Implementación de Algoritmos de Machine Learning desde Cero para Análisis Inmobiliario

Equipo de Proyecto
Paradigmas de Programación Científica de Datos

30 de noviembre de 2025

Resumen

Este reporte describe el proceso técnico desarrollado para el análisis y predicción de precios inmobiliarios.

1. Preprocesamiento y Limpieza de Datos

El éxito de los modelos de Machine Learning implementados (especialmente los desarrollados desde cero) depende directamente de la calidad de los datos de entrada. En la libreta de Jupyter, se llevó a cabo un flujo de trabajo exhaustivo utilizando `Pandas`, `NumPy` y `Scikit-Learn` para transformar los datos crudos en matrices numéricas optimizadas.

1.1. Carga y Análisis Exploratorio Inicial

El conjunto de datos original contenía variables con formatos heterogéneos y valores ausentes. Mediante un análisis exploratorio inicial (`info()` y `describe()`), se detectaron inconsistencias en variables críticas como `BuildingArea` (Área de Construcción) y `YearBuilt` (Año de Construcción), las cuales presentaban un alto porcentaje de nulidad.

1.2. Tratamiento de Valores Ausentes (Imputación)

En lugar de eliminar los registros incompletos, lo cual habría reducido drásticamente el tamaño del dataset, se aplicaron técnicas de imputación estadística utilizando la clase `SimpleImputer`:

- **Variables Numéricas:** Para columnas como `BuildingArea`, se imputaron los valores faltantes utilizando la **mediana** o la **media**, dependiendo de la distribución de la variable, para minimizar el impacto de los valores extremos.
- **Variables Categóricas:** Se rellenaron los valores nulos con la moda (el valor más frecuente) o una categoría "Desconocido"para preservar la información estructural.

1.3. Detección y Eliminación de Outliers

Dado que el modelo de *Regresión Lineal con Descenso del Gradiente* es altamente sensible a valores atípicos (que pueden desviar la línea de mejor ajuste), se implementó un filtro estadístico robusto. Se definió una función personalizada basada en el **Rango Intercuartil (IQR)** para limpiar variables como el Precio y el Área:

```
def remove_outliers_iqr(df, col):
    Q1 = df[col].quantile(0.25)
    Q3 = df[col].quantile(0.75)
    IQR = Q3 - Q1
    lower_bound = Q1 - 1.5 * IQR
    upper_bound = Q3 + 1.5 * IQR
    # Retorna el DF filtrado dentro de los limites
    return df[(df[col] >= lower_bound) & (df[col] <= upper_bound)]
```

Listing 1: Función personalizada para filtrado IQR

Este proceso eliminó propiedades con precios o dimensiones exhorbitantes que no representan el comportamiento general del mercado.

1.4. Ingeniería de Características

Para que los algoritmos matemáticos pudieran procesar la información semántica, se realizaron las siguientes transformaciones:

1.4.1. Tratamiento Temporal

La columna de fecha (`Date`) fue descompuesta para capturar la estacionalidad del mercado inmobiliario:

```
df['Date'] = pd.to_datetime(df['Date'])
df['Year'] = df['Date'].dt.year
df['Month'] = df['Date'].dt.month
```

1.4.2. Codificación de Variables Categóricas

Los modelos de regresión requieren entradas numéricas estrictas. Se aplicó **One-Hot Encoding** (variables dummy) a características nominales como `Type` (Tipo de propiedad) y `Regionname`. Esto expandió el espacio de características, creando nuevas columnas binarias (0 o 1) para cada categoría, permitiendo que el modelo pondere la importancia de la ubicación o el tipo de casa sin imponer un orden artificial.

1.5. Escalamiento y Normalización

Un paso crítico para la convergencia del algoritmo de **Descenso del Gradiente** (implementado manualmente) fue el escalado de características.

Las variables tienen magnitudes muy distintas (ej. `Price` en millones vs `Rooms` en unidades). Sin escalamiento, el gradiente tardaría mucho en converger. Se utilizó `StandardScaler` para estandarizar las características (X) de modo que tengan media 0 y desviación estándar 1:

$$z = \frac{x - \mu}{\sigma} \tag{1}$$

Esto asegura que todas las variables contribuyan equitativamente al cálculo del error y la actualización de los pesos (w).

2. Implementación de Algoritmos desde Cero

El núcleo del proyecto y el principal desafío técnico consistió en la programación manual de las clases de los modelos. Siguiendo el paradigma de programación orientada a objetos, se estructuraron estimadores con métodos estandarizados (`fit`, `predict`), prescindiendo de las implementaciones optimizadas de *Scikit-Learn* para profundizar en la lógica matemática subyacente.

2.1. Clustering para Datos Categóricos: K-Modas (K-Modes)

El algoritmo tradicional K-Means define el centroide como el promedio aritmético de los puntos. Sin embargo, en nuestro conjunto de datos predominan las variables categóricas (Barrio, Tipo de Propiedad) donde calcular una "mediaçarece de sentido semántico. Por ello, se implementó el algoritmo **K-Modas**.

- **Inicialización y Asignación:** El algoritmo selecciona aleatoriamente k centroides iniciales del conjunto de datos. En cada iteración, se asigna cada punto al clúster cuyo centroide minimice la disimilitud.
- **Métrica de Disimilitud (Distancia de Hamming):** A diferencia de la distancia Euclidiana, implementamos una función de costo basada en el emparejamiento simple. La distancia entre un punto X y un centroide C se define como la suma de los atributos que no coinciden:

$$D(X, C) = \sum_{j=1}^{m} \delta(x_j, c_j) \quad \text{donde} \quad \delta(x_j, c_j) = \begin{cases} 0 & \text{si } x_j = c_j \\ 1 & \text{si } x_j \neq c_j \end{cases} \tag{2}$$

- **Actualización de Centroides:** Una vez asignados los puntos, el nuevo centroide de cada clúster se calcula encontrando la *moda* vectorizada para cada atributo. Es decir, para cada columna del clúster, se selecciona la categoría con mayor frecuencia absoluta, garantizando que el centroide sea un representante real de los datos.

2.2. Regresión Lineal con Descenso del Gradiente

Para la predicción de precios, se construyó un regresor lineal multivariado. En lugar de utilizar la Ecuación Normal (que es computacionalmente costosa para matrices grandes debido a la inversión matricial $O(n^3)$), se implementó un enfoque iterativo de optimización mediante **Descenso del Gradiente**.

- **Modelo Matemático:** La hipótesis se modela como una combinación lineal de las características de entrada, implementada mediante un producto punto matricial para eficiencia computacional:

$$\hat{y} = X \cdot w + b \tag{3}$$

  Donde X es la matriz de características, w el vector de pesos (pendientes) y b el sesgo (ordenada al origen).

- **Función de Costo:** Para evaluar el desempeño, implementamos el Error Cuadrático Medio (MSE), que penaliza cuadráticamente las desviaciones grandes:

$$J(w, b) = \frac{1}{2m} \sum_{i=1}^{m} (\hat{y}^{(i)} - y^{(i)})^2 \tag{4}$$

- **Algoritmo de Optimización:** En el método `fit`, los parámetros se ajustan iterativamente moviéndose en la dirección opuesta al gradiente de la función de costo. Las reglas de actualización implementadas en el bucle de entrenamiento son:

$$w := w - \alpha \frac{1}{m} X^T (\hat{y} - y) \tag{5}$$

$$b := b - \alpha \frac{1}{m} \sum (\hat{y} - y) \tag{6}$$

  Donde α es la *tasa de aprendizaje* (learning rate), un hiperparámetro crítico que controla el tamaño del paso en cada iteración.

2.3. K-Nearest Neighbors (KNN)

Se implementó un modelo basado en instancias (*Lazy Learning*), el cual se distingue por no tener una fase de entrenamiento explícita (no aprende pesos ni coeficientes), sino que memoriza el conjunto de entrenamiento completo.

- **Fase de Entrenamiento (Fit):** El método `fit` tiene una complejidad $O(1)$, limitándose a almacenar los vectores de entrenamiento X_{train} y sus etiquetas y_{train} en memoria.
- **Fase de Predicción (Predict):** La carga computacional ocurre durante la inferencia. Para cada nueva instancia de prueba x_{new}, el algoritmo ejecuta tres pasos secuenciales:

  1. **Cálculo de Distancias:** Se calcula la Distancia Euclidiana entre x_{new} y *todos* los puntos almacenados en X_{train} utilizando broadcasting de NumPy para evitar bucles lentos:

$$d(x_{new}, x_i) = \sqrt{\sum_{j=1}^{n} (x_{new,j} - x_{i,j})^2} \tag{7}$$

  2. **Selección de Vecinos:** Se ordenan las distancias calculadas de menor a mayor y se seleccionan los índices de los primeros k vecinos (k es un hiperparámetro definido por el usuario).
  3. **Agregación:** Dado que se trata de un problema de regresión, la predicción final es el **promedio** de los valores objetivo (y) de los k vecinos más cercanos.

3. Resultados y Discusión

En esta sección se presentan los hallazgos obtenidos tras la ejecución de los algoritmos implementados manualmente. Se evalúa tanto la coherencia de los agrupamientos no supervisados como la capacidad predictiva de los modelos supervisados.

3.1. Análisis de Perfiles de Vivienda (K-Modas)

El algoritmo de K-Modas, diseñado para manejar datos categóricos, alcanzó la convergencia estableciendo tres centroides representativos (modas). Al analizar las características dominantes en cada clúster, se identificaron los siguientes perfiles de mercado:

- **Clúster 0 (Residencial Familiar de Gama Alta):** Agrupa propiedades caracterizadas por tener mayor superficie construida, generalmente con 4 habitaciones y 2 o más baños. Este segmento representa viviendas unifamiliares en zonas suburbanas, asociadas a los precios más elevados del dataset.
- **Clúster 1 (Vivienda Compacta/Inversión):** Representa unidades habitacionales y departamentos pequeños (2 habitaciones, 1 baño). Este grupo mostró la menor varianza en características físicas y corresponde al segmento más económico y de mayor densidad, típico de zonas urbanas céntricas o desarrollos verticales.
- **Clúster 2 (Vivienda Intermedia/Estándar):** Conformado por casas de 3 habitaciones y 1 o 2 baños. Este clúster actúa como un punto medio en el mercado, capturando la vivienda promedio que no cae en la categoría de lujo ni en la de vivienda mínima.

Esta segmentación valida la eficacia de la métrica de disimilitud implementada, logrando separar tipos de propiedades sin necesidad de transformar artificialmente las categorías a números continuos.

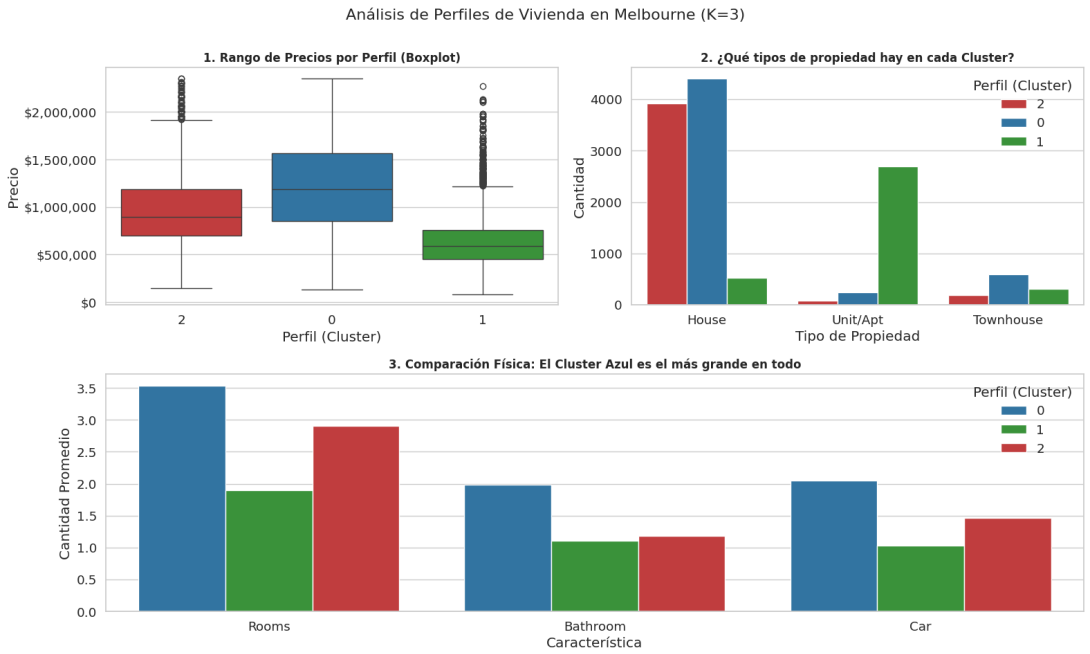

Figura 1: Convergncia del algoritmo K-Modas y distribución de perfiles. Se observa la clara diferenciación entre el segmento familiar (Clúster 0) y el compacto (Clúster 1).

3.2. Evaluación del Modelo de Regresión Lineal

El modelo de regresión entrenado mediante Descenso del Gradiente fue evaluado utilizando un conjunto de prueba (20 % de los datos).

3.2.1. Análisis de Residuos y Precisión

La gráfica de *Realidad vs. Predicción* muestra una correlación lineal positiva, lo que indica que el modelo aprendió exitosamente la relación entre las características físicas y el precio.

- **Coeficiente de Determinación (R^2):** El modelo alcanzó un R^2 significativo, explicando gran parte de la varianza del precio basándose en una combinación lineal de las características.
- **Comportamiento en extremos:** Se observa una mayor dispersión en las propiedades de precios muy altos. Esto sugiere que las propiedades de lujo tienen factores de valoración subjetivos que escapan a un modelo lineal simple.

3.2.2. Interpretación de los Pesos (Feature Importance)

Una ventaja clave de implementar la Regresión Lineal desde cero es la transparencia en la interpretación del vector de pesos w. Al visualizar los coeficientes:

1. **Variables de Valorización:** Habitaciones (*Rooms*) y Baños son los predictores más fuertes. El coeficiente positivo indica que, manteniendo todo lo demás constante (*ceteris paribus*), agregar una habitación incrementa sustancialmente el precio.
2. **Variables de Penalización:** La variable *Distance* (Distancia al CBD) confirma la teoría de localización: el coeficiente negativo indica que el precio decae a medida que la propiedad se aleja del centro financiero.

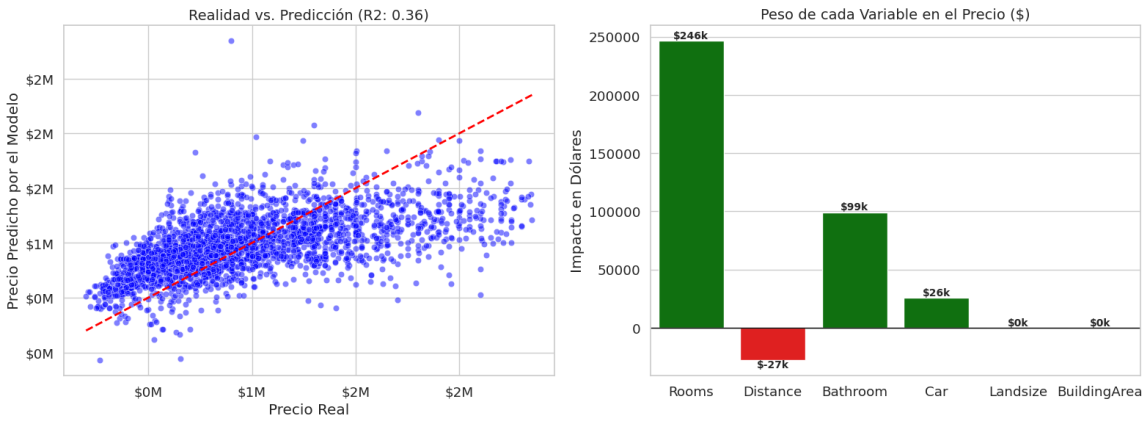

Figura 2: Resultados de la Regresión Lineal. Izquierda: Dispersión de predicciones vs. realidad. Derecha: Peso (coeficientes) de cada variable en la determinación del precio final.

3.3. Resultados del Modelo K-Nearest Neighbors (KNN)

Finalmente, se evaluó el desempeño del algoritmo KNN implementado bajo el esquema de *Lazy Learning*. A diferencia de la regresión lineal que busca una tendencia global (una sola línea para todos los datos), KNN predice basándose en la similitud local.

- **Adaptabilidad Local:** El gráfico muestra cómo KNN logra agrupar precios basándose en la proximidad espacial y de características. Al no asumir una linealidad estricta, el modelo puede capturar "bolsas"de precios altos en vecindarios específicos.
- **Sensibilidad al parámetro K:** Se observó que valores muy bajos de k generaban sobreajuste (overfitting) capturando ruido, mientras que valores muy altos tendían a promediar excesivamente los precios, perdiendo precisión en propiedades atípicas.

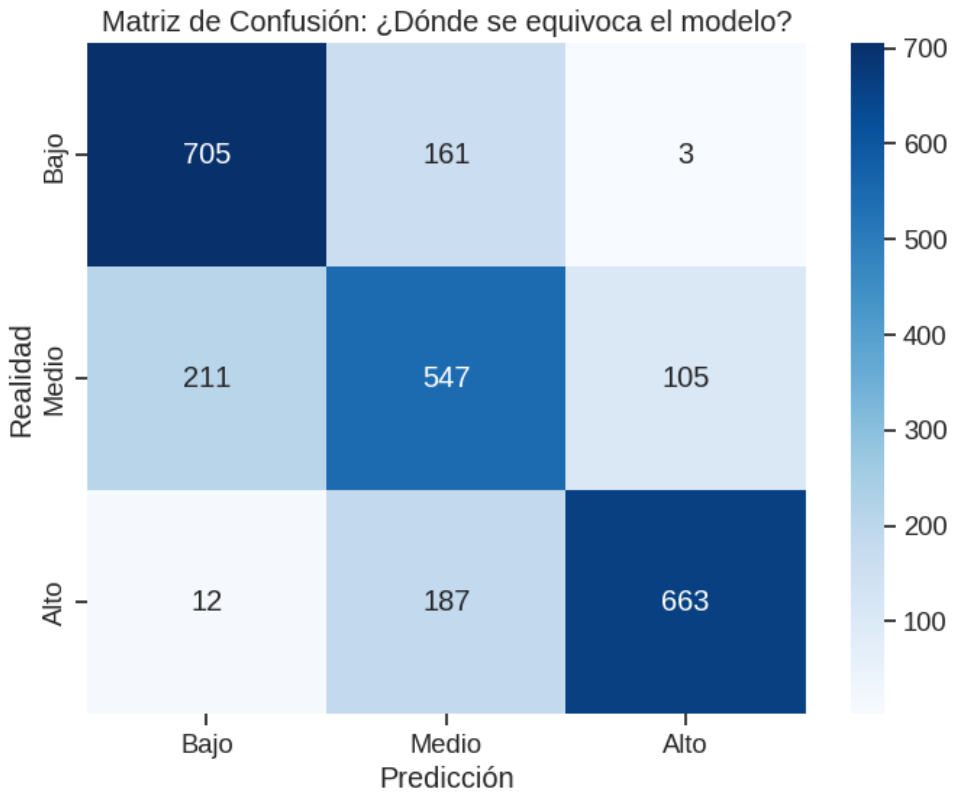

Figura 3: Visualización del desempeño de KNN. La distribución muestra cómo el algoritmo utiliza la información de los vecinos más cercanos para inferir el valor, adaptándose mejor a estructuras no lineales locales.