Modelo de Clasificación para leads en Pokemon VGC

Diego Lira García - A01710369

Abstract

El trabajo aborda la necesidad de predecir cuáles serán los dos Pokémon iniciales que utilizará el rival, a partir de sus seis posibles Pokémon en un combate VGC. El objetivo es obtener más del 50% de aciertos por cada uno de los dos pokémones, triplicando la probabilidad de acierto respecto a escoger al azar, que es del 16.6%. Para ello se aplicaron y compararon varios modelos basados en redes neuronales fully connected, utilizando diferentes tipos de entrada: one-hot encoding, embedding, embedding manual construido a partir de las estadísticas de los Pokémon, y una combinación de embedding más estadísticas. Los resultados muestran que es posible obtener un acierto aproximado del 60% de las veces usando únicamente un embedding estándar. Concluimos que el modelo es útil para mejorar la probabilidad de predecir los Pokémon iniciales del rival al inicio de una batalla.

Glosario de Pokémon

- VGC: Video Game Championships. Es el formato oficial para batallas pokémon
- Lead: Pokémon con el cual inicia la partida el jugador o rival.
- Open Team Sheet: Formato en donde se puede ver la información de los Pokémon del equipo rival.
- Team preview: Fase del juego donde se elige que Pokémon vas a traer a la batalla.
- Regulación: Regla que limita ciertos Pokémon de usarse

Trabajos similares

Predicting competitive Pokémon VGC leads using Latent Semantic Analysis: a data-driven approach to team matchups *[4]*, por Bruno Luvizotto Carli, es el trabajo que dio inspiración a este proyecto. El trabajo de Luvizotto utiliza Latent Semantic Analysis

(LSA) para predecir los leads en partidas de Pokémon VGC, basándose en patrones implícitos entre los equipos de los jugadores.

El enfoque difiere un poco respecto a este trabajo, ya que en el proyecto se intenta aprender los leads de todos los Pokémon en todo el VGC, mientras que el trabajo de Luvizotto se centra únicamente en un nivel alto, usando como referencia solo los Pokémon que aparecieron en el torneo NAIC (North America International Championships).

Además, existen pequeñas diferencias metodológicas, como:

- Luvizotto emplea un enfoque no supervisado, basándose en la similitud semántica de equipos representados como texto.

- Este proyecto usa redes neuronales fully connected supervisadas, con entradas que incluyen one-hot encoding, embeddings y estadísticas de los Pokémon.

Se podría decir que este trabajo es usando aprendizaje supervisado y redes neuronales, mientras que el otro es no supervisado y usa LSA.

Introducción

El mundo del Pokémon competitivo está profundamente ligado a las matemáticas y al análisis de probabilidades, lo cual puede resultar abrumador para jugadores nuevos. Para quienes no juegan de manera competitiva, puede parecer trivial decidir con qué Pokémon iniciar una partida, pero en realidad esta elección es mucho más compleja e importante. Como lo establece Aaron Zheng:

“...a good team preview phase can give you an early advantage...” (Zheng, 2020) [1]

Para entender por qué no es tan sencillo como elegir tu Pokémon más fuerte, es necesario explicar las reglas básicas del formato oficial de Pokémon competitivo. Las partidas oficiales de Nintendo siguen ciertas reglas: el juego consiste en batallas de 2 contra 2, donde de los seis Pokémon que cada jugador lleva, solo se utilizan cuatro en la batalla. En otras palabras, no jugarás con todos tus Pokémon, por lo que es necesario decidir cuáles cuatro traer en cada batalla.

En el formato del juego se aplica el Open Team Sheet, lo que significa que ambos jugadores conocen la información del equipo rival. Es decir, cada jugador sabe cuáles son los seis Pokémon del oponente. Este conocimiento hace que elegir tus Pokémon sea más complicado, ya que no quieres traer un Pokémon que sea débil frente al equipo rival.

Teniendo toda la información de los Pokémon disponibles, es posible identificar patrones: al observar qué Pokémon trae el rival y cuáles traes tú, se puede aprender la probabilidad de que el oponente elija ciertos dos Pokémon como leads para iniciar la partida.

Dataset

El dataset utilizado en este proyecto proviene de los registros de partidas del servicio Pokémon Showdown, un simulador de batallas Pokémon de código abierto, donde los usuarios pueden publicar sus partidas. Un agradecimiento especial al usuario “beelzebruno” por publicar su cuaderno en Kaggle [2] y compartir el código utilizado para descargar las partidas de Pokémon Showdown, así como por proporcionar un script base para obtener la información necesaria para la construcción del dataset, al cual se le hicieron ciertas modificaciones faltantes para este proyecto.

Los registros de las partidas incluyen todas las acciones realizadas durante cada enfrentamiento, así como los Pokémon iniciales de ambos jugadores y sus leads. Mediante el uso de regex, se logró extraer un total de 4,935 partidas válidas. El dataset contiene las siguientes columnas:

- **j1_1:** Primer Pokémon del jugador 1
- **j1_2**: Segundo Pokémon del jugador 1
- **j1_3**: Tercer Pokémon del jugador 1
- **j1_4**: Cuarto Pokémon del jugador 1
- **j1_5**: Quinto Pokémon del jugador 1
- **j1_6**: Sexto Pokémon del jugador 1
- **j2_1**: Primer Pokémon del jugador 2
- **j2_2**: Segundo Pokémon del jugador 2
- **j2_3**: Tercer Pokémon del jugador 2
- **j2_4**: Cuarto Pokémon del jugador 2
- **j2_5**: Quinto Pokémon del jugador 2
- **j2_6**: Sexto Pokémon del jugador 2
- **j1_l1**: Lead 1 del jugador 1
- **j1_l2**: Lead 2 del jugador 1
- **j2_l1**: Lead 1 del jugador 2
- **j2_l2**: Lead 2 del jugador 2

Primera Iteración: Modelo (One hot - encoding)

Se entrenó un modelo de red neuronal fully connected con dos salidas softmax para predecir los dos Pokémon iniciales (leads) de un jugador en partidas de Pokémon VGC.

La arquitectura consistió en una capa de entrada cuyo tamaño correspondía al número total de características del dataset después de aplicar one-hot encoding. Para transformar las columnas categóricas en formato one-hot, se utilizó la función get_dummies() de la librería pandas.

Este proceso generó una nueva columna por cada valor único encontrado. Por ejemplo, si la columna j1_1 contenía los Pokémon Pikachu, Charizard y Gengar, entonces se crearon las columnas:

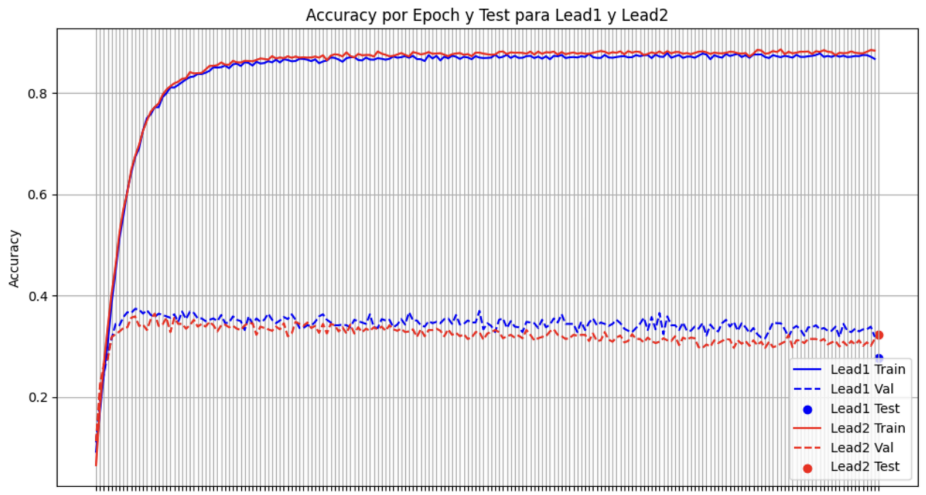
- j1_1_Pikachu
- j1_1_Charizard
- j1_1_Gengar

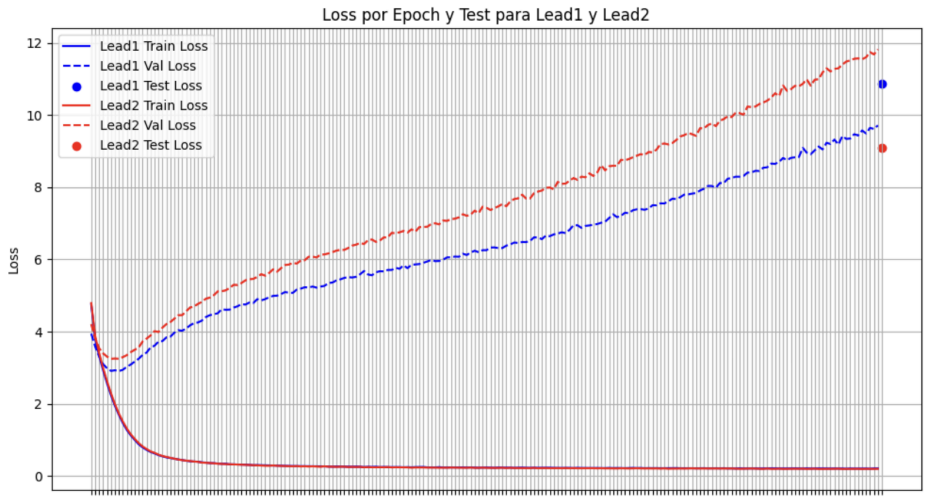
En cada fila, el valor correspondiente a la columna del Pokémon presente era igual a 1, mientras que el resto se mantenían en 0. Debido a la gran cantidad de Pokémon, el número total de características fue de 2690 columnas.

La red neuronal utilizó estas columnas como entrada, seguida de dos capas densas de 64 neuronas con activación ReLU. Finalmente, el modelo tenía dos salidas softmax, cada una con tamaño igual al vocabulario de Pokémon, correspondientes a la predicción del primer y segundo lead.

La evaluación se realizó considerando que no importa si un Pokémon está en la posición de lead 1 o lead 2, es decir, se calculó un accuracy desordenado que mide si los Pokémon predichos coinciden con los reales sin importar el orden.

Los resultados obtenidos fueron los siguientes:





Resultados del accuracy sin importar el orden

Conjunto de datos	Lead 1	Lead 2
Entrenamiento	90.68%	91.17%
Validación	32.03%	31.62%
Test	28.07%	32.66%

El alto accuracy en entrenamiento indica que la red aprendió patrones presentes en los datos de entrenamiento. Pero la baja accuracy en validación y test es sobreajuste, es decir, el modelo memoriza los datos de entrenamiento y no generaliza bien a ejemplos nuevos.

Segunda Iteración: Modelo Embedding

Este modelo es igual al anterior en arquitectura, excepto que, en lugar de usar one-hot encoding, utiliza un embedding de dimensión 64. La idea de pasar de one-hot encoding a embeddings es facilitar que el modelo generalice mejor entre Pokémon, ya que cada

Pokémon deja de ser tratado como un nombre discreto y se representa como un vector de características aprendidas.

En este sentido, funciona de manera similar a un problema de procesamiento de lenguaje natural (NLP), donde los embeddings se utilizan para capturar relaciones y similitudes entre palabras. Sin embargo, en este caso los embeddings aprenden patrones útiles para la predicción de leads, pero no reflejan necesariamente similitudes semánticas entre los Pokémon.

En este modelo no se utilizó Skip-gram, ya que el orden en el que aparecen los Pokémon no tiene relación con sus similitudes. Por ejemplo, si un equipo incluye a Pikachu y Whimsicott juntos, esto no implica que sean similares; de hecho, estos dos Pokémon se juegan de manera diferente y cumplen roles distintos dentro del equipo.

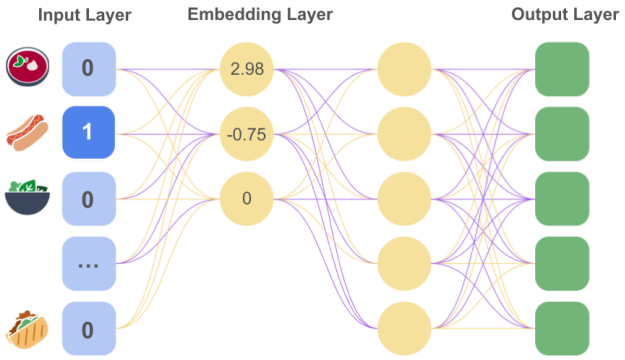
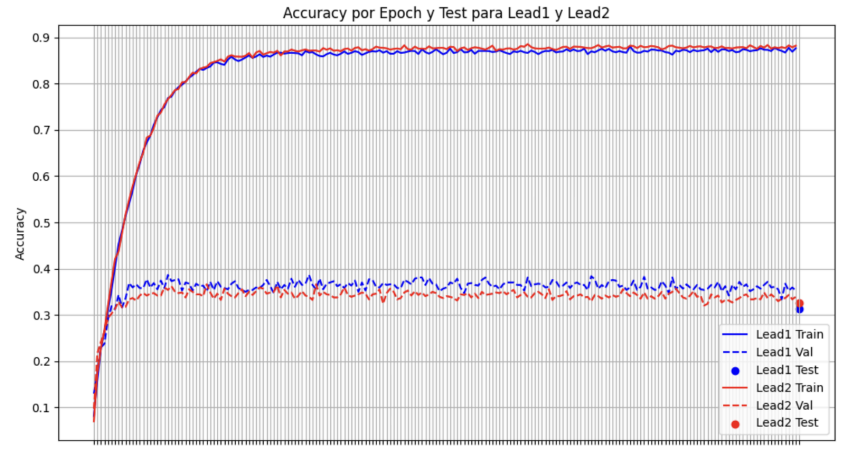
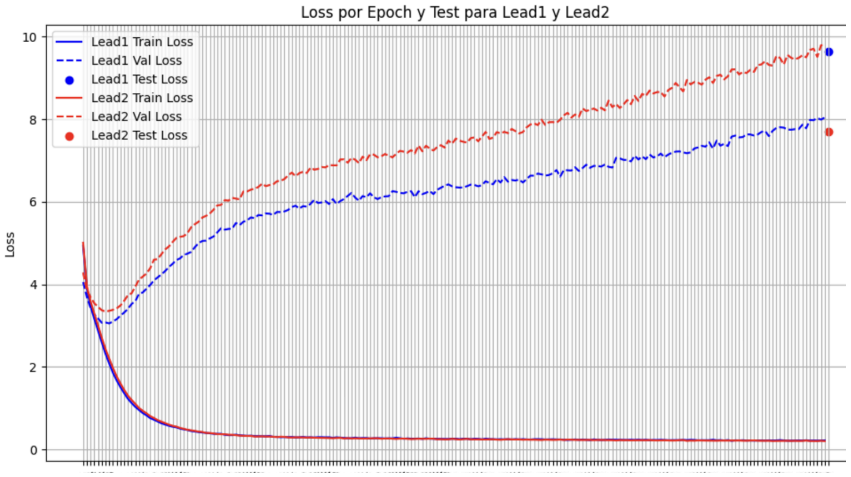


Figura 1: One hot traducido a un vector por una capa Embedding [3]

Para evaluar el desempeño del modelo con embeddings se calcularon dos tipos de accuracy: Donde no importa el orden, si Pokémon predicho para lead1 es el valor correcto de la lead1 o la lead2 se da por buena. El otro tampoco importa el orden, pero después de predecir selecciona únicamente el Pokémon con la posibilidad más alta que sea un Pokémon que tiene el rival.





Resultados del accuracy sin importar el orden

Conjunto	Lead 1	Lead 2
Train	91.28%	92.70%
Validation	43.78%	42.84%
Test	38.73%	39.81%

Al comparar los resultados del modelo con embeddings contra los obtenidos con el modelo basado en one-hot encoding, se observa una mejora tanto en el conjunto de validación como en el de prueba.

- En validación, el accuracy del lead 1 pasó de 32.03% (one-hot) a 43.78% (embedding).
- En test, el accuracy del lead 1 pasó de 28.07% (one-hot) a 38.73% (embedding).

Esto indica que el uso de embeddings permitió al modelo generalizar mejor a equipos que no vio durante el entrenamiento. Sin embargo, a pesar de esta mejora, el modelo aún muestra signos evidentes de sobreajuste.

Resultados del accuracy con solo Pokémon válidos

Conjunto	Lead 1	Lead 2
Train	91.20	92.18
Validation	48.78	48.78
Test	44.13	47.50

Como era de esperarse, al limitar las predicciones únicamente a los Pokémon que están presentes en el equipo del rival, el accuracy aumenta aproximadamente un 5% tanto en el conjunto de validación como en el de prueba.

Tercera Iteración: Aumentar el dataset

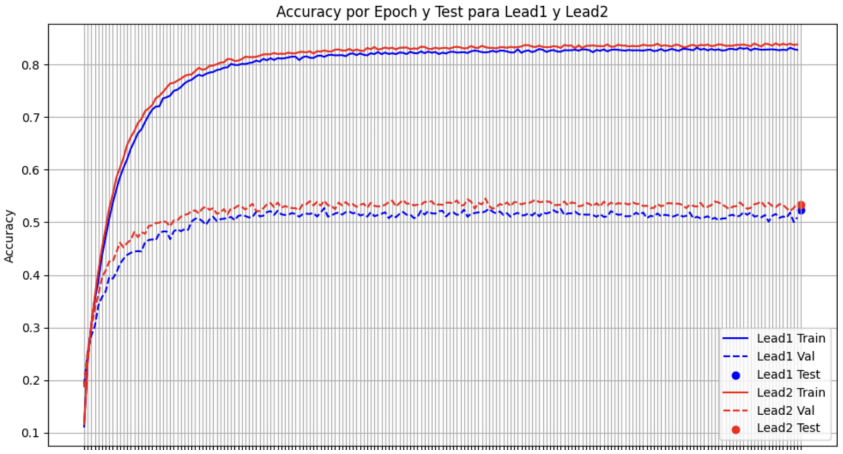
Al pasar de one-hot encoding a embeddings, el modelo mejoró su capacidad de abstracción, sin embargo, existe un límite en cuánto puede mejorar únicamente mediante un aumento del tamaño del vector o añadiendo más capas. Incrementar la dimensionalidad del embedding o la profundidad de la red no garantiza un mejor rendimiento, especialmente cuando el problema principal radica en la calidad y diversidad del dataset. Por esta razón, la siguiente mejora propuesta fue precisamente ampliar y refinar el conjunto de datos.

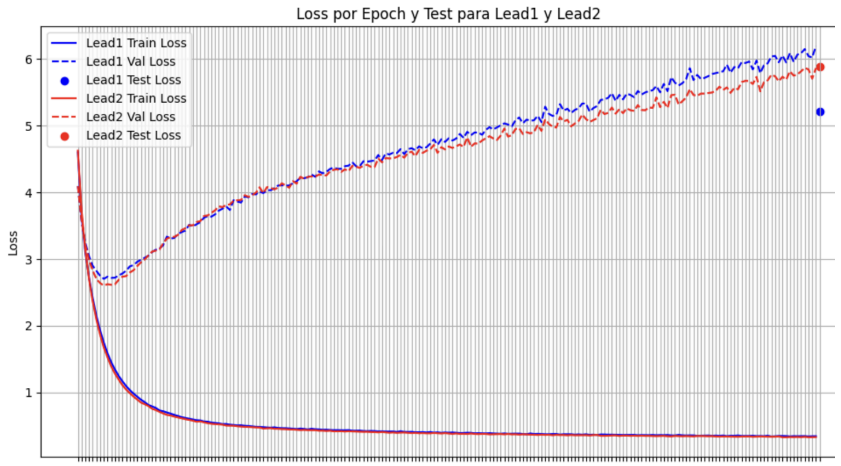
El dataset original proviene de los registros de partidas de Pokémon Showdown. Desafortunadamente, existe un límite en la cantidad de partidas accesibles. Se pueden consultar únicamente las últimas 100 páginas por formato, y en cada página se encuentran los IDs de las partidas disponibles. En este proyecto se descargaron estas 100 páginas y, mediante regex, se extrajeron los IDs de las partidas. Luego, a partir de esos IDs, se hizo una petición al servidor para obtener el registro completo de cada partida para obtener la información necesaria para construir el dataset.

Sin embargo, más allá de esa cantidad no es posible acceder a partidas adicionales dentro de la misma regulación. Esto nos lleva al concepto de "regulación" dentro de Pokémon VGC. Una regulación define qué Pokémon están permitidos y cuáles están prohibidos durante un periodo de tiempo específico, aunque las reglas del formato permanecen iguales. El dataset utilizado originalmente corresponde a la Regulación I, pero es posible ampliar los datos incorporando partidas de las regulaciones F y G.

Es importante tener en cuenta que las regulaciones F y G incluyen Pokémon que no aparecían en la regulación I, lo cual puede generar dificultades para el modelo al no haber suficientes de ellos. No obstante, también incluyen Pokémon que sí estaban permitidos desde la Regulación I, pero que tenían baja representación en ese dataset original. Incluir estas regulaciones adicionales puede ayudar al modelo a generalizar mejor, ya que aumenta la variedad.

La arquitectura para esta iteración se mantuvo igual la anterior iteración.





Resultados del accuracy sin importar el orden

Conjunto	Lead 1	Lead 2
Train	88.7%	89.3%
Validation	55.9%	58.9%
Test	57.6%	59.5%

Resultados del accuracy con solo Pokémon válidos

Conjunto	Lead 1	Lead 2
Train	88.6%	88.5%
Validation	59.3%	61.9%
Test	61.3%	62.8%

Como era de esperarse, tanto en validación como en test la accuracy aumentó hasta aproximadamente un 60%, cumpliendo el objetivo de triplicar la probabilidad de predecir correctamente el lead del rival en comparación con elegirlo de forma aleatoria.

Claro que el modelo aún puede mejorar si se contara con una mayor cantidad de datos o, en su defecto, si se filtraran los Pokémon con baja frecuencia de aparición en el dataset, tal como lo hizo Luvizotto en su trabajo.

Sin embargo, el propósito de este modelo es distinto: se busca que funcione para todo tipo de niveles en VGC, no únicamente para entornos competitivos de alto rendimiento o torneos profesionales.

Desarrollo Experimental del Modelo

En esta sección se muestras resultados de experimeinots realizados y sus resultados.

Vector manual con estadísticas del Pokémon

En este experimento se reemplazó el embedding de dimensión 64 por un vector manual construido a partir de las seis estadísticas base de cada Pokémon: Salud, Ataque, Defensa, Ataque Especial, Defensa Especial y Velocidad.
La hipótesis era evaluar si un vector más pequeño pero con significado explícito podía capturar mejor la información relevante para predecir los leads.

Arquitectura

Debido a que el vector de entrada se redujo de 64 dimensiones a solo 6, se aumentó la capacidad del modelo para permitir una mejor generalización.
La arquitectura utilizada fue:

Dos capas densas de 128 neuronas

Seguidas por dos capas densas de 64 neuronas

Dos salidas softmax (lead1 y lead2)

Resultados

- Accuracy train 1: 0.5394
- Accuracy train 2: 0.5599
- Accuracy val 1: 0.3508
- Accuracy val 2: 0.3639

Conclusión

El uso de un vector tan pequeño (solo 6 características) no permitió al modelo aprender los patrones presentes en el dataset.
A diferencia del embedding de 64, estas estadísticas no capturan relaciones más abstractas entre los Pokémon, por lo que el modelo carece de la capacidad expresiva necesaria.

Se concluye que se requiere un embedding de mayor dimensión para representar adecuadamente la complejidad del problema.

Modelo con estadísticas base + tipo + habilidad + ID

En este experimento se buscó incrementar la cantidad de características que describen a cada Pokémon.
Además de las seis estadísticas base, se integraron tres tipos de información adicional:

- Tipo del Pokémon
- Habilidad
- Un identificador único (ID) asignado manualmente

La hipótesis era que aumentar el número de features ayudaría al modelo a aprender relaciones más complejas relacionadas con composición de equipos, sinergias y patrones.

Arquitectura

Se usó la misma arquitectura que la anterior menos el tamaño de la entrada.

Capas densas:

- 128 neuronas ReLU
- 128 neuronas ReLU
- 64 neuronas ReLU
- 64 neuronas ReLU

Dos capas softmax, una por cada lead.

Esta arquitectura busca capturar patrones entre las diversas características de cada Pokémon y su rol dentro del equipo.

Resultados

- Accuracy train 1: 0.4506
- Accuracy val 1: 0.2933
- Accuracy train 2: 0.4597
- Accuracy val 2: 0.3050

Conclusión

A pesar de añadir más features, el rendimiento del modelo no mejoró, de hecho empeoró. Los resultados indican que incrementar la cantidad de información no es suficiente si esta no es la información correcta o más determinante para predecir leads en VGC.

Se podría incluir features más relevantes desde un punto de vista competitivo.

Por ejemplo:

- Movimientos comunes en ciertas estrategias.
- Sinergias específicas (por ejemplo, Pokémon lentos con compañeros que usan Trick Room).
- Roles del Pokémon (support, sweeper, redirection).
- Items característicos (Eviolite, Assault Vest, Sitrus Berry, etc.).

Este experimento muestra que no basta con agregar datos, sino que se requieren features con verdadero valor competitivo.

El efecto del dataset y las features

Como estoy trabajando con más de 500 clases, hacer una matriz de confusión completa sería prácticamente imposible. Eliminar clases que no aporten no es una opción, porque va en contra del objetivo del modelo. Y aunque sería ideal diseñar una función de pérdida personalizada que obligue al modelo a elegir únicamente entre los Pokémon del equipo rival, esto está fuera del alcance por ahora.

Por eso decidí analizar cómo el dataset influye en las predicciones y observar los valores más frecuentes. Se imprimieron los 10 Pokémon más comunes en el dataset y luego los 10 Pokémon más predichos por el modelo.

	pokemon	id	count
0	incineroar	273	11079
1	urshifurapidstrike	638	9404
2	fluttermane	183	7081
3	rillaboom	485	5832
4	amoonguss	6	5397
5	miraidon	377	5053
6	ragingbolt	466	4891
7	calyrexshadow	61	4766
8	tornadus	615	4409
9	farigiraf	169	4291

	pokemon	id	predicted_count
0	incineroar	273	323
1	miraidon	377	209
2	tornadus	615	195
3	fluttermane	183	161
4	calyrexshadow	61	159
5	indeedeef	275	137
6	whimsicott	669	136
7	urshifurapidstrike	638	127
8	grimmsnarl	234	120
9	lunala	330	119

Podemos observar que 6 de los 10 Pokémon más populares también aparecen dentro de las predicciones más frecuentes. A primera vista podría parecer que el modelo simplemente elige lo más popular, pero si revisamos Pokémon como Whimsicott, Grimmsnarl e

Indeedee-F, notamos que no son especialmente populares en frecuencia general del dataset, pero sí son muy comunes como leads.

Whimsicott y Grimmsnarl tienen la habilidad Prankster, que da prioridad a los movimientos de estatus, y por eso suelen usarse para abrir partida con Tailwind, que duplica la velocidad del equipo por 4 turnos.

Por otro lado, Indeedee-F se utiliza muchísimo por Follow Me, un movimiento que redirige los ataques hacia ella, permitiendo que su compañero utilice un turno para potenciarse sin sufrir daño. Estos resultados fue la razón de agregar más datasets e intentar usar ciertas características de los Pokémon

Conclusiones

Es posible construir un modelo que prediga con más del 50% de precisión los leads iniciales de un combate. Al final el mejor modelo fue el de la segunda iteración, cuando es entrando con un mayor dataset. Sin embargo, el modelo actual aún presenta varias oportunidades de mejora y limitaciones importantes.

La principal mejora pendiente es la necesidad de contar con una mayor variedad de partidas. En los niveles altos del juego, la mayoría de los enfrentamientos utilizan los mismos Pokémon, por lo que el modelo aprende patrones muy repetitivos. Esto provoca que tenga dificultades frente a Pokémon poco populares.

Una forma de mitigar este problema sería representar a los Pokémon con un sistema más complejo que simplemente usar su nombre o los features probados en este trabajo. Añadir estadísticas base, tipo o habilidad no fue suficiente, y agregar más características requiere un entendimiento más profundo del juego competitivo para evitar introducir ruido o features irrelevantes. Idealmente, el modelo debería ser capaz de detectar similitudes entre Pokémon y generalizar, de modo que cuando aparezca un Pokémon poco común, pueda compararlo internamente con otros similares en lugar de tratarlo como un caso totalmente nuevo.

Con esta perspectiva, el problema deja de ser la falta de variedad de Pokémon en el dataset, y pasa a ser la falta de variedad de estilos de juego.

Otra mejora que no se implementó, pero sería valiosa, es una función de pérdida personalizada que obligue al modelo a escoger únicamente entre los Pokémon del equipo rival. Esto reduciría considerablemente las predicciones inválidas y alinearía mejor el comportamiento del modelo con la lógica real del juego.

Referencias

[1] Team Preview — VGC guide. (n.d.). VGC Guide. https://www.vgcguide.com/team-preview

[2] Brunolcarli. (2025, June 25). *Pokemon_VGC_leads_prediction_with_LSA*. https://www.kaggle.com/code/brunolcarli/pokemon-vgc-leads-prediction-with-lsa

[3] *Cómo obtener embeddings*. (n.d.). Google for Developers. https://developers.google.com/machine-learning/crash-course/embeddings/obtaining-embeddings?hl=es-419

[4] *Predicting competitive Pokémon VGC leads using Latent Semantic Analysis: a data-driven approach to team matchups*. (2025, July 12). Journal of Geek Studies. https://jgeekstudies.org/2025/07/11/predicting-competitive-pokemon-vgc-leads-using-latent-semantic-analysis-a-data-driven-approach-to-team-matchups/?utm_source=chatgpt.com