Proyecto del curso
Modelos y Simulación II

Sebastián Amaya Pérez, Jhon Alejandro García Pareja y David Felipe Tovar Zurita

I. Introducción

El aprendizaje supervisado permite encontrar relaciones entre las variables de un conjunto de datos para hacer predicciones a partir de ejemplos ya conocidos. Según lo visto en el curso *Introduction to Machine Learning*, cuando la salida que se quiere estimar es un valor numérico continuo, la forma adecuada de abordar el problema es mediante técnicas de regresión. Entre ellas, la regresión lineal se convierte en el primer punto de referencia gracias a que es fácil de interpretar y sirve como modelo base para comparar métodos más avanzados más adelante.

En este proyecto, el objetivo es predecir el precio de venta de viviendas utilizando el dataset *House Prices: Advanced Regression Techniques* de Kaggle. Para lograrlo, se realizará un análisis exploratorio de los datos, su limpieza, la imputación de valores faltantes y la codificación de variables categóricas, para posteriormente entrenar un modelo de regresión lineal. Una vez entrenado, su rendimiento será evaluado con métricas comunes en problemas de regresión, con el fin de medir qué tan bien logra ajustar la información disponible.

Finalmente, los resultados obtenidos permitirán identificar qué tanto puede aportar un modelo lineal en este tipo de problema y si es necesario acudir a modelos más complejos en posibles futuras etapas del trabajo.

A. Contexto del Problema

En el mercado inmobiliario, estimar el precio real de una vivienda es un reto debido a la cantidad de factores que influyen en su valor, tales como la ubicación, el estado de la construcción, el área total, las mejoras internas y las condiciones económicas del entorno. A pesar de contar con información histórica, en la práctica muchos cálculos siguen basándose en métodos subjetivos o modelos tradicionales que no logran capturar relaciones complejas entre las variables, lo que genera incertidumbre y decisiones poco acertadas en procesos de compra, venta o inversión.

El uso de técnicas de *Machine Learning* ofrece una alternativa más sólida y eficiente para este tipo de problemas. Estos métodos permiten analizar grandes volúmenes de datos, identificar patrones no lineales y generar modelos predictivos con mayor precisión que los enfoques estadísticos convencionales. Desarrollar una solución basada en *Machine Learning* para la predicción del precio de viviendas no solo mejora la exactitud de las estimaciones, sino que también facilita la toma de decisiones informadas, reduce el margen de error y aporta mayor transparencia al mercado inmobiliario.

Fig. 1. *

B. Composición de la Base de Datos

La base de datos utilizada se obtuvo de la plataforma Kaggle en el desafío *House Prices: Advanced Regression Techniques*. Esta base de datos contiene los siguientes cuatro archivos:

- **train.csv:** 1 460 muestras con 81 columnas, incluyendo la variable objetivo `SalePrice`.
- **data_description.txt:** documento que detalla el significado de cada variable, sus valores posibles y notas adicionales.
- **sample_submission.csv:** archivo ejemplo que muestra el formato requerido para la presentación de predicciones en el concurso.

Las variables presentes en la base de datos describen distintos aspectos de cada vivienda. Estas pueden agruparse conceptualmente en las siguientes categorías:

- Características físicas del lote: área, forma, frente del terreno, pendiente, entre otros.
- Características estructurales de la vivienda: número de pisos, año de construcción, área habitable, materiales y estado exterior.
- Características internas: número de habitaciones, baños, sótano, chimenea, cocina, acabados y su nivel de calidad.
- Características adicionales: garaje, piscina, cercas, porches y otras mejoras.
- Información del vecindario: ubicación y clasificación residencial del sector.

1) Existencia de Datos Faltantes: Durante la revisión del conjunto de entrenamiento se identificó la presencia de valores representados como `NA`. Estos casos se dividen en dos situaciones:

1) **Valores `NA` que significan "no aplica":** Aparecen principalmente en variables categóricas asociadas a características que no existen en la vivienda. Por ejemplo, si una propiedad no tiene chimenea, la columna que describe su calidad aparece con `NA`, indicando ausencia de la característica y no pérdida de información.
2) **Valores `NA` que representan datos faltantes reales:** Se presentan en columnas numéricas donde el valor sí debería existir. Un ejemplo es la variable `LotFrontage`,

donde el frente del lote no se encuentra registrado para algunas viviendas.

Con el fin de preparar los datos para la construcción del modelo, se adoptará la siguiente estrategia:

- Para variables numéricas con datos faltantes reales: imputación mediante la mediana, con el fin de evitar distorsiones ocasionadas por valores atípicos.
- Para variables categóricas donde `NA` indica ausencia de característica: sustitución por la categoría `"None"`, manteniendo así la coherencia semántica de la información.
- Eliminación de variables con alta proporción de valores faltantes: en los casos donde una variable presente un porcentaje elevado de `NA` y aporte poca información al modelo, se considerará su eliminación.

2) Codificación de Variables: Debido a la presencia de variables de distintos tipos, se emplearán diferentes estrategias de codificación:

- Variables numéricas: se utilizarán sin codificación adicional.
- Variables categóricas nominales: se empleará *One-Hot Encoding* para evitar la introducción de relaciones inexistentes entre categorías.
- Variables categóricas ordinales: se utilizará *Label Encoding*, respetando el orden natural de sus niveles (por ejemplo: *Poor* $<$ *Fair* $<$ *Typical* $<$ *Good* $<$ *Excellent*).

C. Paradigma de Aprendizaje Seleccionado

El equipo de trabajo decidió abordar el problema bajo el paradigma de aprendizaje supervisado. De acuerdo con el curso, este tipo de aprendizaje se emplea cuando se cuenta con ejemplos donde tanto las entradas como las salidas están claramente identificadas, permitiendo que el modelo aprenda la relación entre ellas con el fin de realizar predicciones sobre nuevos datos.

En este contexto, el proyecto se plantea como un problema de regresión, dado que la variable objetivo corresponde a un valor numérico continuo (el precio de una vivienda). El curso establece que, cuando la salida a predecir es un número real, el enfoque adecuado es la regresión. Por lo tanto, esta configuración se considera apropiada para el tipo de predicción que se desea realizar. A partir de esta definición, el equipo procederá posteriormente a la construcción e implementación del modelo de regresión correspondiente.

II. Estado del arte

Artículo I

Han, Y. (2023). *Price Prediction of Ames Housing Through Advanced Regression Techniques.* BCP Business & Management EMFRM, 38, 1966–1974.

Este trabajo aborda el problema de predicción del precio de venta de viviendas utilizando la base de datos *Ames Housing*, la misma empleada en la competencia *House Prices – Advanced Regression Techniques* de Kaggle. El estudio aplica un paradigma de aprendizaje supervisado de tipo regresión, orientado a estimar el valor de la variable continua *SalePrice*.

El autor implementa y compara diferentes técnicas de aprendizaje automático, entre ellas *LASSO*, *Elastic Net*, *Gradient Boosting*, *XGBoost*, *LightGBM* y un modelo ensamblado (*Stacked Model*). El proceso incluyó una exhaustiva ingeniería de características, imputación de valores faltantes y normalización de variables.

Como metodología de validación, se empleó una validación cruzada de 5 particiones (5-fold CV) sobre el conjunto de entrenamiento. La métrica principal fue el *Root Mean Squared Logarithmic Error (RMSLE)*, definida como:

$$RMSLE = \sqrt{\frac{1}{n}\sum_{i=1}^{n}\left(\log(\hat{y}_i + 1) - \log(y_i + 1)\right)^2} \qquad (1)$$

Esta métrica penaliza de forma equilibrada los errores relativos en precios altos y bajos.

En cuanto a resultados, el mejor desempeño se obtuvo con el modelo *Gradient Boosting*, con un RMSLE de 0.046, seguido de *XGBoost* y *LightGBM* con valores cercanos. El autor concluye que los métodos basados en *boosting* ofrecen el mejor equilibrio entre precisión y eficiencia para este conjunto de datos.

Artículo II

Harsora, H., Ogunleye, B., & Shobayo, O. (2023). *House Price Prediction Using Machine Learning Algorithms. Analytics, 3(1)*, 1–20.

Este estudio aborda el problema de la predicción del precio de viviendas utilizando el *Ames Housing Dataset*, disponible en Kaggle, el mismo conjunto de datos empleado en la competencia *House Prices – Advanced Regression Techniques*. Los autores aplican un paradigma de aprendizaje supervisado basado en regresión, enfocado en estimar el valor continuo de la variable *SalePrice*.

El trabajo compara el desempeño de cinco técnicas de aprendizaje automático: Regresión Lineal Múltiple, Red Neuronal Multicapa (*MLP*), *Random Forest Regressor*, *Support Vector Regression (SVR)* y *XGBoost*. Tras un proceso de limpieza y codificación de variables, se realizó ingeniería de características para optimizar el rendimiento predictivo de los modelos.

Como metodología de validación, se aplicó validación cruzada (*Cross-Validation*) para garantizar la estabilidad del modelo y reducir el riesgo de sobreajuste. Las métricas utilizadas fueron el coeficiente de determinación (R^2), R^2 ajustado, Error Absoluto Medio (*MAE*), Error Cuadrático Medio (*MSE*) y su raíz (*RMSE*), indicadores que permiten evaluar la precisión y generalización del modelo.

En cuanto a resultados, el algoritmo *XGBoost* alcanzó el mejor desempeño con un R^2 de 0.93, un *MSE* de 0.001, un *MAE* de 0.084 y una precisión promedio del 88.94% en la validación cruzada. Los autores concluyen que *XGBoost* constituye el modelo más estable y preciso para la predicción de precios inmobiliarios en el conjunto de datos de *Ames*, superando consistentemente a los métodos tradicionales de regresión.

Artículo III

G. Naga Satish, Ch. V. Raghavendran, M.D.Sugnana Rao, Ch.Srinivasulu. (2019). *House Price Prediction Using Machine Learning*
Este artículo aborda el problema de la predicción de precios de viviendas y, para seleccionar un método adecuado, compara y analiza diferentes enfoques de modelado. Los autores emplean *Lasso Regression* como modelo principal debido a que constituye una metodología adaptable y con fundamentos probabilísticos para la selección de variables. Para efectos de comparación, también se utilizaron modelos como *XGBoost* y redes neuronales; sin embargo, los resultados presentados muestran que *Lasso Regression* ofrece un mejor desempeño, ya que alcanza una mayor precisión y supera de manera consistente a los otros modelos evaluados.

Lasso Regression es una técnica que busca minimizar el error de predicción imponiendo una restricción sobre los parámetros del modelo. Esta restricción reduce los coeficientes de regresión hacia cero al exigir que la suma del valor absoluto de dichos coeficientes sea menor que un valor fijo (λ).

En la práctica, este procedimiento limita la complejidad del modelo y permite excluir automáticamente aquellas variables cuyos coeficientes se reducen a cero. Esto convierte a *Lasso* en un método eficiente tanto para la predicción como para la selección de características [7]

Artículo IV

Iwona Foryś. (2022). *Aprendizaje automático en el análisis de precios de la vivienda: modelos de regresión versus redes neuronales*. Instituto de Economía y Finanzas, Universidad de Szczecin, Mickiewicza 64, 71-101 Szczecin, Polonia
En este estudio se emplea una metodología tradicional en contraste con enfoques basados en redes neuronales para abordar el problema de la predicción de precios de viviendas. Históricamente, ha existido debate en torno a la Valoración de Mercado Automatizada (VMA), motivo por el cual durante mucho tiempo se han preferido metodologías tradicionales debido al limitado conocimiento y confianza en los modelos automatizados. En particular, el estudio compara el desempeño de una regresión múltiple frente a una red neuronal.

Ambos modelos fueron entrenados con el mismo conjunto de datos. La regresión múltiple alcanzó un desempeño de calidad media, mientras que la red neuronal no logró obtener la mejor calidad esperada, incluso después de normalizar los datos en el rango $(0, 1)$. Los autores destacan que, a pesar de que la literatura suele señalar que las redes neuronales presentan una ventaja significativa sobre los modelos tradicionales, en este caso ambos enfoques mostraron una tendencia marcada a sobreestimar los precios de las viviendas en la mayoría de los casos. Esto sugiere que, para este conjunto de datos y bajo las condiciones evaluadas, ambos métodos pueden converger hacia resultados similares.

III. Entrenamiento y evaluación de los modelos

A. Configuración experimental

Antes de describir cada uno de los modelos evaluados, es importante señalar que se adoptó una metodología de validación unificada para garantizar comparabilidad entre los resultados. En todos los casos se aplicó validación cruzada con cinco particiones (*5-fold cross validation*), lo cual permitió entrenar y evaluar cada modelo sobre distintos subgrupos del conjunto de datos, reduciendo la varianza asociada a una única división *train–test*. Tras este proceso, cada modelo fue reentrenado utilizando la totalidad del conjunto de entrenamiento y posteriormente evaluado sobre el conjunto de prueba independiente. Para todos los modelos se emplearon de manera consistente tres métricas de desempeño: el error cuadrático medio (RMSE), el error absoluto medio (MAE) y el coeficiente de determinación (R^2), con el fin de obtener una valoración completa de la calidad predictiva de las estimaciones.

I. CatBoost Base

Para esta etapa del trabajo se entrenaron diversos modelos con el fin de seleccionar aquellos que ofrecieran el mejor desempeño según sus métricas de evaluación. Como punto de partida, se entrenó y evaluó un modelo *CatBoost Base*, un modelo basado en el ensamble de árboles de decisión, tanto con transformación logarítmica como sin ella, para analizar como afectaba esta transformación a los resultados. El modelo fue configurado con 1000 iteraciones, una tasa de aprendizaje de 0.05, una profundidad de 6 y optimización basada en el error cuadrático medio (MSE).

II. XGBoost Base

Por otro lado, se construyó un modelo base utilizando *XGBoost*, un algoritmo de aprendizaje supervisado que pertenece a la familia de métodos basados en el ensamble de árboles de decisión, al igual que el modelo *CatBoost Base* empleado previamente. XGBoost implementa la técnica de *gradient boosting*, en la cual los árboles se agregan de manera secuencial y cada nuevo árbol intenta corregir los errores cometidos por los anteriores mediante la optimización del gradiente de la función de pérdida. Esta estrategia permite capturar relaciones complejas y mejorar progresivamente la capacidad predictiva del modelo.

Para este modelo se utilizó una configuración inicial compuesta por 1000 árboles, una tasa de aprendizaje moderada y una profundidad media, parámetros que permiten un equilibrio entre flexibilidad y control del sobreajuste.

III. Ridge Regressor

Para el modelo paramétrico se eligió un *Ridge Regressor*, una variante de la regresión lineal que resulta especialmente útil cuando existen variables altamente correlacionadas entre sí. En estos casos, la regresión lineal tradicional puede producir coeficientes inestables o excesivamente grandes. Ridge soluciona este problema añadiendo una penalización que reduce la magnitud de los coeficientes, evitando que el modelo se ajuste demasiado al ruido del conjunto de datos. Esta regularización permite obtener un modelo más estable, con menor varianza y mejor capacidad de generalización. En esencia, Ridge busca un balance entre ajustar bien los datos y

mantener un modelo sencillo y robusto. [9]
El modelo *Ridge Regressor* fue configurado utilizando un parámetro de regularización de $\alpha = 1.0$. Este valor controla la intensidad con la que se penalizan los coeficientes del modelo, lo cual ayuda a reducir problemas de multicolinealidad y a evitar el sobreajuste. La elección de $\alpha = 1.0$ proporciona un equilibrio adecuado entre la complejidad del modelo y su capacidad de generalización, ya que al aumentar α estaríamos aumentando la penalización y castigando más a los coeficientes, por lo que decidimos tomar un valor que nos dé más equilibrio, permitiendo obtener predicciones más estables frente a variaciones en los datos.

IV. KNN Regressor

Para el modelo no paramétrico seleccionamos un *KNN Regressor*, un método que estima el valor de una instancia en función de los vecinos más cercanos dentro del conjunto de entrenamiento. En problemas de regresión, el algoritmo identifica los k puntos más próximos en el espacio de características y calcula el valor objetivo como el promedio de dichos vecinos. Esta idea es análoga a su aplicación tradicional en clasificación, donde se asigna la clase predominante del vecindario; sin embargo, en regresión el resultado corresponde a un valor continuo [10]

Antes de entrenar el modelo final, utilizamos *GridSearchCV* con el fin de determinar el número óptimo de vecinos (k). Este procedimiento evalúa de forma sistemática diferentes configuraciones de hiperparámetros mediante validación cruzada, entrenando múltiples variantes del modelo y seleccionando aquella que obtiene el mejor desempeño según la métrica RMSE negativa utilizada en el proceso de búsqueda. Una vez identificado el mejor valor de k, se construyó el modelo final de *KNN Regressor* empleando dicha configuración.

V. MLP Regressor

Para el modelo basado en redes neuronales se empleó un *MLP Regressor* (Multilayer Perceptron), un tipo de red neuronal artificial que aprende a partir de capas de neuronas conectadas entre sí. A diferencia del perceptrón simple, que únicamente puede resolver problemas lineales, un MLP incorpora varias capas intermedias que permiten capturar relaciones más complejas entre las variables. Cada capa realiza combinaciones lineales de las entradas y aplica posteriormente una función de activación no lineal, lo cual convierte al MLP en una extensión del modelo lineal generalizado. Después de atravesar múltiples capas, la red obtiene una representación interna del problema y produce un valor final como predicción. Gracias a esta estructura jerárquica, los MLP pueden emplearse tanto para clasificación como para regresión, adaptándose adecuadamente a patrones no lineales presentes en los datos [11]

Antes de entrenar el modelo, fue necesario escalar las variables predictoras mediante un *StandardScaler*, debido a que los MLP son sensibles a la magnitud de los valores de entrada y requieren que todas las características estén en una escala comparable. El modelo se configuró con dos capas ocultas de tamaños $(64, 32)$, utilizando la función de activación `relu` y el optimizador `adam`. Asimismo, se estableció una tasa de aprendizaje inicial pequeña y un parámetro de regularización α para mitigar el sobreajuste. Se habilitó el *early stopping* con validación interna, lo que permite detener el entrenamiento cuando el rendimiento deja de mejorar, favoreciendo una mejor capacidad de generalización.

Hiperparámetro	**Malla evaluada**
α	{0.01, 0.1, 1, 10, 100}

TABLE I
HIPERPARÁMETROS EVALUADOS PARA RIDGE REGRESSION.

Hiperparámetro	**Malla evaluada**
Número de vecinos (k)	{3, 5, 7, 9, 11, 13}

TABLE II
HIPERPARÁMETROS EVALUADOS PARA EL MODELO KNN REGRESSOR.

Hiperparámetro	**Malla evaluada**
hidden_layer_sizes	{(64,32), (128,64,32)}
activation	{relu, tanh}
solver	{adam}
learning_rate_init	{0.0001, 0.0005, 0.001}
α	{0.0001, 0.0005, 0.001}
max iter	{2000, 3000}
validation fraction	{0.1}
no iter no change	{10, 20}

TABLE III
HIPERPARÁMETROS EVALUADOS PARA EL MODELO MLP REGRESSOR.

Hiperparámetro	**Malla evaluada**
n_estimators	{500, 800, 1000}
learning_rate	{0.01, 0.05, 0.1}
max_depth	{4, 5, 6}
subsample	{0.7, 0.8, 1.0}
colsample_bytree	{0.7, 0.8, 1.0}

TABLE IV
HIPERPARÁMETROS EVALUADOS PARA EL MODELO XGBOOST.

Hiperparámetro	**Malla evaluada**
iterations	{500, 800, 1000}
learning_rate	{0.01, 0.05, 0.1}
depth	{4, 6, 8}
loss_function	{RMSE}

TABLE V
HIPERPARÁMETROS EVALUADOS PARA EL MODELO CATBOOST.

B. Resultados del entrenamiento de modelos

Para consolidar los resultados de todos los modelos evaluados, se construyó un `DataFrame` que resume las métricas de

desempeño obtenidas para cada técnica. Se incluyeron tanto los modelos base como sus versiones optimizadas mediante ajuste de hiperparámetros, y se evaluó su rendimiento en dos escenarios: con y sin transformación logarítmica de la variable objetivo.

Las métricas consideradas fueron RMSE, MAE y R^2, calculadas tanto en validación cruzada (CV) como en el conjunto de prueba. Esto permite comparar de manera directa la capacidad predictiva de cada modelo y analizar cómo el ajuste de hiperparámetros y la transformación de la variable objetivo afectan su desempeño.

Modelo	Transformación	RMSE CV	MAE CV	R^2 CV	RMSE Test	MAE Test	R^2 Test
CatBoost	Sin Log	26,979.82	15,107.88	0.878	27,121.14	15,658.72	0.904
CatBoost	Con Log	27,756.76	14,987.46	0.871	26,966.03	15,315.50	0.905
XGBoost	Sin Log	24,650.90	15,431.38	0.921	24,650.90	15,431.38	0.921
XGBoost	Con Log	29,165.55	15,728.51	0.865	25,875.55	15,974.09	0.913
Ridge	Con Log	35,923.76	20,796.05	0.784	29,290.45	17,967.36	0.888
Ridge	Sin Log	40,869.67	17,781.56	0.720	29,290.45	17,967.36	0.888
KNN	Sin Log	43,593.26	28,310.78	0.681	43,042.09	25,778.19	0.758
KNN	Con Log	44,614.67	28,702.67	0.666	44,788.28	26,410.58	0.738
MLP	Sin Log	132,208.5	91,900.73	-1.930	40,514.76	28,411.30	0.786
MLP	Con Log	8.59e10	2.52e9	-1.24e12	486,935.20	175,906.19	-29.91

TABLE VI
RESUMEN DE MÉTRICAS DE DESEMPEÑO PARA TODOS LOS MODELOS EVALUADOS.

De acuerdo con la tabla anterior, se observan algunas tendencias claras:

- Los modelos basados en ensamble de árboles, como **CatBoost** y **XGBoost**, presentan el mejor desempeño general, con valores de RMSE más bajos y R^2 más altos tanto en CV como en el conjunto de prueba.
- Los modelos paramétricos (**Ridge**) y no paramétricos (**KNN**) presentan un desempeño razonable, aunque inferior al de los modelos basados en árboles.
- La red neuronal (**MLP**) mostró gran sensibilidad a la transformación logarítmica. En el caso con log, los resultados fueron extremadamente inestables, ya que nuestro dataset es pequeño y diverge muy rápido, indicando problemas de convergencia o de sobreajuste.
- Para la mayoría de los modelos, la transformación logarítmica no siempre mejora el desempeño; de hecho, en XGBoost y KNN, los valores de RMSE y MAE son ligeramente mejores sin la transformación.

Para evaluar el efecto del ajuste de hiperparámetros sobre el rendimiento de los modelos, se generaron visualizaciones comparativas usando gráficos de barras. Se construyeron dos barras por cada modelo, correspondientes a su versión base y a su versión ajustada, considerando RMSE Test como métrica de desempeño.
Se compararon los resultados tanto en la escala original como en la escala logarítmica de la variable objetivo. Esta representación permitió identificar de manera clara qué modelos mejoraron con el tuning, facilitando la comparación directa entre configuraciones y destacando cuáles se beneficiaron más del ajuste de hiperparámetros.

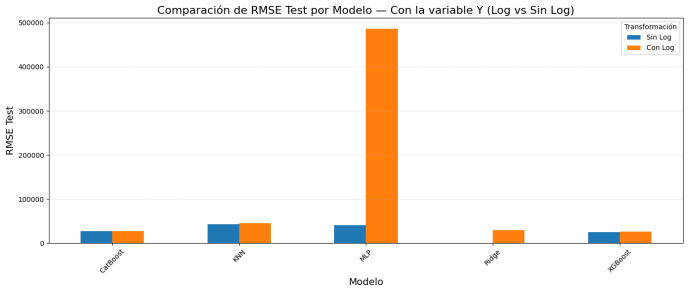

Fig. 2. Comparación de RMSE Test por modelo

De acuerdo con el gráfico se puede observar que:

- Los modelos basados en ensamble de árboles, como **CatBoost** y **XGBoost**, presentan el mejor desempeño general, con valores de RMSE más bajos en test y su transformación logarítmica no marca mucha diferencia.
- El modelo de red neuronal **MLP** si bien no tiene el mejor desempeño, manejarla sin transformación logarítmica nos da un valor en test muchisimo mejor que al transformarla.
- El modelo paramétrico **Ridge** presenta gran diferencia en el valor del RMSE en Test al momento de aplicarle su transformación logarítmica

Para complementar el análisis cuantitativo del desempeño de los modelos, se generaron gráficos de dispersión comparando las predicciones frente a los valores reales para cada modelo evaluado: **CatBoost**, **XGBoost**, **Ridge**, **KNN** y **MLP**.
Cada modelo se presentó en dos versiones: sin transformación logarítmica de la variable objetivo y con log, pero regresando los valores a su escala original para facilitar la interpretación. En cada gráfico, los puntos representan las predicciones del modelo y se contrastan con una línea diagonal que indica la predicción perfecta ($y = \hat{y}$).
Esta visualización permite identificar de manera intuitiva cómo se aproximan las predicciones a los valores reales y evaluar el impacto del ajuste de hiperparámetros, mostrando de forma clara si los modelos optimizados lograron mejorar la precisión respecto a sus versiones base, más allá de las métricas numéricas como RMSE, MAE o R^2.

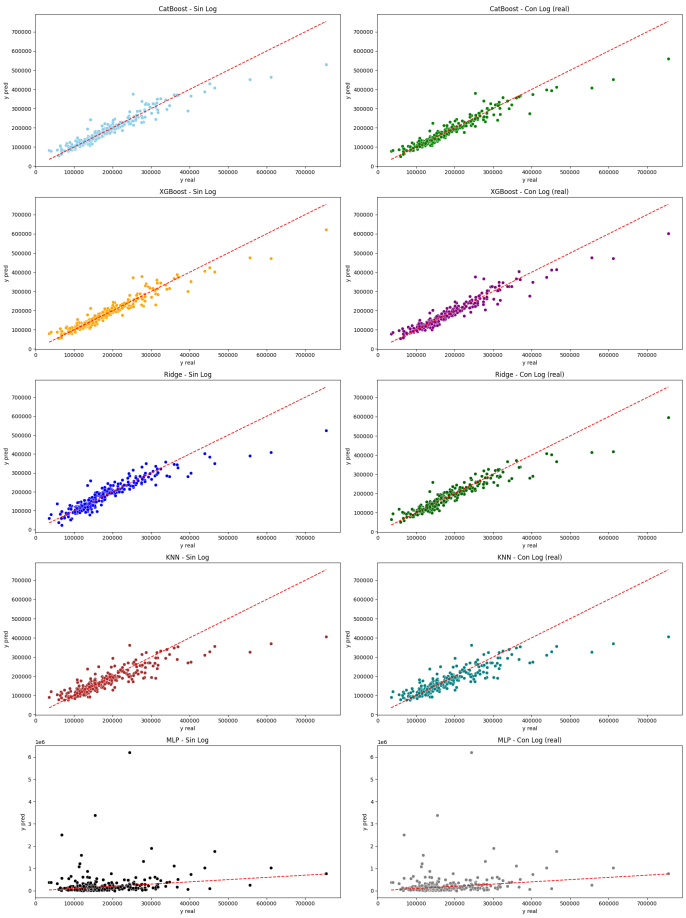

Fig. 3. Gráficos de dispersión de valores reales vs predicciones

De acuerdo a los gráficos podemos afirmar que:

1) **Modelos basados en ensamble de árboles (CatBoost y XGBoost)**
   Los puntos están muy cerca de la línea diagonal, lo que indica predicciones precisas.
   No se aprecia mucha diferencia entre la versión con y sin log, lo que sugiere que estos modelos capturan bien las relaciones no lineales del dataset sin necesidad de transformar la variable objetivo.
   Son los modelos con menor dispersión y menor error, coincidiendo con los valores bajos de RMSE que obtuvimos en la tabla.
2) **Ridge (modelo paramétrico)**
   Los puntos muestran una tendencia lineal, pero hay más dispersión alrededor de la diagonal que en CatBoost y XGBoost.
   La versión con log mejora ligeramente el ajuste para valores pequeños, pero aún tiene errores mayores en viviendas de alto precio.
   Podemos decir que Ridge captura relaciones lineales, pero tiene dificultades con valores extremos o relaciones no lineales complejas.
3) **KNN (modelo no paramétrico)**
   Predicciones bastante cercanas a la diagonal para valores medios, pero se dispersan más en valores altos y bajos.
   La versión con log no mejora significativamente.
   Se refleja que KNN depende de la densidad local de los datos y puede sobreestimar o subestimar en zonas con pocos vecinos similares.
4) **MLP (red neuronal)**
   La versión sin log tiene cierta dispersión, pero predice algunos valores medianamente bien.
   La versión con log produce valores muy distorsionados, con algunas predicciones extremadamente grandes o negativas (lo que coincide con las métricas negativas que obtuvimos de RMSE y R^2).
   Esto indica que la red neuronal no se entrenó de manera estable con la transformación log, por un dataset tan pequeño.

IV. Reducción de dimensión

En esta fase del proyecto se aplicaron distintas técnicas para reducir la dimensionalidad del conjunto de datos y evaluar cómo esto impacta en los modelos predictivos. Primero, se trabajó con PCA (Análisis de Componentes Principales). Todas las variables se normalizaron para que estuvieran en la misma escala, y luego se revisó cuánta varianza explicaba cada componente. Después de varias pruebas, se decidió conservar los componentes necesarios para mantener al menos el 90% de la varianza total. Esto permitió reducir el número de variables manteniendo casi toda la información útil. Se compararon las columnas originales con las reducidas y se observó que se eliminó un número mínimo de variables sin perder información relevante.

Por otro lado, se implementó UMAP (Uniform Manifold Approximation and Projection) para obtener representaciones más compactas, manteniendo la estructura interna de los datos. Se probaron distintas dimensiones, desde 2D hasta 10D. La versión de 10 dimensiones se utilizó para entrenar los modelos, ya que conserva la mayor parte de la información original (87%–91%) y permite un mejor desempeño predictivo. Dimensiones menores, como 2D o 3D, se usaron más para visualización. Se notó que reducir demasiado las dimensiones afecta el rendimiento: por ejemplo, CatBoost y XGBoost pasaron de $R^2 \approx 0.90$–0.92 con los datos originales a $R^2 \approx 0.71$ con UMAP 10D.

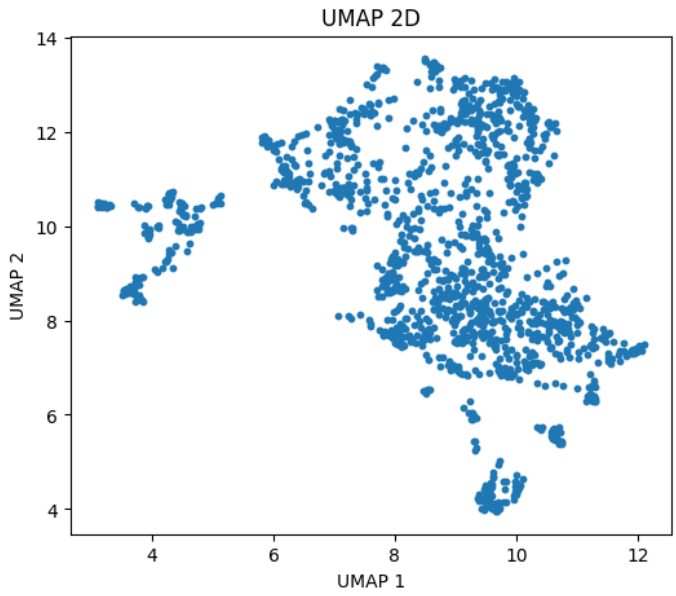

Fig. 4. Gráfica de UMAP 2D

También se hizo un análisis individual de cada variable para identificar cuáles podrían ser eliminadas sin afectar la información:

- **Variables numéricas poco correlacionadas:** si la correlación absoluta con la variable objetivo era menor a 0.1, se consideraba que aportaba poca información y podía eliminarse.
- **Todas las variables con baja información mutua:** si el valor era menor a 0.01, la variable aportaba muy poca información y era candidata a eliminar.

Con este análisis, se identificaron 29 variables candidatas a eliminar, mientras que PCA y UMAP sugerían eliminar 27 variables, mostrando que los resultados son consistentes y la reducción es segura sin perder información crítica.

En conclusión, la elección del número de componentes en PCA y UMAP requiere un balance entre reducción de dimensionalidad y retención de información. Para nuestra base de datos, 10 dimensiones en UMAP resultaron adecuadas para maximizar el rendimiento predictivo sin perder información relevante. Este proceso también demuestra la importancia de evaluar individualmente cada variable y probar distintas configuraciones antes de decidir la reducción final, sobre todo cuando se trabaja con bases de datos relativamente pequeñas pero con posible expansión futura.

V. Conclusiones

A partir del análisis realizado y los experimentos con distintos modelos de regresión para predecir el precio de viviendas, podemos extraer las siguientes conclusiones:

- La Ridge Regression mostró un desempeño aceptable, especialmente para capturar relaciones lineales, pero presentó dificultades con valores extremos. El KNN Regressor fue preciso en rangos de precios medios, pero tuvo mayor dispersión para valores altos o bajos, reflejando su dependencia de la cantidad de vecinos similares en una región
- La red neuronal mostró problemas de convergencia y sensibilidad a la transformación logarítmica, especialmente con la versión transformada, lo que generó predicciones inestables. Esto indica que, con conjuntos de datos relativamente pequeños, los MLP requieren un ajuste cuidadoso de hiperparámetros y un preprocesamiento exhaustivo para entrenar de manera confiable.
- La aplicación de la transformación logarítmica a la variable objetivo no siempre mejora el desempeño; su impacto depende del modelo. En los ensambles de árboles no se observó beneficio significativo, mientras que en Ridge y MLP, los resultados variaron considerablemente, sugiriendo que la elección de aplicar log debe evaluarse dependiendo del modelo que se esté entrenando y evaluando.
- Las técnicas de PCA y UMAP permitieron reducir el número de variables manteniendo la mayoría de la información relevante. Se identificaron entre 27 y 29 variables candidatas a eliminar sin comprometer significativamente el desempeño de los modelos. Sin embargo, la reducción excesiva (por ejemplo, a 2D o 3D en UMAP) afectó la precisión de la predicción, mostrando la importancia de balancear simplificación y retención de información.
- La imputación de valores faltantes, la codificación adecuada de variables categóricas y la estandarización fueron pasos clave para lograr modelos robustos. Este proceso asegura que los modelos puedan aprender correctamente las relaciones subyacentes y evita sesgos o errores derivados de datos incompletos o mal representados.
- Para predicción de precios de vivienda en datasets similares, se recomienda priorizar modelos de ensamble de árboles con ajuste de hiperparámetros, acompañados de una exploración cuidadosa de la reducción de dimensionalidad y un preprocesamiento exhaustivo, como bien vimos los modelos XGBoost y CatBoost tuvieron el mejor desempeño. Los modelos lineales o basados en vecinos pueden ser útiles como referencia o para interpretabilidad, pero generalmente no alcanzan la precisión de los métodos de boosting.

Referencias

[1] A. Cook, "Categorical Variables," *Kaggle*, [Online]. Available: https://www.kaggle.com/code/alexisbcook/categorical-variables

[2] Kaggle, "House Prices – Advanced Regression Techniques," *Kaggle Competition Overview*, [Online]. Available: https://www.kaggle.com/competitions/house-prices-advanced-regression-techniques/overview

[3] J. Darias, "Introducción al Aprendizaje Automático – Unidad 1," *Intro_ML_2025*, [Online]. Available: https://jdariasl.github.io/Intro_ML_2025/titles/U1_description.html

[4] Y. Han, "Price Prediction of Ames Housing Through Advanced Regression Techniques," *BCP Business & Management EMFRM*, vol. 38, pp. 1966–1974, 2023. [Online]. Available: https://www.researchgate.net/publication/369437029_Price_Prediction_of_Ames_Housing_Through_Advanced_Regression_Techniques

[5] H. Harsora, B. Ogunleye, and O. Shobayo, "House Price Prediction Using Machine Learning Algorithms," *Analytics*, vol. 3, no. 1, pp. 1–20, 2023. [Online]. Available: https://www.mdpi.com/2813-2203/3/1/3

[6] G. Naga Satish, Ch. V. Raghavendran, M. D. Sugnana Rao, and Ch. Srinivasulu, "House Price Prediction Using Machine Learning," *International Journal of Innovative Technology and Exploring Engineering (IJITEE)*, vol. 8, no. 9, pp. 717–721, Jul. 2019. DOI: https://doi.org/10.35940/ijitee.I7849.078919.

[7] J. Ranstam and J. A. Cook, "LASSO regression," *British Journal of Surgery*, vol. 105, no. 10, p. 1348, 2018. https://doi.org/10.1002/bjs.10895
[8] I. Foryś, "Aprendizaje automático en el análisis de precios de la vivienda: modelos de regresión versus redes neuronales," *Real Estate Management and Valuation*, vol. 27, no. 4, pp. 34–45, 2019. DOI: https://doi.org/10.2478/remav-2019-0030.
[9] G. C. McDonald, "Ridge Regression," *Wiley Interdisciplinary Reviews: Computational Statistics*, 2009. DOI: https://doi.org/10.1002/wics.14.
[10] M. Steinbach and P.-N. Tan, "kNN: k-Nearest Neighbors," in *The Top Ten Algorithms in Data Mining*, 2009. https://www.taylorfrancis.com/chapters/edit/10.1201/9781420089653-15/knn-nearest-neighbors-michael-steinbach-pang-ning-tan
[11] Y. Qin, C. Li, X. Shi, and W. Wang, "MLP-Based Regression Prediction Model for Compound Bioactivity," *School of Statistics and Mathematics, Zhejiang Gongshang University*, 2019. https://www.frontiersin.org/journals/bioengineering-and-biotechnology/articles/10.3389/fbioe.2022.946329/full