Magic: The Gathering - Clasificación de Rareza de Cartas

Francisco Rodriguez - 22001828

Douglas Pérez - 22003865

Isabel Paíz - 22001411

Introducción

Este proyecto aborda un problema de clasificación en el ámbito de la inteligencia artificial aplicado a datos del juego de cartas coleccionables *Magic: The Gathering*. El objetivo es predecir la rareza (*rarity*) de una carta a partir de diversas estadísticas de juego. La rareza es un atributo fundamental que determina el valor e impacto de una carta y se clasifica en: Mythic (Mítica), Rare (Rara), Uncommon (Poco común) y Common (Común). Este análisis tiene aplicaciones prácticas en contextos como el draft, donde la selección estratégica es esencial.

Descripción del Dataset

Utilizamos un dataset de la expansión *Outlaws of Thunder Junction*, el cual contiene atributos, las cuales se destacan:

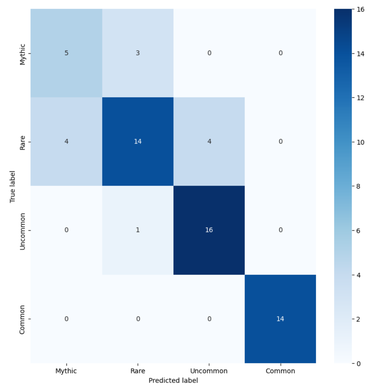
- #Seen: veces que la carta fue vista.
- #Picked: veces que fue seleccionada.
- GP WR: porcentaje de victorias cuando se jugó.
- ALSA: posición promedio de selección,
- #OH: Número de juegos dentro de la mano abierta.
- #GD : Veces que la carta fue robada del mazo hacia la mano.
- #GIH : Games in Hand
- #GNS : Número de partidas no vistas (Game Not Seen)
- Etc.

Además incluye información sobre colores, transformada vía *One Hot Encoding*, y la variable objetivo *Rarity*, que se busca codificarla ordinalmente (M=0, R=1, U=2, C=3). Existen valores nulos, categóricos y númericos en dicho dataset.

Metodología - Arquitectura 1

Para esta primera arquitectura vamos a aplicar algunos métodos de Future Engineering, luego se irán añadiendo más en las otras arquitecturas. Las métricas de este fueron:

- **Preprocesamiento:** Para este modelo se eliminaron las filas que contuvieran valores nulos. El dataset contenía varios y decidimos eliminarlos usando **dropna.** Luego hicimos un mapeo con algunos campos que tenían valores categóricos representados en portentajes. Se usó una función que lo convirtiera a su representación decimal. Luego se aplicó One-Hot-Encoding para el color porque no existía relación de orden, pero en Rarity si, por lo que se aplicó Ordinal Encoding a este feature.
- **Normalización Min-Max:** Se aplicó a variables como #Seen, #Picked, #GP, #OH, #GD, #GIH y #GNS.
- **Modelo:** Consta de una neuronal con 3 capas densas de 32 neuronas y activación *Leaky ReLU*.
- **Entrenamiento:** Para esta primera arquitectura se usó únicamente EarlyStopping como callback para evitar sobreajuste.

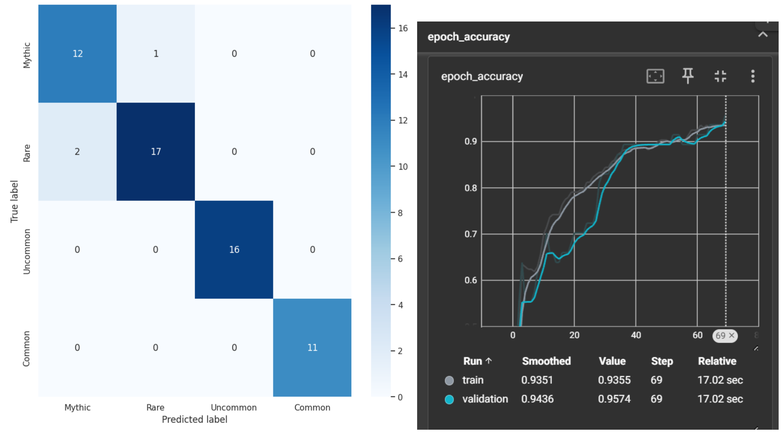


Metodología - Arquitectura 2

Luego de evaluar el modelo, llegamos a la conclusión de que esta es la mejor arquitectura porque presenta resultados más solidos. Para este se aplicaron las siguientes métricas:

- Eliminación de valores nulos: Se quitaron las filas que contuvieran valores nulos
- Mapeo de porcentajes: Los valores categoricos se mapearon a su representación en decimal (entre 0 y 1)
- Ordinal Encoding y One-Hot Encoding: Se aplicó One-Hot Encoding al campo de color porque no había relación de orden entre ellas. Se aplicó Ordinal Encoding para la rareza porque esta si tenía relación de Orden.
- Z-Score Normalization: Centrado en media y desviación estándar. Se aplicaron a los features que tuvieran valores numéricos y tuvieran un rango alto de valores.
- Balance de clases: aplicación de *undersampling* y *oversampling* para equilibrar las clases. Se usó como referencia la clase 1 (que es la clasificación de 'Rare').
- Eliminación de columnas: Según matriz de correlación habían campos que tenian muy poca dependencia, como Name, %GP, GP WR, OH WR, GNS WR y Color
- Modelo: Una capa de entrada y 3 capas densas (32 unidades), activación *tanh*, y capa final *softmax*. 100 epochs.
- Callbacks: EarlyStopping, ModelCheckpoint y TensorBoard.

Entrenamiento: 93.5%
Evaluación: 94.92%

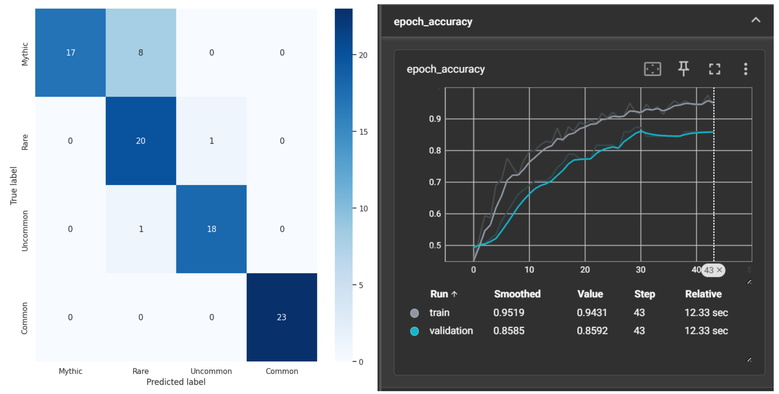


Metodología - Arquitectura 3

- **Preprocesamiento:** Eliminación de valores nulos, transformación de porcentajes a decimale como las otras arquitecturas, y codificación categórica.
- **Balance de clases:** Se aplicó ordinal encoding a la variable "Rarity" y one-hot encoding a la variable "Color".
- **Normalización:** Se utilizó una herramienta nueva llamada RobustScaler, la cual ofrece una resistencia a outliers. Esta herramienta Utiliza la mediana y el rango intercualítico de cada característica para escalar los datos.
- **Reducción de dimensionalidad:** Se descartaron columnas con baja correlación con la variable objetivo, incluida toda la información de color.

Continuación Arquitectura 3

- **Modelo:** Red neuronal con dos capas densas (64 neuronas), activación ReLU, BatchNormalization y Dropout. Dropout es una técnica de regularización que sirve para prevenir el sobreajuste
- **Callbacks:** EarlyStopping, TensorBoard, y ModelCheckpoint para guardar el mejor modelo. TensorFlow tiene un método de Dropout con un rate de 0.3 (30%) para probar elimina el overfitting.



Resultados

A continuación se presentan los resultados de forma resumida de las tres arquitecturas.

Arquitectura	Training	Validation	Epochs
Arq. 1	81.66%	80.33%	75/75
Arq. 2	93.5%	94.92%	70/100
Arq. 3	94%	85%	44/50

Conclusiones

Se realizaron algunas pruebas dentro de las tres arquitecturas para evaluar cual de ellas tiene el mejor rendimiento. La primera arquitectura tanto su entrenamiento y su validación no son tan altos, por lo que este es el menos eficiente. La segunda en la mayoría de las pruebas ofrecía una cantidad casi balanceada entre el entrenamiento y la validación. Habían casos en el que el porcentaje de entrenamiento era mayor, pero por una baja cantidad. La tercera arquitectura en la mayoría de pruebas presentaba overfitting, a pesar de haber aplicado Dropout, por lo que el entrenamiento estaba sólido, pero la validación en algunas pruebas no presentaba el mismo porcentaje del entrenamiento. El análisis comparativo demuestra que el preprocesamiento cuidadoso, junto a la elección adecuada de normalización, balanceo y funciones de activación, influye significativamente en el rendimiento del modelo. La arquitectura 2 fue la más efectiva gracias a su enfoque en balance de clases y uso de técnicas de monitoreo.

Futuras Mejoras

- Explorar modelos como Gradient Boosting o redes profundas con embeddings categóricos.
- Aplicar validación cruzada para obtener resultados más robustos.
- Aumentar el dataset incluyendo otras expansiones de Magic.
- Incorporar interpretabilidad con SHAP o LIME para comprender la relevancia de cada atributo.