Winner in Pokémon Combat Prediction

José A. Rodríguez
Martín Francisco Zelada
Juan Cruz Caceres Miranda

CoderHouse - Data Science

21 de Abríl, 2023

Indice

1. Descripción del mundo Pokémon

Pokémon es una franquicia de medios que comenzó como un videojuego RPG, pero debido a su popularidad ha logrado expandirse a otros medios de entretenimiento como series de televisión, películas, juegos de cartas, ropa, entre otros, convirtiéndose en una marca que es reconocida en el mercado mundial. La misión en estos juegos es capturar y entrenar a los Pokémon, criaturas similares a lo que serían los animales en el mundo real, pero con diversos poderes. El nombre del juego proviene de la contracción de "Pocket Monster" (Monstruos de bolsillo).

2. Tabla de versionado.

Se ensayaron y probaron diferentes hipótesis que no condujeron a buenos resultados. El trabajo actual es resultado de haber tomado la decisión de reiniciar el proyecto, quizás en el momento justo.

3. Objetivos del modelo.

En base a las diferentes variables del dataset se pretende analizar cuales son lo suficientemente fuertes para influir en el resultado de un enfrentamiento.

Habiendo determinado cuales variables tomar, y cuales descartar, se procederá a ensayar diferentes modelos de machine learning, con el objetivo de determinar, o mas bien servir de guía, respecto a cuál Pokémon puede, de antemano, considerarse un probable ganador del enfrentamiento

3.1 Naturaleza del problema

En base a las características de la variable objetivo, la cual es una variable discreta que solo puede tomar dos valores, 0 o 1, el problema a resolver se puede categorizar como un problema de clasificación.

4. Descripción de los datos

Nos valemos de dos fuentes diferentes:

1- De PokeApi, extrajimos la información de los 801 Pokémon, así como un cuadro de doble entrada, en el cual constan las ventajas entre tipos de Pokémon.

2- De Kaggle, obtuvimos un dataset que contiene información sobre enfrentamientos Pokémon, con 50.000 (Cincuenta mil) entradas, correspondiendo cada una a un entrenamiento diferente.

A continuación, ofrecemos información de cada dataframe:

Tabla "pokemon_ds"

- **pokedex_id:** un numero único que identifica a cada especie Pokémon
- **nombre:** El nombre (utilizado internacionalmente, excepto en Japón) del Pokémon
- **tipo1:** La naturaleza principal de los ataques del Pokémon. Se utiliza para calcular ventajas y desventajas sobre otros Pokémon.
- **tipo2:** La naturaleza secundaria de los ataques del Pokémon. Se suele obtener al evolucionar, y no todas las especies poseen una.
- **altura_m:** La altura, expresada en metros.
- **peso_kg:** El peso, expresado en kilogramos.
- **hp:** Los puntos de vida. Durante el combate los puntos bajan con cada ataque recibido, y pueden ser regenerados de diversas formas. Al llegar a cero, el Pokémon queda fuera de combate.
- **ataque:** Los puntos base para ataques normales.
- **defensa:** Los puntos base de defensa contra ataques normales.
- **ataque_esp:** Los puntos base para ataques especiales.
- **defensa_esp:** Los puntos base de defensa contra ataques especiales.
- **velocidad:** La velocidad. Se utiliza principalmente para determinar que Pokémon ataca primero.

Tabla "combat":

- **First_pokemon:** El primer contendiente, identificado por su número de pokedex
- **Second_pokemon:** El segundo contendiente, identificado por su número de pokedex.
- **Winner:** El ganador, identificado por su número de pokedex

Tabla "Multiplicador" (o de ventajas)

	normal	fighting	flying	poison	ground	rock	bug	ghost	steel	fire	water	grass	electric	psychic	ice	dragon	dark	fairy
normal	1.0	1.0	1.0	1.0	1.0	0.5	1.0	0.0	0.5	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0
fighting	2.0	1.0	0.5	0.5	1.0	2.0	0.5	0.0	2.0	1.0	1.0	1.0	1.0	0.5	2.0	1.0	2.0	0.5
flying	1.0	2.0	1.0	1.0	1.0	0.5	2.0	1.0	0.5	1.0	1.0	2.0	0.5	1.0	1.0	1.0	1.0	1.0
poison	1.0	1.0	1.0	0.5	0.5	0.5	1.0	0.5	0.0	1.0	1.0	2.0	1.0	1.0	1.0	1.0	1.0	2.0
ground	1.0	1.0	0.0	2.0	1.0	2.0	0.5	1.0	2.0	2.0	1.0	0.5	2.0	1.0	1.0	1.0	1.0	1.0
rock	1.0	0.5	2.0	1.0	0.5	1.0	2.0	1.0	0.5	2.0	1.0	1.0	1.0	1.0	2.0	1.0	1.0	1.0
bug	1.0	0.5	0.5	0.5	1.0	1.0	1.0	0.5	0.5	0.5	1.0	2.0	1.0	2.0	1.0	1.0	2.0	0.5
ghost	0.0	1.0	1.0	1.0	1.0	1.0	1.0	2.0	1.0	1.0	1.0	1.0	1.0	2.0	1.0	1.0	0.5	1.0
steel	1.0	1.0	1.0	1.0	1.0	2.0	1.0	1.0	0.5	0.5	0.5	1.0	0.5	1.0	2.0	1.0	1.0	2.0
fire	1.0	1.0	1.0	1.0	1.0	0.5	2.0	1.0	2.0	0.5	0.5	2.0	1.0	1.0	2.0	0.5	1.0	1.0
water	1.0	1.0	1.0	1.0	2.0	2.0	1.0	1.0	1.0	2.0	0.5	0.5	1.0	1.0	1.0	0.5	1.0	1.0
grass	1.0	1.0	0.5	0.5	2.0	2.0	0.5	1.0	0.5	0.5	2.0	0.5	1.0	1.0	1.0	0.5	1.0	1.0
electric	1.0	1.0	2.0	1.0	0.0	1.0	1.0	1.0	1.0	1.0	2.0	0.5	0.5	1.0	1.0	0.5	1.0	1.0
psychic	1.0	2.0	1.0	2.0	1.0	1.0	1.0	1.0	0.5	1.0	1.0	1.0	1.0	0.5	1.0	1.0	0.0	1.0
ice	1.0	1.0	2.0	1.0	2.0	1.0	1.0	1.0	0.5	0.5	0.5	2.0	1.0	1.0	0.5	2.0	1.0	1.0
dragon	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	0.5	1.0	1.0	1.0	1.0	1.0	1.0	2.0	1.0	0.0
dark	1.0	0.5	1.0	1.0	1.0	1.0	1.0	2.0	1.0	1.0	1.0	1.0	1.0	2.0	1.0	1.0	0.5	0.5
fairy	1.0	2.0	1.0	0.5	1.0	1.0	1.0	1.0	0.5	0.5	1.0	1.0	1.0	1.0	1.0	2.0	2.0	1.0

La misma se debe entender de la siguiente manera: Los tipos de las filas tienen ventaja "x" sobre los tipos de las columnas. Por ejemplo, el tipo normal tiene ventaja de 1 contra el tipo "normal", "fighting", "flying", "poison", "ground"; de 0,5 contra el tipo "rock"; de 1 contra "bug"; de 0 contra "ghost", etc.

En principio se realizó un análisis de todas las variables. No se detectaron nulos. Se detectaron duplicados solo en la tabla combat. Sin embargo, consideramos que el hecho de que dos Pokémon se enfrenten varias veces, y el resultado sea el mismo, es información relevante y significativa para el modelo.

El dataset contiene 12 variables con 801 registros. En él se encuentra información del tipo, tamaño y peso, así como los stats de los Pokémon.

A continuación, se muestra una tabla resumen de todas las variables, con información de nulos, tipos de datos, y cantidad de valores únicos.

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 801 entries, 0 to 800
Data columns (total 12 columns):
 #   Column       Non-Null Count  Dtype
---  ------       --------------  -----
 0   pokedex_id   801 non-null    int64
 1   nombre       801 non-null    object
 2   tipo1        801 non-null    object
 3   tipo2        801 non-null    object
 4   altura_m     801 non-null    float64
 5   peso_kg      801 non-null    float64
 6   hp           801 non-null    int64
 7   ataque       801 non-null    int64
 8   defensa      801 non-null    int64
 9   ataque_esp   801 non-null    int64
 10  defensa_esp  801 non-null    int64
 11  velocidad    801 non-null    int64
dtypes: float64(2), int64(7), object(3)
memory usage: 81.4+ KB
```

A continuación, se detallan los principales resultados del EDA, de las demás variables estudiadas.

5. EDA: Exploratory Data Analysis

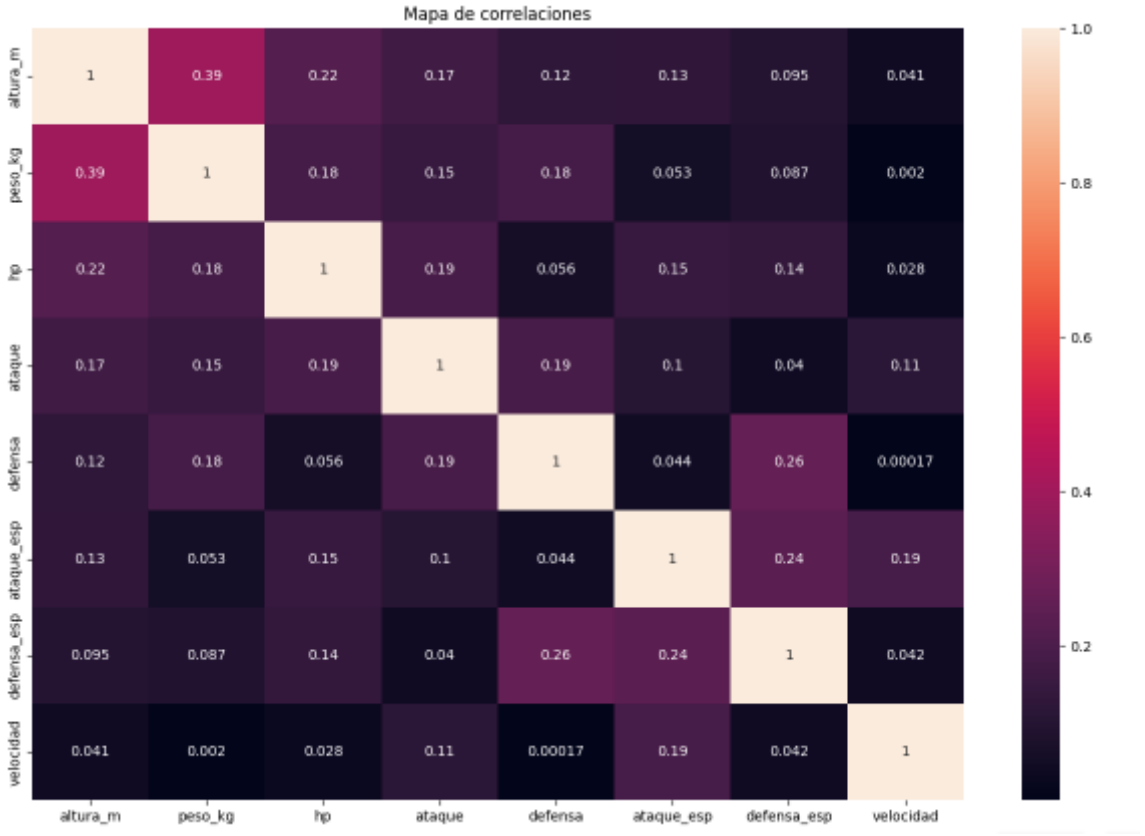
En el análisis exploratorio se estudiaron las distribuciones de las variables numéricas.

Entre los resultados que se destacan, se observaron dos columnas (altura y peso) con una desviación estándar demasiado alta. El análisis posterior demostró que la columna presentaba con una asimetría importante, al igual que la curtosis, debido a una gran cantidad de valores atípicos y extremos, motivo por el cual se la consideró perjudicial para el desarrollo del modelo, descartándola.

Las demás variables presentaban una distribución normal, por lo cual se decidió conservarla.

Dado que la documentación disponible respecto a cada feature, y como interactúan entre ellos es basta y accesible, poco se puede inferir del EDA que no pueda concluirse de leer la documentación del juego.

No obstante, aunque la correlación entre las variables parece, en principio, baja, tal como se observa en el siguiente grafico; luego del proceso de feature engineering podemos encontrar correlaciones mucho mas fuertes, que validan nuestra intuición inicial respecto a la factibilidad de producir un modelo de ML con los objetivos que tenemos en mente.



La unica correlación mas o menos importante que podemos ver es entre peso y altura, dos variables que, como ya mencionamos, descartamos.

6. Creación de nuevas variables

A partir de la información disponible en la tabla “pokemon_ds”, complementamos la tabla “combat”, agregando la siguiente información:

```
(50000, 18)
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 50000 entries, 0 to 49999
Data columns (total 18 columns):
 #   Column          Non-Null Count  Dtype
---  ------          --------------  -----
 0   First_pokemon   50000 non-null  int64
 1   Second_pokemon  50000 non-null  int64
 2   Winner          50000 non-null  int64
 3   First_Adv       50000 non-null  float64
 4   Sec_Adv         50000 non-null  float64
 5   1Ataque         50000 non-null  int64
 6   1Defensa        50000 non-null  int64
 7   1AtaqueEsp      50000 non-null  int64
 8   1DefensaEsp     50000 non-null  int64
 9   1velocidad      50000 non-null  int64
 10  1hp             50000 non-null  int64
 11  2Ataque         50000 non-null  int64
 12  2Defensa        50000 non-null  int64
 13  2AtaqueEsp      50000 non-null  int64
 14  2DefensaEsp     50000 non-null  int64
 15  2velocidad      50000 non-null  int64
 16  2hp             50000 non-null  int64
 17  binwin          50000 non-null  int64
dtypes: float64(2), int64(16)
memory usage: 6.9 MB
None
```

Así, nuestra nueva tabla posee 18 columnas, todas numéricas. La información contenida en las mismas es la siguiente:

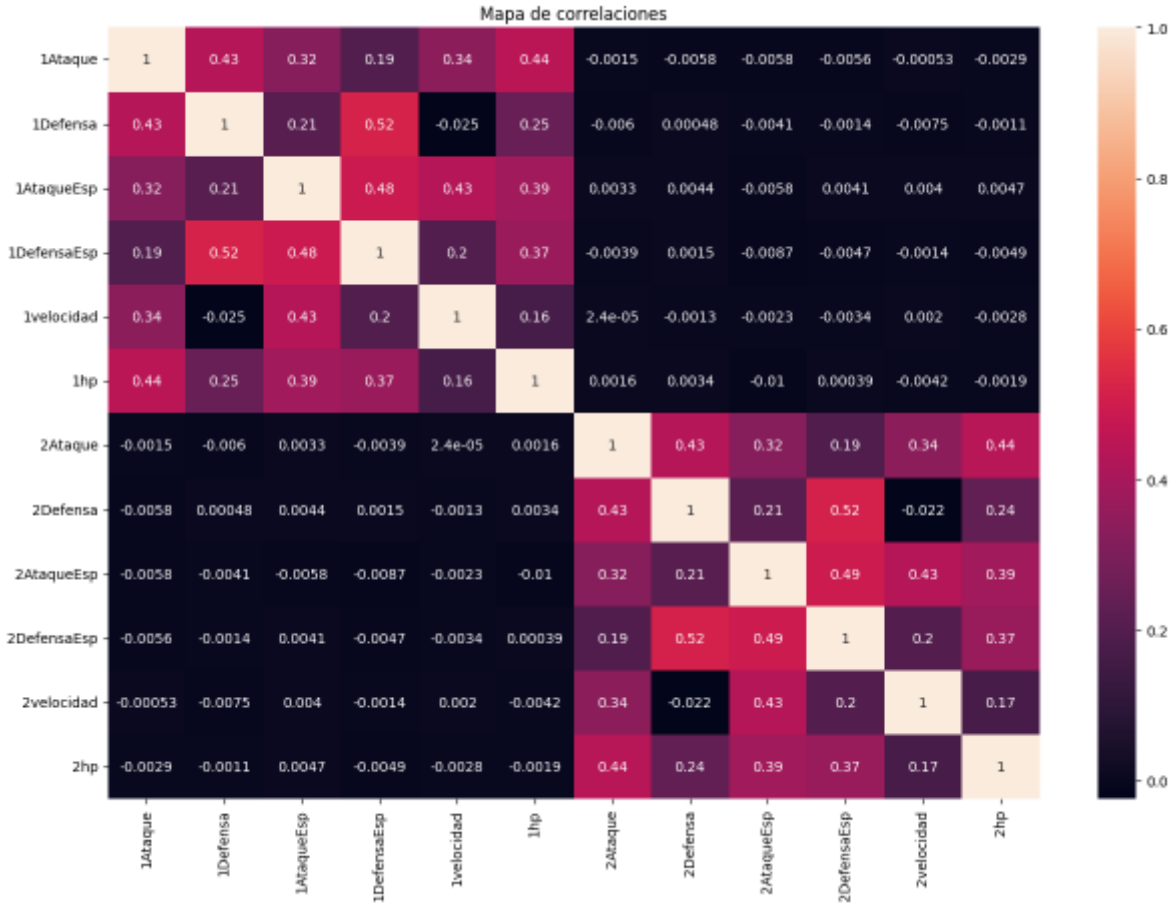
- **First_pokemon:** El número de pokedex del primer contendiente.
- **Second_pokemon:** El número de pokedex del segundo contendiente.
- **Winner:** El número de pokedex del ganador.

Vamos a prescindir de estas tres columnas, pues al ser numéricas, podrían confundir al modelo si las utilizamos para entrenarlas (El modelo podría interpretar que un Pokémon con un numero de pokedex más alto "tiene más" de algo). El motivo por el cual no las eliminamos antes consiste en que necesitamos esas tres columnas para poder construir las siguientes 15:

- **First_Adv:** La ventaja del primer Pokémon sobre el segundo
- **Sec_Adv:** La ventaja del segundo Pokémon sobre el primero
- **1Ataque:** Los puntos de ataque del primer Pokémon
- **1Defensa:** Los puntos de defensa del primer Pokémon
- **1AtaqueEsp:** Los puntos de ataque especial del primer Pokémon

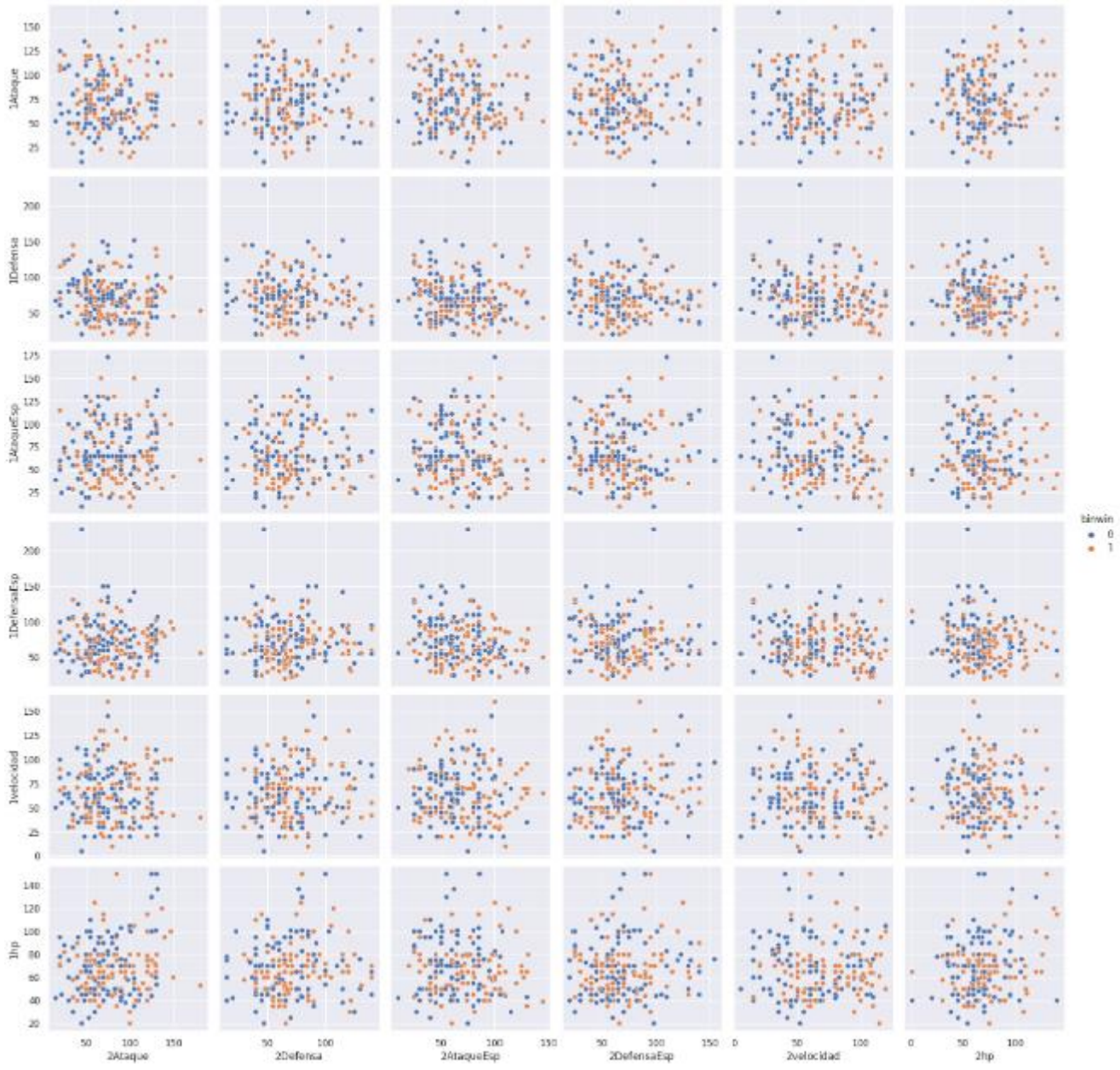
- **1DefensaEsp:** Los puntos de defensa especial del primer Pokémon
- **1velocidad:** Los puntos de velocidad del primer Pokémon
- **1hp:** Los puntos de vida del primer Pokémon
- **2Ataque:** Los puntos de ataque del segundo Pokémon
- **2Defensa:** Los puntos de defensa del segundo Pokémon
- **2AtaqueEsp:** Los puntos de ataque especial del segundo Pokémon
- **2DefensaEsp:** Los puntos de defensa especial del segundo Pokémon
- **2velocidad:** Los puntos de velocidad del segundo Pokémon
- **2hp:** Los puntos de vida del segundo Pokémon
- **binwin:** 0 si el ganador es el primer Pokémon, 1 si es el segundo. Es además la variable a predecir por el modelo

En la tabla resultante podemos observar correlaciones mas fuertes, tal como se ve en la siguiente imagen:

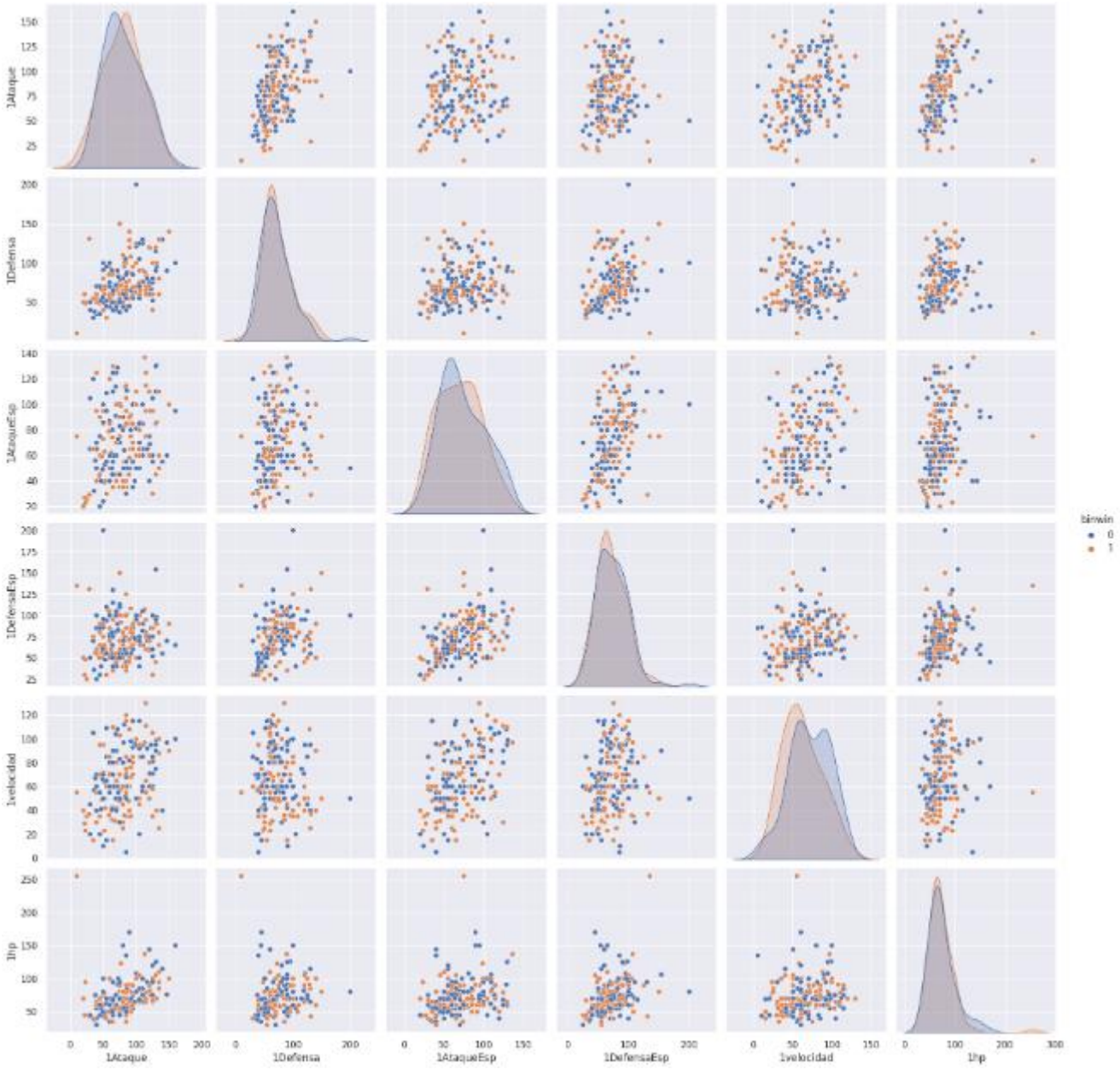


Ademas, en los tres siguientes pairplots, donde se cruzan todas las columnas de la tabla, con todas las demas, podemos observar tendencias, en forma de regiones o lagunas donde todos los

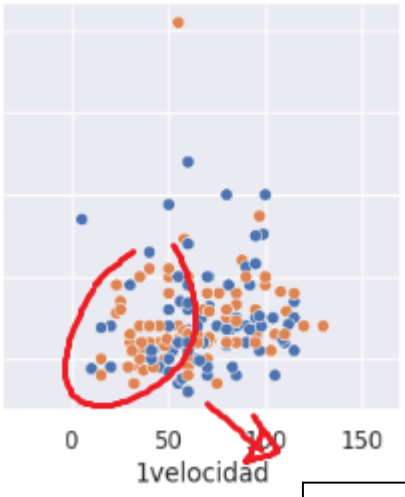
puntos son del mismo color, que si bien no son demasiado pronunciadas, en conjunto deberian ser suficientes para aproximarnos a un resultado.



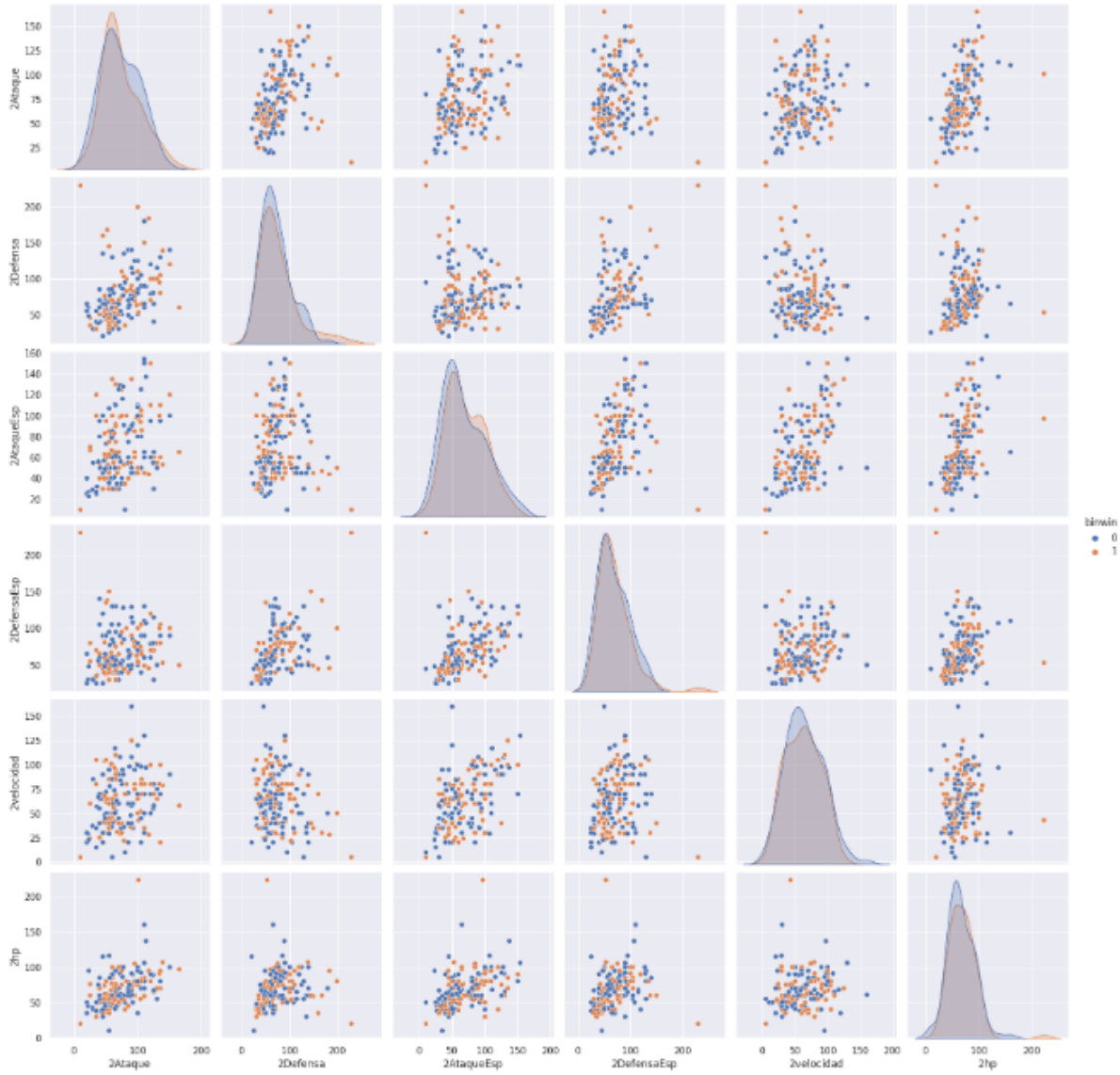
Pairplot 1. En el eje vertical, los stats del primer Pokémon. En el eje horizontal, del segundo.



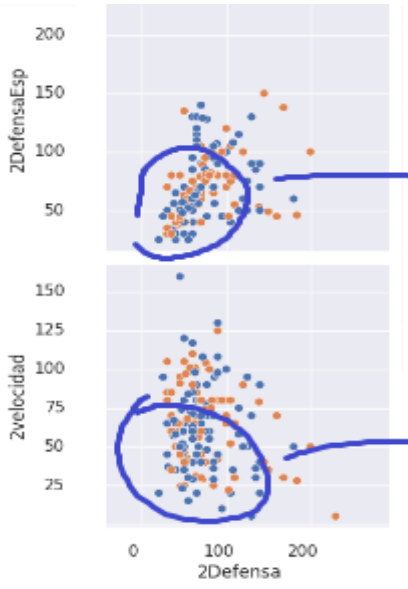
Pairplot 2. En ambos ejes, los stats del primer Pokémon.



La tendencia es clara, los Pokémon con menos de 50 de velocidad, rara vez ganan un combate.



Pairplot 3. En ambos ejes, los stats del segundo Pokémon.



Los Pokémon que tienen, a la vez, poca defensa común y especial, tampoco parecen ganar demasiado seguido.

Nuevamente, observamos la tendencia de la velocidad. Para los Pokémon con poca defensa, ese umbral parece aumentar hasta cerca de 75.

7. Algoritmo Elegido.

Tal como dijimos en el apartado anterior, la variable a predecir es “binwin”.
Los modelos utilizados para predecir la variable fueron los siguientes:

1. **Random Forest**
2. **Red Neuronal**
3. **XGBoost**

En un primer lugar se testearon los modelos con el dataset completo, filtrando las variables que se mencionaron en la sección anterior, tomando una división de Train/Test con una relación de 80/20.
Luego de analizar las métricas, y con el fin de optimizar recursos, se descarto la red neuronal, y se procedió a optimizar los otros dos modelos desde un ajuste en los hiperparámetros con la técnica “random search”.

Tal como se menciona en la notebook, la métrica que se intenta optimizar es la F1-Score. Esto responde a un motivo meramente epistemológico: dada la naturaleza de los datos y el modelo construido, así como los objetivos, no hay diferencia esencial entre los falsos positivos y los falsos negativos.

Se detallan los modelos implementados:

Random Forest:

Se aplicó el modelo de regresión logística con los parámetros de scikit-learn por defecto. Se utilizo el algoritmo “Random Forest Regressor”, en vez del Classifier. Originalmente esto ocurrió como resultado de un descuido, pero luego se observaron dos ventajas respecto del mismo. La primera, es que las métricas eran superiores. La segunda ventaja, era que el modelo era mas transparente al inclinarse por tal o cual Pokémon, pues su respuesta no era binaria, sino que se acercaba mas o menos a algún resultado. Claro está, que para evaluar el modelo se debió transformar cada predicción a 0 o 1. El límite que se tomó, arbitrariamente, fue el de 0.5, ya que se

consideró que es el que mejor representaría el razonamiento si la decisión fuese tomada por un humano. Mencionamos algunas consideraciones técnicas y epistemológicas al respecto más adelante.

Red Neuronal:

El proceso para ajustar las variables se hizo, en primera instancia, de forma manual y siguiendo un proceso bastante intuitivo de prueba y error, según el cual se iban ajustando los parámetros, y revirtiendo estos ajustes si se notaba que el F1 disminuía.

Este modelo se descarto para el proceso de optimizado por presentar métricas demasiado bajas (cercanas al .5, lo cual, dado que la variable a predecir es binaria, sería equivalente a tirar una moneda)

XGBoost:

Se seleccionó también el modelo XGBoost, uno de los modelos más utilizados en la actualidad para objetivos de clasificación.

Entre sus parámetros se adoptó objective = "binary: logistic", es decir una técnica de regresión logística para clasificación binaria, ya que la variable a predecir es del tipo "0/1".

Debido a que parte del código es reciclado de ejercicios anteriores, y que al primer intento encontramos una métrica bastante elevada, decidimos pasar al proceso de "random search" sin más.

8. Métricas de Desempeño del Modelo.

A continuación, se presentan los resultados obtenidos (redondeados a 4 decimales)

- **Modelos sin optimizar**

	Random Forest	Red Neuronal	XGboost
Accuracy	0.7661	0.5997	0.5687
Precision	0.7675	0.6078	0.5561
Recall	0.7990	0.6814	0.9072
F1- Score	0.7830	0.6425	0.6895

- **Modelos optimizados**

	Random Forest	XGboost
Accuracy	0.7754	0.8186
Precision	0.7757	0.8232
Recall	0.8083	0.8359
F1- Score	0.7881	0.8295

9. Cross Validation.

Se realizo un proceso de Cross Validation sobre el modelo elegido finalmente, para corroborar que no exista un overfitting.

```
from xgboost import XGBClassifier
from sklearn.model_selection import cross_val_score
from sklearn.datasets import make_classification

X, y = make_classification(n_samples=1000, n_features=10, random_state=33)

# Crear modelo XGBoost
model = XGBClassifier()

# Realizar validación cruzada de 50 veces
scores = cross_val_score(model, X, y, cv=50)

# Imprimir resultados
print("Precisión de la validación cruzada: %0.2f (+/- %0.2f)" % (scores.mean(), scores.std() * 2))
```

```
Precisión de la validación cruzada: 0.94 (+/- 0.10)
```

10. Métricas finales del Modelo Optimizado.

El modelo considerado óptimo según los resultados de las métricas obtenidas, fue el XGboost en su versión optimizada, para el cual se obtuvo:

	XGboost optimizado
Accuracy	0.8186
Precision	0.8232
Recall	0.8359
F1- Score	0.8295

11. Futuras líneas

Se ha mencionado que el modelo es abstracto respecto de valores que se saben relevantes.
En ese sentido, se sugieren dos líneas de investigación posibles, según los objetivos del modelo:

Si se pretendiera hacer un modelo más preciso, una posible forma de proceder sería trabajar en el mismo para que considere datos tales como el nivel de los Pokémon, o los objetos que cada uno posee, etc. Se podría incluso trabajar en un deployment del modelo donde se pudiera incluir mas o menos datos, y la predicción fuese mas o menos precisa según esto.

Por otro lado, si el objetivo del modelo sigue siendo el de funcionar como una guía que anticipe que tan difícil será el enfrentamiento, se podría experimentar con el límite dentro del intervalo desde el cual el modelo ofrece tal o cual respuesta. Por ejemplo, uno podría querer tomar precauciones extra fijando ese limite en 0.65 en vez de 0.5. Claro está, que esto tendría repercusiones en la efectividad del modelo, que el data scientist deberá sopesar.

12. Conclusiones:

Al comparar las métricas de los modelos desarrollados, concluimos que el XGboost (optimizado) es el más efectivo para determinar el eventual ganador de un combate Pokémon.

Respecto de la eficacia de nuestro modelo, debemos mencionar que el modelo abstrae muchos factores que sabemos, son relevantes para el resultado.

Algunos ejemplos de esto son: la diferencia de nivel entre los Pokémon, los movimientos aprendidos, o la habilidad de cada jugador. Cualquier jugador de Pokémon sabe, independientemente de su experiencia, que una diferencia de 20 niveles entre los Pokémon rivales es sencillamente lapidaria. En el mismo sentido, un jugador experimentado puede lograr vencer a un Pokémon contra el cual el suyo tiene desventaja, si su habilidad es ampliamente superior a la del rival.

Reformulando el objetivo de este proyecto, el modelo no debe predecir a ciencia cierta, sino funcionar como una guía que permita discriminar los combates más difíciles para el jugador, de los más fáciles.

En el apéndice se evalúa una hipótesis que no es estrictamente necesaria para el modelo, pero que es epistemológicamente relevante. La pregunta que guía el punto 5 es la siguiente: Si suponemos dos enfrentamientos entre los mismos Pokémon, solo que en un caso los Pokémon X e Y están identificado como principal y secundario respectivamente, mientras que en otro caso están identificados al revés, la predicción ¿Será igual en ambos casos?

Para evaluar esa hipótesis se construyo un segundo set de prueba, con todas las métricas invertidas.

Los resultados muestran que las predicciones son iguales en ambos casos.