Predictor de Precios de cartas de Magic the Gathering

by Itsazain M. Bilbao

Concepto

Magic: The Gathering es un juego de cartas coleccionables (TCG) en el que las cartas tienen un valor tanto como objeto coleccionable como elemento de juego. Este valor económico depende de diversos atributos de la carta, que pueden ser abstractos y varían con la demanda y el tiempo. El proyecto tiene como objetivo construir un modelo que prediga el precio de una carta, considerando sus atributos y utilizando procesamiento de lenguaje natural (NLP) para interpretar los efectos en texto plano de la carta.

Índice

Dataset

Los datos provienen de la API de Scryfall, una reconocida base de datos de *Magic: The Gathering*. El dataset original contiene hasta 86 columnas, pero en la primera etapa del proyecto se realizó una selección de atributos relevantes. Esta selección se basó en la revisión y análisis de las descripciones de cada columna, conservando sólo aquellas que aportan valor potencial para el objetivo del proyecto.

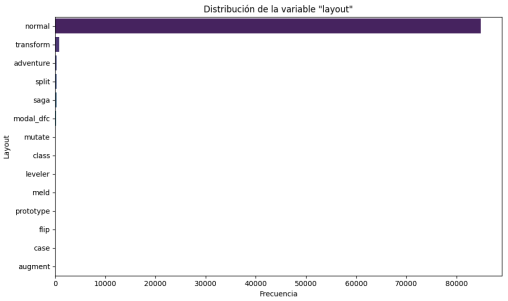
Atributo	Tipo	Descripción breve	Transformación	Atributo	Tipo	Descripción breve	Transformación
`name`	string	Nombre de la carta	Conservar como ID	`released_at`	fecha	Fecha de lanzamiento	Transformar a datetime/ordinal
`layout`	string	Tipo de diseño de carta	Evaluar codificación	`mana_cost`	string	Costo de maná con símbolos `{}`	Tal vez ignorar si hay `cmc`
`cmc`	float	Valor convertido del costo de maná	Rellenar nulos	`type_line`	string	Línea de tipo (e.g., “Creature — Elf”)	Procesar texto o codificar
`oracle_text`	string	Texto de reglas de la carta	PLN y tratamiento de nulos	`colors`	lista	Colores que tiene la carta	Dummies (junto con `color_identity`)
`color_identity`	lista	Identidad de color (todos los colores posibles)	Dummies junto con `colors`	`keywords`	lista	Habilidades clave (e.g., “Flying”)	Separar y codificar como dummies
`produced_mana`	lista	Tipos de maná que puede producir	Contar tipos o booleano	`legalities`	dict	Legalidad por formato	One-hot por formato
`reserved`	boolean	Si está en la Reserved List	Ignorar (poca varianza)	`game_changer`	boolean	Si cambió el juego significativamente	Ignorar (poca varianza)
`foil`	boolean	Disponible en foil	Conservar	`nonfoil`	boolean	Disponible en no-foil	Conservar
`promo`	boolean	Si es una carta promocional	Ignorar (poca varianza)	`reprint`	boolean	Si es una reimpresión	Conservar
`rarity`	string	Rareza de la carta (`common`, `rare`, etc.)	Label encoding manual	`textless`	boolean	Sin texto (puede ayudar con nulos en `oracle`)	Útil para imputar nulos
`booster`	boolean	Si aparece en sobres booster	Posible inclusión	`power`	string/float	Poder de criatura	Nulos a -1, imputar "X"/otros
`toughness`	string/float	Resistencia de criatura	Nulos a -1, imputar "X"/otros	`edhrec_rank`	int/null	Popularidad en Commander (menor es mejor)	Difícil imputación de nulos
`prices`	dict	Precios por moneda	Extraer `eur` como target (`log`)				

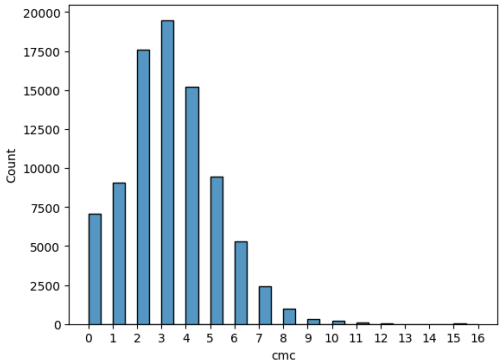
Después de escoger las columnas el nuevo dataframe se guarda en “dataCards.csv”.

Limpieza y Feature Engineering

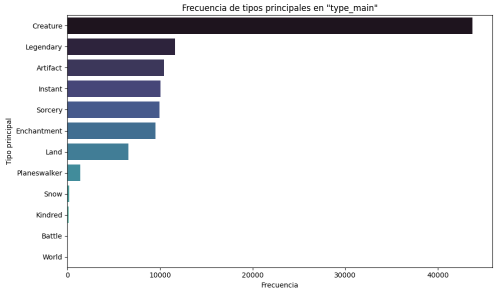
Se carga el archivo `dataCards.csv` y se revisa columna por columna para detectar valores nulos y definir estrategias de tratamiento. Las acciones principales han sido:

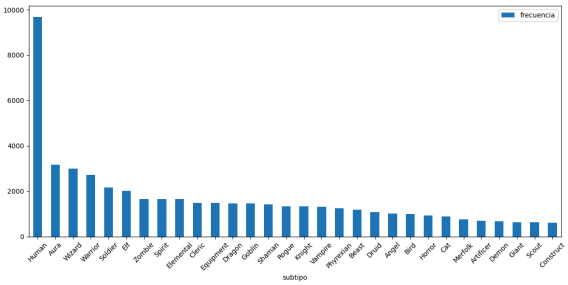
- **Eliminación de ruido**: Se descartan cartas irrelevantes para el análisis de precios, como *Tierras Básicas*, *Tokens*, y cartas exclusivas de coleccionismo que no se utilizan en juego real.
- **`released_at`**: Se transforma esta columna para conservar sólo el **año de publicación**, ya que el mes o día no aportan valor adicional. Esto también justifica no incluir la columna `set`.
- **`layout`**: Se utilizó inicialmente para filtrar cartas ilegales o promocionales. Al comprobar que la mayoría son de tipo "normal", se decide **eliminarla** por baja varianza.





- **`mana_cost` y `cmc`**: Dado que están correlacionadas, se **elimina `mana_cost`**, ya que transformarla implicaría una alta dimensionalidad. Se conserva `cmc`, eliminando valores extremos (cartas de broma), para mantener una distribución más realista.
- **`type_line`**: Contiene información relevante sobre el tipo de carta. Se identificaron los **tipos principales más comunes** y se aplicó **One-Hot Encoding** sobre ellos. Los tipos secundarios se descartaron por su alta variedad y baja frecuencia.





- **`colors` y `color_identity`**: Cómo están correlacionadas, se conserva solo `color_identity`, que no tiene nulos y es más fácil de transformar. Se aplica **One-Hot Encoding** para los cinco colores (`B`, `G`, `R`, `U`, `W`). Las cartas con todos los valores a cero se consideran **incoloras**, y aquellas con más de un color, **multicolores**. Esta transformación ayuda a reducir la dependencia directa entre color y `cmc`.
- **`legalities`**: Originalmente un diccionario con el estado legal de cada carta en distintos formatos. Se seleccionaron los **formatos principales** y se transformaron sus valores en una **codificación numérica ordinal**: `legal` → 1 `restricted` → 0 `not_legal` → -1 `banned` → -2

Esta representación busca aprovechar la ordinalidad para aportar valor al modelo.

- `keywords`: Se extraen dos variables:
    a. Una columna binaria que indica si la carta tiene **habilidades de evasión** (`1` si tiene, `0` si no).

    b. Una columna numérica que indica la **cantidad total de keywords** que posee la carta.
- `produced_mana`: Se transforma de lista a valor **numérico**, representando la cantidad de tipos distintos de maná que puede generar una carta. Si no genera maná, se asigna un **0**.
- `power` **y** `toughness`: Ambas se tratan en conjunto. Los valores `*` se imputan con la **moda** (valor más frecuente). En casos de valores vacíos o no numéricos, se reemplazan con un valor **por debajo del mínimo observado** para mantener orden ordinal.
- `rarity`: Se convierte de categórica a **ordinal numérica**, con la siguiente codificación: `common` = 0, `uncommon` = 1, `rare` = 2, `mythic` = 3, etc.
- `oracle_text`: Contiene el efecto de la carta en texto plano. Se **embebe usando BERT**, generando un vector de ~400 dimensiones, luego **reducido con PCA** para facilitar su uso en modelos.
- `edhrec_rank`: Rango de popularidad en el formato Commander. Se invierte el orden para que **mayor valor indique mayor popularidad**. Los nulos se imputan con el **mínimo valor** observado.
- `prices`: Originalmente un diccionario con múltiples precios según formato y divisa. Se extrae el **precio en euros** a la columna **final_price_eur**. En caso de nulo:

    1. Se toma el valor en **dólares** y se convierte en euros.

    2. Si no hay, se usa **precio foil en euros**, y luego foil en dólares si no existe el anterior..

    3. Si tampoco existe, se verifica si la carta está duplicada y se elimina.
    Si no está duplicada, se mueve a un **dataset aparte** para predicción futura.

Finalmente, el dataframe limpio se guarda como dataCardsClean.csv

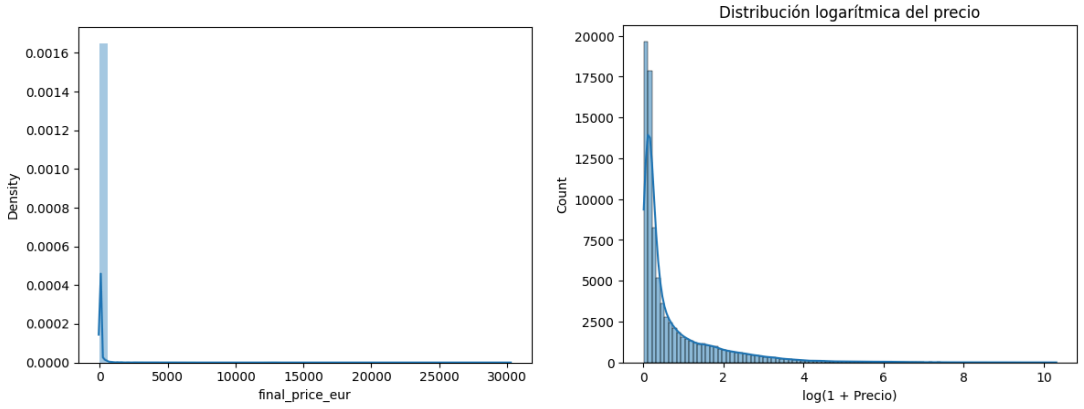
Análisis Exploratorio

Análisis del target: `precios_final_eur` y `log_price`

Con el dataset ya limpio, se analiza la variable objetivo `precios_final_eur`. Esta presenta una **distribución altamente sesgada a la derecha**, donde el 75% de las cartas cuestan menos de 1.5 €, pero existen valores extremos que superan los 30.000 €.

Para **reducir el impacto de estos outliers** y mejorar la distribución para los modelos, se aplica una **transformación logarítmica**. Aunque no normaliza completamente, **suaviza significativamente la cola** y concentra mejor los valores centrales.

Estadístico	Distribución Normal (`precios_final_eur`)	Distribución Logarítmica (`log_price`)
Media	13.20	0.75
Desviación estándar	248.82	1.06
Mínimo	0.01	0.00995
Percentil 25	0.12	0.1133
Mediana (P50)	0.31	0.2700
Percentil 75	1.56	0.94
Máximo	30,194.40	10.32



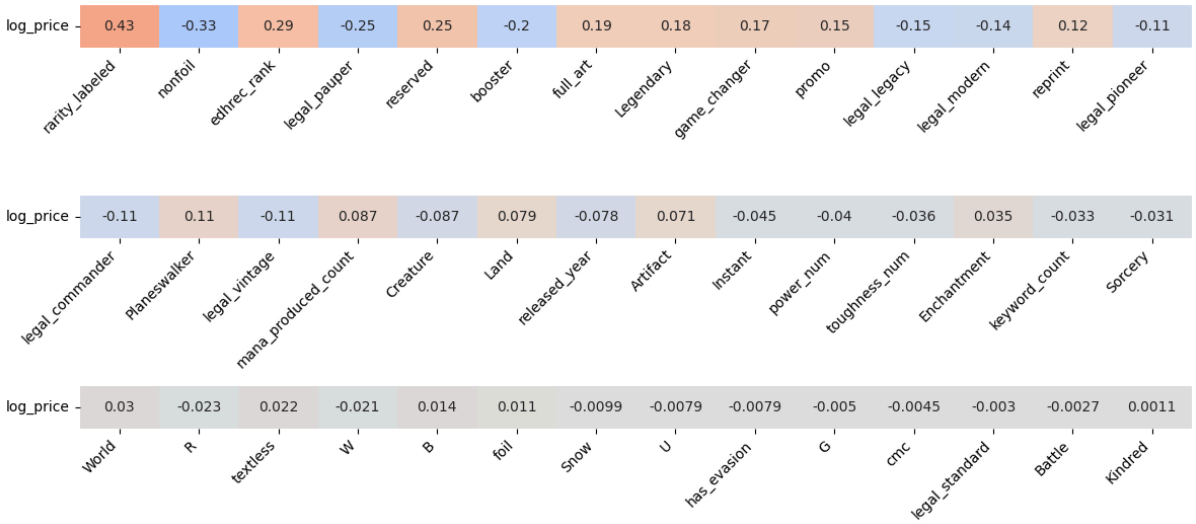
Correlación entre el target y las features

Se analiza la correlación de las variables con el target (`log_price`). Algunas variables presentan **correlaciones significativas**, lo que valida ciertas intuiciones del juego:

- **Correlación positiva**:
  - `rarity_labeled`: Las cartas más raras tienden a ser más caras.

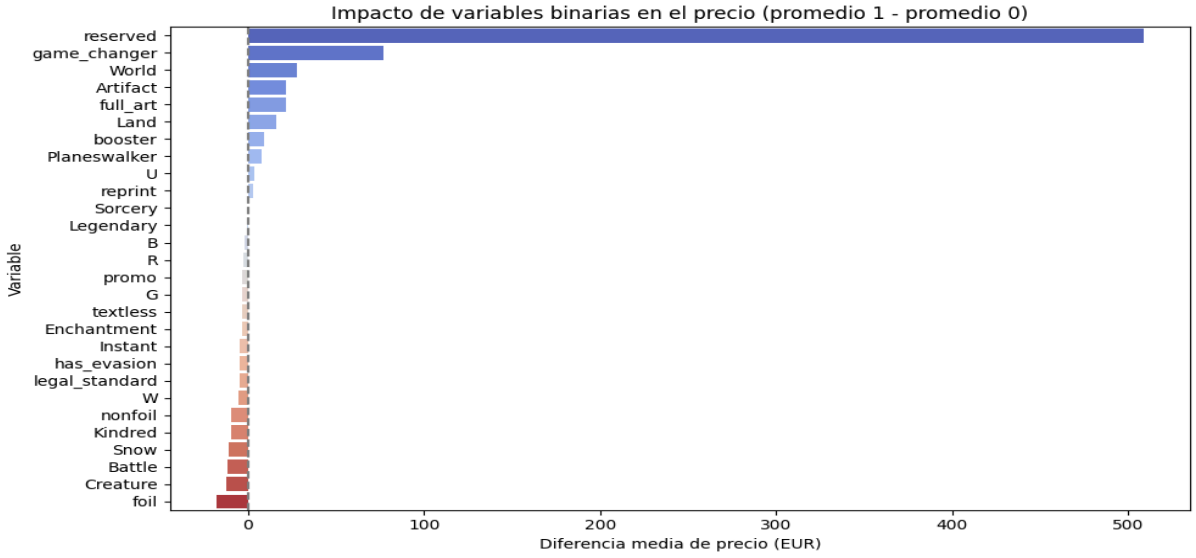
    - `edhrec_rank`: Las cartas más populares también tienden a tener precios más altos.
- **Correlación negativa**:
    - `nonfoil`: Las cartas que **no tienen versión foil** suelen tener menor valor.
    - `legal_pauper`: Las cartas legales en el formato **Pauper** (económico) tienden a ser más baratas.

Estas correlaciones iniciales sugieren que las variables capturan **tendencias reales del mercado** y pueden aportar valor predictivo al modelo.



- `reserved` y `game_changer` también muestran una **correlación positiva** con el target.
    - Las cartas **Reserved** no se han reimpreso en décadas, lo que **aumenta su escasez y valor**.
    - Las cartas marcadas como **game_changer** suelen tener un gran impacto en la partida, lo que **justifica su mayor precio**.

Estas variables refuerzan la relación entre **exclusividad o poder** y **valor económico**.

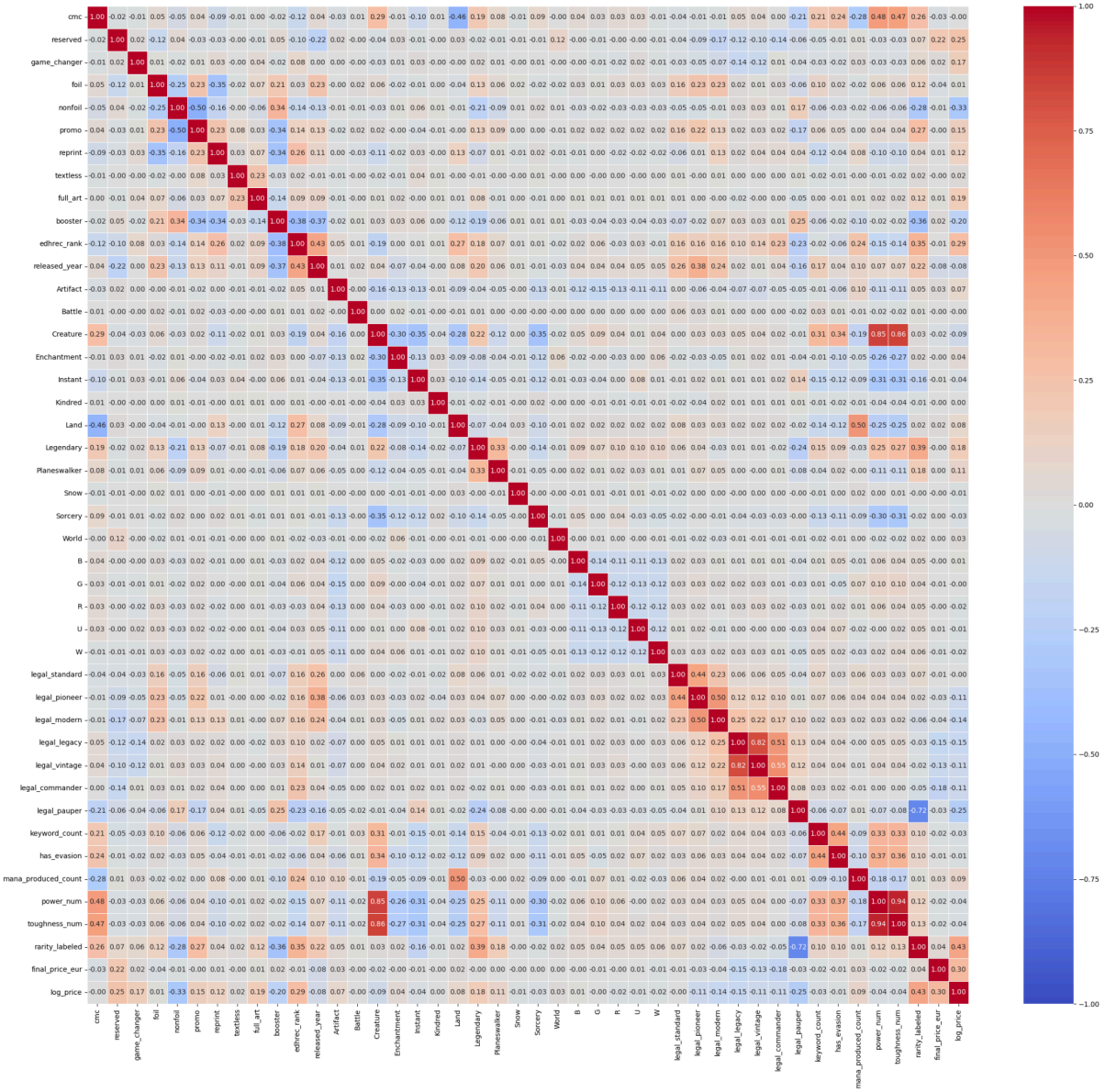


Matriz de correlación: colinealidades relevantes

El análisis de la matriz de correlación revela **colinealidades esperadas** entre ciertas variables:

- `Creature` ↔ `power` / `toughness`: Fuerte correlación positiva, ya que **sólo las criaturas** poseen estos atributos.
- `legal_pauper` ↔ `rarity_labeled`: Correlación **negativa alta**, ya que las cartas **menos raras** son las que **se permiten en Pauper**.
- `cmc` ↔ `power` / `toughness`: Relación positiva moderada, dado que **el coste de maná** suele crecer con **el poder y la resistencia** de la carta.

Estas relaciones ayudan a entender posibles **redundancias** o dependencias entre features, útiles para el diseño del modelo.



Diseño y producción del modelo

Estrategia general

- Se utiliza **`log_price` como target**, aprovechando su distribución suavizada.
- Se divide el dataset según la fecha de publicación:
  - **`df_past`**: Cartas publicadas **antes de 2025** (para entrenamiento y validación).
  - **`df_future`**: Cartas de **2025**, usadas como **segmento ciego** para predicción.
- Procesamiento
  - `df_past` se divide en:
    - `df_past_train`
    - `df_past_test`
  - A las variables numéricas se les aplica **MinMaxScaler** para normalizar rangos.
- Enfoque dual de modelado
  - Se entrena un modelo especializado en **cartas baratas** (`log_price < 4`) y otro en **cartas caras**.
  - Los modelos que se entrenan son diferentes y por pares hasta encontrar los que tengan mejores métricas.
- Evaluación
  - Ambos modelos se **evalúan en `df_past_train`, `df_past_test` y `df_future`**.
  - Las predicciones se **exponencian** para volver a euros.
  - La métrica de evaluación principal es el **MAE (Mean Absolute Error)** en su **unidad original (€)**.

Evaluación de los modelos

Experimento	Modelo	Grupo	MAE Train	MAE Test	MAE 2025
Fresno I	RandomForest	cheap	0.59	1.21	2.15
	RandomForest	expensive	207.33	355.84	130.18
	XGBoost	cheap	0.97	1.18	2.10
	XGBoost	expensive	93.47	344.74	144.97
	GradientBoosting	cheap	1.57	1.55	2.25
	GradientBoosting	expensive	313.74	376.67	123.60
Fresno II	RandomForest	cheap	0.59	1.21	2.05
	RandomForest	expensive	186.77	318.85	145.31
	XGBoost	cheap	1.12	1.24	2.13
	XGBoost	expensive	149.44	287.43	182.42

Almendro IV	RandomForest	cheap	0.65	1.35	2.19
	RandomForest	expensive	243.65	332.97	172.66
	XGBoost	cheap	1.27	1.36	2.17
	XGBoost	expensive	216.38	319.49	228.70
Castaño III	RandomForest	cheap	0.23	1.20	2.04
	RandomForest	expensive	0.61	320.75	138.36
	XGBoost	cheap	0.24	1.18	2.15
	XGBoost	expensive	0.60	288.21	175.78
Exedra III	XGBoost	cheap	0.24	1.17	2.20
	XGBoost	expensive	0.68	328.14	133.10

Producción del Multimodelo final

A partir de los **hiperparámetros óptimos de Exedra III**, se construyó el modelo definitivo empleando una estrategia de **multimodelo segmentado por rangos de log_price**. Se definieron tres rangos:

- **Cheap**: log_price < 2
- **Mid**: 2 ≤ log_price < 4
- **Expensive**: log_price ≥ 4

Aunque los análisis previos sugerían cortes más bajos (1 y 4), los mejores resultados se obtuvieron con cortes en **2 y 4**, en términos de acierto por segmento.

Segmento	Rango Log-Price	Muestras con mejor predicción	Porcentaje del total
Cheap	< 2	14,316	86.9%
Mid	2 – 4	1,799	10.9%
Expensive	≥ 4	353	2.1%

Implementación Técnica

- Se crea una clase propia llamada `MultiModelRegressor`, que gestiona internamente tres modelos distintos, según el segmento.
- Se encapsula esta clase en un `Pipeline` de scikit-learn.
- Se usan los **hiperparámetros de Exedra III**, previamente optimizados.
- Finalmente, el modelo se entrena y se exporta con `pickle` para su reutilización en producción.

Producto Final: Programa de Predicción Predictor2.0

Se desarrolló una aplicación que, dada la entrada del nombre de una carta (en inglés o español), consulta la base de datos y utiliza el modelo final entrenado para predecir su precio, comparándolo con el valor real.

Oportunidades y Mejoras Futuras

- Adaptar la aplicación para permitir la predicción de precios de cartas inventadas o no existentes en la base de datos.

- Desarrollar un modelo avanzado basado en técnicas de procesamiento de lenguaje natural (NLP) para extraer características funcionales de las cartas, con el fin de crear sistemas de recomendación o clasificación basados en la teoría del cuadrante.

- Incorporar variables adicionales relacionadas con la popularidad de las cartas en múltiples formatos de juego, más allá del formato EDH, para enriquecer el modelo predictivo.