# **Introducción**

En el presente informe se documenta el desarrollo de un proceso completo de **Descubrimiento de Conocimiento en Bases de Datos (KDD, por sus siglas en inglés)**, aplicado a un conjunto de datos relacionado atributos de autos y su precio sugerido o MSRP (Manufacturer's Suggested Retail Price). Este trabajo se enmarca en el estudio de técnicas de Minería de Datos y tiene como finalidad no solo aplicar algoritmos de aprendizaje automático, sino también comprender y reflexionar sobre cada una de las fases que conforman el proceso KDD: desde la preparación de los datos hasta la interpretación de los resultados obtenidos.

En el trabajo se desarrolla un modelo de regresión para predecir el precio sugerido de vehículos (MSRP) a partir de sus características técnicas y de mercado, usando técnicas de preprocesamiento, visualización exploratoria, codificación de variables, y algoritmos de machine learning. La herramienta utilizada es Jupyter Notebooks en Python.

Selección de datos (Selection)

Se cargo un dataset del repositorio Kaggle: <https://www.kaggle.com/datasets/CooperUnion/cardataset> , el mismo trata sobre los atributos y el precio sugerido de los autos en el mercado de EEUU.

Se utilizó un archivo .csv con 3830 registros y 16 columnas iniciales.

En python se importo la librería pandas y se guardaron los datos en un dataframe: df = pd.read\_csv("carPrices.csv")

Variables disponibles: Make, Model, Year, Engine Fuel Type, Engine HP, Engine Cylinders, Transmission Type, Driven\_Wheels, Number of Doors, Market Category, Vehicle Size, Vehicle Style, highway MPG, city mpg, Popularity, MSRP.

**Explicación de cada variable**

Make : Marca del auto (por ejemplo: Ford, BMW, Toyota).

Model: Modelo específico del auto (por ejemplo: Focus, 3 Series).

Year: Año de fabricación del auto.

Engine Fuel Type: Tipo de combustible que usa el motor (nafta, diésel, híbrido, eléctrico, etc).

Engine HP: Caballos de fuerza del motor (más HP = más potencia).

Engine Cylinders: Cuántos cilindros tiene el motor (más = más potencia y consumo).

Transmission Type: Tipo de caja de cambios (manual, automática, CVT, etc).

Driven\_Wheels: Qué ruedas impulsan el auto:

Number of Doors: Número de puertas (2, 4, etc).

Market Category: Categoría de mercado: si es "lujo", "deportivo", "camioneta", etc.

Vehicle Size: Tamaño general del vehículo (pequeño, mediano, grande).

Vehicle Style: Estilo del auto: sedán, SUV, convertible, hatchback, etc.

highway MPG: Consumo en carretera (millas por galón — más alto = más eficiente).

city mpg: Consumo en ciudad (millas por galón).

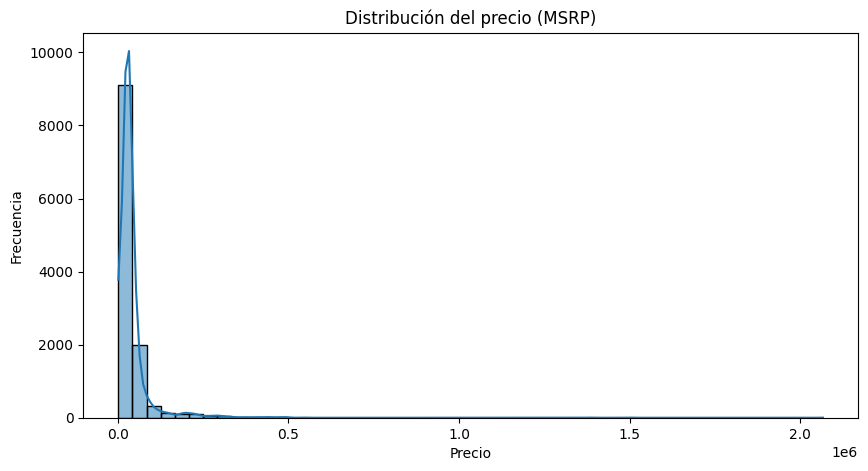
Popularity: Qué tan popular es ese auto (medido en cantidad de búsquedas o ventas).

MSRP: Precio sugerido de venta por el fabricante (lo que queremos predecir).

## Visualización Exploratoria

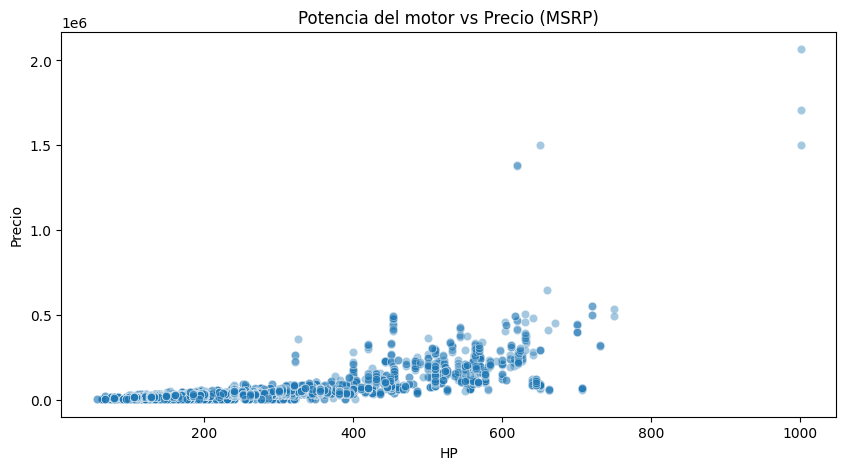
Se utilizaron gráficos para comprender la distribución y relación de variables clave con el precio, los mismos se construyeron con funciones de las librerías matplotlib y seaborn en Python.

Distribución de precios (MSRP): Para analizar a simple vista la distribución de los datos y si hay grandes outliers, podrían haber precios imposiblemente altos o .bajos, es decir, posibles errores en el dataset.



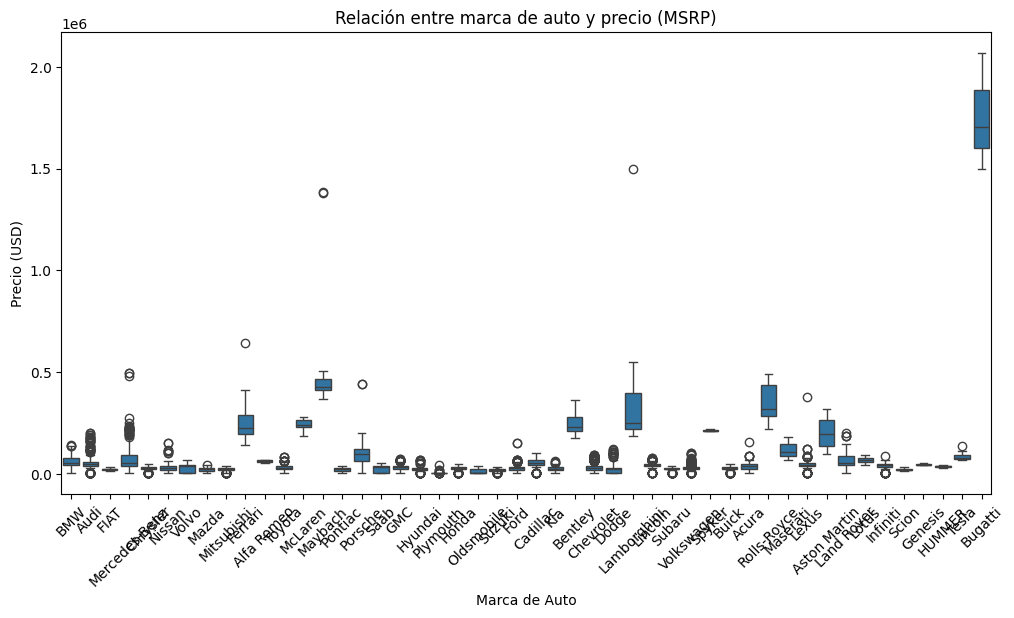
No se detectaron precios negativos y el precio máximo fue 2 x 10^ 6, es decir 2 millones de dólares, lo cual no es descabellado.

Relación entre potencia (Engine HP) y precio: para ver si a simple vista hay una correlacion fuerte entre ambos.



Deducimos que hay una correlación positiva entre ambos, pero esta es mas fuerte a partir de 400 hp en adelante.

Comparación de precios promedio por marca: Tambien se compararon los precios por marca, para ver si las marcas distinguen mucho el precio, esto es importante para analizar si incluir todas las marcas o no.



Se puede ver que una gran cantidad de marcas rondean precios similares mientras que marcas de autos de lujo como Bugatti y Lamborghini claramente tienen precios mucho mayores. De esto deducimos que la marca debe ser incluida, la discusión es sobre si usar todos los valores o dividir estos en grupos (Lujo, Caro, Mediano, Budget).

### **Limpieza de los datos**

Como primer paso para empezar la limpieza de datos se contó cuántos valores nulos de cada variable habia, estos fueron los resultados:

Market Category 3742

Engine HP 69

Engine Cylinders 30

Number of Doors 6

Engine Fuel Type 3

Todas las demás variables 0

Con esto se concluye que Market Category es por lejos el atributo con más nulos y que al ser casi todos nulos sus datos se tomó la decisión de eliminarlo del dataset de entrenamiento.

Con los demás atributos se usaron las siguientes técnicas para reemplazar los valores nulos:

**Para variables numéricas** se las reemplazó por la mediana de ese atributo.

Engine HP, Engine Cylinders, Number of Doors

**Para variables categóricas** se las reemplazó con la moda (valor más frecuente).

Engine Fuel Type

**Nueva Selección de Atributos**

En esta etapa se decidieron los atributos iniciales con los cuales probar el modelo, los atributos eliminados fueron Market Category.

Después se evaluó eliminar atributos con el criterio de evitar la Multicolinealidad, se dice que ocurre esto cuando una variable predictora puede ser expresada como una combinación lineal de otras. Esto puede llevar a problemas en la interpretación de los resultados del modelo, ya que es difícil determinar el impacto individual de cada variable predictora sobre la variable dependiente. Por otra parte, se buscó reducir la cantidad de variables para simplificar el modelo e intentar evitar el overfitting con variables de poco valor.

Se usaron funciones para averiguar la correlación entre variables que a simple vista sospechábamos como muy parecidas, por ejemplo:

| **city mpg** | **highway MPG** |  |
| --- | --- | --- |
| city mpg | 1.000000 | 0.886829 |
| highway MPG | 0.886829 | 1.000000 |

Correlación entre Engine HP y Engine Cylinders: 0.78

Agregando a lo anterior se eliminó el atributo Model ya que sobrecargar demasiado la cantidad de atributos y nos pareció que con la marca ya se podía distinguir los efectos más importantes. Por último se eliminó por intuición y comprobando en el gráfico que sus valores eran similares, el atributo Vehicle Size.

Finalmente se eliminaron las siguientes columnas:

df = df.drop(columns=[

"Model", "Market Category", "Engine Cylinders", "Vehicle Size", "city mpg",

])

### **Integración**

No fue necesario realizar integración de datos provenientes de múltiples fuentes. El análisis se llevó a cabo a partir de un único archivo de entrada, lo que evitó problemas de compatibilidad, duplicación o reconciliación de datos.

**Transformación de Variables**

La transformación de datos fue una etapa clave en el desarrollo del modelo predictivo, ya que permitió adaptar la información original del conjunto de datos a un formato que los algoritmos de machine learning pudieran interpretar de manera eficaz. Esta fase consistió en convertir, ajustar y codificar los datos con el objetivo de mejorar tanto el rendimiento como la interpretabilidad de los modelos.

### **1. Conversión de variables categóricas a variables numéricas**

Los algoritmos de aprendizaje automático, como los modelos de regresión o los árboles de decisión, no pueden trabajar directamente con variables categóricas en formato de texto. Por eso, una de las transformaciones más importantes fue la **codificación de estas variables**. En este caso, se optó por la técnica conocida como **One-Hot Encoding**, que consiste en crear una nueva columna binaria (0 o 1) para cada categoría presente en las variables categóricas.

Por ejemplo, la columna Make, que indica la marca del vehículo, incluía decenas de fabricantes diferentes (como Ford, Audi, Toyota, etc.). Aplicar One-Hot Encoding sobre esta variable generó una nueva columna para cada marca, permitiendo que el modelo identificara el impacto específico de cada fabricante en el precio final (MSRP).

Se aplicó esta codificación a variables como:

* **Marca del vehículo (Make)**
* **Tipo de tracción (Driven\_Wheels)**
* **Estilo del vehículo (Vehicle Style)**
* **Tipo de combustible agrupado (Fuel Group)**

Aunque este proceso aumentó considerablemente la cantidad de columnas (de 16 a más de 70), fue esencial para preservar la información de las categorías de forma comprensible para los algoritmos.

### **2. Agrupación y simplificación de variables**

En algunos casos, se consideró que las categorías de ciertas variables eran demasiado específicas o numerosas. Esto ocurría, por ejemplo, con las marcas de vehículos o los tipos de combustible. Para evitar una explosión de variables innecesarias y sobreajuste, se evaluó agrupar valores similares.

En particular:

* Se agruparon los tipos de combustible en categorías más generales como “Regular”, “Premium” y “Especial”, considerando que muchas variantes (como flex-fuel o combinaciones con gas natural) eran poco frecuentes y generaban ruido.
* Para la marca del vehículo (Make), si bien se exploró la posibilidad de agrupar fabricantes en grupos como “budget”, “premium” y “luxury” según el precio promedio, finalmente se decidió mantener todas las categorías debido a que el modelo de Random Forest tolera bien gran cantidad de variables, y estas mostraban tener un peso relevante. Más adelante en un nuevo intento, se volvió a correr los modelos con esta agrupación.

**Escalado y estandarización**

Para ciertos algoritmos (como la regresión lineal o Ridge), es necesario que las variables numéricas estén en la misma escala, ya que de lo contrario, los coeficientes estimados pueden verse distorsionados. Por ejemplo, variables como Engine HP (caballos de fuerza) o Popularity (popularidad del modelo) pueden tener escalas muy distintas y afectar el comportamiento del modelo.

Por esta razón, en los modelos lineales se aplicó un proceso de **escalado**, que consiste en transformar todas las variables numéricas para que tengan media cero y desviación estándar uno. Esta transformación no afecta al modelo de Random Forest, que es insensible a las escalas, pero sí es imprescindible en modelos lineales con regularización como Ridge.

**Minería De Datos**

Para la construcción del modelo de regresión, se utilizaron principalmente dos algoritmos: **Random Forest Regressor** y **Linear Regression**. Más adelante se probaron también modelos adicionales como **Ridge Regression** y **XGBoost Regressor**, K-Nearest Neighbours.

Se definió desde el comienzo que la evaluación de los modelos se realizaría mediante **validación cruzada (cross-validation) con 5 folds**. Esta técnica consiste en dividir el conjunto de datos en cinco partes iguales, entrenar el modelo con cuatro y validar con la quinta, repitiendo el proceso cinco veces alternando los conjuntos. Esto permite obtener métricas más robustas, reducir el riesgo de **overfitting** y mitigar el impacto de la distribución de los datos.

Los datos fueron **aleatorizados** previamente para evitar que existan patrones de orden que puedan sesgar la validación. Todos los algoritmos fueron evaluados bajo esta misma estructura.

### **Métricas utilizadas**

Las métricas seleccionadas para evaluar los modelos fueron:

* **R² (Coeficiente de Determinación):** mide qué tan bien las predicciones del modelo explican la variabilidad de la variable objetivo. Su valor va de 0 a 1, siendo 1 una predicción perfecta.
* **RMSE (Root Mean Squared Error):** representa el error medio de las predicciones, penalizando más los errores grandes. Cuanto menor, mejor.

### **Descripción de los algoritmos**

* **Linear Regression:** busca ajustar una función matemática lineal (una combinación ponderada de los atributos) que mejor aproxime el valor de salida. Es un modelo interpretable, pero sensible a relaciones no lineales y a valores atípicos.
* **Random Forest Regressor:** es un ensamble de múltiples árboles de decisión entrenados sobre subconjuntos aleatorios del dataset. Cada árbol da una predicción, y el resultado final se obtiene promediando. Esta técnica es más robusta frente a datos complejos y no lineales.

### 

### **Resultados obtenidos**

#### **Linear Regression (con validación cruzada)**

* **R² por fold:** [0.8406, 0.8556, 0.8219, 0.7603, 0.8892]
* **R² promedio:** 0.8335
* **RMSE por fold:** [19,488; 25,069; 19,982; 36,925; 19,393]
* **RMSE promedio:** 24,171

#### **Random Forest Regressor (con validación cruzada)**

* **R² por fold:** [0.9795, 0.9767, 0.9610, 0.8558, 0.9568]
* **R² promedio:** 0.9460
* **RMSE por fold:** [6,986; 10,052; 9,341; 28,635; 12,108]
* **RMSE promedio:** 13,424

### 

Los resultados muestran que el Random Forest Regressor fue claramente el modelo más eficaz, con una correlación R² promedio de 0.946, lo cual indica una capacidad de predicción muy alta. Además, su error promedio (RMSE) fue de aproximadamente 13,424 USD, un valor bastante bajo considerando que el precio de los autos en el dataset puede llegar al millón.

En comparación, Linear Regression obtuvo un desempeño aceptable, con un R² promedio de 0.833, pero con un error medio casi el doble, lo que sugiere que no logra capturar adecuadamente las complejidades del conjunto de datos.

También se evaluaron los algoritmos Ridge Regression y XGBoost Regressor como variantes de los modelos principales. Ridge Regression es una forma regularizada de regresión lineal que penaliza los coeficientes grandes para evitar el sobreajuste, pero mantiene la misma lógica de una función lineal con pesos. XGBoost, por su parte, es una versión optimizada y más compleja de los árboles de decisión y funciona de forma muy similar al Random Forest, pero con un enfoque secuencial (boosting) en lugar de paralelo. Ambos modelos arrojaron resultados muy similares a los modelos originales (Linear Regression y Random Forest), lo que confirma que el conjunto de datos se ajusta bien a esas estructuras base y que no se gana una mejora sustancial al usar estos algoritmos más avanzados en este caso particular.

Resultados Ridge Regression:

R² por fold: [0.8406437 0.83231595 0.8221178 0.70824942 0.90345708]

R² promedio: 0.8213567909917494

RMSE por fold: [19489.40479088 27013.2336359 19971.8536005 40743.07895685

18103.36890366]

RMSE promedio: 25064.187977558042

Resultados XXGBoost Regressor Cross-Validation

R² scores: [0.94754219 0.98524356 0.95699954 0.88521957 0.9439593 ]

Mean R²: 0.9437928318977356

RMSE scores: [11181.98694329 8013.48563361 9819.48186006 25555.3523161

13792.75113964]

Mean RMSE: 13672.611578539136

El algoritmo K-Nearest Neighbors (KNN) realiza predicciones en base a la similitud entre observaciones. Para estimar el precio de un auto, identifica los *k* autos más parecidos en el conjunto de entrenamiento (en base a sus características) y promedia sus precios.

KNN R2 scores: [0.91897005 0.8338841 0.87945213 0.70725047 0.89853351]

KNN Mean R2: 0.8476180532582245

KNN RMSE scores: [13897.50167803 26886.62541146 16441.13859408 40812.77168927

18559.25380475]

KNN Mean RMSE: 23319.458235519305

### **Tuning del Modelo**

De los algoritmos con mejores resultados, elegimos continuar con **Random Forest Regressor** en lugar de XGBoost debido a nuestra mayor familiaridad con su funcionamiento y facilidad de ajuste.  
 En esta última etapa realizamos distintos ajustes (*tweaks*) para **mejorar la performance general del modelo** y, al mismo tiempo, **reducir el riesgo de sobreajuste** (*overfitting*). Esto es importante, ya que un modelo con un rendimiento excesivamente bueno en los datos de entrenamiento puede estar simplemente memorizando ejemplos concretos en lugar de aprender patrones generales que se trasladen bien a nuevos datos.

Evaluación de Importancia de Atributos:

Sobre Random Forest Regressor se aplicó una funcion para ver la importancia que tiene cada atributo a la hora de predecir, estos fueron los resultados principales:

Feature Importance

1 Engine HP 0.626457

3 highway MPG 0.091346

10 Make\_Bugatti 0.084270

4 Popularity 0.063314

0 Year 0.038056

42 Make\_Rolls-Royce 0.015819

32 Make\_Maybach 0.012810

64 Vehicle Style\_Convertible 0.010355

9 Make\_Bentley 0.006653

72 Vehicle Style\_Sedan 0.006131

17 Make\_Ferrari 0.005036

35 Make\_Mercedes-Benz 0.004976

2 Number of Doors 0.004670

58 Driven\_Wheels\_rear wheel drive 0.004664

26 Make\_Lamborghini 0.003964

Se puede ver como por una diferencia astronómica el Engine HP es el mas importante con mas del 60%, highway mpg, popularity y la marca bugatti estan en el rango de 6 a 9%, otros cómo Year y ciertas marcas le siguen. Esta información es muy valiosa para decidir que atributos priorizar y cuáles podemos eliminar. Se decide eliminar los tipos de autos y otros atributos que no aparecen acá como Transmission Type.

Para controlar este sobreajuste, aplicamos una técnica de **poda de árboles** mediante el parámetro max\_depth, limitando la profundidad de los árboles a 10 niveles.  
 La **poda** es un método que restringe el crecimiento de los árboles de decisión para evitar que se ajusten demasiado a ruidos o detalles irrelevantes del conjunto de entrenamiento. Esto permite que el modelo generalice mejor. También se exploraron diferentes valores de k en la validación cruzada (por ejemplo, k=3 y k=10) para verificar la estabilidad de los resultados.

Además, retomando la etapa de transformación de datos, enfrentamos el problema de una gran cantidad de categorías únicas en la variable **"marca del auto" (Make)**. En lugar de mantener todas las categorías originales —lo cual aumentaba en exceso la dimensionalidad del modelo— decidimos agrupar las marcas en **tres grandes grupos** según su precio promedio:

* **Luxury** (lujo)
* **High-End** (caro)
* **Standard** (normal)

Este enfoque nos permitió simplificar el modelo sin perder la información valiosa que aporta el prestigio o segmento de mercado de una marca.

Los atributos finales de entrenamiento que quedaron fueron: 'Engine HP', 'Year', 'highway MPG', 'Popularity', 'Make Group\_High-End', 'Make Group\_Luxury', 'Make Group\_Other', 'Make Group\_Standard'

Atributo a predecir: 'MSRP'

Durante el desarrollo del modelo de regresión para predecir el precio de los autos, implementamos diversas técnicas y ajustes con el objetivo de mejorar la precisión y evitar el sobreajuste. Aunque inicialmente se alcanzaron altos valores de R² con un modelo Random Forest sin restricciones, al aplicar técnicas de poda (limitación de la profundidad del árbol) y simplificar la cantidad de variables, el desempeño del modelo mostró una leve disminución, alcanzando un R² promedio de aproximadamente 0.91 con un error de

Este comportamiento es esperado y razonable, ya que la poda reduce la complejidad del modelo para favorecer su capacidad de generalización en datos nuevos, evitando que el modelo memorice patrones específicos del conjunto de entrenamiento. Asimismo, agrupar y reducir variables también puede implicar una pérdida parcial de información, aunque mejora la interpretabilidad y evita el ruido.

En resumen, un modelo con un R² cercano a 0.91 y menor complejidad podría ser preferible ya que equilibra precisión y robustez para futuras predicciones, sin arriesgar tanto el overfitting.

Por otro lado, existe la posibilidad de que la primera versión del modelo era mejor en general y se podaron demasiado los atributos y las ramas. Si se quiere ajustar más a los datos exactos del dataset claramente los resultados fueron mejores.