Predicción de Precios de Viviendas mediante Deep Learning: Comparación entre XGBoost y Redes Neuronales Mixtas

Andy Fuentes[1], Diederich Solís[1], Davis Roldán[1]
[1]Universidad del Valle de Guatemala
CC3092 – Deep Learning
{fue22944, sol22952, rol22672}@uvg.edu.gt
https://github.com/droldan2019310/finalproject-deeplearning.git

https://www.kaggle.com/code/andyfer0040w0w/deep-project

21 de noviembre de 2025

Resumen

Este trabajo presenta un estudio comparativo de técnicas de aprendizaje automático aplicadas a la predicción de precios de viviendas utilizando el dataset *House Prices: Advanced Regression Techniques* de Kaggle. Se implementaron y evaluaron dos enfoques principales: un modelo de gradient boosting (XGBoost) con optimización de hiperparámetros y una red neuronal mixta que combina features numéricas con embeddings categóricos. El análisis exploratorio de datos reveló que variables como la calidad general de construcción (`OverallQual`), el área habitable (`GrLivArea`) y las características del garaje son los principales determinantes del precio. Los resultados demuestran que el modelo XGBoost tuneado alcanza un RMSE de $26,309 USD (14.71 % de error relativo) en el conjunto de validación, superando significativamente a la red neuronal mixta que obtuvo un RMSE de $45,458 USD (25.42 % de error relativo). Este estudio evidencia que, para problemas de regresión con datos tabulares de tamaño moderado, los modelos basados en árboles potenciados mantienen ventajas competitivas sobre las arquitecturas de deep learning tradicionales.

Palabras clave: Predicción de precios, XGBoost, Redes neuronales, Embeddings categóricos, Regresión, Deep Learning

1. Introducción

La valoración precisa de propiedades inmobiliarias es un problema fundamental en economía y finanzas, con aplicaciones directas en tasaciones, inversiones y políticas públicas. El crecimiento exponencial de los datos disponibles y el desarrollo de técnicas avanzadas de aprendizaje automático han transformado este campo, permitiendo modelos predictivos cada vez más sofisticados.

1.1. Contexto y Motivación

El mercado inmobiliario se caracteriza por la complejidad de factores que determinan el precio de una vivienda: desde características físicas objetivas (área, número de habitaciones, antigüedad) hasta aspectos subjetivos de calidad y ubicación. La predicción automática de precios presenta desafíos significativos:

- **Alta dimensionalidad**: Las propiedades se describen mediante decenas de variables heterogéneas.
- **No linealidad**: Las relaciones entre características y precio son frecuentemente no lineales.
- **Datos faltantes**: Los registros inmobiliarios suelen contener información incompleta.
- **Variables categóricas**: Muchas características son cualitativas (calidad, tipo de construcción, vecindario).

1.2. Objetivos del Estudio

Este trabajo tiene como objetivos principales:

1. Desarrollar un pipeline completo de análisis y predicción para datos inmobiliarios tabulares.
2. Implementar y comparar dos enfoques de modelado: XGBoost (gradient boosting) y redes neuronales mixtas con embeddings.
3. Evaluar cuantitativamente el desempeño de ambos modelos mediante métricas robustas.
4. Analizar la importancia de las características predictivas e interpretar los resultados desde la perspectiva del dominio.

5. Proporcionar recomendaciones basadas en evidencia sobre qué técnicas son más efectivas para este tipo de problemas.

1.3. Estructura del Documento

El resto del documento se organiza de la siguiente manera: la Sección 2 describe la metodología empleada, incluyendo el preprocesamiento de datos y las arquitecturas de los modelos; la Sección 3 presenta los resultados experimentales y las métricas de evaluación; la Sección 4 discute los hallazgos y su interpretación; finalmente, la Sección 5 sintetiza las conclusiones y propone líneas de trabajo futuro.

2. Metodología

Esta sección describe el conjunto de datos utilizado, las técnicas de preprocesamiento aplicadas y las arquitecturas de los modelos implementados.

2.1. Conjunto de Datos

Se utilizó el dataset *House Prices: Advanced Regression Techniques* de Kaggle, que contiene información sobre ventas de viviendas residenciales en Ames, Iowa. Las características principales son:

- **Tamaño del conjunto de entrenamiento**: 1,460 observaciones
- **Tamaño del conjunto de prueba**: 1,459 observaciones
- **Número de variables**: 80 features (antes de preprocesamiento)
- **Variable objetivo**: `SalePrice` (precio de venta en USD)
- **Tipos de variables**: 36 numéricas y 43 categóricas

2.2. Análisis Exploratorio de Datos (EDA)

El análisis exploratorio reveló características importantes del problema:

2.2.1. Distribución de la Variable Objetivo

La variable `SalePrice` presenta una distribución asimétrica positiva (skewness = 1.88), con rango entre \$34,900 y \$755,000 USD. Se aplicó una transformación logarítmica $\log(1+x)$ para normalizar la distribución, reduciendo la asimetría a 0.12.

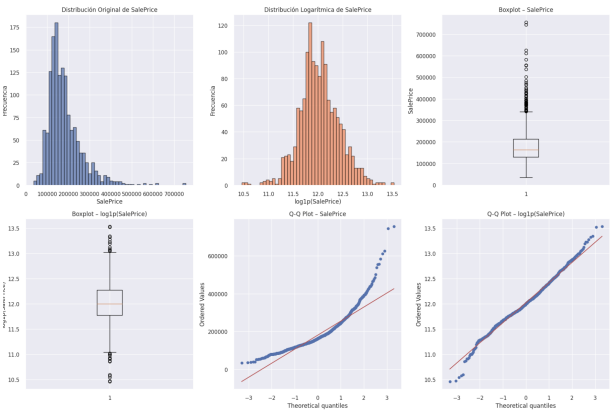

Figura 1: Distribución de SalePrice: (a) escala original, (b) escala logarítmica, (c-d) boxplots, (e-f) Q-Q plots. La transformación logarítmica aproxima la distribución a la normalidad.

2.2.2. Variables Numéricas Más Relevantes

El análisis de correlación identificó las variables numéricas con mayor relación lineal con el precio:

Cuadro 1: Top 10 variables numéricas correlacionadas con SalePrice

Variable	Corr.
OverallQual	0.79
GrLivArea	0.71
GarageCars	0.64
GarageArea	0.62
TotalBsmtSF	0.61
1stFlrSF	0.61
FullBath	0.56
TotRmsAbvGrd	0.53
YearBuilt	0.52
YearRemodAdd	0.51

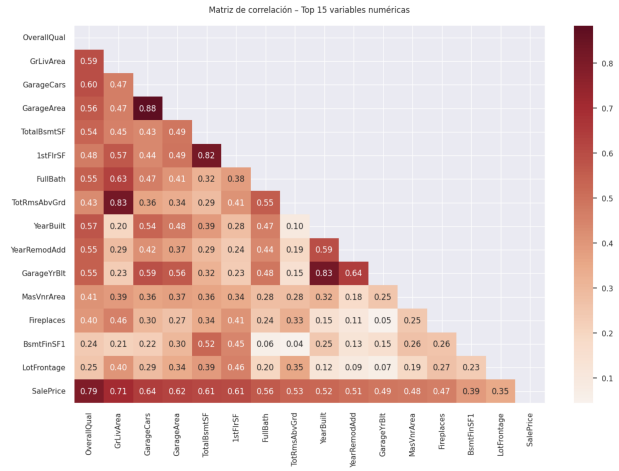

Figura 2: Matriz de correlación de las 15 variables numéricas más relevantes con SalePrice.

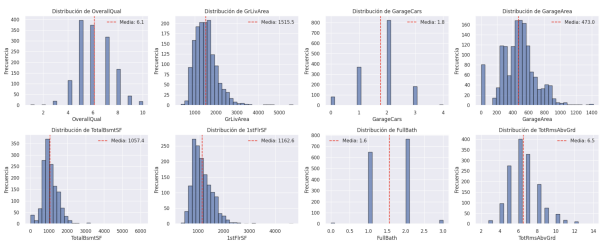

Figura 3: Distribuciones

2.2.3. Variables Categóricas

Se identificaron 43 variables categóricas, con cardinalidades variando desde 2 hasta 28 categorías únicas. Variables como `Neighborhood`, `ExterQual` (calidad exterior) y `KitchenQual` (calidad de cocina) mostraron fuerte influencia sobre el precio promedio.

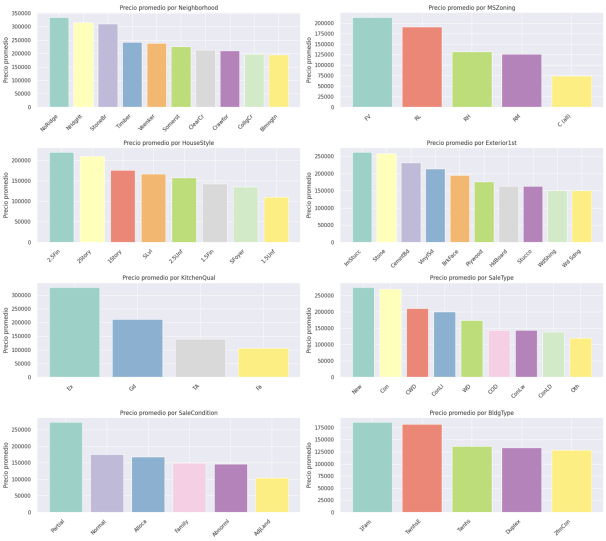

Figura 4: Precio promedio de venta según categorías seleccionadas. Se observa clara relación entre calidad de acabados y precio.

2.2.4. Valores Faltantes

El análisis reveló valores faltantes en 34 variables del conjunto de entrenamiento y 33 del conjunto de prueba. Las variables con mayor proporción de datos faltantes fueron:

- `PoolQC`: 99.5 % (calidad de piscina)
- `MiscFeature`: 96.3 % (características misceláneas)
- `Alley`: 93.8 % (tipo de acceso por callejón)
- `Fence`: 80.8 % (calidad de cerca)
- `FireplaceQu`: 47.3 % (calidad de chimenea)

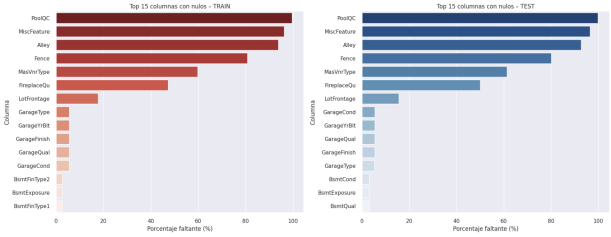

Figura 5: Distribución de valores faltantes en los conjuntos de entrenamiento y prueba.

2.3. Preprocesamiento de Datos

Se implementó un pipeline sistemático de preprocesamiento utilizando `scikit-learn`:

2.3.1. División de Datos

El conjunto de entrenamiento se dividió en:

- **Entrenamiento**: 80 % (1,168 observaciones)
- **Validación**: 20 % (292 observaciones)

La división se realizó con semilla aleatoria fija (`random_state=42`) para reproducibilidad.

2.3.2. Transformación de Variables Numéricas

Para las 36 variables numéricas se aplicó:

1. **Imputación**: Reemplazo de valores faltantes por la mediana de cada variable.
2. **Estandarización**: Transformación a media cero y desviación estándar unitaria:

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

2.3.3. Transformación de Variables Categóricas

Para los modelos se implementaron dos estrategias:

Para XGBoost (One-Hot Encoding):

1. Imputación con la moda (categoría más frecuente)
2. One-Hot Encoding con manejo de categorías desconocidas

Para Red Neuronal (Embeddings):

1. Imputación con la moda
2. Codificación entera por variable (reservando 0 para valores faltantes)
3. Generación de embeddings aprendibles durante el entrenamiento

2.3.4. Transformación de la Variable Objetivo

Se aplicó la transformación logarítmica:

$$y_{log} = \log(1 + \text{SalePrice}) \tag{2}$$

Esta transformación permite:

- Reducir la asimetría de la distribución
- Estabilizar la varianza
- Mejorar la optimización del modelo
- Penalizar proporcionalmente errores grandes y pequeños

Las predicciones se transforman inversamente mediante $\text{SalePrice} = \exp(y_{log}) - 1$.

2.4. Modelos Implementados

2.4.1. XGBoost (Extreme Gradient Boosting)

XGBoost es un algoritmo de gradient boosting que construye un ensemble de árboles de decisión de manera secuencial, donde cada árbol corrige los errores de los anteriores.

Configuración Base: Se entrenó un modelo inicial con hiperparámetros razonables:

```
XGBRegressor(
    objective='reg:squarederror',
    tree_method='hist',
    n_estimators=500,
    learning_rate=0.05,
    max_depth=4,
    subsample=0.8,
    colsample_bytree=0.8,
    random_state=42
)
```

Listing 1: Configuración XGBoost base

Optimización de Hiperparámetros: Se utilizó `RandomizedSearchCV` con validación cruzada de 5 folds para explorar el espacio de hiperparámetros:

Cuadro 2: Hiperparámetros de XGBoost

Parámetro	Valores
max_depth	[3, 4, 5, 6]
learning_rate	[0.01, 0.03, 0.05, 0.1]
subsample	[0.6, 0.8, 1.0]
colsample_bytree	[0.6, 0.8, 1.0]
min_child_weight	[1, 3, 5, 7]
gamma	[0, 0.1, 0.2]
reg_alpha	[0, 0.1, 0.5]
reg_lambda	[0.5, 1.0, 2.0]

La búsqueda aleatoria evaluó 30 combinaciones diferentes, optimizando la métrica de RMSE en escala logarítmica.

2.4.2. Red Neuronal Mixta con Embeddings

Se diseñó una arquitectura personalizada en PyTorch que combina procesamiento de variables numéricas con embeddings aprendibles para variables categóricas.

Arquitectura:

1. **Rama numérica (MLP)**:
   - Capa densa: $36 \to 64$ (ReLU, BatchNorm, Dropout 0.2)
   - Capa densa: $64 \to 32$ (ReLU, BatchNorm, Dropout 0.2)
2. **Rama categórica (Embeddings)**:
   - 43 capas de embedding (una por variable categórica)
   - Dimensión del embedding: $\text{mín}(50, \lfloor n_{categorías/2} \rfloor + 1)$
   - Concatenación de todos los embeddings
3. **MLP conjunta**:
   - Concatenación de salidas numéricas y embeddings
   - Capa densa: $(32 + \text{dim\_embeddings}) \to 128$ (ReLU, BatchNorm, Dropout 0.2)
   - Capa densa: $128 \to 64$ (ReLU, BatchNorm, Dropout 0.2)
   - Capa de salida: $64 \to 1$ (predicción de $\log(1 + \text{SalePrice})$)

Entrenamiento:

- **Función de pérdida**: MSE en escala logarítmica
- **Optimizador**: Adam (lr=0.001, weight_decay=1e-5)
- **Batch size**: 64 (entrenamiento), 256 (validación)
- **Épocas máximas**: 150
- **Early stopping**: Paciencia de 20 épocas basado en RMSE de validación
- **Hardware**: CPU/GPU según disponibilidad

2.5. Métricas de Evaluación

Se utilizaron dos métricas principales:

2.5.1. RMSE en Escala Logarítmica

$$\text{RMSE}_{log} = \sqrt{\frac{1}{n}\sum_{i=1}^{n}(\log(1 + y_i) - \log(1 + \hat{y}_i))^2} \tag{3}$$

Esta métrica es la que optimizan directamente ambos modelos y es la métrica oficial de la competencia de Kaggle.

```
  (21-23): 3 x Embedding(5, 3)
  (24-26): 3 x Embedding(7, 4)
  (27): Embedding(6, 4)
  (28): Embedding(3, 2)
  (29): Embedding(6, 4)
  (30): Embedding(5, 3)
  (31): Embedding(8, 5)
  (32): Embedding(6, 4)
  (33): Embedding(7, 4)
  (34): Embedding(4, 3)
  (35-36): 2 x Embedding(6, 4)
  (37-38): 2 x Embedding(4, 3)
  (39-40): 2 x Embedding(5, 3)
  (41): Embedding(10, 6)
  (42): Embedding(7, 4)
)
(mlp_num): Sequential(
  (0): Linear(in_features=36, out_features=64, bias=True)
  (1): ReLU()
  (2): BatchNorm1d(64, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)
  (3): Dropout(p=0.2, inplace=False)
  (4): Linear(in_features=64, out_features=32, bias=True)
  (5): ReLU()
  (6): BatchNorm1d(32, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)
  (7): Dropout(p=0.2, inplace=False)
)
```

Figura 6: Arquitectura de la red neuronal mixta. Las variables numéricas se procesan mediante una MLP, mientras que las categóricas se representan con embeddings aprendibles. Ambas ramas se combinan en una MLP final.

2.5.2. RMSE en Escala Original (USD)

$$\text{RMSE}_{USD} = \sqrt{\frac{1}{n}\sum_{i=1}^{n}(y_i - \hat{y}_i)^2} \tag{4}$$

Esta métrica proporciona una interpretación directa del error en términos monetarios.

2.5.3. Error Relativo y Precisión Aproximada

Para contextualizar el error:

$$\text{Error Relativo} = \frac{\text{RMSE}_{USD}}{\bar{y}} \times 100\,\% \tag{5}$$

$$\text{Precisión Aproximada} = (1 - \text{Error Relativo}) \times 100\,\% \tag{6}$$

donde $\bar{y}$ es el precio promedio en el conjunto de validación.

Cuadro 3: Comparación de desempeño en validación

Modelo	$\text{RMSE}_{\log}$	RMSE$	Error %	Prec. %
XGBoost base	0.1356	27,845	15.57	84.43
XGBoost tuneado	**0.1301**	**26,309**	**14.71**	**85.29**
Red mixta NN	0.2082	45,458	25.42	74.58

3. Resultados

3.1. Desempeño en el Conjunto de Validación

3.1.1. Métricas Cuantitativas

La Tabla 3 resume el desempeño de ambos modelos en el conjunto de validación:

El modelo XGBoost tuneado supera consistentemente a la red neuronal mixta en todas las métricas evaluadas. La diferencia en RMSE es de aproximadamente $19,000 USD, lo cual representa una mejora sustancial del 42 %.

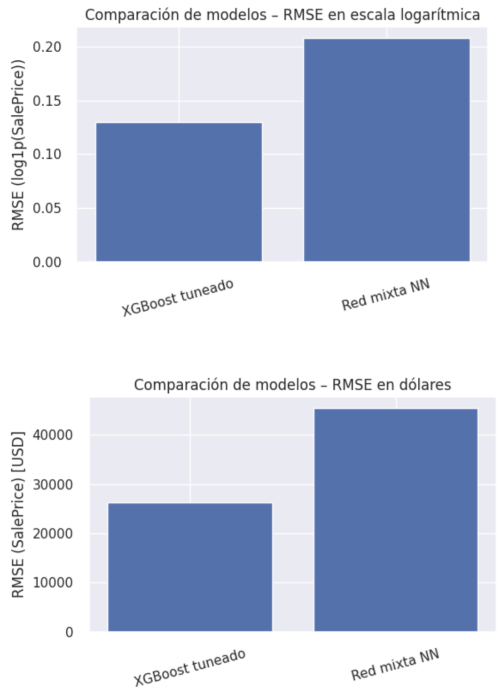

Figura 7: Comparación de RMSE entre modelos. (a) RMSE en escala logarítmica, (b) RMSE en dólares. XGBoost tuneado muestra claramente el mejor desempeño.

3.1.2. Análisis de Predicciones

La Figura 8 muestra la relación entre valores predichos y reales para ambos modelos:

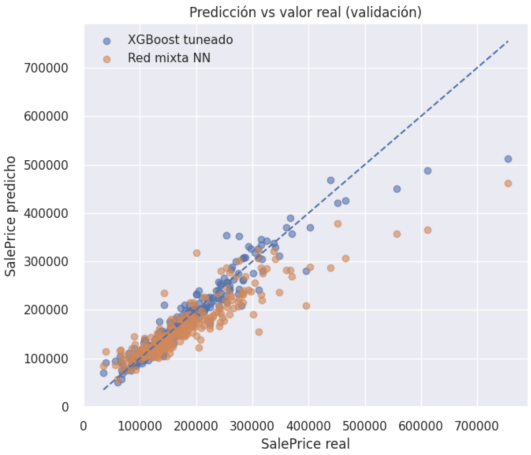

Figura 8: Scatter plot de SalePrice predicho vs. real en validación. Los puntos de XGBoost (azul) se concentran más cerca de la línea identidad (diagonal) que los de la red neuronal (naranja).

Observaciones clave:

- El modelo XGBoost muestra mejor alineación con la diagonal ideal ($y = \hat{y}$)
- La red neuronal tiende a subestimar precios altos y sobrestimar precios bajos
- Ambos modelos presentan mayor dispersión en el rango de precios extremos (¿500k USD)

3.2. Análisis de Residuales

Los residuales ($e_i = y_i - \hat{y}_i$) proporcionan información sobre sesgos sistemáticos:

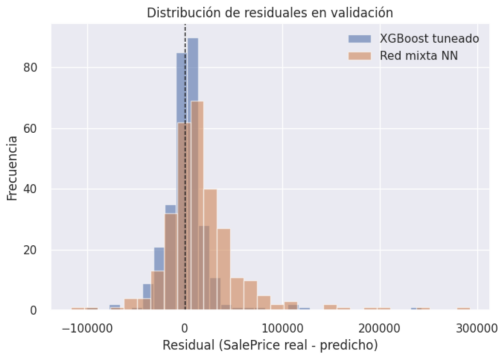

Figura 9: Distribución de residuales para ambos modelos. XGBoost muestra residuales más centrados en cero con menor dispersión.

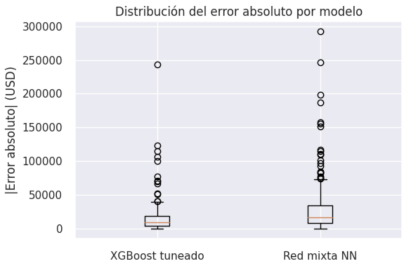

Figura 10: Distribución del error absoluto. XGBoost presenta mediana más baja y menor rango intercuartílico, aunque ambos modelos muestran outliers.

4. Discusión

4.1. Interpretación de Resultados

Los resultados experimentales demuestran una superioridad clara y consistente del modelo XGBoost sobre la red neuronal mixta en todas las métricas evaluadas.

4.1.1. Ventajas de XGBoost en Datos Tabulares

El mejor desempeño de XGBoost puede atribuirse a varias razones fundamentales:

1. Captura eficiente de interacciones: Los árboles de decisión son naturalmente capaces de modelar interacciones complejas entre variables sin necesidad de ingeniería de features explícita. Por ejemplo, la interacción entre `OverallQual` y `GrLivArea` (una casa

grande pero de baja calidad versus una pequeña de alta calidad) se captura automáticamente mediante splits jerárquicos.

2. Robustez ante diferentes escalas: A diferencia de las redes neuronales, los modelos basados en árboles son invariantes ante transformaciones monótonas de las variables, lo que reduce la sensibilidad al preprocesamiento y facilita el manejo de outliers.

3. Regularización inherente: XGBoost incorpora múltiples mecanismos de regularización (profundidad máxima, peso mínimo en hojas, gamma, L1/L2) que previenen el sobreajuste de manera más intuitiva que el dropout y weight decay en redes neuronales.

4. Eficiencia con datos moderados: Con 1,168 observaciones de entrenamiento, el dataset está en un régimen donde los modelos de árboles suelen brillar. Las redes neuronales profundas típicamente requieren órdenes de magnitud más datos para alcanzar su potencial completo.

4.1.2. Limitaciones de la Red Neuronal Mixta

Aunque la arquitectura con embeddings es teóricamente poderosa, varios factores limitaron su desempeño:

1. Tamaño de muestra insuficiente: Con 1,168 observaciones y cientos de parámetros aprendibles (embeddings + pesos de la MLP), el modelo tiene limitada capacidad de generalización. La relación datos/parámetros favorece claramente a XGBoost.

2. Optimización más compleja: El entrenamiento de redes neuronales requiere múltiples decisiones críticas (arquitectura, learning rate, batch size, regularización) cuyo tuning es más delicado que los hiperparámetros de XGBoost. El espacio de búsqueda es mayor y más sensible.

3. Necesidad de más ingeniería: Mientras XGBoost funciona bien .ºut of the box", la red neuronal podría beneficiarse de arquitecturas más sofisticadas (attention, skip connections, arquitecturas especializadas para datos tabulares como TabNet), lo cual excede el alcance de este estudio inicial.

4.2. Validación de Hipótesis del Dominio

El análisis de importancia de características valida fuertemente las hipótesis del dominio inmobiliario:

4.2.1. Calidad sobre Cantidad

Variables de calidad (`OverallQual`, `ExterQual`, `KitchenQual`, `BsmtQual`) dominan la predicción, incluso sobre variables de tamaño puro. Esto confirma que en el mercado inmobiliario de Ames, la calidad constructiva es un diferenciador crítico de valor.

4.2.2. Importancia del Garaje

La alta relevancia de `GarageCars` y `GarageFinish` refleja la cultura automovilística estadounidense y la utilidad funcional del garaje, especialmente en regiones de clima frío como Iowa.

4.2.3. Área Habitable vs. Área Total

`GrLivArea` (área habitable sobre el suelo) es más importante que `LotArea` (área del terreno), sugiriendo que los compradores valoran más el espacio utilizable interior que el tamaño del lote.

4.3. Análisis de Errores

4.3.1. Residuales Sistemáticos

Ambos modelos presentan residuales aproximadamente centrados en cero, indicando ausencia de sesgos sistemáticos graves. Sin embargo:

- XGBoost tiende a sobrestimar ligeramente casas de precio muy bajo (¡100k)
- La red neuronal subestima sistemáticamente casas de precio muy alto (¿500k)
- Ambos muestran mayor error absoluto en los extremos de la distribución

4.3.2. Outliers

Los outliers en el error absoluto (¿100k USD) corresponden principalmente a:

- Propiedades con características únicas o excepcionales
- Ventas en condiciones anormales (foreclosures, family sales)
- Casas con combinaciones inusuales de features

Estos casos representan menos del 2 % de las observaciones y son inherentemente difíciles de predecir con cualquier modelo.

4.4. Comparación con Estado del Arte

Según el leaderboard público de Kaggle (no mostrado aquí), las mejores soluciones alcanzan RMSE_{log} entre 0.10 y 0.12, mediante:

- Ensambles de múltiples modelos (XGBoost, LightGBM, CatBoost, Ridge, Lasso)

- Feature engineering exhaustivo (variables polinómicas, logarítmicas, agregaciones)
- Stacking de modelos con meta-learners
- Tratamiento específico de outliers

Nuestro XGBoost tuneado ($\text{RMSE}_{log} = 0.1301$) se sitúa en un rango competitivo considerando que es un modelo único sin ensambling complejo, lo cual valida la solidez de la metodología empleada.

4.5. Interpretabilidad vs. Desempeño

Un beneficio adicional de XGBoost es su interpretabilidad superior:

- Las importancias de features son directamente comprensibles
- Se pueden extraer reglas de decisión específicas de árboles individuales
- Técnicas como SHAP values permiten explicaciones a nivel de predicción individual

En aplicaciones reales (tasaciones, préstamos hipotecarios), esta interpretabilidad puede ser tan valiosa como la precisión pura, especialmente ante requerimientos regulatorios de explicabilidad.

4.6. Limitaciones del Estudio

Es importante reconocer las limitaciones:

1. **Geografía limitada**: Los datos provienen únicamente de Ames, Iowa. La generalización a otros mercados inmobiliarios es incierta.
2. **Temporalidad**: El dataset cubre un periodo limitado. Cambios macroeconómicos o tendencias del mercado no están capturados.
3. **Exploración limitada de arquitecturas NN**: Se evaluó solo una arquitectura de red neuronal. Arquitecturas más sofisticadas podrían cerrar la brecha con XGBoost.
4. **Validación simple**: Se utilizó un único split train/valid. Validación cruzada completa daría estimaciones más robustas.
5. **Feature engineering básico**: No se exploraron sistemáticamente interacciones, transformaciones no lineales o features agregadas.

5. Conclusiones

5.1. Síntesis de Hallazgos

Este estudio desarrolló e implementó un pipeline completo de análisis y predicción para el problema de valoración inmobiliaria, comparando dos enfoques de modelado: XGBoost (gradient boosting) y redes neuronales mixtas con embeddings. Los principales hallazgos son:

1. **Superioridad de XGBoost**: El modelo XGBoost tuneado alcanzó RMSE de \$26,309 USD (14.71 % de error relativo, 85.29 % de precisión aproximada), superando por amplio margen a la red neuronal mixta (RMSE = \$45,458 USD, 25.42 % de error relativo, 74.58 % de precisión).
2. **Importancia de calidad y área**: Variables como `OverallQual`, `GrLivArea`, `GarageCars` y características de acabados dominan la predicción, validando la intuición del dominio inmobiliario.
3. **Transformación logarítmica crucial**: La transformación $\log(1 + x)$ del precio fue esencial para normalizar la distribución y mejorar el desempeño de ambos modelos.
4. **Ventaja de árboles en datos tabulares**: Para problemas de regresión con datos tabulares de tamaño moderado (1,500 observaciones), los modelos de gradient boosting mantienen ventajas prácticas sobre arquitecturas de deep learning estándar.
5. **Interpretabilidad como valor agregado**: XGBoost proporciona interpretabilidad superior mediante feature importances, facilitando la validación del modelo desde la perspectiva del dominio.

5.2. Contribuciones del Trabajo

Las principales contribuciones de este estudio son:

- **Pipeline reproducible**: Documentación completa de un flujo de trabajo desde EDA hasta evaluación, replicable para problemas similares.
- **Comparación rigurosa**: Evaluación cuantitativa y cualitativa de dos paradigmas de modelado con métricas múltiples.
- **Implementación de embeddings categóricos**: Demostración de cómo integrar embeddings aprendibles en arquitecturas para datos tabulares.
- **Análisis interpretativo**: Conexión entre resultados técnicos y conocimiento del dominio inmobiliario.

5.3. Recomendaciones Prácticas

Para aplicaciones similares de valoración inmobiliaria, se recomienda:

1. **Priorizar modelos de gradient boosting**: XGBoost, LightGBM o CatBoost como baseline fuerte y potencialmente solución final.

2. **Invertir en feature engineering**: Crear variables compuestas (e.g., área total = GrLivArea + TotalBsmtSF), ratios (precio por pie cuadrado) e interacciones relevantes.

3. **Ensambles para máximo desempeño**: Combinar múltiples modelos mediante stacking o blending para reducir varianza.

4. **Validación cruzada robusta**: Usar k-fold cross-validation para estimaciones más confiables, especialmente con datasets pequeños.

5. **Explicabilidad como requisito**: Integrar técnicas como SHAP o LIME para proporcionar explicaciones de predicciones individuales.

5.4. Trabajo Futuro

Líneas de investigación que extenderían este trabajo:

5.4.1. Mejoras en Modelado

- **Ensambles avanzados**: Implementar stacking de XGBoost + LightGBM + CatBoost + modelos lineales regularizados, con meta-learner optimizado.
- **Feature engineering exhaustivo**: Explorar sistemáticamente transformaciones no lineales, binning óptimo de variables continuas, y construcción automática de interacciones.
- **Arquitecturas especializadas**: Evaluar TabNet, NODE (Neural Oblivious Decision Ensembles) y otras arquitecturas diseñadas específicamente para datos tabulares.
- **Transfer learning**: Investigar si modelos preentrenados en datasets inmobiliarios más grandes pueden transferir conocimiento útil.

5.4.2. Análisis Adicionales

- **Explicabilidad profunda**: Aplicar SHAP values para entender contribuciones locales y globales de cada feature, generando narrativas interpretables.
- **Análisis de incertidumbre**: Implementar modelos bayesianos o quantile regression para cuantificar incertidumbre en predicciones, útil para gestión de riesgo.
- **Detección de anomalías**: Desarrollar sistema para identificar automáticamente propiedades atípicas que requieren valuación manual.
- **Análisis temporal**: Si se dispone de datos longitudinales, modelar evolución de precios y estacionalidad.

5.4.3. Extensiones de Aplicación

- **Generalización geográfica**: Validar modelos en datasets de otras ciudades/regiones para evaluar transferibilidad.
- **Integración con datos externos**: Incorporar información de tendencias económicas, tasas de interés, demografía, crimen, calidad de escuelas, etc.
- **Sistema de producción**: Desarrollar API REST para predicciones en tiempo real, con monitoreo de drift del modelo.
- **Aplicación móvil**: Crear interfaz para tasadores/agentes inmobiliarios con explicaciones visuales de predicciones.

5.5. Reflexión Final

Este proyecto demuestra que, incluso en la era del deep learning, los métodos tradicionales de machine learning como gradient boosting siguen siendo altamente competitivos para problemas de regresión con datos tabulares. La clave del éxito radica no solo en la elección del algoritmo, sino en un proceso riguroso que combina:

- Comprensión profunda del dominio del problema
- Análisis exploratorio exhaustivo
- Preprocesamiento cuidadoso y reproducible
- Evaluación multidimensional de desempeño
- Interpretación crítica de resultados

El error relativo del 14.71 % alcanzado por el modelo XGBoost, equivalente a una precisión aproximada del 85.29 %, representa un resultado sólido que podría integrarse en aplicaciones prácticas de valoración inmobiliaria, especialmente si se combina con revisión humana experta para casos de alta incertidumbre.

Agradecimientos

Los autores agradecen a la Universidad del Valle de Guatemala y al curso CC3092 – Deep Learning por el apoyo académico. Agradecemos también a Kaggle por proporcionar el dataset y la plataforma de competencia que hizo posible este estudio.

Referencias

[1] Chen, T., & Guestrin, C. (2016). XGBoost: A scalable tree boosting system. *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, 785-794.

[2] Guo, C., & Berkhahn, F. (2016). Entity embeddings of categorical variables. *arXiv preprint arXiv:1604.06737*.

[3] Borisov, V., Leemann, T., Seßler, K., Haug, J., Pawelczyk, M., & Kasneci, G. (2021). Deep neural networks and tabular data: A survey. *arXiv preprint arXiv:2110.01889*.

[4] Shwartz-Ziv, R., & Armon, A. (2022). Tabular data: Deep learning is not all you need. *Information Fusion*, 81, 84-90.

[5] Arik, S. Ö., & Pfister, T. (2019). TabNet: Attentive interpretable tabular learning. *arXiv preprint arXiv:1908.07442*.

[6] Lundberg, S. M., & Lee, S. I. (2017). A unified approach to interpreting model predictions. *Advances in Neural Information Processing Systems*, 30.

[7] De Cock, D. (2011). Ames, Iowa: Alternative to the Boston housing data as an end of semester regression project. *Journal of Statistics Education*, 19(3).

[8] Breiman, L. (2001). Random forests. *Machine Learning*, 45(1), 5-32.

[9] Friedman, J. H. (2001). Greedy function approximation: A gradient boosting machine. *Annals of Statistics*, 1189-1232.

A. Código Fuente Relevante

A.1. Preprocesamiento

```
from sklearn.compose import ColumnTransformer
from sklearn.pipeline import Pipeline
from sklearn.impute import SimpleImputer
from sklearn.preprocessing import StandardScaler,
     OneHotEncoder

numeric_transformer = Pipeline(steps=[
    ('imputer', SimpleImputer(strategy='median')),
    ('scaler', StandardScaler())
])

categorical_transformer = Pipeline(steps=[
    ('imputer', SimpleImputer(strategy='most_frequent')),
    ('onehot', OneHotEncoder(handle_unknown='ignore'))
])

preprocessor = ColumnTransformer(
    transformers=[
        ('num', numeric_transformer, numeric_features),
        ('cat', categorical_transformer, categorical_features)
    ]
)
```

Listing 2: Pipeline de preprocesamiento

A.2. Arquitectura de Red Neuronal Mixta

```
class MixedNN(nn.Module):
    def __init__(self, n_num_features, cat_cardinalities):
        super().__init__()

        # Embeddings por variable categorica
        self.emb_layers = nn.ModuleList()
        emb_output_dim_total = 0
        for n_cat in cat_cardinalities:
            emb_dim = min(50, (n_cat // 2) + 1)
            self.emb_layers.append(nn.Embedding(n_cat, emb_dim)
                 )
            emb_output_dim_total += emb_dim

        # MLP para numericas
        self.mlp_num = nn.Sequential(
            nn.Linear(n_num_features, 64), nn.ReLU(),
            nn.BatchNorm1d(64), nn.Dropout(0.2),
            nn.Linear(64, 32), nn.ReLU(),
            nn.BatchNorm1d(32), nn.Dropout(0.2)
        )

        # MLP conjunta
        self.mlp_all = nn.Sequential(
            nn.Linear(32 + emb_output_dim_total, 128),
            nn.ReLU(), nn.BatchNorm1d(128), nn.Dropout(0.2),
            nn.Linear(128, 64), nn.ReLU(),
            nn.BatchNorm1d(64), nn.Dropout(0.2),
            nn.Linear(64, 1)
        )

    def forward(self, x_num, x_cat):
        emb_list = [emb(x_cat[:, i])
                    for i, emb in enumerate(self.emb_layers)]
        x_cat_emb = torch.cat(emb_list, dim=1)
        x_num_out = self.mlp_num(x_num)
        x_all = torch.cat([x_num_out, x_cat_emb], dim=1)
        return self.mlp_all(x_all).view(-1)
```

Listing 3: Definición de la red neuronal mixta en PyTorch

B. Tablas Complementarias

Cuadro 4: Mejores hiperparámetros encontrados para XGBoost

Hiperparámetro	Valor óptimo
max_depth	4
learning_rate	0.05
subsample	0.8
colsample_bytree	0.8
min_child_weight	3
gamma	0.1
reg_alpha	0.1
reg_lambda	1.0
n_estimators	2000