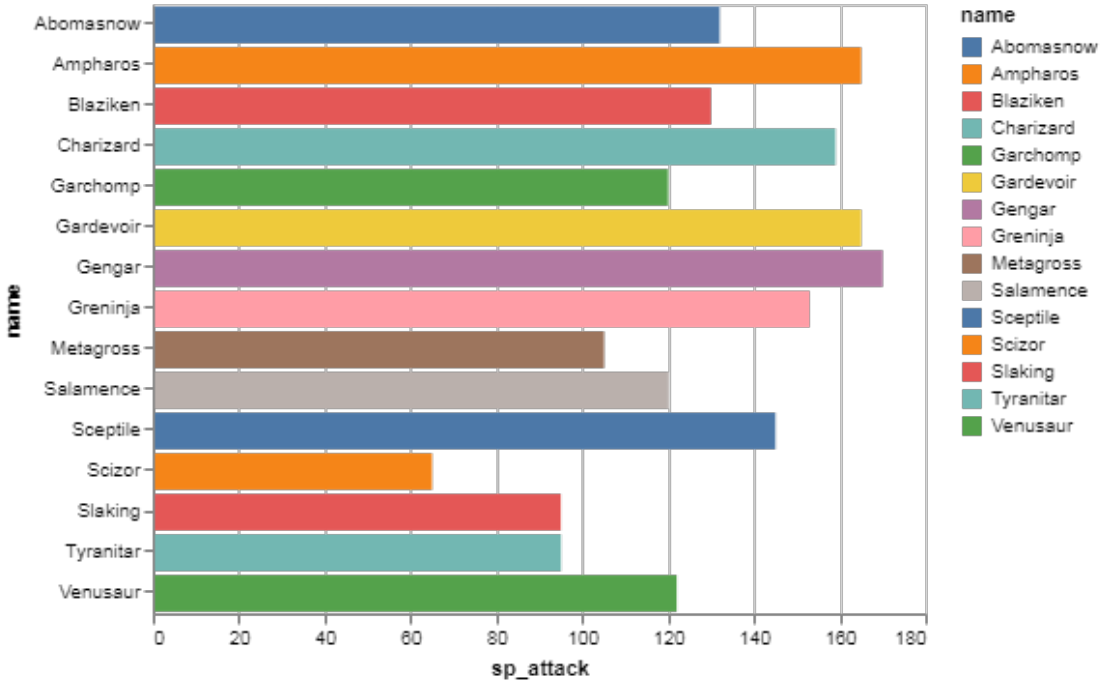
```
        y='name',
        color='name',
    )
```

Out[95]:



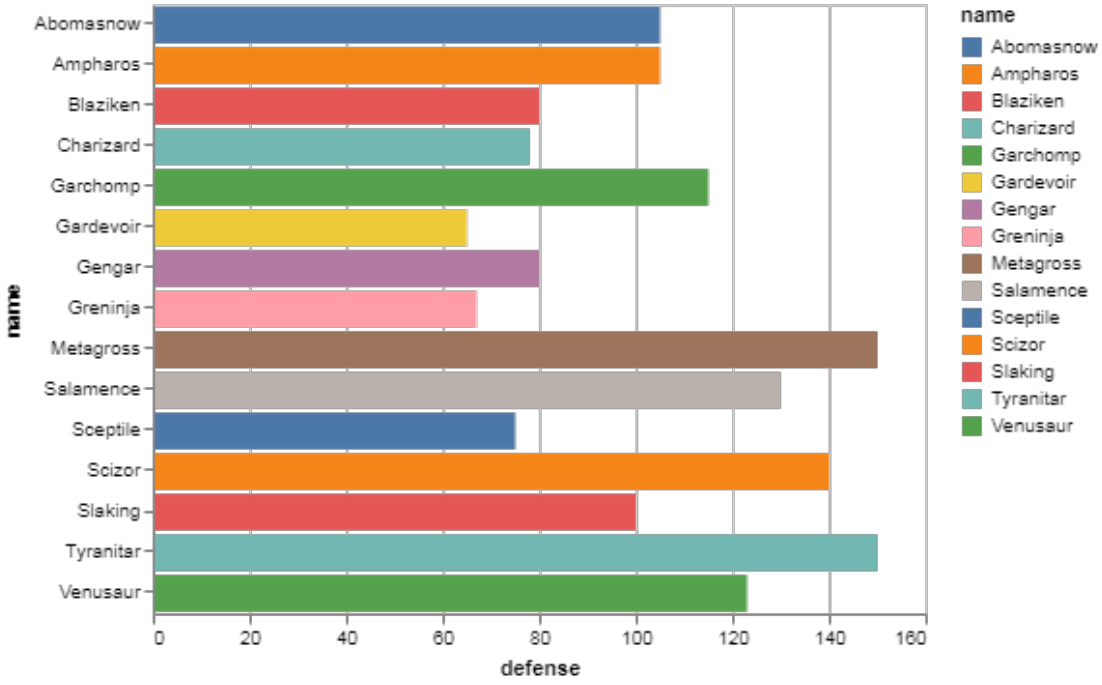

In [98]:
```
#special attack chart for strongest Pokemon of each type
alt.Chart(bytype).mark_bar().encode(
    x='sp_attack',
    y='name',
    color='name',
)
```

Out[98]:



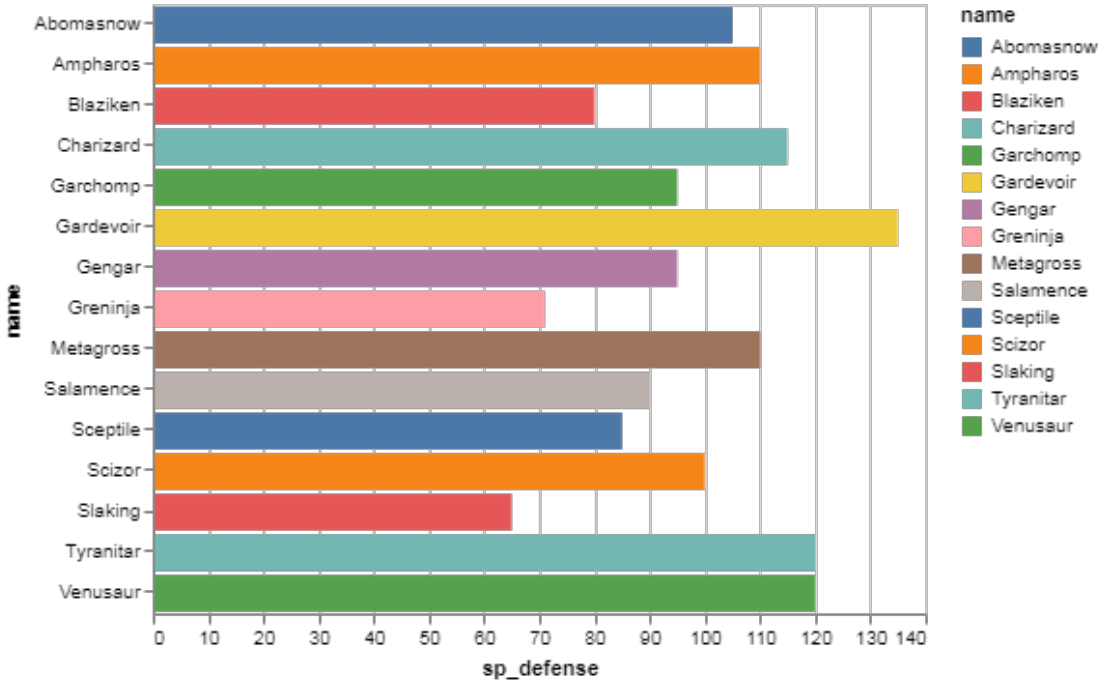

In [99]:
```
#defense chart for strongest Pokemon of each type
alt.Chart(bytype).mark_bar().encode(
    x='defense',
    y='name',
    color='name',
)
```

Out[99]:



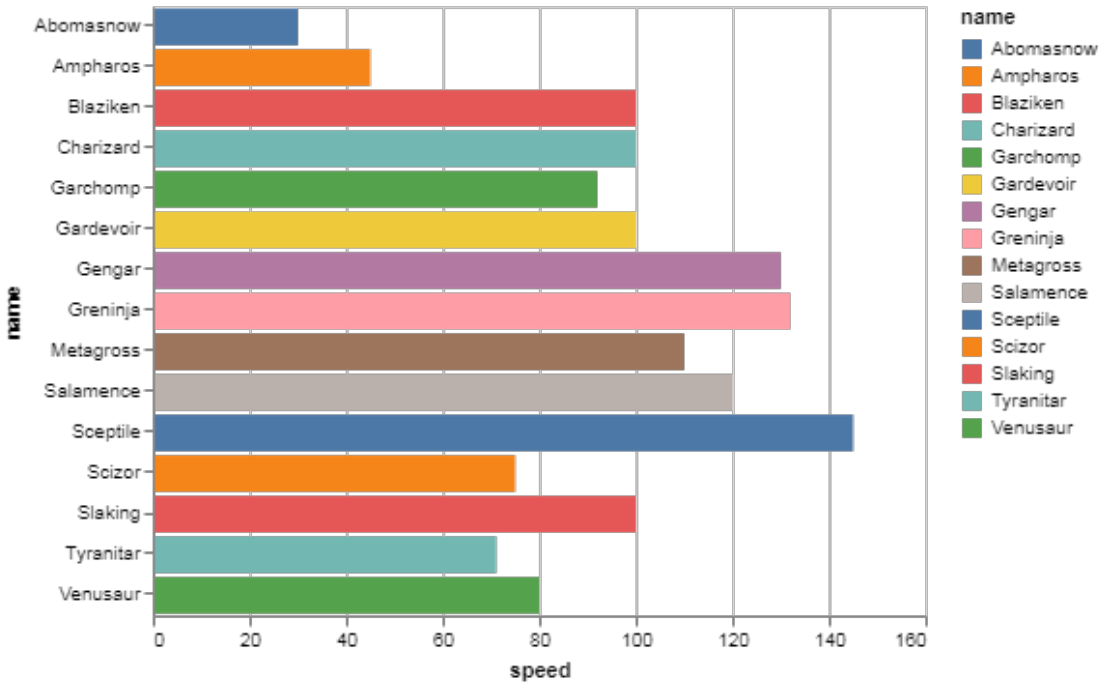

In [100]:
```
#special defense chart for strongest Pokemon of each type
alt.Chart(bytype).mark_bar().encode(
    x='sp_defense',
    y='name',
    color='name',
)
```

Out[100]:



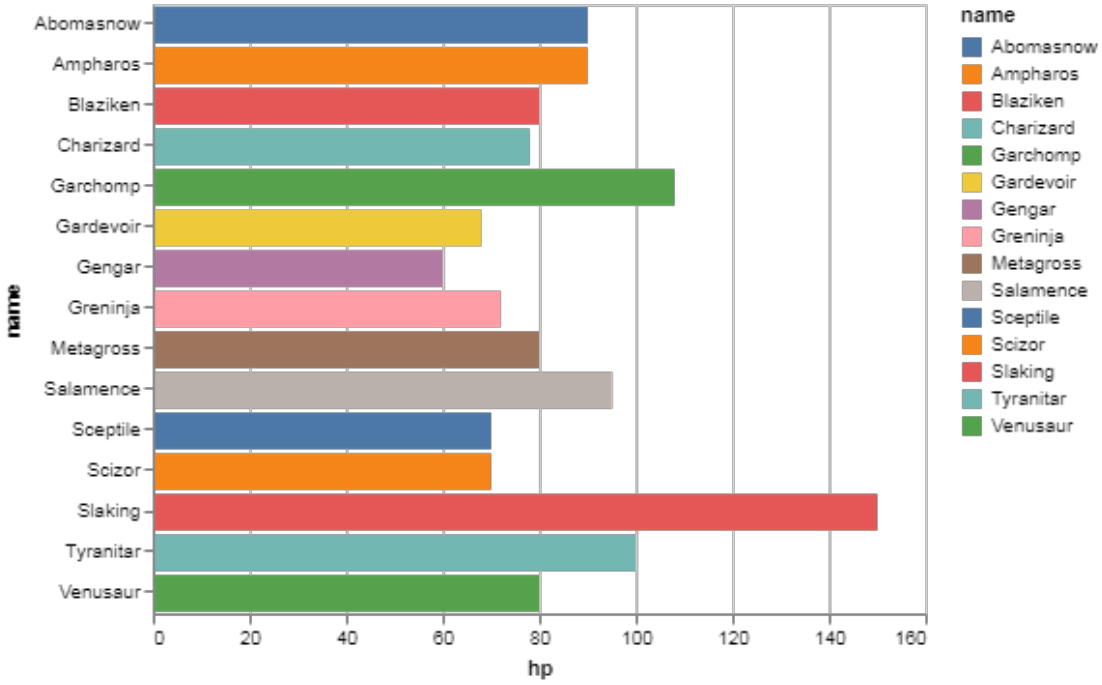

In [101]:
```
#speed chart for strongest Pokemon of each type
alt.Chart(bytype).mark_bar().encode(
    x='speed',
    y='name',
    color='name',
)
```

Out[101]:




In [102]:
```
#health point chart for strongest Pokemon of each type
alt.Chart(bytype).mark_bar().encode(
    x='hp',
    y='name',
    color='name',
)
```

Out[102]:




All of them have a base total of 550 or more, 2/3 of them have attacks over 120, and all but four have a special attack near or greater than 120. However, most of them do not seem very high in defense; only 5 have a defense greater than 120, and only 3 have a special defense near or greater than 120. Only 3 have health points near or greater than 100. Speed seems to have the most normal looking distribution.

These numbers are somewhat arbitrary; I've used them as a point of comparison because they visually look like a reasonable cutoff for "stronger" vs "weaker" Pokemon in a specific statistic.

From looking at this, it seems there aren't clear trends in what makes a Pokemon strong; there are a variety of combinations of stats that can make each Pokemon as strong as it is. Out of the strongest of each type, it seems that most of them have higher attack stats, but a few of them seem to rely more strongly on defense. There are over 800 Pokemon, so it makes sense that there is such a variety of combinations of battle stats that make each species unique.

## 1.3 Creating Teams

What are the six Pokemon with the highest total stats? If we want to create a Pokemon team with them, which Pokemon would they be?

```
In [103]: #Making a dataframe for the team of Pokemon with highest stats
          high_list = poke.sort_values(['base_total'], ascending = False)
          high = high_list.head(6)
          high[['name','base_total','type1','type2','is_legendary']]
```

```
Out[103]:          name  base_total    type1   type2  is_legendary
          149    Mewtwo         780  psychic     NaN             1
          383  Rayquaza         780   dragon  flying             1
```

```
382   Groudon          770   ground     NaN             1
381    Kyogre          770    water     NaN             1
492    Arceus          720   normal     NaN             1
717   Zygarde          708   dragon  ground             1
```

This is definitely interesting, but notice that all the Pokemon are legendary. I particularly want to look at non-legendary Pokemon, since they are more widely available to catch in games. Let's look at the same thing, but without legendary Pokemon.

```
In [104]: #dataframe excluding legendary Pokemon
          nonleg=poke[poke.is_legendary==False]

          #highest base total stats excluding legendary Pokemon
          high_nonleg = nonleg.sort_values(['base_total'], ascending = False)
          highest = high_nonleg.head(6)
          highest[['name','base_total','type1','type2','is_legendary']]
```

Out[104]:

| | name | base_total | type1 | type2 | is_legendary |
|---|---|---|---|---|---|
| 372 | Salamence | 700 | dragon | flying | 0 |
| 375 | Metagross | 700 | steel | psychic | 0 |
| 444 | Garchomp | 700 | dragon | ground | 0 |
| 247 | Tyranitar | 700 | rock | dark | 0 |
| 288 | Slaking | 670 | normal | NaN | 0 |
| 129 | Gyarados | 640 | water | flying | 0 |

This is a fairly diverse team in terms of Pokemon types. However, is it really the best team in specific situations? Let's pick a Pokemon game and find a Champion to battle. I'll use the example of Pokemon LeafGreen, where the Champion's name is Green. His team consists of the following Pokemon: Pidgeot, Alakazam, Rhydon, Gyarados, Arcanine, and Venusaur. We will make a dataframe for his team, and then see if we can build an ideal team to fight his based off of type.

```
In [105]: #Champion Green's team
          Pidgeot = poke[poke.name=="Pidgeot"]
          Alakazam = poke[poke.name=="Alakazam"]
          Rhydon = poke[poke.name=="Rhydon"]
          Gyarados = poke[poke.name=="Gyarados"]
          Arcanine = poke[poke.name=="Arcanine"]
          Venusaur = poke[poke.name=="Venusaur"]

          green = pd.concat([Pidgeot, Alakazam, Rhydon, Gyarados, Arcanine, Venusaur])

          #green = pd.concat((poke[poke.name=="Pidgeot"],poke[poke.name=="Alakazam"],poke[poke
          green[['name','type1','type2','attack','defense','sp_attack','sp_defense','speed',]]
```

Out[105]:

| | name | type1 | type2 | attack | defense | sp_attack | sp_defense | speed |
|---|---|---|---|---|---|---|---|---|
| 17 | Pidgeot | normal | flying | 80 | 80 | 135 | 80 | 121 |
| 64 | Alakazam | psychic | NaN | 50 | 65 | 175 | 105 | 150 |
| 111 | Rhydon | ground | rock | 130 | 120 | 45 | 45 | 40 |

```
129  Gyarados    water  flying     155      109         70         130     81
58   Arcanine     fire     NaN     110       80        100          80     95
2    Venusaur    grass  poison     100      123        122         120     80
```

What types are each Pokemon weakest against? From the dataset, we will see columns labeled "against_" followed by a Pokemon type. If the value is less than 1, then that type is not very effective against the particular Pokemon; if the value is 1, then it has normal effectiveness; if the value is greater than 1, then that type is super effective.

```
In [106]: green=green.rename(columns={'against_fight':'against_fighting'})
          gagainst=green[['name','against_bug','against_dark','against_dragon','against_electri
          gagainst
```

Out[106]:

```
          name  against_bug  against_dark  against_dragon  against_electric  \
17    Pidgeot          0.5           1.0             1.0               2.0
64   Alakazam          2.0           2.0             1.0               1.0
111    Rhydon          1.0           1.0             1.0               0.0
129  Gyarados          0.5           1.0             1.0               4.0
58   Arcanine          0.5           1.0             1.0               1.0
2    Venusaur          1.0           1.0             1.0               0.5

     against_fairy  against_fighting  against_fire  against_flying  \
17             1.0               1.0           1.0             1.0
64             1.0               0.5           1.0             1.0
111            1.0               2.0           0.5             0.5
129            1.0               0.5           0.5             1.0
58             0.5               1.0           0.5             1.0
2              0.5               0.5           2.0             2.0

     against_ghost  against_grass  against_ground  against_ice  \
17             0.0           0.50             0.0          2.0
64             2.0           1.00             1.0          1.0
111            1.0           4.00             2.0          2.0
129            1.0           1.00             0.0          1.0
58             1.0           0.50             2.0          0.5
2              1.0           0.25             1.0          2.0

     against_normal  against_poison  against_psychic  against_rock  \
17              1.0            1.00              1.0           2.0
64              1.0            1.00              0.5           1.0
111             0.5            0.25              1.0           0.5
129             1.0            1.00              1.0           2.0
58              1.0            1.00              1.0           2.0
2               1.0            1.00              2.0           1.0

     against_steel  against_water
17             1.0            1.0
64             1.0            1.0
```

```
111          2.0          4.0
129          0.5          0.5
58           0.5          2.0
2            1.0          0.5
```

From this, it seems Pidgeot is weakest against electric, ice, and rock.

Alakazam is weakest against bug, dark, and ghost.

Rhydon is weak against fighting, ground, ice, and steel, but super weak against grass and water.

Gyarados is weak against rock and super weak against electric.

Arcanine is weak against ground, rock, and water.

Venusaur is weak against fire, flying, ice, and psychic.

So, the types we want on our team are electric, ice, rock, bug, dark, ghost, grass, water, ground, fire, flying, and psychic. However, we do not need all of these types. If we have electric, water, ice, and either bug, dark, or ghost types on our team, then we've covered the weaknesses of each of Green's Pokemon.

```
In [110]: leaf = pd.concat([electric[electric.is_legendary==False].sort_values(['base_total'],
          leaf[['name','type1','type2']]
```

```
Out[110]:          name     type1   type2
          180   Ampharos  electric     NaN
          657   Greninja     water    dark
          459  Abomasnow     grass     ice
          211     Scizor       bug   steel
          247  Tyranitar      rock    dark
          93      Gengar     ghost  poison
```

Cool. We have a well-rounded team, equipped to take on Champion Green. We've accomplished our goal!

But wait! Each Pokemon game has a certain generation of Pokemon available. In Pokemon LeafGreen, there is only generation 1 Pokemon present. Using the information about types we want to use against Green, what would an ideal team of Pokemon realistically look like in the game, when Pokemon from other generations aren't present?

```
In [111]: leafgreen = pd.concat([electric[electric.generation==1].sort_values(['base_total'], a
          leafgreen[['name','type1','type2','is_legendary','generation']]
```

```
Out[111]:          name     type1   type2  is_legendary  generation
          144    Zapdos  electric  flying             1           1
          129  Gyarados     water  flying             0           1
          143  Articuno       ice  flying             1           1
          126    Pinsir       bug     NaN             0           1
          52    Persian    normal    dark             0           1
          93     Gengar     ghost  poison             0           1
```

We now have a team that's all from Generation 1, so we could conceivably have this team in Pokemon LeafGreen. However, we didn't select for only nonlegendary Pokemon, so it would be a little difficult to get this team. Let's get our final team by choosing the strongest nonlegendary Pokemon from Generation 1 of each of the types we identified as Green's weakness.

```
In [112]: #The types of Pokemon Green's team is weak against
          gteam = 'electric','water','ice','bug','dark','ghost'

          #Creating a dataframe where we filter by legendary status, generation, and type, and
          lg = pd.DataFrame()
          for i in gteam:
              i = nonleg[(nonleg.type1==i) | (nonleg.type2==i)]
              lg = lg.append([i[i.generation==1].sort_values(['base_total'],ascending=False).he
          lg[['name','type1','type2','is_legendary','generation']]
```

Out[112]:

| | name | type1 | type2 | is_legendary | generation |
|---|---|---|---|---|---|
| 134 | Jolteon | electric | NaN | 0 | 1 |
| 129 | Gyarados | water | flying | 0 | 1 |
| 130 | Lapras | water | ice | 0 | 1 |
| 126 | Pinsir | bug | NaN | 0 | 1 |
| 52 | Persian | normal | dark | 0 | 1 |
| 93 | Gengar | ghost | poison | 0 | 1 |

We finally made it! Our end result tells us that the best team to fight Champion Green with in Pokemon LeafGreen is: Jolteon, Gyarados, Lapras, Pinsir, Persian, and Gengar.

### 1.4 Conclusion

We looked at Pokemon battle stats and tried to find correlations between stats, and then between battle stats vs other Pokemon properties. In both cases, there was no correlation present. Because of this, we could not predict one property based off of others. This makes sense because Pokemon species are meant to be diverse to give every player a unique experience, and having clear trends in what makes a "good" Pokemon would mean everyone would gravitate toward the same ones.

After that, we defined what it means to be a strong Pokemon, which we did simply by summing across battle stats for each Pokemon; we could have taken a more sophisticated route, but this sum was easy to use, especially since it was already present in the dataset. We then used that definition to create a few teams; the first was a team of the six "strongest" Pokemon, which all happened to be legendary Pokemon; the second was a team of the six "strongest" non-legendary Pokemon; the third team was created to beat Champion Green from the Pokemon LeafGreen game, using our definition of a strong Pokemon and the information we had about type strength and weaknesses from the dataset; the fourth team built on that principle and let us build a team we could realistically have in-game; and the fifth team brought everything together to select only the strongest non-legendary Pokemon of Generation 1 that were strongest against Green. Despite not gaining much clear information about battle stats, we were able to create some well-rounded and realistic Pokemon teams. Perhaps I will use this final team myself the next time I play Pokemon LeafGreen.